Robust Model-Based In-Hand Manipulation with Integrated Real-Time Motion-Contact Planning and Tracking

Yongpeng Jiang

***Abstract*— Robotic in-hand manipulation, involving fingers making and breaking contacts, advances toward human-like dexterity in real-world robotic interactions. While learning-based approaches have recently shown promising performance, they face bottlenecks due to high data requirements and lengthy training times. Although model-based methods have the potential to overcome these limitations, they struggle with efficient online planning and handling modeling errors, which limits their real-world applications. This paper proposes a novel approach for in-hand manipulation that addresses the limitations of both learning-based and model-based methods. The key feature of our approach is the integrated real-time motion-contact planning and tracking, achieved through a hierarchical structure. At the high level, finger motion and contact force references are jointly generated using contact-implicit model predictive control (CIMPC). At the low level, these combined references are tracked with tactile feedback. Extensive experiments demonstrate that our approach outperforms existing methods in terms of accuracy, robustness, and real-time performance. It successfully completes all 6 challenging tasks in real-world environments, even under significant external disturbances.**

I. Introduction

In-hand manipulation refers to changing the position of grasped objects using fingers, the palm, and external contacts [1], [2],. This capability is essential for enabling versatile and dexterous robotic interaction with the real world [3], [4]. In-hand manipulation can be categorized into in-grasp manipulation [5], where hand-object contacts are maintained, and manipulation involving finger making and breaking contacts (regrasping) [2], [6]. This paper focuses on the latter, which is challenging due to two key aspects. First, modeling errors are unavoidable due to the difficulty of accurately modeling the nonlinear, contact-rich dynamics [7]–[9], compounded by sensor noise and variability in object properties and hand structures. Robust planning and control, incorporating contact state monitoring with tactile feedback, as well as visual and proprioceptive signals, are essential. Second, real-time planning is challenging due to the high degrees of freedom in multi-fingered hands and the need to coordinate numerous contacts. External disturbances and stochastic contact dynamics [10] require fast online re-planning to update contact sequences and recover from perturbations, especially in long-horizon tasks with regrasping.

To address these issues, considerable works have been reported, divided into learning-based and model-based methods. Reinforcement learning (RL) achieves state-of-the-art performance through parallel simulation and domain randomization [11]–[14]. However, RL's generalization requires extensive data, posing significant challenges for further deployment. In contrast, model-based methods offer training-free generalization. These methods can be further categorized into contact-explicit and contact-implicit methods. Contact-explicit methods transform manipulation into a hybrid problem, solving discrete contact sequences and continuous control inputs [15]–[18]. However, to avoid the locality of solutions, problems are often solved considering the complete manipulation sequence, making online re-planning time-consuming and susceptible to perturbations. Contact-implicit methods, on the other hand, plan directly through contacts without explicitly considering contact sequences, using complementary constraints that are difficult to solve [19]–[22]. To ensure real-time performance, they typically use simplified models with poorer physical fidelity. Hence, combining the adaptability of contact-implicit methods with the robustness of contact-explicit methods is a promising direction.

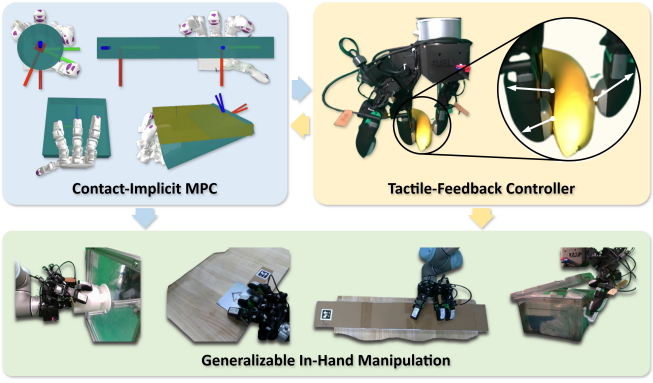

Fig. 1. Overview of the proposed framework for generalizable in-hand manipulation. The framework is model-based and organized as a hierarchical structure. In the high level, a contact-implicit MPC generates real-time motion-contact plans where the fingers make and break contacts. In the low level, a tactile-feedback controller tracks the high-level plans while compensating for the modeling errors by exerting desired contact force.

This paper addresses the in-hand manipulation problem with regrasping, emphasizing robust, long-horizon manipulation in the presence of external disturbances and significant object pose changes. It proposes a hierarchical framework combining real-time integrated motion-contact planning and tactile-feedback tracking control. At the high level, a contact-implicit model predictive control (CIMPC) scheme computes reference finger motions and contact forces using a differential dynamic programming (DDP) algorithm and an implicit contact model. At the low level, these references

Y. Jiang is with the Department of Automation, Tsinghua University, China.

are tracked with MPC-based hybrid force-motion control (HFMC) incorporating tactile feedback. The high-level module enables real-time planning, while the low-level module ensures robust execution and addresses modeling errors like the force-at-a-distance effect[1] caused by modeling errors. We conduct extensive simulations and real-world experiments to validate the accuracy, robustness, and real-time performance of the proposed framework, with the first two metrics outperforming existing methods. Videos and codes are available on the project website[2].

II. Method

This section outlines our method for synergistically planning and tracking finger movements and contact forces in dexterous in-hand manipulation. For a comprehensive overview, see Appendix A.

A. Assumptions and Notation

This paper considers the task of quasi-dynamic in-hand manipulation through rigid frictional contacts. We make the following assumptions: 1) The in-hand manipulation is performed under *quasi-dynamic* conditions [23], which means negligible inertia effects. 2) The object is modeled as a single rigid body. The geometry of the object and fingers is known, while the inertia and surface parameters are approximately estimated. In this article, we use superscripts a, u for the robot (actuated) and the object (unactuated). And $\|\cdot\|_{\boldsymbol{W}}$ is the weighted quadratic norm. Besides, $[\boldsymbol{a};\boldsymbol{b}]$ is the vertical concatenation of vectors $\boldsymbol{a},\boldsymbol{b}$, $[\boldsymbol{A};\boldsymbol{B}]$ is the vertical stacking of matrices $\boldsymbol{A},\boldsymbol{B}$, and blkdiag$(\boldsymbol{A}_1,\cdots,\boldsymbol{A}_m)$ is the block diagonal matrix composed of $\boldsymbol{A}_1,\cdots,\boldsymbol{A}_m$. In addition, $\oplus,\ominus$ are the generalized addition and subtraction involving operations on the SE(3) group. Notations used in this paper are summarized in Tab. I.

TABLE I
Notations Used in this Article

High-Level Module	
$\boldsymbol{x}$	state of the full hand-object system $[\boldsymbol{x}^u;\boldsymbol{x}^a]$
$\boldsymbol{x}^u$	object pose composed of position and quaternion
$\boldsymbol{x}^a$	hand joints
$\boldsymbol{u}$	control input (i.e., delta desired joints)
Low-Level Module	
$\boldsymbol{s}$	low-level state $[\boldsymbol{q};\boldsymbol{q}_d;\boldsymbol{\Lambda}_{\text{ext}}]$
$\boldsymbol{q},\boldsymbol{q}_d$	actual and desired joints
$\boldsymbol{\lambda}_{\text{ext}},\boldsymbol{\Lambda}_{\text{ext}}$	the contact force of a single contact and all contacts
$\boldsymbol{u}$	control input (i.e., the change rate of desired joints $\dot{\boldsymbol{q}}_d$)
$\boldsymbol{J}$	stacked hand kinematic Jacobian of all contacts
$\boldsymbol{G_o}$	the grasp matrix
$\boldsymbol{K}_P,\boldsymbol{K}_D$	stiffness and damping of the hand's joint PD model
Others	
n_c	number of active contacts
n_q	number of joints
N	prediction horizon for the high-level discrete system

[1]Also refered to as the "boundary layer" effect in [19]

[2]https://director-of-g.github.io/in_hand_manipulation/. This manuscript is an abstract version of a journal paper that will soon be submitted. The website, video, and code are coming soon and are expected to be available before the workshop.

B. Real-Time Motion-Contact Planning

The high level module aims at generating adaptive finger motion and contact force references based on desired object motion. This process can be realized by solving the following OCP in a receding horizon fashion:

$$\begin{aligned}\min_{\boldsymbol{X},\boldsymbol{U}} \quad & \sum_{i=0}^{N-1} l(\boldsymbol{x}_i,\boldsymbol{u}_i)+l_f(\boldsymbol{x}_N)\\ \text{s.t.} \quad & \boldsymbol{x}_0=\boldsymbol{x}_{\text{init}}\\ & \boldsymbol{x}_{i+1}=\boldsymbol{f}(\boldsymbol{x}_i,\boldsymbol{u}_i)\\ & \boldsymbol{x}_i\in\mathcal{X}_{\text{jnt}}\cap\mathcal{X}_{\text{sc}}\\ & \boldsymbol{u}_i\in\mathcal{U}\end{aligned} \tag{1}$$

where $\boldsymbol{X}=\{\boldsymbol{x}_0,\cdots,\boldsymbol{x}_N\},\boldsymbol{U}=\{\boldsymbol{u}_0,\cdots,\boldsymbol{u}_{N-1}\}$ are the state and control variables, $\boldsymbol{x}_{\text{init}}$ is the known initial state. The feasible sets $\mathcal{X}_{\text{jnt}}$, $\mathcal{X}_{\text{sc}}$ represents the joint limit and self-collision constraint, respectively, and $\mathcal{U}$ is the input bound. We adopt the CQDC model (Appendix B) as the dynamics $\boldsymbol{f}$. In addition, $\mathcal{X}_{\text{jnt}},\mathcal{X}_{\text{sc}},\mathcal{U}$ are converted to soft constraints:

$$\begin{aligned}l(\boldsymbol{x},\boldsymbol{u})&=l_{\text{reg}}(\boldsymbol{x})+l_{\text{reg}}(\boldsymbol{u})+l_{\text{jnt}}(\boldsymbol{x})+l_{\text{sc}}(\boldsymbol{x})\\ l_f(\boldsymbol{x}_N)&=l_{\text{reg}}(\boldsymbol{x}_N)\end{aligned} \tag{2}$$

The running cost l is composed of the regulation cost l_{reg}, the joint limits cost l_{jnt} and the self-collision cost l_{sc}; While the final cost l_f consists of only the regulation cost of states. Please refer to Appendix C for the specific form of each cost. More details of the high-level MPC formulation can be found in Appendix D.

C. Tactile-Feedback Motion-Contact Planning

This section introduces a tactile-feedback controller for synergistically tracking finger movements and contact forces.

1) Contact Force-Motion Model with Coupling Effect: During in-hand manipulation, suppose that the object is not fully constrained, the motion of each finger also affects the contact forces of the other fingers through the object's movement (i.e., the coupling effect), we introduce the grasp matrix $\boldsymbol{G}_o\in\mathbb{R}^{6\times 3n_c}$, and the following mappings hold:

$$\boldsymbol{w}_{\text{ext}}=\boldsymbol{G}_o\boldsymbol{\Lambda}_{\text{ext}},\quad \delta\boldsymbol{P}_c=\boldsymbol{G}_o^\top\delta\boldsymbol{x}^u \tag{3}$$

where $\boldsymbol{w}_{\text{ext}}$ is the resultant wrench applied on the object, $\boldsymbol{\Lambda}_{\text{ext}},\boldsymbol{P}_c$ are stacked contact forces and locations of all n_c contacts. The coupled stiffness is derived as:

$$\begin{aligned}\boldsymbol{K}_{\text{coup}}=\frac{\partial\boldsymbol{\Lambda}_{\text{ext}}}{\partial\boldsymbol{P}_d}&=\bar{\boldsymbol{K}}\left(\boldsymbol{I}-\frac{\partial\boldsymbol{P}_c}{\partial\boldsymbol{x}^u}\frac{\partial\boldsymbol{x}^u}{\partial\boldsymbol{P}_d}\right)\\ &=\bar{\boldsymbol{K}}+\bar{\boldsymbol{K}}\boldsymbol{G}_o^\top\left(\boldsymbol{G}_o\bar{\boldsymbol{K}}\boldsymbol{G}_o^\top\right)^{-1}\boldsymbol{G}_o\bar{\boldsymbol{K}}\end{aligned} \tag{4}$$

where $\bar{\boldsymbol{K}}$ is the decoupled stiffness described in Appendix E.1 and E.2. Note that the second term is the correction term due to the coupling effect. And the force-motion model is formulated as

$$\begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\boldsymbol{q}}_d\\ \dot{\boldsymbol{\Lambda}}_{\text{ext}}\end{bmatrix}=\begin{bmatrix}\boldsymbol{u}+\boldsymbol{K}_D^{-1}\left(\boldsymbol{K}_P(\boldsymbol{q}_d-\boldsymbol{q})-\boldsymbol{J}(\boldsymbol{q})^\top\boldsymbol{\Lambda}_{\text{ext}}\right)\\ \boldsymbol{u}\\ \boldsymbol{K}_{\text{coup}}\boldsymbol{J}(\boldsymbol{q}_d)\boldsymbol{u}\end{bmatrix} \tag{5}$$

TABLE II

COMPARISON BETWEEN DIFFERENT METHODS IN THE SPHERE ROTATION TASK IN MUJOCO SIMULATION.

Methods	Success rate ↑	Average task error (rad) ↓	Average task error of successful cases (rad) ↓	Average task error S.D. of successful cases ($\times 10^{-1}$rad) ↓
ours (planning)[a]	100 / 100	4.954×10^{-5}	4.954×10^{-5}	0.003
ours	**100 / 100**	**0.024**	**0.024**	0.034
openloop	14 / 100	0.644	0.069	**0.004**
MJPC (CEM)	83 / 100	0.099	0.050	4.505
MJPC (iLQG)	14 / 100	0.464	0.084	5.601

Methods	Average task time of successful cases (s) ↓	Average joint acceleration (rad/s^2) ↓	Average control frequency high-level/low-level (Hz)
ours (planning)[a]	28.824	0.247	10.007 / N/A
ours	35.550	**0.618**	9.884 / 30.006
openloop	47.801	0.651	9.456 / N/A
MJPC (CEM)	**21.926**	174.578	99.996 / N/A
MJPC (iLQG)	24.999	198.478	99.998 / N/A

[a] ours (planning) refers to testing the high-level planning module without MuJoCo simulation.

Define $\boldsymbol{s} \triangleq [\boldsymbol{q}; \boldsymbol{q}_d; \boldsymbol{\Lambda}_{\text{ext}}] \in \mathbb{R}^{2n_q+3n_c}$, and the above equation can be represented as the continuous time dynamics $\dot{\boldsymbol{s}} = \boldsymbol{g}(\boldsymbol{s}, \boldsymbol{u})$. Details of the force-motion model can be found in Appendix E.

2) Model Predictive Control: With the proposed force-motion model (5), we can now derive the low-level controller with MPC.

$$\begin{aligned}\min_{\boldsymbol{u}} \int_{t_0}^{t_0+T} \Big(&\|\boldsymbol{q} - \boldsymbol{q}_{\text{ref}}\|^2_{\boldsymbol{W_q}} + \sum_{i=1}^{n_c} \|\text{FK}_i(\boldsymbol{q}) \ominus \text{FK}_i(\boldsymbol{q}_{\text{ref}})\|^2_{\boldsymbol{W}_{\boldsymbol{p}_i}} \\ &+ \|\boldsymbol{\Lambda}_{\text{ext}} - \boldsymbol{\Lambda}_{\text{ext, ref}}\|^2_{\boldsymbol{W}_{\boldsymbol{\Lambda}}} + \|\boldsymbol{u}\|^2_{\boldsymbol{W}_{\boldsymbol{u}}} d\tau \Big) \\ \text{s.t.} \quad &\dot{\boldsymbol{s}} = \boldsymbol{g}(\boldsymbol{s}, \boldsymbol{u})\end{aligned} \tag{6}$$

where $\boldsymbol{q}_{\text{ref}}$ and $\boldsymbol{\Lambda}_{\text{ext, ref}}$ are the reference finger motions and contact forces, respectively. We follow the treatment in [24] to directly solve the problem with efficient third-party solvers. We find this choice avoids discretization errors while maintaining the control frequency at real-time rates. Details of the low-level MPC formulation can be found in Appendix. F.

III. SIMULATIONS

We choose the **Rotate Sphere** task to compare different methods. Two representative approaches from existing work were chosen: 1) executing generated finger motions in an open-loop manner (**openloop**); 2) predictive sampling (PS) using Cross Entropy Methods (**CEM**) or gradient-based methods (**iLQG**). Detailed implementation of the baselines can be found in Appendix G.6. We generate 100 random target orientations, of which the rotation from the initial orientation is no more than 90 degrees. For each target orientation, we record the sphere orientation and joint positions within the first 60s. The evaluation metrics are discussed in Appendix G.7.

1) Results and Discussions: As shown in Tab. II, the proposed method achieves the highest precision and success rate with the smoothest finger motions. Compared with the **openloop** baseline, our method has a lower task error since the tactile-feedback controller tracks desired contact forces and avoids missing contacts. The **openloop** baseline demonstrates poorer performance compared to [19]. This is attributed to the omission of the additional trajectory optimization process, ensuring a fair comparison as the experiments are conducted in real-time. Furthermore, we employ the more accurate MuJoCo simulator instead of a quasi-static simulator[3]. For the same reason, the **openloop** baseline has the lowest task error S.D. Compared with MJPC (CEM), our method has a lower task error, since sampling-based methods have poorer performance especially near convergence. However, MJPC quickly reduces task error, resulting in the shortest task time. This is attributed to the precise dynamics model utilized by MJPC. Besides, our method has much lower task error S.D. and joint acceleration, which show a potential advantage for hardware deployment. MJPC (iLQG) has the worst performance, since the gradients computed through finite difference often vanish for contact-rich manipulation, which indicates the importance of smoothing.

Remark. *Compared to manipulation tasks, MJPC (iLQG) performs relatively better on locomotion tasks [25], [26], as foot contacts are easily established due to gravity and the gradients are typically non-zero.*

Additional simulation results can be found in Appendix G.

IV. REAL-WORLD EXPERIMENTS

A. Experiment Setup

We attach the LEAP Hand to the flange of a UR5 arm, which serves as a movable base, and the wrist movements are not utilized during in-hand manipulation. The original fingertips of LEAP Hand are replaced with four vision-based tactile sensors Tac3D [27], which estimate the contact normals and forces at 30Hz. The objects are tracked with AprilTags [28] and one RealSense D405 camera at 30Hz. We only track relative movements with no need for calibration. The parallelization of different modules and the hardware access are implemented with ROS2 [29]. The high-level motion generation runs at 10Hz for all tasks, and the low-level controller computes and sends joint commands at 30Hz.

[3]In [19], an additional trajectory optimization with a smaller time step refines the planned trajectory to mitigate the "boundary layer" effect. Moreover, [19] employs a custom quasi-static simulator, which is less accurate than alternatives.

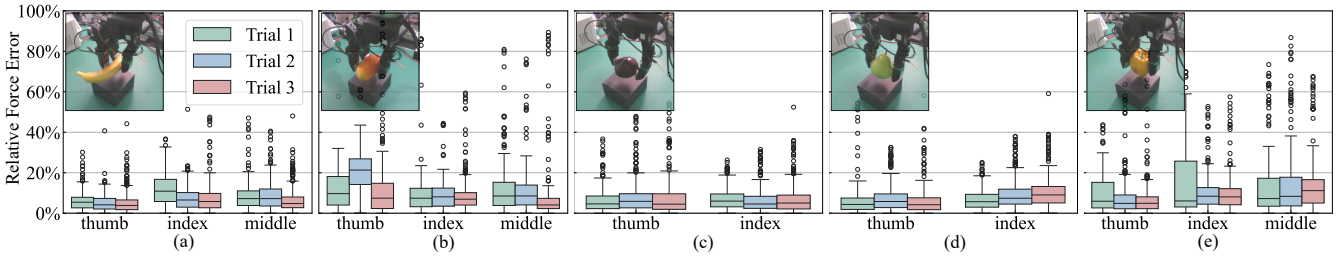

Fig. 2. Relative force tracking error and snapshots of the grasping experiment. The objects are: (a) banana, (b) mango, (c) onion, (d) pear, (e) pepper.

Besides, the joint states of LEAP Hand are queried at no more than 60Hz. Further real-world experimental results and additional details are available in Appendix H.

B. Real-World Grasping Experiment

We test the low-level controller in a grasping experiment similar to Appendix G.10, and the results are shown in Fig. 2. We execute grasping 3 times for each of the 5 objects. The relative force tracking error is small and remains consistent between multiple trials, which accords with the simulation. The results demonstrate that the proposed tactile-feedback controller is reliable to track desired contact forces. Extended experiments of this task can be found in Appendix H.2.

C. Experiments of the Open Door Task

We customize a door model with cylindrical handle and a hinge joint to accommodate of the LEAP Hand, as shown in Fig. 3 (a)(b). The task first requires in-hand manipulation to turn the door handle 180 degrees so that the notch is aligned with the door latch, which requires high precision. The hand then pulls open the door as shown in the last column of the snapshots. The rotation of the door handle during four trials is shown in Fig. 3 (c). We exert human interference in Trial 1 and 2, as shown in the peaks. The door handle is turned to the target orientation even under disturbances. The planned and real contact forces during Trial 4 are visualized in Fig. 3 (d)(e). The boxes with solid borders and the shadows filled in corresponds to time intervals where the planned and real contact forces exceed the given threshold $0.1\,\mathrm{N}$ (time intervals that last less than $1\,\mathrm{s}$ are regarded as noises and are neglected). If the planned forces are ideally tracked, the shadows will fill up 100% of the boxes. However, there is an obvious tracking delay especially when the planned forces do not maintain for sometime (i.e., high-frequency oscillation). The reasons for the imperfection include the low sampling frequency and the servo characteristics which are not accurately modeled.

V. Conclusions

This paper introduces a hierarchical model-based approach for generalizable in-hand manipulation. The high-level module generates real-time motion-contact plans, while the low-level module uses tactile feedback to correct modeling errors and track these plans. Both modules run in parallel, enabling robust and precise long-horizon manipulation. At the high level, we solve a contact-implicit MPC problem using the CQDC model and DDP solver, with strategies for real-time performance. At the low level, we use an MPC-based HFMC scheme to balance tracking desired motion and contact forces based on tactile feedback. Experiments show that our method outperforms existing approaches in accuracy, robustness, and real-time performance, generalizing to various tasks without pre-training. Future work will involve replacing the CQDC model with explicit contact-based dynamics and learned models to enhance control frequency and enable more dynamic motions.

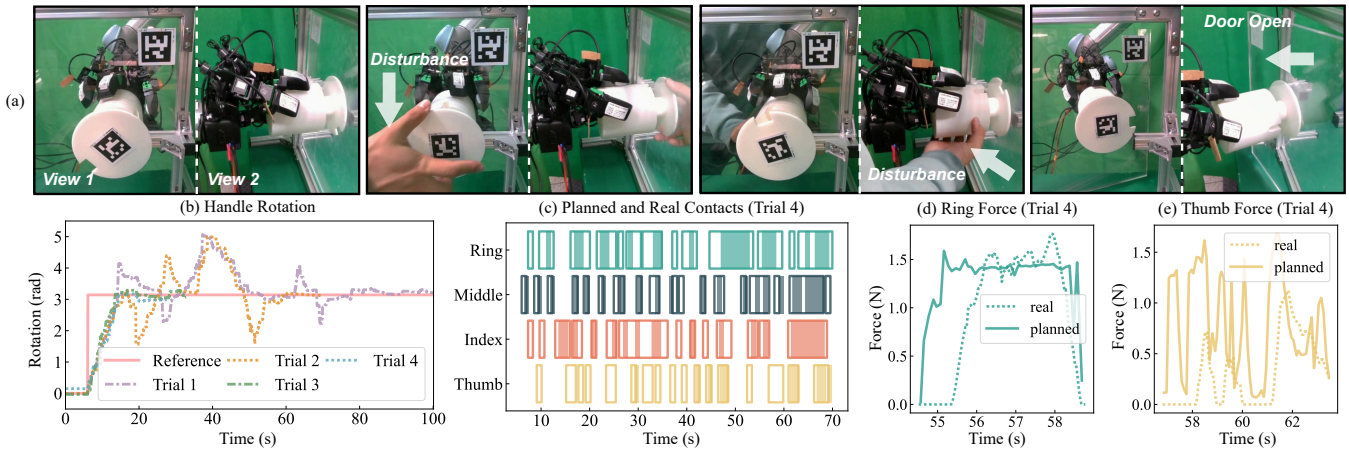

Fig. 3. Experiment results of the open door task with disturbance. (a) Snapshots from two different views. (b) Door handle rotation of different trials. (c) Planned and real contacts (Trial 4) represented as time intervals where the contact forces exceed a given threshold. (d)(e) The planned and real contact forces.

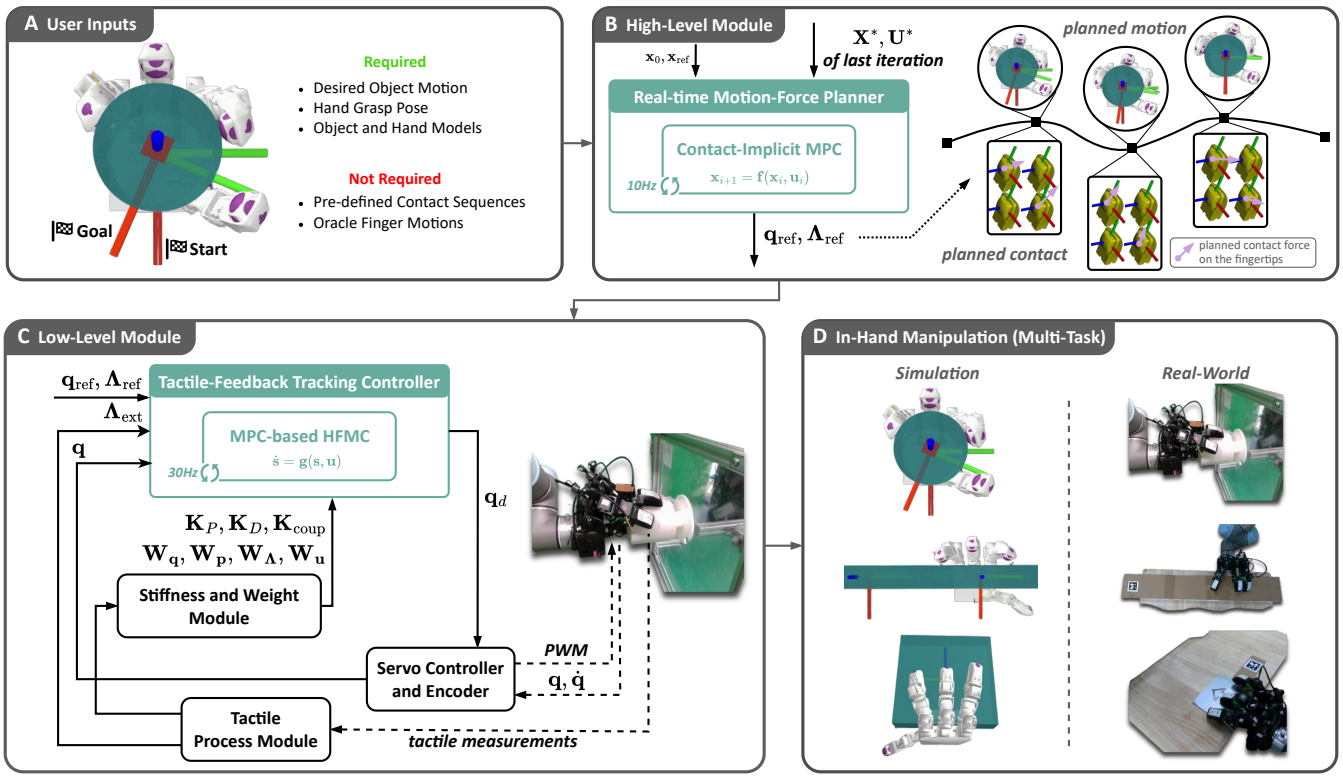

Fig. 4. The proposed integrated motion-contact planning and tracking framework. (A) The user inputs the desired object motion, hand grasp pose, and models for both. (B) The high-level real-time motion-contact planner employs contact-implicit MPC to generate motion-contact references from the initial state $\boldsymbol{x}_0$, state reference $\boldsymbol{x}_{\text{ref}}$, and the previous iteration's solution $\boldsymbol{X}^*, \boldsymbol{U}^*$. (C) The low-level tactile-feedback tracking controller uses tactile feedback to track these references jointly. The core algorithm is an MPC-based HFMC. (D) Together, these modules ensure robust and precise in-hand manipulation across multiple tasks.

APPENDIX

A. Overview of the Proposed Framework

This section provides an overview of the proposed framework for dexterous in-hand manipulation, as illustrated in Fig. 4. The framework takes as input the desired object motion, the hand grasp pose, and the models of both the object and hand. It eliminates the need for predefined contact sequences or predetermined finger motions, thanks to real-time planning with implicit contact models. The framework has a hierarchical structure, including the high-level real-time integrated motion-contact planner and the low-level tactile-feedback tracking controller. The two modules run in parallel, and are related through the integrated motion-contact references. Note that different dynamics models are used at different levels. At the high level, we care about the full system dynamics with joint motion-contact planning ability. Thus the smoothed CQDC model (8) $\boldsymbol{f}$ is used; At the low level, we care about the local force-motion relationship and the computation speed. Since the high-level model (8) is optimization-based, we use a simpler compliant contact model $\boldsymbol{g}$. The dynamics models will be discussed in the later sections. In addition, the models are updated with proprioception and object perception.

We then discuss the coordination between the motion-contact planner and controller. The high-level planner generates coarse finger motions to establish specific contacts and

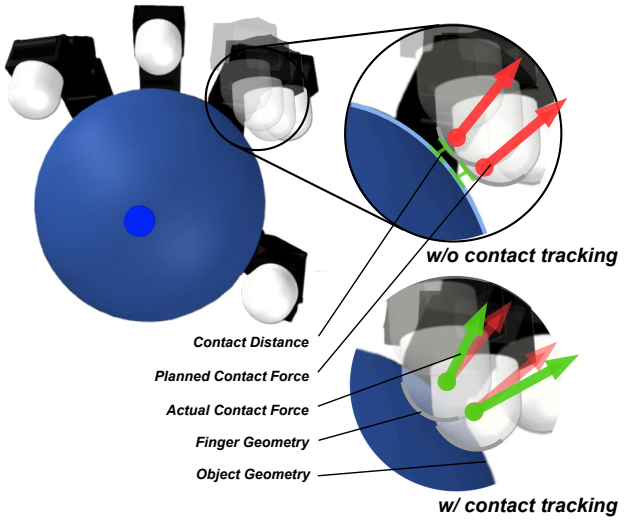

Fig. 5. This detailed view of the proposed framework. In the top figures, the high-level integrated motion-contact planning module generates real-time finger motions (with only the index finger visualized) and contact information, such as locations and forces (with only force visualized). The right figures illustrate how smoothed contact dynamics lead to the force-at-a-distance effect, where non-zero planned forces appear even when contact is inactive. Modeling errors, including this effect, can be mitigated through low-level joint motion-contact tracking.

drive the object to follow the desired motion. Finger motions, contact locations, and forces are jointly planned. However, due to smoothed contact dynamics, the force-at-a-distance effect can occur, leading to insufficient contact force and sliding during pure motion tracking. The low-level module addresses modeling errors, including the force-at-a-distance effect, by jointly tracking planned motions and contact forces using tactile feedback. Consequently, the desired motion deviates from the high-level plan, ensuring the actual contact forces closely match the planned forces. The output of the low-level module is then converted into position commands for the hardware.

B. CQDC Model

1) CQDC Model Formulation: Pang et al. [19] proposed an implicit time stepping for general multi-contact systems. Under quasi-dynamic assumptions, the system dynamics is constructed as the KKT conditions of the following Second Order Cone Program (SOCP):

$$\begin{aligned}\delta\boldsymbol{x}^* = \min_{\delta\boldsymbol{x}} \quad & \frac{1}{2}\delta\boldsymbol{x}^\top\boldsymbol{Q}\delta\boldsymbol{x} + h\cdot\boldsymbol{b}^\top\delta\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{J}_i\delta\boldsymbol{x} + \begin{bmatrix}\phi_i \\ \mathbf{0}_2\end{bmatrix} \in \mathbf{FC}^*(\mu_i), \forall i \in \{1,\dots,n_c\}\end{aligned} \tag{7}$$

where h is the discrete time step, n_c is the number of active contacts, and $\mathbf{FC}^*(\mu) = \left\{(\alpha,\boldsymbol{\beta}) \in \mathbb{R}\times\mathbb{R}^2 | \mu\|\boldsymbol{\beta}\|_2 \le \alpha\right\}$ is the dual friction cone with friction coefficient μ. The state-dependent signed distance $\phi(\boldsymbol{x})$ and contact Jacobian $\boldsymbol{J}(\boldsymbol{x})$ are computed with collision detection. Besides, $\boldsymbol{Q}$ and $\boldsymbol{b}(\boldsymbol{x},\boldsymbol{u})$ are computed from the pre-set robot stiffness and object inertia (i.e., model parameters). In this paper, only contacts with $\phi \le 0.1$ m (i.e., active contacts) are considered, following the approach in [19]. The implicit time stepping is derived as $\boldsymbol{x}^+ = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{x} \oplus \delta\boldsymbol{x}(\boldsymbol{x},\boldsymbol{u})$. In this article, we refer to $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$ as the CQDC model.

2) Smoothing of the Contact-Based Dynamics Model: Due to the rigid contact modeling, directly solving (7) results in shortsighted outcomes that inadequately explore possible contact modes. In [19], the original dynamics (7) is smoothed with a barrier function, resulting in the following problem:

$$\begin{aligned}\delta\boldsymbol{x}^* = \min_{\delta\boldsymbol{x}} \quad & \frac{1}{2}\delta\boldsymbol{x}^\top\boldsymbol{Q}\delta\boldsymbol{x} + h\cdot\boldsymbol{b}^\top\delta\boldsymbol{x} \\ & -\frac{1}{\kappa}\sum_{i=1}^{n_c}\log\left(\left\|\boldsymbol{J}_i\delta\boldsymbol{x} + \begin{bmatrix}\phi_i \\ \mathbf{0}_2\end{bmatrix}\right\|^2_{\boldsymbol{W}_{\mathrm{FC}}}\right)\end{aligned} \tag{8}$$

where $\boldsymbol{W}_{\mathrm{FC}} = \mathrm{blkdiag}(1, -\mu_i^2\boldsymbol{I}_2)$, and κ controls the degree of smoothing. Due to this smoothing, a small κ (e.g., on the order of hundreds) can generate non-negligible forces even when the constraint remains inactive. This phenomenon is known as the force-at-a-distance effect. While beneficial for planning, it can be detrimental to control by causing sliding and missed contacts, which the proposed framework addresses.

3) The Gradient Computation: An important property of the CQDC model is the differentiability, which is required by the backward pass of DDP. According to [19], the partial derivatives over state and control are

$$\begin{aligned}\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}} &= \boldsymbol{I} + \frac{\partial\delta\boldsymbol{x}}{\partial\boldsymbol{x}} \\ &= \boldsymbol{I} + \frac{\partial\delta\boldsymbol{x}}{\partial\boldsymbol{b}}\frac{\partial\boldsymbol{b}}{\partial\boldsymbol{x}} + \sum_{i=1}^{n_c}\left(\frac{\partial\delta\boldsymbol{x}}{\partial\boldsymbol{J}_i}\frac{\partial\boldsymbol{J}_i}{\partial\boldsymbol{x}} + \frac{\partial\delta\boldsymbol{x}}{\partial\phi_i}\frac{\partial\phi_i}{\partial\boldsymbol{x}}\right) \\ \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{u}} &= \frac{\partial\delta\boldsymbol{x}}{\partial\boldsymbol{u}} = \frac{\partial\delta\boldsymbol{x}}{\partial\boldsymbol{b}}\frac{\partial\boldsymbol{b}}{\partial\boldsymbol{u}}\end{aligned} \tag{9}$$

The partial derivatives can be obtained through sensitivity analysis or automatic differentiation. However, the CQDC model is restricted to relatively simple geometries (e.g., spheres, boxes) due to the inherently non-differentiable nature of collision detection, as reflected in the contact Hessian $\frac{\partial\boldsymbol{J}}{\partial\boldsymbol{x}}$.

In this article, we propose to approximate the contact Hessian by numerical differentiation.

$$\frac{\partial\boldsymbol{J}}{\partial\boldsymbol{x}_i} \approx \frac{\boldsymbol{J}(\boldsymbol{x}+\boldsymbol{\Delta}_i) - \boldsymbol{J}(\boldsymbol{x})}{\boldsymbol{\Delta}_i} \tag{10}$$

where $\boldsymbol{x}_i$ is the i^{th} dimension of $\boldsymbol{x}$ and $\boldsymbol{\Delta}_i$ denotes the perturbation to $\boldsymbol{x}_i$. Most elements of the Jacobian matrix are zero, and this sparsity is predictable due to the tree structure of the multi-fingered hand. Thus, numerical differentiation only slightly increases computation time. Compared to smoothing over geometries, numerical differentiation is easy to implement, introduces no extra parameters, and remains effective.

C. Cost Terms of the High-Level DDP

1) Regulation Cost: The regulation cost includes two parts. The state-dependent part $l_{\text{reg}}(\boldsymbol{x})$ encourages contact-rich plans that lead the object to follow reference motions; The control-dependent part $l_{\text{reg}}(\boldsymbol{u})$ penalizes excessive control input and improves numerical conditions of the control problem. The regulation cost is formulated as the weighted quadratic norm:

$$\begin{aligned}l_{\text{reg}}(\boldsymbol{x}) &= \|\boldsymbol{x}^a - \boldsymbol{x}^a_{\text{ref}}\|^2_{\boldsymbol{W}^a_{\boldsymbol{x}}} + \|\boldsymbol{x}^u \ominus \boldsymbol{x}^u_{\text{ref}}\|^2_{\boldsymbol{W}^u_{\boldsymbol{x}}} \\ l_{\text{reg}}(\boldsymbol{u}) &= \|\boldsymbol{u}\|^2_{\boldsymbol{W}_u} \\ l_f(\boldsymbol{x}_N) &= \|\boldsymbol{x}^a_N - \boldsymbol{x}^a_{N,\text{ref}}\|^2_{\boldsymbol{W}^a_{\boldsymbol{x}_N}} + \|\boldsymbol{x}^u_N \ominus \boldsymbol{x}^u_{N,\text{ref}}\|^2_{\boldsymbol{W}^u_{\boldsymbol{x}_N}}\end{aligned} \tag{11}$$

where $\boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{N,\text{ref}}$ are the state references. If the desired object motion is continuous, then $\boldsymbol{x}^u_{\text{ref}}$ is increased from $\boldsymbol{x}^u_{\text{init}}$ with constant velocity. If the desired object motion is a fixed target pose, $\boldsymbol{x}^u_{\text{ref}}$ is then constructed by interpolating from $\boldsymbol{x}^u_{\text{init}}$ to the target.

Merely using the object tracking cost $\|\boldsymbol{x}^u \ominus \boldsymbol{x}^u_{\text{ref}}\|^2_{\boldsymbol{W}^u_{\boldsymbol{x}}}$ often causes the system to fall into local optima, i.e., the fingers simply follow the object motion and fail to actively break contacts. Typically, the expected finger motion accords with the periodic pattern (e.g., finger gaiting) [11]. Thus another term $\|\boldsymbol{x}^a - \boldsymbol{x}^a_{\text{ref}}\|^2_{\boldsymbol{W}^a_{\boldsymbol{x}}}$ is introduced to encourage local exploration and contact breaking of the fingers. We emphasize that $\boldsymbol{x}^a_{\text{ref}}$ is not a predefined trajectory, but rather a hand configuration that remains fixed during the manipulation. In practice, $\boldsymbol{x}^a_{\text{ref}}$ can be generated with grasp synthesis methods

or tuned with GUI. The basic idea behind $\boldsymbol{x}^a_{\text{ref}}$ is to imitate the way humans manipulate objects. Specifically, humans tend to pre-grasp the object and establish potential contacts for the following manipulation. Besides, this design aligns with previous works using contact-implicit methods [20], [26].

The weighting matrices $\boldsymbol{W}$ are designed as diagonal matrices, and the terminal weight $\boldsymbol{W}_{\boldsymbol{x}_N}$ is increased to improve convergence. The gradients that accumulate on $\boldsymbol{x}^a$ contain two parts: directly from the finger tracking cost and indirectly from the object tracking cost. The latter is passed down through the contact-implicit dynamics $\boldsymbol{f}$.

2) Joint Limits Cost: Dexterous hands typically have mechanical joint limits, for example, the joint limits of LEAP Hand are reported in [30]. Violating joint limits will cause the mechanical structures to collide and result in overloads, since the current is positively related to the output torque for common actuators. In addition, the differences between actual and predicted joint angles will result in significant modeling errors. Thus we design the joint limits cost as follows:

$$l_{\text{jnt}}(\boldsymbol{x}) = \frac{w_{\text{jnt}}}{2}\left(\|\min(\boldsymbol{x}^a - \underline{\boldsymbol{x}^a}, \boldsymbol{0})\|_2^2 + \|\max(\boldsymbol{x}^a - \overline{\boldsymbol{x}^a}, \boldsymbol{0})\|_2^2\right) \tag{12}$$

where w_{jnt} is the weighting parameter, and $\min, \max$ are element-wise operations. From (12), the joint limits cost is designed as a barrier function. In other words, l_{jnt} remains zero if $\boldsymbol{x}^a$ stays within the limits, and penalizes the dimensions that violate limits.

3) Self-Collision Cost: Using joint limits cost only will not be able to avoid collisions between fingers, especially when only part of the collision geometries are used. Thus we introduce the self-collision cost defined as:

$$l_{\text{sc}}(\boldsymbol{x}) = w_{\text{sc}} \sum_{\substack{i,j\in\{1,\cdots,n_l\}\\ i\neq j}} \sigma\left(d\left(\text{FK}_i(\boldsymbol{x}), \text{FK}_j(\boldsymbol{x})\right)\right) \tag{13}$$

where w_{sc} is the weighting parameter, n_l is the number of monitored links. To reduce the computational burden, instead of including all links in (13), we can first solve (1) without l_{sc} and only monitor the links in collision. Besides, FK_i is the forward kinematics of the i^{th} link, where the collision geometry is attached, and $d(\cdot,\cdot)$ denotes the translational distance. The activation function σ is in quadratic barrier form:

$$\sigma(a) = \begin{cases} \frac{1}{2}(a-\gamma)^2, & a < \gamma \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

According to (13), the state $\boldsymbol{x}$ is penalized only if the distance between any two links falls below the threshold γ.

D. Details of the High-Level MPC

The proposed OCP (1) with soft constraints (2) is solved with the Control-Limited DDP [31]. The optimal state and control trajectories are denoted as $\boldsymbol{X}^*, \boldsymbol{U}^*$. Meanwhile, the contact force trajectory is formulated as $\boldsymbol{\Lambda}^* = \{\boldsymbol{\lambda}_1^*, \cdots, \boldsymbol{\lambda}_{N-1}^*\}$, where $\boldsymbol{\lambda}^*$ is the dual solution corresponding to the conic constraint of (7). Note that $\boldsymbol{\lambda}^*$ excludes the moment component since the constraint is constructed assuming point contact. Several strategies are proposed to improve the generalization ability, convergence speed and smoothness of solutions.

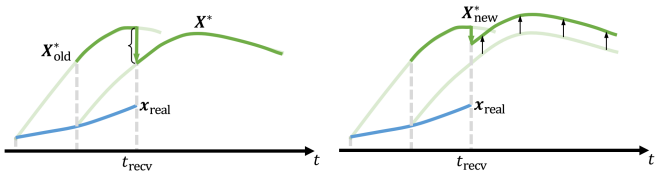

Fig. 6. Illustration of the trajectory interpolation and shifting. DDP produces trajectories $\boldsymbol{X}^*$ at fixed intervals, shown as the light green trajectory. The low-level module interpolates the previous solution $\boldsymbol{X}^*_{\text{old}}$ until a new one is received at t_{recv}. The interpolated trajectory is shown in dark green, and the arrow denotes the discontinuity. To reduce the discontinuity, we shift the new solution to $\boldsymbol{X}^*_{\text{new}}$. The interpolated trajectory is transformed into joint commands by the low-level module. Note that due to multiple delays, the real system states $\boldsymbol{x}_{\text{real}}$ always fall behind the commanded ones.

1) Warm Start with Shifted Trajectory: DDP typically needs a tolerable initial guess since it does only local optimization. If a trivial initial guess such as zero or random $\boldsymbol{u}$ is provided, DDP rarely converges. Thus we propose to warm-start DDP with the previous solution, based on the method's predictive ability. Specifically, we construct a spline with the previous solution $\boldsymbol{U}^*$ as control points. The spline is shifted $\beta_1\tau$ along the time axis, where τ is the time passed since the last iteration and β_1 influences the speed of finger motion. The initial guess $\boldsymbol{U}_0$ is interpolated on the new spline and padded with zeros at the last time step. Note that the solution might diverge without zero padding. Then, the initial guess $\boldsymbol{X}_0$ is established from the rollout of $\boldsymbol{U}_0$, starting at $\boldsymbol{x}_{\text{init}}$. This ensures the initial guess exactly starts from the current system state while sharing the same trend as the previous solution.

2) Discontinuity in the Joint Commands: Once DDP returns a new solution, the low-level module interpolates and updates the solution to get desired joint commands, until the next solution arrives. There is a natural discontinuity when the new solution replaces the old one, as shown in Fig. 6(a). Such discontinuity results in obvious jitters in hardware, which is unwanted in contact-rich manipulation. Thus we propose to reduce the discontinuity by shifting the new solution. This is described as $\boldsymbol{X}^*_{\text{new}} = \boldsymbol{X}^* + \beta_2\left(\boldsymbol{X}^*_{\text{old}}(t_{\text{recv}}) - \boldsymbol{X}^*(t_{\text{recv}})\right)$, where $\boldsymbol{X}^*_{\text{new}}$ is the shifted solution, $\boldsymbol{X}^*$ and $\boldsymbol{X}^*_{\text{old}}$ are the new and old solutions, and t_{recv} is the time when $\boldsymbol{X}^*_{\text{new}}$ will replace $\boldsymbol{X}_{\text{old}}$. Note that β_2 balances the smoothness and timeliness of solution.

The processing of reference trajectory $\boldsymbol{X}^*, \boldsymbol{U}^*, \boldsymbol{\Lambda}^*$ and the computation of joint commands will be discussed in Appendix F.

E. Details of the Contact Force-Motion Model

1) Single Contact Case: We make the following assumptions: 1) The finger's built-in controller can be described by the joint space PD, with stiffness $\boldsymbol{K}_P$ and damping $\boldsymbol{K}_D$; 2) There are mechanical compliance with both the object and the fingers, with environment stiffness $\boldsymbol{K}_e$; 3) The inertia,

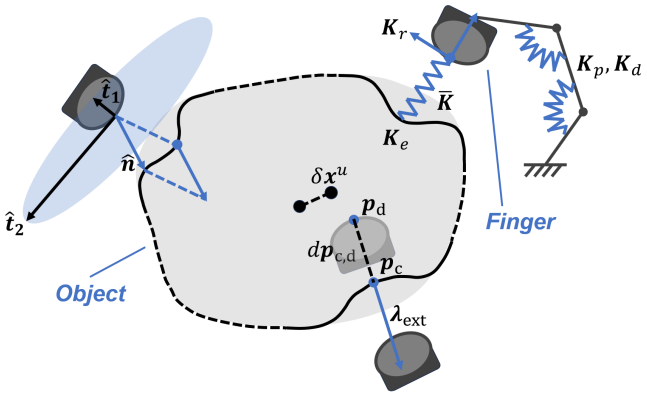

Fig. 7. Notations of the force-motion model. The finger is modeled with joint PD, i.e., $\boldsymbol{K}_p, \boldsymbol{K}_d$. At the contact point, the environment stiffness $\boldsymbol{K}_e$ and the finger's Cartesian stiffness $\boldsymbol{K}_r$ result in the equivalent stiffness $\bar{\boldsymbol{K}}_s$. The stiffness of single contact relates contact force $\boldsymbol{\lambda}_{\text{ext}}$ with desired motions $d\boldsymbol{p}_{c,d}$. The forces and motions of all contacts are correlated due to the object displacement $\delta \boldsymbol{x}^u$. The force controlled direction $\hat{\boldsymbol{n}}$ and motion controlled directions $\{\hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2\}$ are determined with the contact normal.

Coriolis and centripetal effects are neglected at the scale of contact forces; 4) The point contact model is considered and the contact wrenches are ignored. Under the single contact case. Gold et al. [24] proposed the relationship between contact force and finger motion:

$$\boldsymbol{\lambda}_{\text{ext}} = \bar{\boldsymbol{K}}_s d\boldsymbol{p}_{c,d} = \left(\boldsymbol{I} + \boldsymbol{K}_e \boldsymbol{K}_r^{-1}\right)^{-1} \boldsymbol{K}_e d\boldsymbol{p}_{c,d} \tag{15}$$

where $\boldsymbol{\lambda}_{\text{ext}} \in \mathbb{R}^3$ is the contact force, $d\boldsymbol{p}_{c,d} = \boldsymbol{p}_d - \boldsymbol{p}_c$, and $\boldsymbol{p}_c, \boldsymbol{p}_d \in \mathbb{R}^3$ are contact point positions on the object and robot finger, respectively. Note that $\boldsymbol{p}_d$ is associated with the *desired* finger configurations. Due to the controller compliance and contact force, the *desired* and *real* finger configurations are usually different. Besides, $\boldsymbol{K}_r$ is defined as the Cartesian stiffness of the finger at the contact point, where $\boldsymbol{K}_r^{-1} = \boldsymbol{J}_s(\boldsymbol{q}) \boldsymbol{K}_P^{-1} \boldsymbol{J}_s(\boldsymbol{q})^\top$ is configuration-dependent. Please refer to Fig. 7 for all the notations. Besides, the following equation is derived from joint space PD control:

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_d + \boldsymbol{K}_D^{-1} \left(\boldsymbol{K}_P(\boldsymbol{q}_d - \boldsymbol{q}) - \boldsymbol{J}_s^\top \boldsymbol{\lambda}_{\text{ext}}\right) \tag{16}$$

2) Force-Motion Model: Full Hand Case: The force-motion model in Appendix E.1 applies to the single contact case. There are possibly multiple contacts between the dexterous hand and the object, and we assume there are n_c contacts. The force-motion model of the complete system can be derived by repeating (15) for all n_c contacts.

$$\boldsymbol{\Lambda}_{\text{ext}} = \bar{\boldsymbol{K}} \left(\boldsymbol{P}_d - \boldsymbol{P}_c\right), \tag{17}$$

The joint motion equation (16) is now

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_d + \boldsymbol{K}_D^{-1} \left(\boldsymbol{K}_P(\boldsymbol{q}_d - \boldsymbol{q}) - \boldsymbol{J}^\top \boldsymbol{\Lambda}_{\text{ext}}\right) \tag{18}$$

where the matrices are defined as

$$\boldsymbol{\Lambda}_{\text{ext}} = \begin{bmatrix} \boldsymbol{\lambda}_{\text{ext},1} \\ \vdots \\ \boldsymbol{\lambda}_{\text{ext},n_c} \end{bmatrix}, \boldsymbol{P}_{\text{d}} = \begin{bmatrix} \boldsymbol{p}_{d,1} \\ \vdots \\ \boldsymbol{p}_{d,n_c} \end{bmatrix}, \boldsymbol{P}_{\text{c}} = \begin{bmatrix} \boldsymbol{p}_{c,1} \\ \vdots \\ \boldsymbol{p}_{c,n_c} \end{bmatrix},$$
$$\bar{\boldsymbol{K}} = \text{blkdiag}\left(\bar{\boldsymbol{K}}_{s,1}, \cdots, \bar{\boldsymbol{K}}_{s,n_c}\right), \boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{s,1} \\ \vdots \\ \boldsymbol{J}_{s,n_c} \end{bmatrix}. \tag{19}$$

The dimensions are $\boldsymbol{\Lambda}_{\text{ext}} \in \mathbb{R}^{3n_c \times 1}, \bar{\boldsymbol{K}} \in \mathbb{R}^{3n_c \times 3n_c}, \boldsymbol{P}_d, \boldsymbol{P}_c \in \mathbb{R}^{3n_c \times 1}, \boldsymbol{J}_{\text{stack}} \in \mathbb{R}^{3n_c \times n_q}$.

3) Details of the Coupled Force-Motion Model: We ignore the local deformation and assume the contact points move with the object. The resultant wrench should be constant under quasi-static assumptions and constant external force (i.e., gravity), $\boldsymbol{w}_{\text{ext}} = \text{Const}$. The following equation is then obtained with (17):

$$\delta \boldsymbol{w}_{\text{ext}} = \boldsymbol{G}_o \delta \boldsymbol{\Lambda}_{\text{ext}} = \boldsymbol{G}_o \bar{\boldsymbol{K}} \left(\delta \boldsymbol{P}_d - \delta \boldsymbol{P}_c\right) = 0 \tag{20}$$

Apply the implicit function theorem to (20) and we get

$$\begin{aligned} \frac{\partial \boldsymbol{x}^u}{\partial \boldsymbol{P}_d} &= \left[\frac{\partial(\delta \boldsymbol{w}_{\text{ext}})}{\partial \boldsymbol{x}^u}\right]^{-1} \frac{\partial(\delta \boldsymbol{w}_{\text{ext}})}{\partial \boldsymbol{P}_d} \\ &= -\left(\boldsymbol{G}_o \bar{\boldsymbol{K}} \boldsymbol{G}_o^\top\right)^{-1} \boldsymbol{G}_o \bar{\boldsymbol{K}} \end{aligned} \tag{21}$$

The coupled stiffness (4) can be obtained accordingly.

The analysis in Sec. II-C naturally applies to the case where the object moves freely. If the object is partially constrained (i.e., by a hinge or a supporting surface), (3)∼(4) still hold, except that the dimensions of some variables and matrices are adjusted accordingly. Specifically, we keep as many rows of $\boldsymbol{G}_0$ as possible, and ensures $\boldsymbol{G}_0 \bar{\boldsymbol{K}} \boldsymbol{G}_0^\top$ is full-rank. And $\boldsymbol{w}_{\text{ext}}$ represents the external wrench applied on these dimensions. Thus, $\boldsymbol{x}^u, \boldsymbol{w}_{\text{ext}}$ and $\boldsymbol{G}_o$ have the same number of rows.

F. Details of the Low-Level Tactile-Feedback Controller

1) Remarks on the Low-Level MPC Formulation: A schematic of the proposed feedback controller is visualized in Fig. 4 (C). In (6), $\text{FK}_i(\boldsymbol{q})$ refers to the forward kinematics of the i^{th} contact, and $\ominus$ is used to represent pose error. Note that if the pose error and contact force error terms are activated with proper $\boldsymbol{W}_{\boldsymbol{p}}$ and $\boldsymbol{W}_{\boldsymbol{\Lambda}}$, the MPC is akin to the HFMC. We keep the joint error term since the MPC-based HFMC easily gets into local minimum if the reference pose is not correctly defined, especially for large-range movements (i.e., reaching the grasping pose). When the joint error term is activated, the MPC is akin to Joint Space Control (JSC). Even though additional constraints are not considered in the problem, the MPC provides a unified and convenient framework to implement multiple controllers.

2) Determination of the Matrices: The MPC is initialized with measured joint position $\boldsymbol{q}$, last joint command $\boldsymbol{q}_d$ and measured force $\boldsymbol{\lambda}_{\text{ext}}$ collected from all contacts. The determination of state references $\boldsymbol{q}_{\text{ref}}, \boldsymbol{\Lambda}_{\text{ext, ref}}$ and weighting matrices will be discussed in this section. As for the reference motion, we construct a spline with the reference

trajectory $\boldsymbol{q}_{\text{ref}}\left(t; \boldsymbol{X}^*, h\right), t \in [t_0, t_0 + T]$ and sample on the spline. The same applies to the contact force reference. As for the weighting matrices, note that $\boldsymbol{W}_{\boldsymbol{\Lambda}} = \text{blkdiag}\left(\boldsymbol{W}_{\boldsymbol{\Lambda}_1}, \cdots, \boldsymbol{W}_{\boldsymbol{\Lambda}_{n_c}}\right)$. Thus the weighting matrices could be determined if all $\boldsymbol{W}_{\boldsymbol{p}_i}$ and $\boldsymbol{W}_{\boldsymbol{\Lambda}_i}$ are worked out. These sub-matrices are constructed from the contact normals. The inline collision detection of high-level MPC generates the contact normals of all active contacts. For a specific contact, the averaged contact normal over all time steps is denoted as $\hat{\boldsymbol{n}}$. The orthogonal basis $\left\{\hat{\boldsymbol{n}}, \hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2\right\}$ is then obtained by QR decomposition, and we define $\hat{\boldsymbol{T}} \triangleq \left[\hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2\right]$, as shown in Fig. 7. Following that, the matrices are defined as:

$$\boldsymbol{W}_{\boldsymbol{\Lambda}} = \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top, \quad \boldsymbol{W}_{\boldsymbol{p}} = \begin{bmatrix} \hat{\boldsymbol{T}}\hat{\boldsymbol{T}}^\top & \boldsymbol{0} \\ \boldsymbol{0} & w_{\text{ori}}\boldsymbol{I} \end{bmatrix} \tag{22}$$

where w_{ori} controls the weight of orientation. And the environment stiffness $\boldsymbol{K}_e$ is equal to a multiply of $\boldsymbol{W}_{\boldsymbol{\Lambda}}$, corresponding to the steepest growing direction of contact force. Note that the force controlled direction aligns with the contact normal. In other words, only the *normal contact force* is tracked to maintain the planned contact; While *tangential movements* are important for in-hand manipulation and should lie in the motion controlled subspace.

It is worth noting that the linear stiffness model (15) only takes effect for the active contacts. Thus all contacts are classified into two categories according to the force threshold $\underline{\boldsymbol{\lambda}}$. The matrices $\boldsymbol{W}_{\boldsymbol{\Lambda}}, \boldsymbol{W}_{\boldsymbol{p}}, \boldsymbol{K}_e$ in (22) are designed for contacts with $\|\boldsymbol{\lambda}_{\text{ext, ref}}\|_2 \geq \underline{\boldsymbol{\lambda}}$; While for contacts with $\|\boldsymbol{\lambda}_{\text{ext, ref}}\|_2 \leq \underline{\boldsymbol{\lambda}}$, the matrices are defined as $\boldsymbol{W}_{\boldsymbol{\Lambda}} = \boldsymbol{0}, \boldsymbol{W}_{\boldsymbol{p}} = \boldsymbol{I}, \boldsymbol{K}_e = \boldsymbol{0}$, since subtle joint motions may not cause changes in the contact forces, and the controller performs position tracking only. In addition, we set $\boldsymbol{W}_{\boldsymbol{q}} = \boldsymbol{0}$ for HFMC. As for JSC, we set $\boldsymbol{W}_{\boldsymbol{q}} = \boldsymbol{I}$ and $\boldsymbol{W}_{\boldsymbol{p}} = \boldsymbol{0}$. The determination of weighting matrices is summarized in Table III.

Once (6) is solved and the optimal control $\boldsymbol{u}^*$ is obtained, the hardware level joint commands are computed as $\boldsymbol{q}_d(t_0) + \boldsymbol{u}^*(t_0)\Delta t$, where Δt is the sampling time used for solving (6). The next MPC iteration is warm-started by shifting $\boldsymbol{u}^*$.

G. Other Simulation Results

1) Simulation Setup: We use MuJoCo [8] for rigid body simulation due to its compliant contact model, and use ROS to parallelize the high-level and low-level modules, resulting in a simulation environment which highly resembles the real-world setup. The high-level module generates references for finger motions and contact forces at 10 Hz, while the low-level module interpolates and adjusts the finger motions at 30 Hz. We select seven benchmark tasks for evaluation in the simulation and on the hardware, as shown in Fig. 8. Note that tasks 3) and 7) are completed using the Allegro Hand exclusively in simulation, whereas the other tasks are performed with the LEAP Hand, providing both simulation and real-world results.

- **Open Door**: The door handle rotates around its z-axis and is rotated by the dexterous hand either continuously or towards a target orientation. The system has $n_x = 17$, where n_x denotes the total DoFs of the hand and the object combined.
- **Open Door (With Wrist Movement)**: This task is similar to Open Door except that the wrist is allowed to translate freely and rotate around the handle's x-axis within -30°∼30°. The hand executes a three-fingered grasp, utilizing the thumb and two additional fingers. The system has $n_x = 17$.
- **Rotate Valve**: The capsule-shaped valve rotates around its z-axis and is rotated by the dexterous hand continuously. The system has $n_x = 17$.
- **Rotate Card**: The card moves freely on the table. The dexterous hand rotates the card around its z-axis either continuously or towards a fixed target orientation, while minimizing the card's translation. The thumb is not used in this task. The system has $n_x = 19$.
- **Slide Board**: The board moves freely on the table. The dexterous hand slides the board (with three fingers as in Rotate Card) along its y-axis either continuously or towards a target location. The translation along the z-axis and the rotation around the x-axis should be minimized. The system has $n_x = 19$.
- **Open Box**: The box is fixed on the table and the lid is lifted and rotated around the hinge joint. The dexterous hand should make contact with the lid using fingers other than the thumb and open it to a specified angle. The system has $n_x = 17$.
- **Rotate Sphere**: The sphere rotates freely in the SO(3) space with its center fixed. The dexterous hand rotates the sphere towards a target SO(3) orientation. The system has $n_x = 19$.

2) Implementation Details of the Contact-Implicit MPC: The high-level contact-implicit MPC is implemented with the Box-DDP solver in Crocoddyl [32] and the CQDC model from the codebase of [19]. In our Python implementation, we use the bindings of the C++-based Crocoddyl and CQDC model. We choose an MPC horizon $N = 10$ and the maximum solver iterations of 2 to guarantee adequate solution speed. Besides, we set $h = 0.1s$ and $\kappa = 100$ in (8). In the experiments described in Appendix G.3, G.4 and G.5, we solely evaluate the performance of the high-level motion-contact planner, where the first planned state is considered as the subsequent system state.

3) Solution Speed of Different Tasks: The elapsed time of different tasks during key sub-steps is shown in Fig. 9.

TABLE III
DETERMINATION OF THE WEIGHTING MATRICES

$\boldsymbol{W}$	HFMC		JSC
	$\|\boldsymbol{\lambda}_{\text{ext, ref}}\|_2 \geq \underline{\boldsymbol{\lambda}}$	$\|\boldsymbol{\lambda}_{\text{ext, ref}}\|_2 \leq \underline{\boldsymbol{\lambda}}$	
$\boldsymbol{W}_{\boldsymbol{q}} \in \mathbb{R}^{n_q \times n_q}$	$\boldsymbol{0}$		$\boldsymbol{I}$
$\boldsymbol{W}_{\boldsymbol{p}} \in \mathbb{R}^{6\times 6}$	$[\hat{\boldsymbol{T}}\hat{\boldsymbol{T}}^\top \ \ \boldsymbol{0};\ \ \boldsymbol{0} \ \ w_{\text{ori}}\boldsymbol{I}]$	$\boldsymbol{I}$	$\boldsymbol{0}$
$\boldsymbol{W}_{\boldsymbol{\Lambda}}, \boldsymbol{K}_e \in \mathbb{R}^{3\times 3}$	$\hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$	$\boldsymbol{0}$	$\hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^\top$
$\boldsymbol{W}_{\boldsymbol{u}} \in \mathbb{R}^{n_u \times n_u}$	$\boldsymbol{I}$		

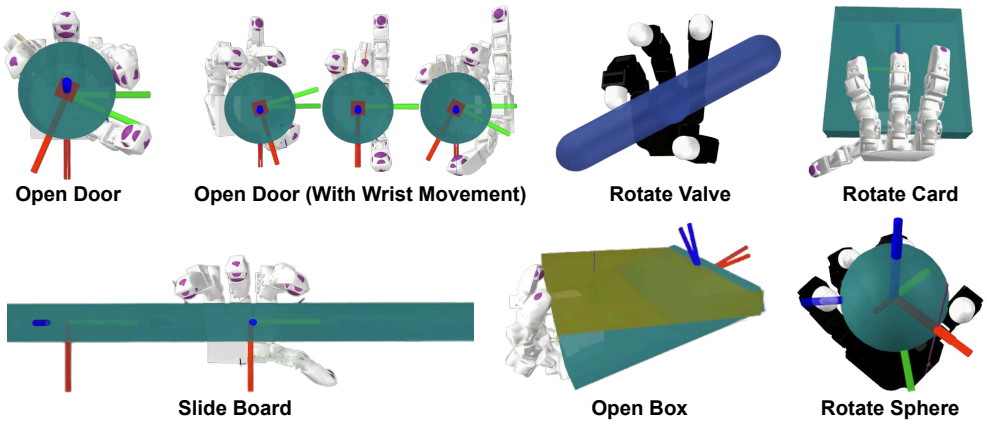

Fig. 8. Snapshots of the benchmark tasks used for evaluation in the simulation and on the hardware. The dexterous hands have simplified visual geometries and the purple spheres attached at each fingertip are the vollision geometries.

The results represent the time consumed in a single iteration, which is averaged over the whole manipulation sequence. The major time-consuming processes are the forward dynamics **calc** and the gradient computation **calc_diff**. Since DDP needs multiple rollouts to determine the proper step size, and it is difficult to truly parallelize these rollouts due to Python's Global Interpreter Lock mechanism [33], the computation of forward dynamics takes up most of the iteration time. The motion generation of more complex systems (i.e., with larger n_x) tends to consume longer time in each iteration. However, our selected tasks can generate motion-contact plans at approximately 8 to 15 Hz, which is sufficient for real-time planning. In real-world experiments, the low-level tactile-feedback controller operates at a higher frequency to track the planned references.

4) Influence of Different MPC Horizons: The **Valve Rotation** task is used to study the influence of choosing different MPC horizons. The fingers need to switch between different movement patterns to accomplish the task, such as stepping over and pushing the valve. The average rotation speed and the solution time per iteration are shown in Fig. 10. The valve hardly rotates under shorter horizons $N \leq 4$, and the fingers push the valve without stepping over and regrasping it, which indicates that shorter horizons lack predictive ability and easily lead to local optima. On the one hand, the rotation speed increases with longer horizons, which indicates improved solution quality. On the other hand, the solution time increases linearly with the MPC horizon, which results in degraded control frequency. Thus we choose $N = 10$ for all the tasks in the experiments, where the growth of rotation speed starts to decelerate.

5) Influence of the Smoothness in Forward Dynamics: We investigate the influence of the smoothness factor κ in the **Open Door** task, with the door handle replaced by a sphere of radius $0.06\,\mathrm{m}$. Different κ values are applied in the forward dynamics computation (8) to achieve varying smoothness levels. As shown in Fig. 11, excessively large κ ($\kappa \geq 500$) results in poor convergence and minimal door handle rotation due to overly conservative DDP step sizes. Conversely, very small κ ($\kappa = 10$) results in excessive smoothing, which enhances nonphysical behavior and also leads to poor performance. An appropriate κ ($\kappa = 100$) facilitates finger-gaiting and consistent rotation speed. Thus, we demonstrate that high-level modeling errors due to smoothing cannot be corrected by more realistic forward

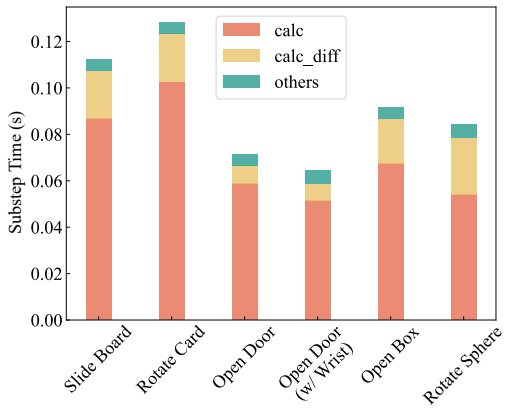

Fig. 9. The elapsed time of the high-level motion generation during key sub-steps. The 'calc' and 'calc_diff' represents the forward dynamics and gradient computation of the CQDC model, respectively. The total height of the bars indicate the average iteration time for each task.

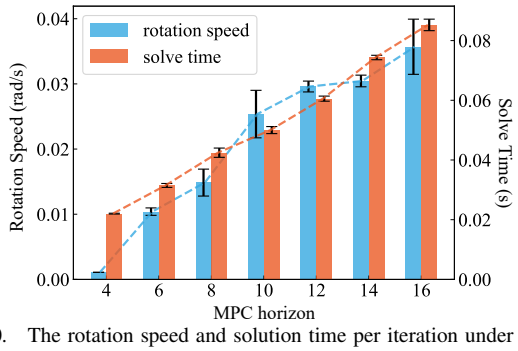

Fig. 10. The rotation speed and solution time per iteration under different MPC horizon in the valve rotation task. The error bars indicate the standard deviation. The dashed lines represent the average values.

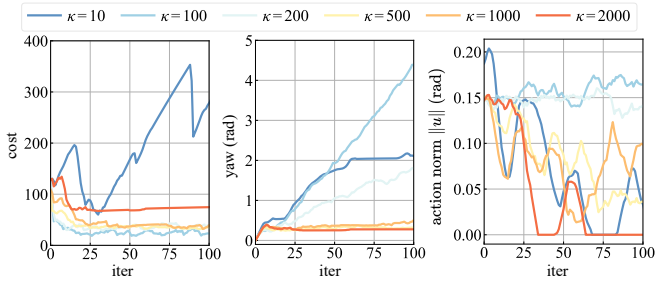

Fig. 11. The relationship between κ and optimization results, including the cost of DDP, the sphere's rotation (yaw angle), and the action norm. Warmer colors indicate a weaker smoothing effect.

dynamics alone. This underscores the necessity of the low-level controller for contact tracking, which compensates for high-level errors and improves manipulation performance, as shown in the following sections.

6) Implementation of the Baselines: First, the **openloop** baseline is selected from the state-of-the-art model-based framework [19], which first plans an offline trajectory, optimizes it with trajectory optimization, and then directly executes the trajectory in an open-loop fashion. In contrast, the proposed method improves the baseline through online planning, which is more robust to disturbances. For fair comparison, we implement the **openloop** baseline by running the proposed method without the low-level tactile-feedback tracking controller. The **openloop** baseline runs at 10 Hz. Second, the **CEM** and **iLQG** baselines are implemented with the codebase of MuJoCo MPC (MJPC) [25]. One major drawback of MuJoCo MPC is the need for accurate dynamics models. We run the algorithms in a ROS2 node and build a separate MuJoCo simulation environment for evaluation. Since these methods sample in the joint space and generate highly dynamic motions, which are harmful to potential hardware deployment, we clamp the delta joint positions between adjacent timestamps within $[-0.1, 0.1]\text{rad}$. This treatment only slightly degrades the performance. The **CEM** and **iLQG** baselines run at 100 Hz.

7) Evaluation Metrics in Simulation: The following metrics are used for comparison:

- **Success Rate**: The number of successful trials over all 100 trials. The trial is judged a success if the minimum orientation error is below 8 degree.
- **Average Task Error**: The average minimum orientation error among all 100 trials or the successful trials.
- **Task Error S.D.**: The standard deviation of the orientation error after the trial succeeds. This metric reflects whether the sphere can stabilize at the target orientation. Besides, this metric is averaged over all successful trials.
- **Average Task Time**: The average time consumed before the trial succeeds. This metric is averaged over all successful trials.
- **Average Joint Acceleration**: The average joint acceleration over all joints and all 100 trials. We sample the joint positions of different methods at a uniform frequency for fair comparison. This metric reflects the smoothness of actions.

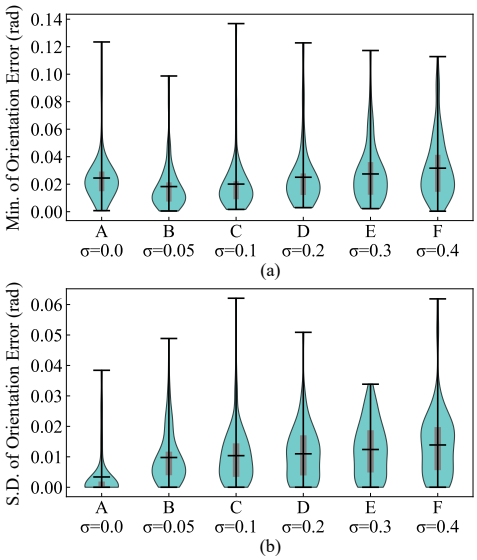

Fig. 12. The task error distribution for the sphere rotation task with noisy object orientation. The x-axis labels indicate the standard deviation of the added Gaussian noise. The bars represent the mean and extreme values, while the gray stripes denote the intervals between the first and third quartiles.

8) Discussions on the Learning-Based Baselines: Here we do not compare with learning-based methods since they cannot efficiently generalize to new tasks. In contrast, the proposed method seamlessly generalizes to various tasks in Fig. 8 with different object geometries and dexterous hands, without any additional training. The users can propose new prototype tasks by modifying a small number of hyper-parameters, without laborious reward engineering. However, learning-based methods should be adept at more dynamic tasks [12], [13], which is challenging for the proposed method with simplified models and relatively low control frequency.

9) Robustness Under Noise and Modeling Errors: First, to study the performance of the proposed method under sensory noise, we randomly perturb the sphere's observed orientation at each time step. The perturbation is composed of a random axis and the angle obeys normal distribution $\mathcal{N}(0, \sigma^2)$. We again track the 100 random target orientations in Sec. **??**. The task error and task error S.D. under different σ are shown in Fig. 12. The task error slightly increases as the noise increases. And the task error S.D. grows notably with larger noise. The results show that sensory noise causes unwanted finger and object motions. The stochasticity also leads to error reduction when $\sigma \leq 0.1$. Overall, the proposed method is robust to sensory noises, owing to the online planning and long-term predictive ability which reduce the sensitivity to noisy observations.

Second, to explore the influence of modeling errors, we vary the radius of the sphere in the simulation from 0.05m to 0.064m. We compare the proposed method with the MJPC (CEM) baseline. The nominal sphere radius retains 0.06m in the planning module for both methods. The distribution of task error among all 100 trials is shown in Fig. 13. For the

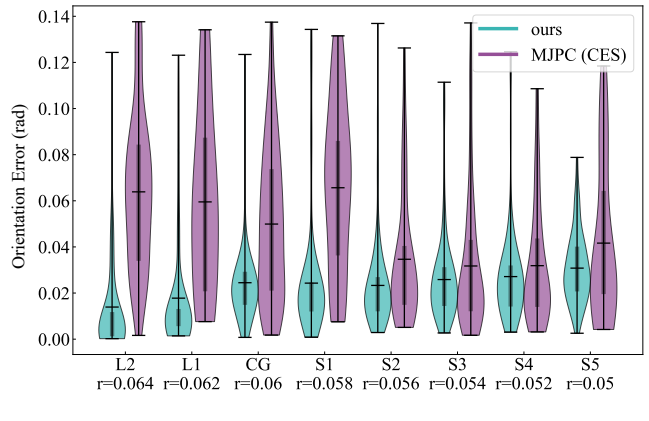

Fig. 13. The task error distribution for the sphere rotation task with modeling error. We use a consistent radius of $r = 0.06\text{m}$ in the planning model across different groups, while the x-axis labels indicate the actual radius used in the MuJoCo simulation. The control group (CG) has zero modeling error. Groups L1 to L2 have a larger radius, whereas groups S1 to S5 use a smaller radius for the simulated sphere.

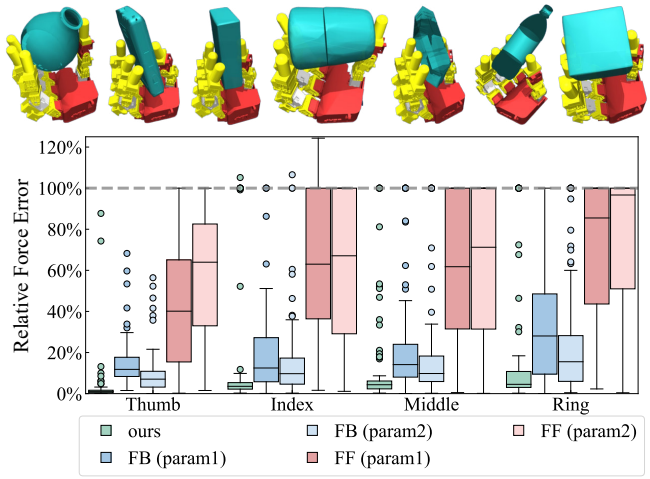

Fig. 14. Relative force tracking error when executing the grasps generated using BODex [34]. Examples of the hand configuration at the pre-grasping stage are shown on the top row. Among the methods, ours refers to the MPC-based feedback controller, and FB and FF refer to the baseline feedback and feedforward controllers, respectively. Besides, param1 and param2 refer to two representative sets of parameters.

proposed method, when the simulated sphere gets smaller, the task error increases since the fingers should deviate more from the planned trajectory to generate sufficient contact forces. Thus the task error decreases when the simulated sphere gets larger. We do not consider $r > 0.064\text{m}$ since the fingers will penetrate the simulated sphere. In contrast, the task error of MJPC (CEM) has stronger randomness, which is shown in the wider range of the error distribution. The task error decreases when $r > 0.058\text{m}$ because the fingers contact the sphere less often, resulting in less unwanted motions. Overall, the proposed method achieves higher accuracy and lower variance under modeling errors, compared to the predictive sampling baseline.

10) Analysis of the Tactile-Feedback Tracking Controller: We conduct two experiments to individually evaluate the proposed tactile-feedback tracking controller. First, the static grasping experiment is designed to validate the capability of force tracking. We use BODex [34] to generate 100 grasp poses with different objects in the test set. Since we are not proposing a grasping controller, we add the objects with damped free-floating joints and disable the gravity. In this way, we can focus on force tracking and neglect other effects. The static grasping is divided into three stages: 1) **Pre-Grasping**: The hand is initialized at the grasp pose with fingers slightly open to avoid contacts; 2) **Normal Estimation**: The fingers close gradually until the contact forces reach the pre-defined threshold. And the stacked contact normals $\boldsymbol{N} = [\boldsymbol{n}_{\text{thumb}}, \boldsymbol{n}_{\text{index}}, \boldsymbol{n}_{\text{middle}}, \boldsymbol{n}_{\text{ring}}] \in \mathbb{R}^{3\times n_c}$ are obtained; 3) **Grasp Control**: The proposed controller tracks desired contact forces. The desired forces are in the directions of the contact normals, and the magnitudes $\boldsymbol{f} \in \mathbb{R}^{n_c}$ are decided by solving the following optimization:

$$\min_{\boldsymbol{f}} \quad \|\boldsymbol{G}_o \boldsymbol{N}_{\text{aug}} \boldsymbol{f}\| + \lambda \|\boldsymbol{f} - m\mathbf{1}\| \tag{23}$$

$\mathbf{1}$ is an all-ones vector, $\boldsymbol{N}_{\text{aug}} \in \mathbb{R}^{3n_c\times n_c}$ is the augmentation of $\boldsymbol{N}$, and λ is a weighting factor. Note that (23) penalizes unbalanced wrench and regulates the force magnitudes around pre-set value m.

We compare the proposed controller with two baselines and the relative force tracking error is shown in Fig. 14. The baselines include the naive feedback controller (FB) and the feedforward controller (FF). These two controllers project Cartesian force to joint motions using the kinematic Jacobian. The feedback controller projects the error of contact force, while the feedforward controller projects the desired contact force. As shown in Fig. 14, the proposed controller achieves the smallest tracking error. The feedback baseline has slightly worse performance, and needs careful parameter tuning to avoid oscillations. The MPC design of our method suppresses these oscillations through penalty on the actions. The feedforward baseline has the worst performance and often generates excessive contact force. Although feedforward controller performs fairly well in locomotion tasks [20], [26], using feedback controller is necessary for manipulation where excessive force might damage the objects. Besides, our MPC design allows the integration of constraints, such as joint limits.

Second, the in-grasp object movement experiment is designed to evaluate the ability to jointly track finger forces and motions. In this experiment, the LEAP Hand grasps a cylinder using three fingers, excluding the middle finger. The marker at the cylinder's top center is directed to sequentially reach waypoints at the eight vertices of a 3 cm cube. After each waypoint, the marker returns to its initial position. Finger motions are generated using the code from [35], with desired forces re-estimated before each waypoint. We compare the proposed controller to the baseline from [35], which focuses solely on finger motion tracking. Normal contact force and position error are illustrated in Fig. 15. The proposed controller accurately tracks the target normal force, while the baseline exhibits significant fluctuations. While

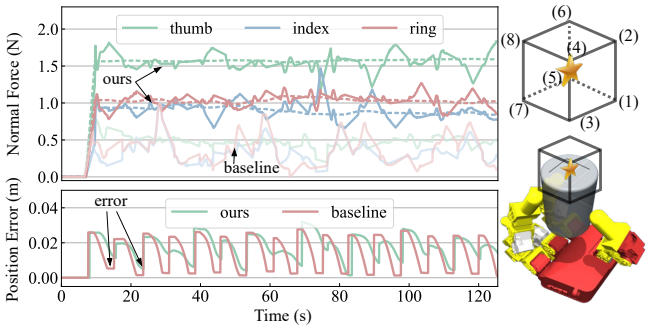

Fig. 15. Results of the in-grasp object movement experiment. **Right: Task snapshot**. The marker at the top center of the cylinder successively moves to 8 corners of the cube and back to the initial position. **Upper Left: Normal contact force**. Dashed lines represent the desired values, while solid lines show the execution results. The three dark lines indicate the results of the proposed controller, and the three light lines represent the baseline results. **Lower Left: Position tracking error**. The local minimums indicate tracking errors when the marker reaches the waypoints.

the baseline achieves high precision in waypoint tracking, the proposed method sacrifices some position accuracy to prioritize force tracking. This trade-off is reasonable, as precision is primarily ensured by the high-level planner in our approach. Overall, the proposed low-level controller effectively tracks both finger motion and contact forces.

H. Details of the Real-World Experiments

1) Implementation Details: The parameters of the high-level module are set as: $h = 0.1\,\mathrm{s}, \kappa = 100, N = 10, \beta_1 = 0.5, \beta_2 = 0.75$. We do not calibrate dynamic parameters such as mass and friction, which shows the generalization ability and robustness of the proposed method. And the nominal friction coefficient of each contact is set as $\mu_i = 1$. We refer the readers to [19] for more discussion on the influence of CQDC model parameters. The parameters of the low-level module are set as: $\boldsymbol{K}_P = 4\boldsymbol{I}, \boldsymbol{K}_D = \boldsymbol{I}, \boldsymbol{K}_e = 200\boldsymbol{I}, T = 0.3\,\mathrm{s}, \Delta t = 0.03\,\mathrm{s}, \underline{\boldsymbol{\lambda}} = 0.5\,\mathrm{N}$. To avoid excessive contact force, the magnitude of high-level reference force $\|\boldsymbol{\lambda}\|$ is softly thresholded by the function:

$$L\left(\frac{2}{1+e^{-k\|\boldsymbol{\lambda}\|}}-1\right) \tag{24}$$

where $k = 0.04, L = 2$.

2) Extension of the Open Door Task: We fix the wrist pose and rotate the door along the vertical axis 30 degrees counterclockwise (Fig. 16 (a)) or clockwise (Fig. 16 (b)). We use different sets of three fingers and perform the open door task twice for each configuration. The door handle rotation of different trials is shown in Fig. 16. The results show that the handle rotates at almost the same speed and converges to the target orientation under different conditions. For learning-based methods, it typically requires re-training the policy network to perform the task with different fingers or palm poses. In contrast, the proposed method generalizes to different conditions without additional training.

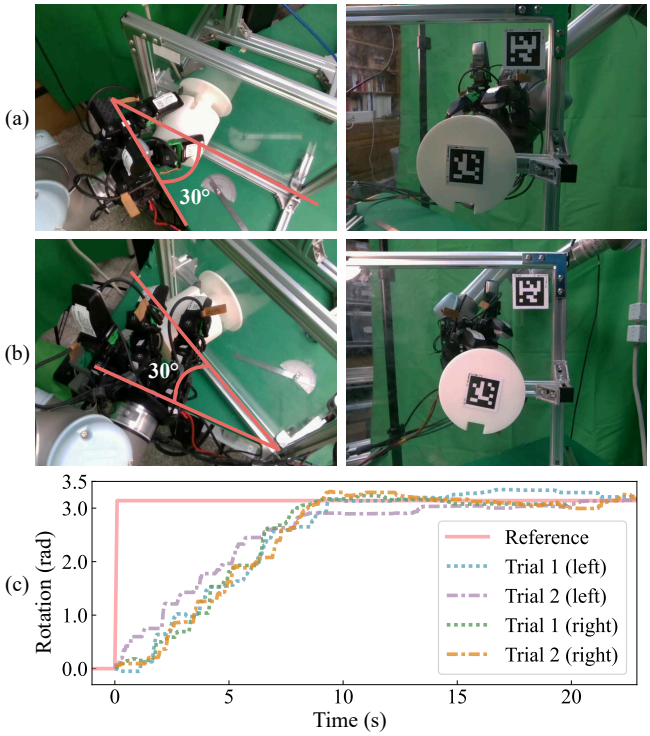

Fig. 16. (a) The door is rotated 30 degrees counterclockwise. The ring finger is not used. (b) The door is rotated 30 degrees clockwise. The index finger is not used. (c) Door handle rotation of different trials.

3) Experiments of the Rotate Card Task: The rotate card task requires the papery card to be rotated 180 degrees, while keeping the COM position stationary. The start pose and goal pose are visualized as the physical card and the ghost white border in Fig. 17 (a), respectively. We attach the AprilTag on the extended part of the card to avoid occlusions. The task is challenging since it relies on the coordination between fingers to produce torques that form the correct center of rotation. To test the robustness of the algorithm, we exert disturbances (rotational and translational) as shown in Fig. 17 (a). The card rotation of six trials is displayed in Fig. 17 (b). Note that Trial 1∼4 are implemented with the HFMC version of the proposed controller, while Trial 5 and 6 are implemented with the JSC version (Tab. III). The HFMC version rotates slightly slower since the Jacobian singularities of the stretched fingers make it difficult to track Cartesian motions along the z-axis, which in turn affects the manipulation. Nevertheless, all six trials successfully rotate the card to its target pose even under disturbances, as shown in Fig.17 (b). The card translation on the yOz plane is plotted in Fig. 17 (c). The error decreases immediately after disturbances, since the high-level motion generator switches between movement patterns according to the cost function. In addition, the planned and measured contact forces of the ring, middle, and index fingers are shown in Fig. 17 (d) (from the top down). Overall, the measured forces follow the references well. The planned middle finger force is almost always non-zero, while the index and ring fingers alternately make contacts, showing the auto-generated finger gaiting.

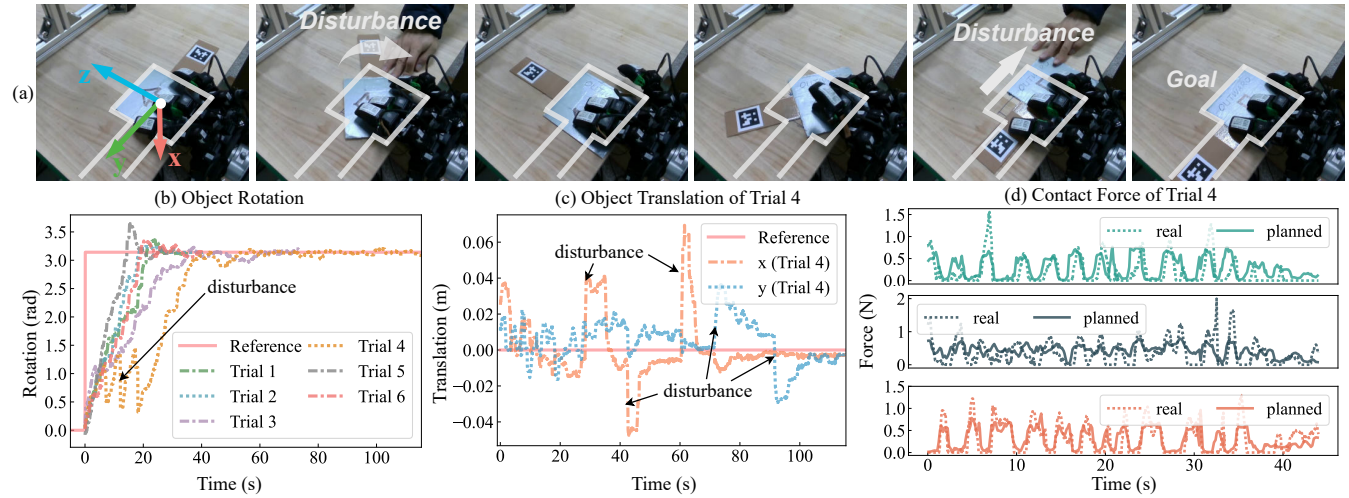

Fig. 17. Experiment results of the rotate card task with disturbance. (a) Snapshots of the task. The white border represents the target pose. Rotational and translational disturbances are exerted by the human operator. (b)(c) Object rotation and translation referred to the coordinate system shown in the first snapshot in (a). (d) Planned and measured contact forces of the ring, middle, and index fingers (from the top down).

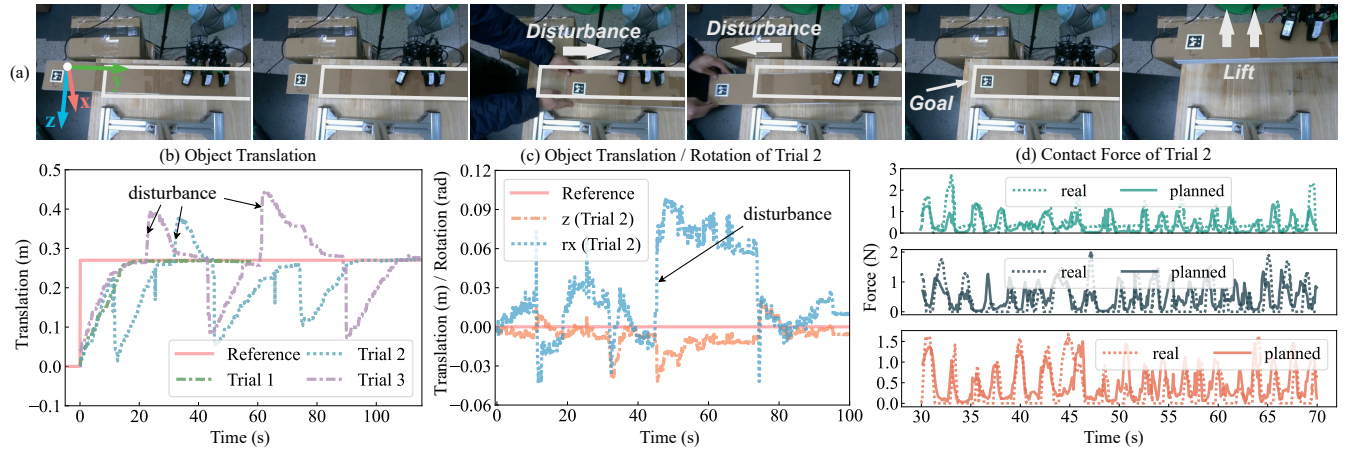

Fig. 18. Experiment results of the slide board task with disturbance. (a) Snapshots of the task. The white border represents the target pose. External disturbances are exerted by the human operator. (b)(c) Object rotation and translation referred to the coordinate system shown in the first snapshot in (a). (d) Planned and measured contact forces of the ring, middle, and index fingers (from the top down).

4) Experiments of the Slide Board Task: The slide board task requires the board to be translated around 27cm along the y-axis, so that the board can be lifted by grasping the middle. The snapshots are shown in Fig. 18 (a). The target pose is visualized by the white boarder. Since the fingertips apply relatively small forces to the board due to the servo's load limit, we customize a light board with foam core and cardboard shell to reduce the frictional wrench. The board translation and rotation are shown in Fig. 18 (b) and (c). The board quickly recovers from translational disturbance but recovers slowly from rotational disturbance, since the frictional torque is relatively large. The rotational disturbance changes suddenly at around 74s in Fig. 18 due to another human intervention. The planned and measured forces are shown in Fig. 18, where the finger gaiting is visualized.

5) Experiments of the Open Box Task: To validate the ability of the proposed method to generate non-periodic finger motions, we design the open box task. The snapshots and results are shown in Fig. 19. The task requires the fingers to lift up the lid using friction to a certain angle, and then lift the lid with one of the fingers, as shown in the simulation snapshots. The angle is set different for different boxes. It is worth noting that large modeling error exists in the task since the contact parts of the lid have significantly different geometries from the simplified cube in the model, as shown in the snapshots. We observe that the task has low success rate (fluctuates between 20%∼50% for different boxes). For each box, we plot the results of two successful trials in Fig. 19 (a)∼(d). Box D is the most difficult since the lid is heavy and requires the largest fingertip movement (the radius of rotation is large). Despite all the difficulties, the proposed method still finishes the task with generalization ability and robustness against modeling errors.

REFERENCES

[1] S. Cruciani, B. Sundaralingam, K. Hang, V. Kumar, T. Hermans, and D. Kragic, "Benchmarking in-hand manipulation," *IEEE Robotics and*

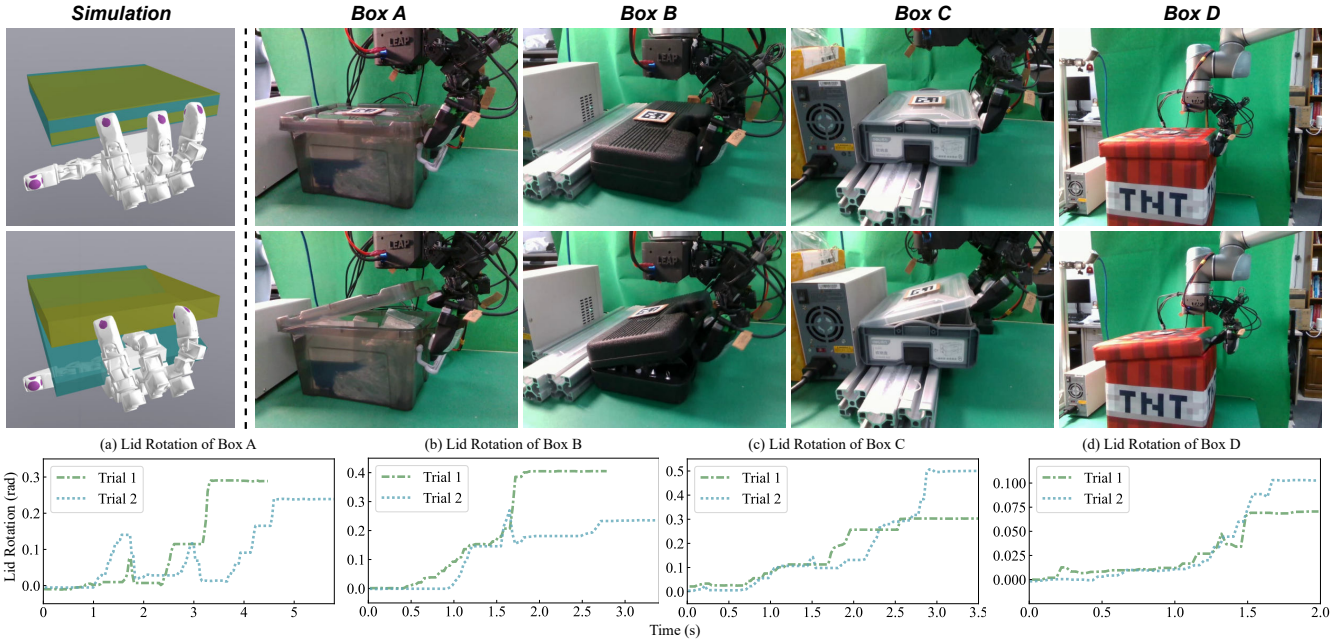

Fig. 19. Experiment results of the open box task. At the top are snapshots of the simulation and the real-world experiments on four different boxes A∼D. The first row shows the initial state, and the second row shows the state when the box is opened. (a)∼(d) Lid rotation of the trials with different boxes.

Automation Letters, vol. 5, no. 2, pp. 588–595, 2020.

[2] R. R. Ma and A. M. Dollar, "On dexterity and dexterous manipulation," in *2011 15th International Conference on Advanced Robotics (ICAR)*. IEEE, 2011, pp. 1–7.

[3] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.

[4] Z. Gu, J. Li, W. Shen, W. Yu, Z. Xie, S. McCrory, X. Cheng, A. Shamsah, R. Griffin, C. K. Liu, *et al.*, "Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning," *arXiv preprint arXiv:2501.02116*, 2025.

[5] B. Sundaralingam and T. Hermans, "Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation," *Autonomous Robots*, vol. 43, pp. 469–483, 2019.

[6] ——, "Geometric in-hand regrasp planning: Alternating optimization of finger gaits and in-grasp manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 231–238.

[7] Q. Le Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, "Contact models in robotics: a comparative analysis," *IEEE Transactions on Robotics*, 2024.

[8] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[9] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[10] L. Röstel, J. Pitz, L. Sievers, and B. Bäuml, "Estimator-coupled reinforcement learning for robust purely tactile in-hand manipulation," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[11] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "General in-hand object rotation with vision and touch," in *Conference on Robot Learning*. PMLR, 2023, pp. 2549–2564.

[12] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, p. eadc9244, 2023.

[13] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5977–5984.

[14] M. Yang, chenghua lu, A. Church, Y. Lin, C. J. Ford, H. Li, E. Psomopoulou, D. A. Barton, and N. F. Lepora, "Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch," in *CoRL Workshop on Learning Robot Fine and Dexterous Manipulation: Perception and Control*, 2024. [Online]. Available: https://openreview.net/forum?id=Bt5upUsbvu

[15] S. Cruciani, C. Smith, D. Kragic, and K. Hang, "Dexterous manipulation graphs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2040–2047.

[16] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, "Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8262–8268.

[17] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason, "Enhancing dexterity in robotic manipulation via hierarchical contact exploration," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 390–397, 2023.

[18] H. Zhu, A. Meduri, and L. Righetti, "Efficient object manipulation planning with monte carlo tree search," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 628–10 635.

[19] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on robotics*, 2023.

[20] V. Kurtz, A. Castro, A. Ö. Önol, and H. Lin, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," *arXiv preprint arXiv:2309.01813*, 2023.

[21] A. Aydinoglu, A. Wei, W.-C. Huang, and M. Posa, "Consensus complementarity control for multi-contact mpc," *IEEE Transactions on Robotics*, 2024.

[22] Y. Jiang, M. Yu, X. Zhu, M. Tomizuka, and X. Li, "Contact-implicit model predictive control for dexterous in-hand manipulation: A long-horizon and robust approach," *arXiv preprint arXiv:2402.18897*, 2024.

[23] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[24] T. Gold, A. Völz, and K. Graichen, "Model predictive interaction control for robotic manipulation tasks," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 76–89, 2022.

[25] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv preprint arXiv:2212.00541*, 2022.

[26] G. Kim, D.-K. Kang, J. ha Kim, S. Hong, and H.-W. Park, "Contact-implicit model predictive control: Controlling diverse quadruped

motions without pre-planned contact modes or trajectories," *The International Journal of Robotics Research*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:266209996

[27] L. Zhang, Y. Wang, and Y. Jiang, "Tac3d: A novel vision-based tactile sensor for measuring forces distribution and estimating friction coefficient distribution," *arXiv preprint arXiv:2202.06211*, 2022.

[28] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4193–4198.

[29] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[30] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," *Robotics: Science and Systems (RSS)*, 2023.

[31] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[32] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[33] loco 3d, "Issue #1042 - crocoddyl," https://github.com/loco-3d/crocoddyl/issues/1042, 2025, accessed: 2025-04-11.

[34] J. Chen, Y. Ke, and H. Wang, "Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization," *arXiv preprint arXiv:2412.16490*, 2024.

[35] M. Yu, Y. Jiang, C. Chen, Y. Jia, and X. Li, "Robotic in-hand manipulation for large-range precise object movement: The rgmc champion solution," *IEEE Robotics and Automation Letters*, pp. 1–8, 2025.