Article | Received 22 November 2025; Revised 5 February 2026; Accepted 17 March 2026; Published 31 March 2026
https://doi.org/10.55092/rl20260008

MoDex: planning high-dimensional dexterous control via learning neural internal models

Tong Wu[1], Shoujie Li[2], Chuqiao Lyu[1], Kit-Wa Sou[1], Wang-Sing Chan[3] and Wenbo Ding[1,*]

[1] Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2] School of Mechanical and Aerospace Engineering, Nanyang Technology University, Singapore
[3] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pennsylvania, USA

* Correspondence author; E-mail: ding.wenbo@sz.tsinghua.edu.cn.

Highlights:

- A neural internal model is introduced to learn intrinsic hand dynamics for high-dimensional control.
- Efficient in-hand manipulation and gesture generation are enabled based on the internal model.
- Extensive experiments in bot simulation and real-world supports the efficacy of our method.

Abstract: Controlling dexterous hands in high-dimensional action spaces remains challenging, whereas humans naturally achieve such control through internal models that predict and adapt body dynamics. Inspired by this concept, we present MoDex, a neural internal model framework that learns the intrinsic dynamics of dexterous hands via coupled forward and inverse networks. MoDex enables efficient bidirectional planning through integration with a Cross-Entropy Method (CEM) optimizer, achieving superior data efficiency and faster decision-making compared to model-free and model-based baselines. Furthermore, the pretrained internal model serves as a transferable module: when combined with an external dynamics model, it improves data efficiency in in-hand object manipulation, and when coupled with a large language model (LLM), it enables few-shot gesture generation in both simulation and the real world. Extensive experiments across multiple robotic hands demonstrate MoDex's versatility and effectiveness in high-dimensional control.

Keywords: model-based learning; dexterous manipulation; high-dimensional control

1. Introduction

The broad functional potential of dexterous hands has motivated extensive research in areas such as dexterous grasping [1–3], in-hand manipulation [4–7] and even musculoskeletal manipulation [8,9]. Despite this progress, controlling high degree-of-freedom (DoF) hands remains a major challenge, as the data required to train a policy grows with the action dimension [10]. Deep reinforcement learning (DRL), a trial-and-error-based approach, struggles with this problem due to its need for an unreasonably large number of interactions in high-dimensional settings [11].

One common approach to addressing this challenge is reducing the dimensionality of the action space by leveraging synergy, a concept from neuroscience that describes the coordinated control of multiple actuators [12]. Previous works have applied this idea to enhance DRL efficiency by exploiting the action redundancy in specific tasks through Manifold learning [13] and covariance analysis [14,15]. While synergy-based methods can accelerate learning, they inherently shape a task-related reduced action space, leaving the transferring of muscle synergies largely unexplored [14]. Additionally, synergies themselves vary significantly depending on the task; for instance, chopstick using and cube rotating differ in both the pattern and quantity of synergies. These characteristics may affect the performance across different tasks.

The internal model, another concept from neuroscience, suggests that rather than mapping observations directly to actuator controls, the nervous system relies on a pair of models: a forward model and an inverse model [16,17]. The forward model predicts the outcomes of actions based on observed dynamics, while the inverse model infers the necessary actuator commands to achieve a target state. The concept of internal model was first introduced to explicitly represent control systems [18]. In biological motor control, growing evidence suggests that the human body maintains an internal model to predict and adapt movements [19,20]. Since real-time sensory feedback is often unreliable due to occlusions or noise, motor planning primarily relies on an internal model constructed from prior perception [16]. For instance, when vision is obstructed or the environment is dark, humans still accurately estimate hand position, a phenomenon best explained by an internal model [21]. Similarly, motor adaptation experiments reveal that humans adjust to external disturbances by continuously updating their internal models and, even after the disturbances are removed, momentarily retain compensatory responses [22]. This raises the question: can we endow robots with similar high-dimensional control capabilities using the internal model?

In this paper, we investigate whether the internal model can facilitate high-dimensional dexterous control. Unlike traditional model-based approaches that focus on learning environment dynamics [23,24], our approach, MoDex, addresses that the dexterous hand itself should be studied as an independent system. Specifically, a pair of NN-based forward model and inverse model are trained through random exploration, which provides robots with an understanding of hand dynamics. To accelerate the decision-making process, we propose a bidirectional planning method by integrating the neural internal model with a Cross-Entropy Method (CEM) [25] planner. Draw inspiration from neuroscience experiments, we evaluate MoDex on fingertip control to reach target positions using four dexterous hands in simulation. Results show that our approach is significantly more data efficient compared to model-free methods while also demonstrating faster planning compared to model-based baselines.

To further demonstrate its versatility, we utilize the pretrained neural internal model as a plug-and-play module for in-hand manipulation and few-shot gesture generation in both simulation and real world. In the first case, we factorize the environment dynamics into the pretrained internal model and an external dynamics model, leading to improved data efficiency in object reorientation. In the second case, we integrate MoDex to a large language model (LLM) by prompting it to generate cost functions. Our approach successfully generates a variety of gestures, such as “Scissorshand” and “Rock&Roll”. These examples indicate MoDex’s potential for diverse high-dimensional control tasks.

2. Methods

2.1. Learning neural internal models

Controlling a hand is exceptionally complex, yet humans manage it effortlessly, suggesting the presence of internal model planning [26]. This observation underscores the importance of explicitly modeling the body system for dexterous control. Accordingly, we propose to learn a neural internal model to enable high-dimensional dexterous control. The neural internal model consists of:

2.1.1. Forward model for dexterous hand dynamics

To achieve precise high-dimensional control, we introduce a neural forward model that approximates the dynamics of the dexterous hand. Specifically, given the current state s_t and action a_t, the forward model predicts the next state:

$$\hat{s}_{t+1} = f_\theta(s_t, a_t), \tag{1}$$

where $s_t, \hat{s}_{t+1} \in \mathbb{R}^H$ represents the hand state and the imagined next state respectively, which we define as the positions of all fingertips. $a_t \in \mathbb{R}^K$ denotes the action generated by actuators, with K being the action dimension. $f_\theta(\cdot)$ is a parametric function modeled as a multi-layer perceptron (MLP) with parameters θ.

To train the forward model, we collect a dataset of transitions $\mathscr{D} = \{(s_t, a_t, s_{t+1})_i\}$ using random exploration. The forward model is then trained to minimize a multi-step prediction loss:

$$L_{\text{forward}} = \sum_{i=0}^{S} \alpha^i ||s_{t+i+1} - f_\theta(\hat{s}_{t+i}, a_{t+i})||^2, \tag{2}$$

where S is the prediction horizon, controlling how many future steps are predicted, and α is a discount factor, weighting short-horizon errors more than long-horizon errors. $\hat{s}_t$ is set to s_t for initial alignment. This multi-step objective prevents compounding errors and encourages the model to generalize beyond short-term predictions, making it more reliable for downstream planning tasks.

In the field of dexterous control, sequential tasks like in-hand manipulation are a major area of focus. However, non-sequential tasks, where the static result is more important than the dynamic process, also play a crucial role. Examples include gesture generation and dexterous grasping. To address these quasi-static dexterous control tasks, we also propose a quasi-static forward model which disregards the current hand state s_t and utilizes only the action to predict the subsequent hand state:

$$\hat{s}_{t+1} = f'_{\theta'}(a_t). \tag{3}$$

To collect the training data, we reset the hand to its default state and record the subsequent states following the executed actions $\mathscr{D}' = \{(a_t, s_{t+1})\}_{i=0,1,2,\ldots}$. Multi-step MSE loss now collapses to MSE loss:

$$L'_{forward} = ||s_{t+1} - f'_{\theta'}(a_t)||^2. \tag{4}$$

2.1.2. Inverse model for action generation

The inverse model operates in the reverse direction of the forward model, predicting an action that moves the hand toward a desired target state. Formally, this process is defined as:

$$\hat{a}_t = g_\phi(s_t, s_T), \tag{5}$$

where g_ϕ is an MLP parameterized by ϕ, and $s_T \in \mathbb{R}^H$ represents the target hand state. However, action prediction is inherently ambiguous since multiple valid action sequences may lead to the same target state. Additionally, the accuracy of the predicted action depends on the information contained in the state. To address this, we model the predicted action as a Gaussian distribution rather than a deterministic output. Specifically, we define:

$$\hat{a}_t \sim \mathcal{N}\big(g_\phi(s_t, s_T), \Sigma)\big), \tag{6}$$

where Σ is a diagonal covariance matrix, and σ, its square root of diagonal vector is estimated from the training dataset as:

$$\sigma = \mathbb{E}_{(s_t, a_t, s_T)\sim\mathscr{D}}\Big[\big|a_t - g_\phi(s_t, s_T)\big|\Big]. \tag{7}$$

This distribution is then utilized to generate a set of action samples, which are refined through additional optimization as described in the following section.

We train the inverse model on the collected dataset $\mathscr{D}$ and use L1 loss function:

$$L_{\text{inverse}} = \mathbb{E}_{(s_t, a_t, s_T)\sim\mathscr{D}}\Big[\big|g_\phi(s_t, s_{t+t_0}) - a_t\big|\Big], \tag{8}$$

where t_0 is a hyperparameter controlling the target timestamp shift, which we set to 1 in this work.

Similarly, to address quasi-static situations, we generate action proposals directly from the target hand state:

$$\hat{a}_t \sim \mathcal{N}(g'_{\phi'}(s_T), \mathbb{E}_{\mathscr{D}}[|a_t - g'_{\phi'}(s_T)|]) \tag{9}$$

since the reset hand state is fixed. $\mathscr{D}'$ is leveraged to train the model:

$$L'_{inverse} = |g'_{\phi'}(s_{t+1}) - a_t|. \tag{10}$$

2.1.3. Bidirectional planning strategy

When executing complex movements, the central nervous system (CNS) decomposes motion into an initial rapid movement followed by corrective sub-movements [27,28]. Inspired by this, our bidirectional planning strategy first generates an initial action distribution using the inverse model, then refines it iteratively.

Given a target hand state, the inverse model produces an action distribution, which is extended over a planning horizon T_0 to initialize the sampling process. A CEM planner iteratively refines the action sequence by sampling candidate actions from the distribution, predicting their outcomes via the forward model, selecting the top-performing samples, and updating the distribution based on their mean and variance [25]. For sequential tasks, we further integrate Model Predictive Control (MPC) [23] by executing the first action. At the next step, we replan using the updated state. The whole process is shown in Algorithm 1.

Algorithm 1 Bidirectional Planning with MPC

Require: Current state s_t, target state s_T, inverse model g_ϕ, forward model f_θ, planning horizon T_0, number of CEM iterations N_{cem}, number of samples N_s, number of elites N_{elites}, filtering coefficient (moving average coefficient) β

while task is not complete **do**
  Initialize: Generate initial action distribution:
    $\hat{a} \sim \mathcal{N}(g_\phi(s_t, s_T), \Sigma)$
  Expand the distribution to planning horizon T_0: $\{\hat{a}\}_{i=t}^{t+T_0-1}$
  for iteration $i = 1$ to N_{cem} **do**
    Sample N_s candidate action sequences from the current distribution
    Initialize: $\hat{s}_t = s_t$
    for each candidate action sequence $\{a_t\}_{t=0}^{T_0-1}$ **do**
      Predict trajectory using forward model:
        $\hat{s}_{i+1} = f_\theta(\hat{s}_i, a_i), \quad \forall i \in [t, t+T_0-1]$
      Compute trajectory cost J based on task objective
    end for
    Select top N_{elites} action sequences with lowest costs (elites)
    Calculate the mean and variance of elite trajectories
    Update action distribution using moving average method
  end for
  Execute first action a_t
  Update state to s_{t+1}
end while

2.2. Applications of pretrained internal model

To demonstrate the versatility of a pretrained neural internal model, we explore two application cases: factorized dynamics learning for in-hand manipulation and few-shot gesture generation.

2.2.1. Factorized dynamics learning

Instead of modeling the hand and external environment jointly for in-hand manipulation, we adopt a hierarchical factorization. The complete system dynamics are decomposed into (i) a pretrained internal model and (ii) an external dynamics model capturing interactions with objects.

Given an action, we first predict the next hand state using the pretrained model (Equation (1)), which serves as an implicit prior since humans acquire such models through experience. The external dynamics model then updates the prediction by incorporating the interaction with the object:

$$s_{t+1}, x_{t+1} = f_\psi(\hat{s}_{t+1}, x_t), \tag{11}$$

where x_t and x_{t+1} are the external states before and after interaction, while $\hat{s}_{t+1}$ and s_{t+1} denote the hand states in imagination and after interaction. The external model f_ψ, parameterized by an MLP, learns the dynamics of interaction. Theoretically, a complete system model requires learning an input-output mapping of dimension $(H+O+K) \times (H+O)$ where H, O, and K are the dimensions of hand state, object state, and action space, respectively. When the action dimension increases, the dimension of the input-output mapping scales proportionally by a factor of $(H+O)$, leading to higher model complexity and data consumption. Instead, factorized dynamics reduces this to $(H+O) \times (H+O)$ by freezing the

internal model and only train the external model, effectively eliminating direct dependence on action dimension. To learn a specific task, we employ an online adaptive learning strategy [29] that iteratively rolls out actions, collects new transition data, and updates the models. Multi-step loss is applied to enhance long-horizon prediction robustness.

2.2.2. Few-shot gesture generation

We formulate gesture generation from text input as an optimization problem, leveraging LLMs' common knowledge and coding ability to provide the objective function without post-training.

Given a linguistic gesture request, we require an LLM to generate a heuristic function that quantifies the cost of the current hand state to the target gesture. For example, the "OK" gesture with a five-fingered hand can be described by:

$$\mathscr{J}(s) = -\|s_0 - s_1\|_2 + \sum_{i>2}(s_i \cdot \hat{n}_i), \tag{12}$$

where s_i represents the i-th fingertip position, and $\hat{n}_i$ denotes the preferred direction for straightening each finger. It encourages thumb-index contact while keeping other fingers extended. This cost function is presented in python function as shown in Figure 1. To enable few-shot ability, we integrate in-context learning [30], supplying the LLM with multiple pre-defined cost functions as exemplars. Finally, MoDex is used to generate the action for the target gesture, where the LLM-generated cost function serves as the objective.

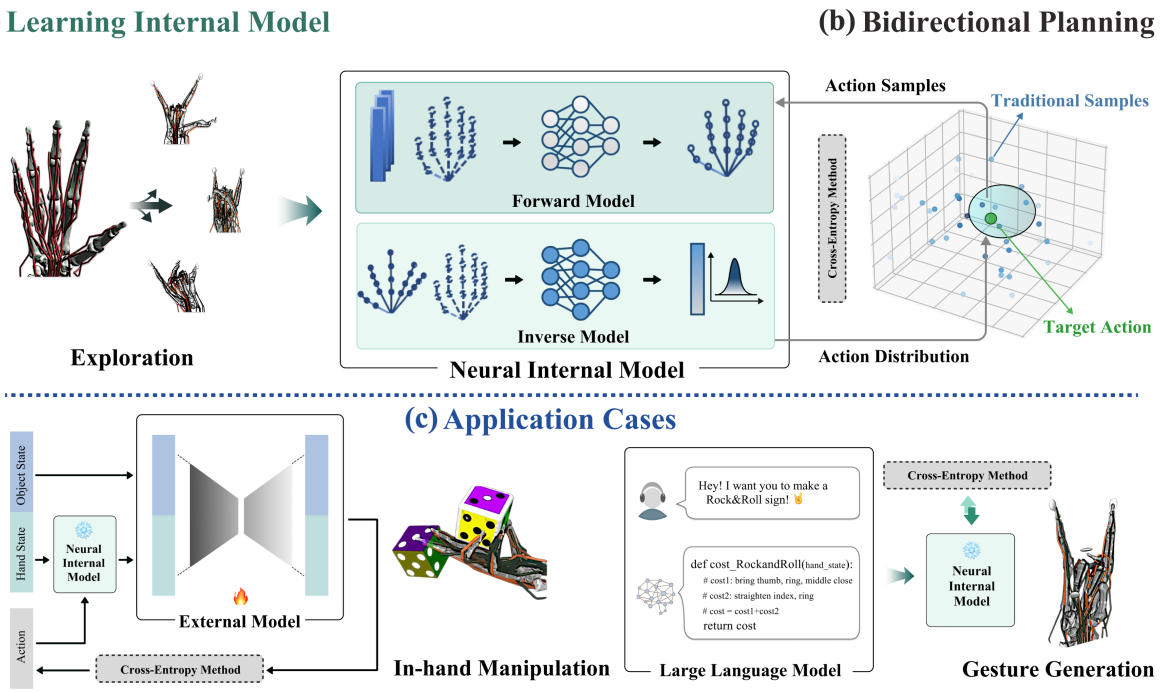


Figure 1. Method overview: We first explore the action space and collect dynamics data to train the internal model. Using these models, we employ CEM-based bidirectional planning to optimize actions. Applications: We decompose the system dynamics into a hand model and an external model, enhancing learning efficiency in in-hand manipulation task. Additionally, we leverage LLM to generate a cost function from textual inputs, guiding action optimization for gesture generation.

2.3. Evaluation protocols

2.3.1. Fingertip reach

We evaluate a fingertip reach task in simulation, inspired by neuroscience studies, using four dexterous robotic hands from IsaacGym [31] and Myosuite [9], which are listed in the Table 1. The objective is to move each fingertip to its designated spatial target. Two experimental settings are considered: (1) Quasi-static—a single-step reaching scenario, and (2) Sequential—a multi-step reaching task requiring dynamic coordination. Performance is evaluated over 100 randomized episodes using three metrics: S.R. (Success Rate), defined as the percentage of episodes achieving a target distance threshold; R.E. (Reach Error), the mean Euclidean distance between fingertips and targets; and P.S. (Planning Samples), the average number of samples utilized per planning step.

Table 1. Dexterous hands.

Name	Driven	Actuation	NoF	DoF	Dimension
Robotiq	Joint-driven	Fully-actuated	3	11	11
Allegro	Joint-driven	Fully-actuated	4	16	16
Shadowhand	Joint-driven	Under-actuated	5	24	20
Myohand	Tendon-driven	Over-actuated	5	23	39

We compare MoDex against two categories of baselines: (1) Model-free methods—Soft Actor-Critic (SAC) [32] and the synergy-based approach (SAR) [14]; (2) Model-based methods—Forward Model+Random Shooting (FM+RS), Forward Model+Batch Gradient Descent (FM+BGD), and FM+CEM [29]. For model-free methods, policies are learned online. Model-based methods, however, involve a training phase where data is collected using a random policy to train a forward model of the hand. In MoDex, the same data is also used to train an inverse model. During evaluation, the forward model is frozen, and different planning methods are applied.

2.3.2. In-hand manipulation

We evaluate factorized dynamics learning to three in-hand manipulation tasks using Myohand [9]:

- Reorient target2. A Myohand reorients a cube to one of two goal orientations, $\pm\frac{\pi}{2}$ around z axis. Each episode starts with a randomly initialized goal and cube position. The task is completed when $\cos(\text{rot}_{\text{cube}}, \text{rot}_{\text{goal}}) > 0.95$ and $|\text{pos}_{\text{cube}} - \text{pos}_{\text{goal}}| < 0.075$.
- Reorient 8object. A Myohand manipulates one of four geometries—ellipsoid, cuboid, cylinder, or capsule—each with two scaled variants. The object pose and goal orientation are randomly initialized at the start of each episode. The object is randomly selected from the 8 geometry variants.
- Reorient 100object. A Myohand reorients objects sampled from 100 geometric variants (25 variants for each geometry). The object pose and goal orientation are randomly initialized at the start of each episode. The object is randomly selected from the 100 geometry variants.

The reward (or negative cost) function used is defined as:

$$R := -\lambda_1 |\text{pos}_{\text{object}} - \text{pos}_{\text{goal}}|_2 + \lambda_2 \cos(\text{rot}_{\text{object}}, \text{rot}_{\text{goal}}) - \lambda_3 \cdot \mathbb{1}_{\text{dropped}}, \tag{13}$$

where $\lambda_1 = 1$, $\lambda_2 = 1$, and $\lambda_3 = 5$. The first term penalizes the Euclidean distance between the object and goal positions, the second term rewards rotational alignment via cosine similarity, and the final term imposes a

penalty if the object is dropped, indicated by the indicator function $\mathbb{1}_{\text{dropped}}$. To demonstrate the plug-and-play ability, we utilized the internal model pretrained in sequential setting of fingertip reach experiment.

We compare our method against three baselines: (1) Soft Actor-Critic (SAC) [32], (2) Planning with Deep Dynamics Models (PDDM) [29], which learns a single dynamics model for the entire manipulation system, and (3) PDDM+Multi-Step Loss (PDDM+MSL), an extension of PDDM incorporating a multi-step loss to improve long-term prediction accuracy. We also evaluate the end-to-end training version of our method Ours (end2end) by unfreezing the pretrained model.

2.3.3. Gesture generation

We evaluate MoDex's capability in gesture generation using AllegroHand, ShadowHand, and MyoHand, excluding Robotiq due to its unsuitability for gesture posing. To assess transferability, we directly employ the pretrained internal model from the quasi-static setting of the fingertip reach experiment. Additionally, to demonstrate few-shot learning, we utilize ChatGPT as the core model for generating cost functions, providing only two example cost functions in the system prompt. We provide the system prompt for Allegro as follows.

Prompt of gesture generation

You are acquired to generate code to control a robotic hand to make gestures, here are examples:

```
def gesture_thumbup_allegro(fingertip_position):
'''
num_envs (1) × finger_num × 3
indices: indexs: 0-thumb, 1-ring finger&little finger, 2-middle finger, 3-index finger
'''
# loss that forces the thumb to be straight
loss1 = -fingertip_position[:,0,1]
# loss that forces the index finger, middle finger and ring finger to curl towards palm
loss2 = fingertip_position[:,1:, 2].mean(-1)
# combine the losses for all fingers
loss = 1. * loss1 + 1. * loss2
return loss

def gesture_ok_allegro(fingertip_position):
'''
num_envs (1) × finger_num × 3
indices: 0-thumb, 1-ring finger&little finger, 2-middle finger, 3-index finger
'''
# loss that forces the tips of thumb and index finger to be close together
loss1 = torch.norm(fingertip_position[:,0] - fingertip_position[:,3], dim=-1)
# loss that forces the middle finger and ring finger to be straight
loss2 = -fingertip_position[:,1:3, 2].mean(dim=-1)
# combine the losses for all fingers
loss = 1. * loss1 + 1. * loss2
return loss
```

Refer to the examples and generate gestures according to the users' request.

2.4. *Real-world deployment*

We validate MoDex in real-world in-hand manipulation and gesture generation. For in-hand manipulation, we use the Xhand (five-fingered dexterous hand) to manipulate a cube, orange, and cylinder at 10 Hz. Object states (position and orientation) are tracked via AprilTags using an Intel RealSense D415, as shown in Figure 2. The hand state is represented by joint positions, with 20 minutes of random exploration collected for pretraining. Object positions are computed in the image coordinate frame and normalized to the range $[0,1]$. We constrain orientation tracking to the rotation around the tag's local z-axis. During evaluation, random z-axis rotations in $[0°, 90°]$ are used, with a success threshold of 3°. The episode ends if the object falls from the hand or the number of task steps exceeds 50. In the online adaptive learning stage, we maintain an on-policy to random exploration ratio of 3:1 to reduce overfitting.

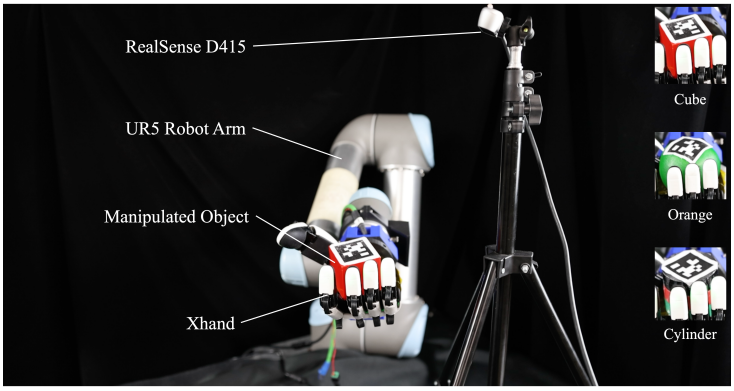
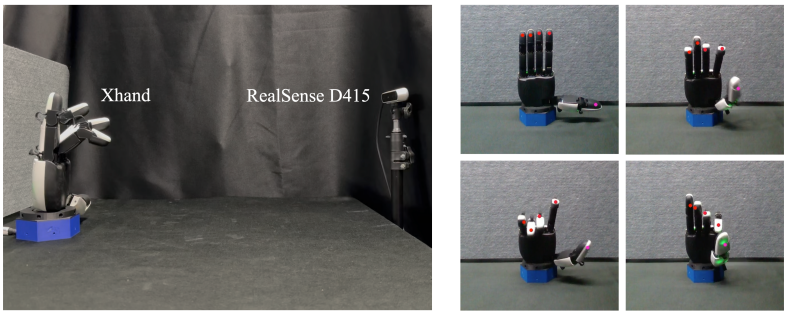


Figure 2. Experimental setup for real-world in-hand manipulation.

For gesture generation, we begin by recording video footage of the hand during exploration. We manually annotate three keypoints per finger across selected frames. These keypoints are then tracked throughout the video using CoTracker3 [33]. To obtain a robust estimate of each finger's position, we compute the average position of its tracked keypoints, weighting by their visibility confidence scores. The tracking results are shown in Figure 3b. We utilize the 2D positions in the image coordinate frame, as this information is sufficient to reliably infer diverse hand gestures given our setup. Notably, no camera-to-hand calibration is required for either in-hand manipulation or gesture generation, due to the design of our experimental setup.



(a) (b)

Figure 3. Experimental framework for gesture generation and fingertip tracking: **(a)** Setup for real-world gesture generation; **(b)** 2D fingertip positions tracked over time using CoTracker3 [33].

3. Results

3.1. Fingertip reach experiment

As shown in Table 2, MoDex consistently outperforms both model-free and model-based baselines across almost all robotic hands and settings. In the quasi-static setting, MoDex achieves the highest success rate on all hands, improving over the strongest baseline (FM+CEM) by an average of 3.5%. At the same time, it attains the lowest reach error, demonstrating superior precision in fingertip positioning. Importantly, MoDex requires substantially fewer planning samples, indicating strong planning efficiency and fast convergence.

Table 2. Evaluation on fingertip reach task. We exclude model-free baselines from this quasi-static setting due to their inability to succeed. Experimental results demonstrate that our method achieves the lowest fingertip reach error while requiring the fewest planning samples, highlighting both its accuracy and sample efficiency.

Methods	Robotiq			Allegro			Shadowhand			Myohand		
	S.R. ↑	R.E. ↓	P.S. ↓	S.R. ↑	R.E. ↓	P.S. ↓	S.R. ↑	R.E. ↓	P.S. ↓	S.R. ↑	R.E. ↓	P.S. ↓
Quasi-static Setting												
FM+RS	79	0.80	10 k	49	1.47	10 k	22	1.78	10 k	34	1.58	10 k
FM+BGD	82	0.76	38.4 k	66	1.33	38.4 k	45	1.50	38.4 k	66	1.13	38.4 k
FM+CEM	88	0.77	1.2 k	70	1.34	1.2 k	**55**	**1.45**	1.2 k	77	1.01	2.0 k
Ours	**91**	**0.72**	**0.8 k**	**72**	**1.32**	**1.0 k**	54	1.47	**1.0 k**	**87**	**0.87**	**2.0 k**
Sequential Setting												
SAC	89	0.74	–	50	1.52	–	38	1.65	–	62	1.52	–
SAR	8	2.03	–	0	2.44	–	2	2.38	–	0	2.54	–
FM+RS	84	0.76	50 k	53	1.50	50 k	45	1.60	50 k	7	2.50	50 k
FM+BGD	80	0.83	38.4 k	62	1.38	38.4 k	43	1.56	38.4 k	85	1.20	38.4 k
FM+CEM	90	0.74	2.0 k	86	1.19	2.0 k	50	1.52	2.4 k	77	1.25	2.4 k
Ours	**96**	**0.51**	**1.2 k**	**89**	**1.16**	**1.2 k**	**56**	**1.48**	**1.2 k**	**88**	**1.18**	**1.8 k**

In the more challenging sequential setting, MoDex further extends its advantage, achieving the highest average success rate of 82.3%, compared to 75.8% for FM+CEM and 59.8% for SAC. Notably, MoDex attains the lowest reach error across all hands (0.51–1.48 *vs.* 0.74–2.50 for baselines), showing enhanced temporal stability and dynamic control. Despite being model-based, MoDex also maintains a low computational burden ($\sim$1.2 k samples per step) compared to FM+RS and FM+BGD (50 k and 38.4 k, respectively).

Overall, MoDex achieves superior accuracy, robustness, and efficiency in both quasi-static and sequential reaching scenarios. Moreover, model-based methods (including MoDex) require only about 1% and 10% of the training data used by model-free methods in the quasi-static and sequential settings, respectively, underscoring their strong data efficiency. In contrast, the synergy-based SAR baseline performs poorly across all hands (S.R. $< 10\%$), suggesting that fixed low-dimensional coordination patterns are insufficient for precise multi-finger control in these low-redundancy tasks. Qualitative results are visualized in Figure 4.

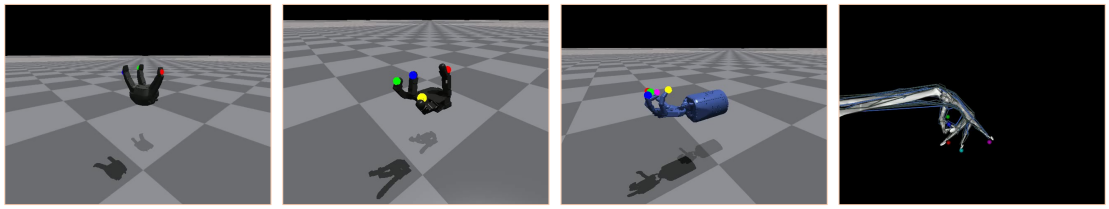


Figure 4. Visualization of fingertip reach: The dexterous hand is tasked with reaching randomly generated target points, where each finger is assigned a specific, color-coded goal.

Relation of action dimension and data volume. We investigate how the action dimensionality and the amount of training data influence the predictive accuracy of the forward dynamics model, using mean squared error (MSE) as the evaluation metric with a fixed 10:1 train/evaluation split. As illustrated in Figure 5a, the logarithm of prediction MSE exhibits an approximately linear increase with action dimension, indicating that higher-dimensional control spaces lead to a proportional degradation in prediction accuracy. Conversely, increasing the training data volume results in a linear decrease in log MSE, as shown in Figure 5b, reflecting improved generalization with more diverse samples. Notably, the near-perfect linear correlations ($r > 0.99$) across both axes imply that model performance follows a predictable scaling law: maintaining a constant prediction error requires the training data volume to grow exponentially with the action dimension. This finding highlights a fundamental data-efficiency limitation in high-dimensional control learning—underscoring the necessity of either data-efficient modeling strategies or dimensionality reduction techniques when scaling to dexterous, multi-DoF manipulators.

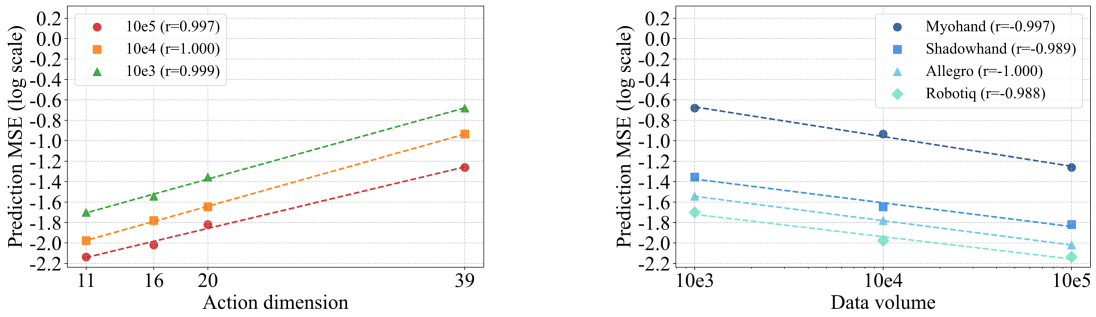


Figure 5. Scaling effects of action dimension and data volume: **(a)** The prediction MSE of forward model grows exponentially with action dimension; **(b)** The prediction MSE of forward model decreases linearly with data volume. Results show that to maintain the same prediction error, the data volume grows exponentially with action dimension.

Why MoDex outperforms other model-based methods? A key question is how bidirectional planning—the primary distinction between MoDex and other model-based optimization methods—contributes to its superior task success rate, particularly when compared to CEM-based variants. We conduct experiments on both quasi-static and sequential settings. As illustrated in Figure 6, FM+CEM achieves performance comparable to MoDex when granted sufficiently large planning budgets in terms of sample size and iteration count. However, under more constrained settings, MoDex exhibits a clear advantage, achieving lower cost values with significantly fewer optimization steps. This behavior indicates that the

inverse model in MoDex provides a strong prior or initialization for the subsequent forward-planning phase (Figure 1), effectively narrowing the search space and accelerating convergence. In contrast, FM+CEM and SIM+CEM rely solely on forward sampling, requiring substantially more samples to reach comparable performance. These results demonstrate that bidirectional planning synergistically combines the strengths of both inverse and forward dynamics reasoning, yielding higher efficiency and robustness, especially under limited computational resources.

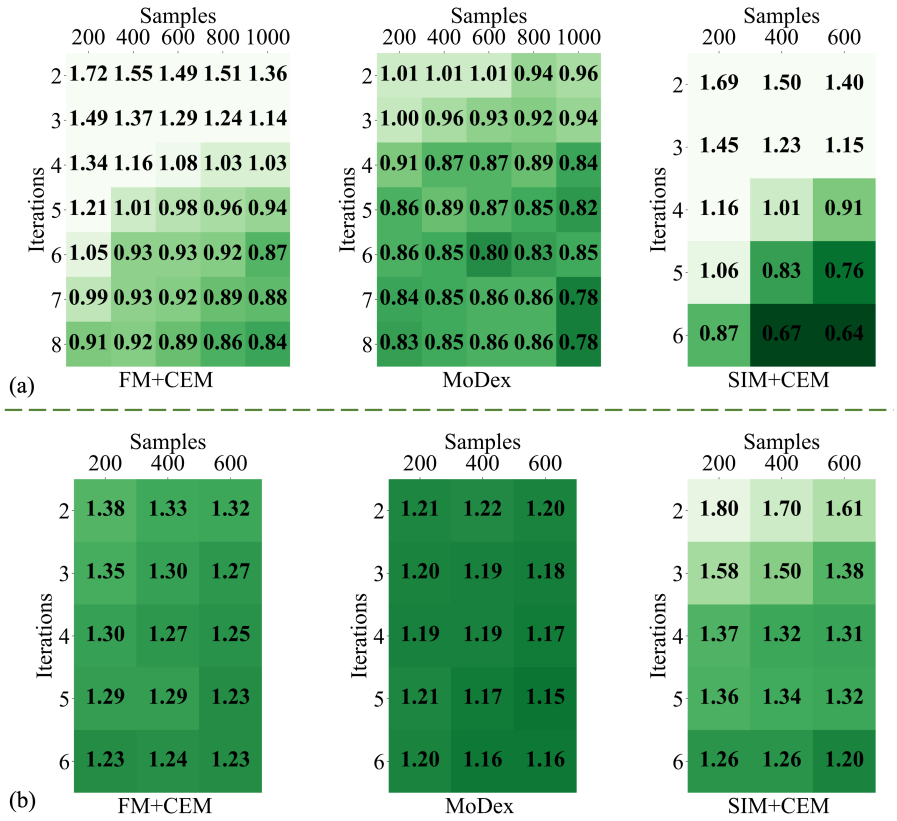


Figure 6. Ablation study on planning method. We compare FM+CEM, SIM+CEM and MoDex's planning iteration steps and samples per iteration in **(a)** quasi-static setting and **(b)** sequential setting. Results demonstrate the inverse process of MoDex can significantly reduce planning samples and iterations.

Adaptability to actuator changes: To evaluate the adaptability of the forward dynamics model to physical changes in the actuation system, we simulate two categories of hand-related perturbations: (i) actuator failures, modeled by setting the control signal of selected actuators to zero, and (ii) muscle fatigue, modeled by scaling actuator commands by a constant factor. For each condition, we collect 1000 samples for fine-tuning and 500 for evaluation. As shown in Figure 7, the neural network-based forward model rapidly recovers its prediction accuracy after the actuator change, indicating strong adaptability through limited online fine-tuning. The model achieves convergence to near-original MSE levels across all tested scenarios. However, the rate of adaptation varies: higher fatigue factors lead to longer convergence times, while the adaptation patterns for actuator failures differ depending on which actuator is affected, likely due to structural asymmetries and differing dynamic dependencies among actuators.

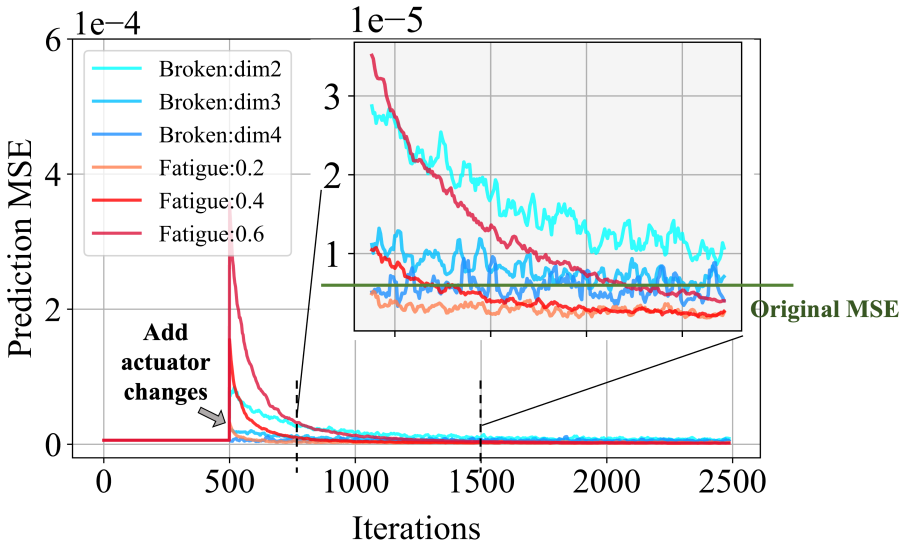


Figure 7. Adaptation of forward model to different actuator changes. Since MoDex is data-driven, it can adapt to actuator changes by learning from new data.

3.2. In-hand manipulation

As shown in Figure 8, our method demonstrates superior data efficiency and generalization across all three reorientation tasks. In the Reorient 8objects task (Figure 8a), MoDex rapidly improves with fewer than 10^4 datapoints, surpassing all baselines by a large margin. While SAC and PDDM exhibit slow and unstable learning, our factorized dynamics approach reaches a success rate of 0.4 with approximately one-tenth of the samples required by model-free reinforcement learning. The improvement highlights the effectiveness of modularized forward-inverse composition, enabling fast adaptation across diverse object geometries.

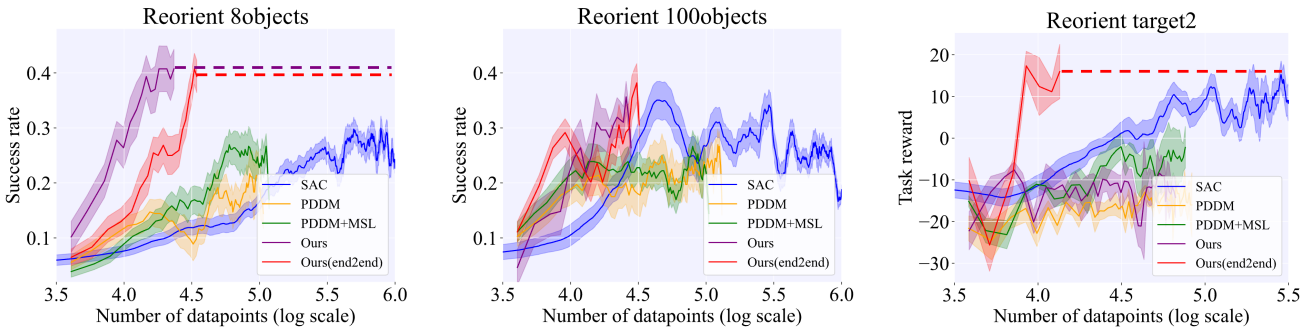


Figure 8. In-hand manipulation experiments. Quantitative evaluation are conducted on three in-hand manipulation tasks: **(a)** Reorient 8objects; **(b)** Reorient 100objects; **(c)** Reorient target2, showing that MoDex achieves the highest success rates with the fewest data.

For the more challenging Reorient 100objects scenario (Figure 8b), MoDex maintains high performance and generalization despite the increased object diversity. Compared to PDDM and PDDM+MSL, which both plateau at about 0.25 success rate, our approach continues to improve with additional data, reaching above 0.35 success rate. This demonstrates that the factorized dynamics representation scales favorably to

large and heterogeneous object sets, suggesting that the learned sub-dynamics can transfer effectively across shape variations.

In contrast, for Reorient target2 (Figure 8c), MoDex initially performs well but experiences performance degradation due to compounding model composition errors, as the task requires high temporal precision and consistent long-horizon prediction. However, when the pretrained internal model is unfrozen (Ours (end2end)), performance recovers and even surpasses all baselines, achieving stable convergence near the optimal reward level. This indicates that end-to-end fine-tuning can effectively mitigate accumulated modeling bias while preserving the data efficiency benefits of modular training.

MoDex demonstrates robust transferability and remarkable data efficiency across diverse in-hand manipulation scenarios. Compared to the model-free SAC baseline, it achieves comparable or superior performance with nearly one order of magnitude fewer samples. The modular training paradigm further enables reusability of pretrained internal models, allowing rapid adaptation to new manipulation tasks without retraining from scratch. This capability highlights the potential of factorized dynamics learning as a scalable and general framework for efficient robot skill acquisition.

3.3. Gesture generation study

As illustrated in Figure 9, MoDex successfully generates a wide range of expressive gestures—including Rock&Roll, Scissorhands, Finger Gun, and Calling—across three distinct hand platforms: AllegroHand, ShadowHand, and MyoHand. The system demonstrates strong transferability by leveraging a pretrained internal model originally trained for fingertip reach tasks in a quasi-static setting, without requiring retraining or hand-specific tuning. To enable few-shot generalization, we employ ChatGPT to synthesize cost functions from linguistic gesture descriptions, using only two example functions provided via in-context learning. The resulting gestures exhibit semantic alignment with the target descriptions, confirming the viability of using LLMs to formulate task objectives for motor control via cost function generation. We provide three generated cost function as below.

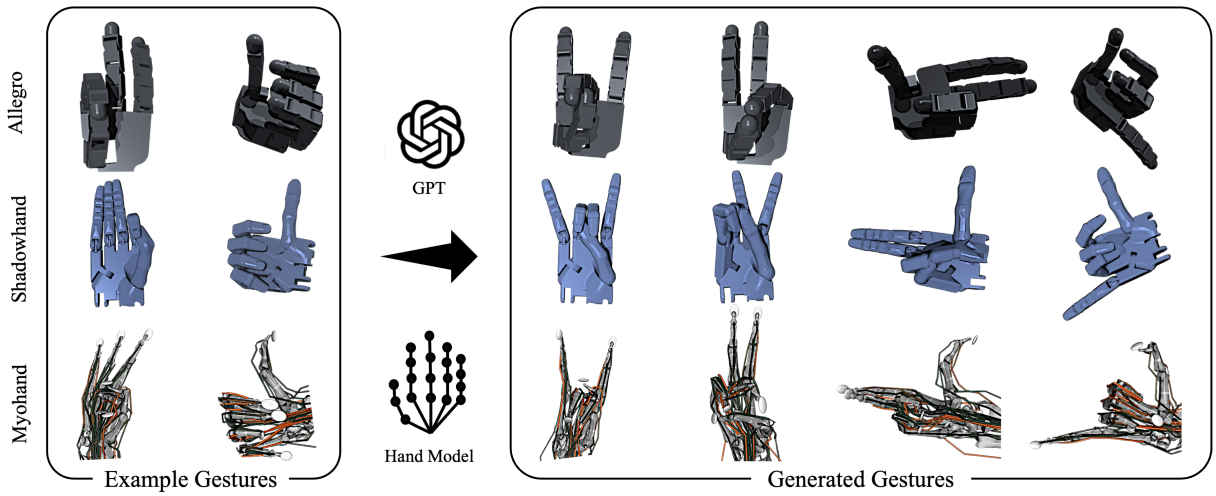


Figure 9. Gesture generation in simulation. We provide two gesture cost functions in prompt as in-context learning and generate four expressive gestures across three different dexterous hands. We exclude Robotiq since its structure is not appropriate for gesture generation.

Generated cost functions

```
def gesture_scissors_allegro(fingertip_position):
"""
num_envs (1) × finger_num × 3
for allegro
indices: 0-thumb, 1-ring finger&little finger, 2-middle finger, 3-index finger
"""
# loss that forces the tips of thumb, ring finger and little finger to be close together
loss1 = torch.norm(fingertip_position[:,0] - fingertip_position[:,1], dim=-1)
# loss that forces the index finger and middle finger to be straight
loss2 = -fingertip_position[:,2:4, 2].mean(dim=-1)
# combine the losses for all fingers
loss = 1. * loss1 + 1. * loss2
return loss

def gesture_rockandroll_allegro(fingertip_position):
"""
num_envs (1) × finger_num × 3
for allegro
indices: 0-thumb, 1-ring finger&little finger, 2-middle finger, 3-index finger
"""
# loss that forces the tips of thumb and middle finger to be close together
loss1 = torch.norm(fingertip_position[:,0] - fingertip_position[:,2], dim=-1)
# loss that forces the index finger, ring finger and little finger to be straight
loss2 = - (fingertip_position[:,1, 2] + fingertip_position[:,3,2])/2
# combine the losses for all fingers
loss = loss1 + 1. * loss2
return loss

def gesture_call_allegro(fingertip_position):
"""
num_envs (1) × finger_num × 3
for allegro
indices: 0-thumb, 1-ring finger&little finger, 2-middle finger, 3-index finger
"""
# loss that forces the tips of thumb to be straight
loss1 = -fingertip_position[:,0,1]
# loss that forces the little finger to be straight
loss2 = fingertip_position[:,1, 0]
# loss that forces the index finger and middle finger to curl towards palm
loss3 = fingertip_position[:,2:, 2].mean(-1)
# combine the losses for all fingers
loss = 1. * loss1 + 1. * loss2 + 1. * loss3
return loss
```

3.4. Real-world deployment

As shown in Figure 10 and Table 3, MoDex achieves successful object rotations within 30 minutes of real-world rollouts, with performance varying by object. The orange is difficult to rotate beyond 30° due to its size, while the cylinder rotates easily but is prone to slipping. The cube shows moderate difficulty, occasionally failing due to edge-related instabilities. MoDex also generates diverse hand gestures (Figure 11), with failures typically arising when gestures fall outside the exploration data distribution, highlighting the importance of coverage during data collection.

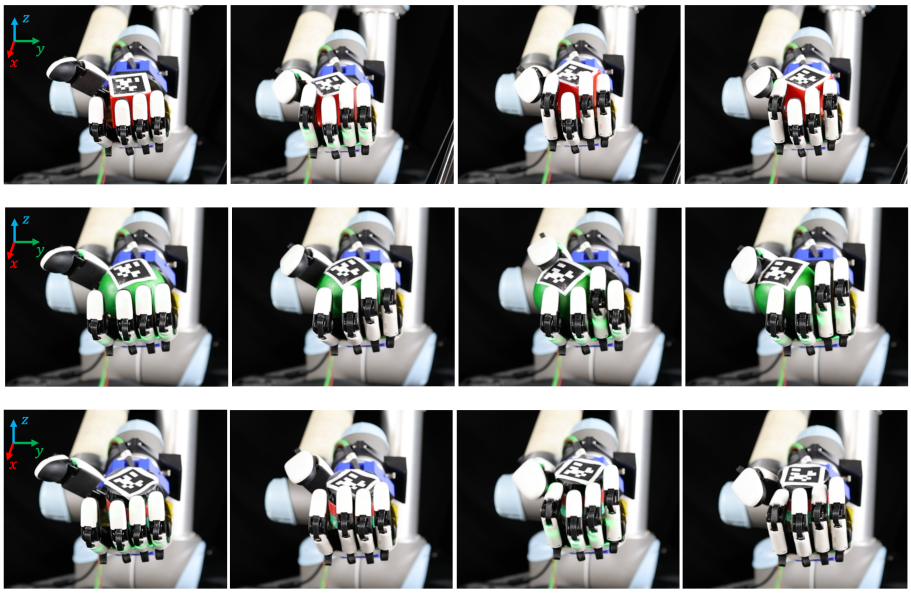


Figure 10. Qualitative evaluation on in-hand rotation in real world. We deploy MoDex to perform in-hand rotation of three objects (cube, orange, and cylinder) around the z-axis.

Table 3. Real-world evaluation on in-hand rotation.

	Rotate Cube	**Rotate Orange**	**Rotate Cylinder**
Success Rate	6/10	2/10	8/10

Figure 11. Qualitative evaluation on gesture generation in real world. MoDex generates diverse hand gestures in the real world. Green indicates successful executions, while red denotes failures. We observe that failures arise when the target gestures fall outside the distribution of the exploration data.

4. Conclusion

This work presents MoDex, a framework for high-dimensional dexterous control that leverages learned neural internal models. By combining a forward model for state prediction and an inverse model for action generation within a bidirectional planning strategy, MoDex achieves superior precision, data efficiency, and robustness in fingertip reaching tasks compared to model-free and model-based baselines. The pretrained internal model further demonstrates versatility by enabling efficient factorized dynamics learning for complex in-hand manipulation and few-shot gesture generation via LLM-synthesized cost functions. In fingertip reach experiment, it achieved an 82.3% success rate, outperforming model-based baselines in precision while using about 1 order of magnitude fewer planning samples. For in-hand manipulation, its factorized dynamics approach accelerated learning, matching model-free RL performance with 10 × less data and generalizing across 100 object variants. In gesture generation, it enabled few-shot gesture generation by translating LLM-generated cost functions into actions using a pretrained model. Finally, real-world experiments confirmed successful deployment for both dynamic cube reorientation and diverse gesture generation, validating the framework's practical efficacy in real scenarios.

Supplementary data

An experiment video is published together with this paper.

Data availability statement

No supplementary or additional data were generated in this study.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors used generative AI tools only to improve language and readability. The authors take full responsibility for the content of the manuscript.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2024YFB3816000), the Shenzhen Science and Technology Program (No. KJZD20240903100905008), and the Meituan Academy of Robotics Shenzhen.

Authors' contribution

Conceptualization: T.W. and S.L.; methodology: T.W. and S.L.; software: T.W.; visualization: T.W.; writing—original draft preparation: T.W.; investigation: T.W.; data curation: T.W.; supervision: W.D.; project administration: D.W.; writing—review and editing: S.L., C.L., K.S., W.C. and W.D. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Wang R, Zhang J, Chen J, Xu Y, Li P, *et al.* DexGraspNet: a large-scale robotic dexterous grasp dataset for general objects based on simulation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, May 29–June 2, 2023, pp. 11359–11366.

[2] Xu Y, Wan W, Zhang J, Liu H, Shan Z, *et al.* UniDexGrasp: universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, June 17–24, 2023, pp. 4737–4746.

[3] Mandikal P, Grauman K. Learning dexterous grasping with object-centric visual affordances. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, May 30–June 5, 2021, pp. 6169–6176.

[4] Handa A, Allshire A, Makoviychuk V, Petrenko A, Singh R, *et al.* DeXtreme: transfer of agile in-hand manipulation from simulation to reality. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, May 29–June 2, 2023, pp. 5977–5984.

[5] Chen T, Tippur M, Wu S, Kumar V, Adelson E, *et al.* Visual dexterity: in-hand reorientation of novel and complex object shapes. *Sci. Robot.* 2023, 8(84):eadc9244.

[6] Yin Z, Huang B, Qin Y, Chen Q, Wang X. Rotating without seeing: towards in-hand dexterity through touch. *arXiv* 2023, arXiv:2303.10880.

[7] Qi H, Yi B, Suresh S, Lambeta M, Ma Y, *et al.* General in-hand object rotation with vision and touch. In *Proceedings of Conference on Robot Learning (CoRL)*, Atlanta, USA, November 6–9, 2023, pp. 2549–2564.

[8] Wang H, Caggiano V, Durandau G, Sartori M, Kumar V. MyoSim: fast and physiologically realistic MuJoCo models for musculoskeletal and exoskeletal studies. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, May 23–27, 2022, pp. 8104–8111.

[9] Caggiano V, Wang H, Durandau G, Sartori M, Kumar V. MyoSuite: a contact-rich simulation suite for musculoskeletal motor control. In *Proceedings of Learning for Dynamics and Control Conference (L4DC)*, Stanford, USA, June 23–24, 2022, pp. 492–507.

[10] Caggiano V, Durandau G, Wang H, Chiappa A, Mathis A, *et al.* MyoChallenge 2022: learning contact-rich manipulation using a musculoskeletal hand. In *Proceedings of the NeurIPS 2022 Competitions Track*, New Orleans, USA, November 28–December 9, 2022, pp. 233–250.

[11] Caggiano V, Dasari S, Kumar V. MyoDex: a generalizable prior for dexterous manipulation. In *Proceedings of International Conference on Machine Learning (ICML)*, Honolulu, USA, July 23–29, 2023, pp. 3327–3346.

[12] Grillner S. Neurobiological bases of rhythmic motor acts in vertebrates. *Science* 1985, 228(4696):143–149.

[13] Zuo C, He K, Shao J, Sui Y. Self model for embodied intelligence: modeling full-body human musculoskeletal system and locomotion control with hierarchical low-dimensional representation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, May 13–17, 2024, pp. 13062–13069.

[14] Berg C, Caggiano V, Kumar V. SAR: generalization of physiological agility and dexterity via synergistic action representation. In *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 10–14, 2023.

[15] He K, Zuo C, Ma C, Sui Y. DynSyn: dynamical synergistic representation for efficient learning and control in overactuated embodied systems. *arXiv* 2024, arXiv:2407.11472.

[16] Kawato M. Internal models for motor control and trajectory planning. *Curr. Opin. Neurobiol.* 1999, 9(6):718–727.

[17] Tin C, Poon CS. Internal models in sensorimotor integration: perspectives from adaptive control theory. *J. Neural Eng.* 2005, 2(3):S147.

[18] Francis BA, Wonham WM. The internal model principle of control theory. *Automatica* 1976, 12(5):457–465.

[19] Egger SW, Remington ED, Chang CJ, Jazayeri M. Internal models of sensorimotor integration regulate cortical dynamics. *Nat. Neurosci.* 2019, 22(11):1871–1882.

[20] Angelaki DE, Shaikh AG, Green AM, Dickman JD. Neurons compute internal models of the physical laws of motion. *Nature* 2004, 430(6999):560–564.

[21] Wolpert DM, Ghahramani Z, Jordan MI. An internal model for sensorimotor integration. *Science* 1995, 269(5232):1880–1882.

[22] Shadmehr R, Mussa-Ivaldi FA. Adaptive representation of dynamics during learning of a motor task. *J. Neurosci.* 1994, 14(5):3208–3224.

[23] Nagabandi A, Kahn G, Fearing RS, Levine S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 21–25, 2018, pp. 7559–7566.

[24] Shi H, Xu H, Huang Z, Li Y, Wu J. RoboCraft: learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks. *Int. J. Robot. Res.* 2024, 43(4):533–549.

[25] Rubinstein RY, Kroese DP. *The Cross-Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation, And Machine Learning*, 1st ed. New York: Springer, 2004.

[26] Ito M. Control of mental activities by internal models in the cerebellum. *Nat. Rev. Neurosci.* 2008, 9(4):304–313.

[27] Woodworth RS. Accuracy of voluntary movement. *Psychol. Rev. Monogr. Suppl.* 1899, 3(3):i–114.

[28] Elliott D, Helsen WF, Chua R. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psychol. Bull.* 2001, 127(3):342.

[29] Nagabandi A, Konolige K, Levine S, Kumar V. Deep dynamics models for learning dexterous manipulation. In *Proceedings of Conference on Robot Learning (CoRL)*, Cambridge, USA, November 16–18, 2020, pp. 1101–1112.

[30] Huang W, Wang C, Zhang R, Li Y, Wu J, *et al.* VoxPoser: composable 3D value maps for robotic manipulation with language models. In *Proceedings of Conference on Robot Learning (CoRL)*, Atlanta, USA, November 6–9, 2023, pp. 540–562.

[31] Makoviychuk V, Wawrzyniak L, Guo Y, Lu M, Storey K, *et al.* Isaac gym: high performance gpu-based physics simulation for robot learning. *arXiv* 2021, arXiv:2108.10470.

[32] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of International Conference on Machine Learning (ICML)*, Stockholmsmässan, Sweden, July 10–15, 2018, pp. 1861–1870.

[33] Karaev N, Makarov I, Wang J, Neverova N, Vedaldi A, *et al.* CoTracker3: simpler and better point tracking by pseudo-labelling real videos. *arXiv* 2023, arXiv:2410.11831.