

# Rearrangement-Based Manipulation via Kinodynamic Planning and Dynamic Planning Horizons

Kejia Ren, Lydia E. Kavraki and Kaiyu Hang

*Abstract*—Robot manipulation in cluttered environments often requires complex and sequential rearrangement of multiple objects in order to achieve the desired reconfiguration of the target objects. Due to the sophisticated physical interactions involved in such scenarios, rearrangement-based manipulation is still limited to a small range of tasks and is especially vulnerable to physical uncertainties and perception noise. This paper presents a planning framework that leverages the efficiency of sampling-based planning approaches, and closes the manipulation loop by dynamically controlling the planning horizon. Our approach interleaves planning and execution to progressively approach the manipulation goal while correcting any errors or path deviations along the process. Meanwhile, our framework allows the definition of manipulation goals without requiring explicit goal configurations, enabling the robot to flexibly interact with all objects to facilitate the manipulation of the target ones. With extensive experiments both in simulation and on a real robot, we evaluate our framework on three manipulation tasks in cluttered environments: grasping, relocating, and sorting. In comparison with two baseline approaches, we show that our framework can significantly improve planning efficiency, robustness against physical uncertainties, and task success rate under limited time budgets.

## I. INTRODUCTION

Research in robotic manipulation has investigated how to reconfigure objects in different task scenarios and robot-object-environment formulations, such as grasping, pick-and-place, in-hand manipulation, dual-arm manipulation, objects sorting, and placement [1]–[3]. Traditionally, most manipulation tasks have been studied as standalone problems without considering the physical interactions with any other objects. For example, grasping is often modeled as a static process where a hand needs to reach a stable grasp on the target object without touching anything else. Although such isolated formulations can simplify the problem and are in many cases sufficient, e.g., in-hand manipulation only concerns a hand and the grasped object, they are inherently oversimplified and not applicable to many real-world settings, since the target objects are not always located in free spaces. Importantly, even if we can sequentially manipulate one object at a time to achieve certain goals, concurrent manipulation of multiple objects has proven a much more efficient strategy for various tasks [4], especially when multiple objects need to be relocated relatively to each other.

As such, rearrangement-based manipulation, defined as a class of problems requiring a robot to concurrently manip-

The authors are with the Department of Computer Science, Rice University, Houston, TX 77005, USA. {kr43, kavraki, kaiyu.hang}@rice.edu.

In this work, KR and KH are supported by NSF CMMI-2133110 and Rice University Funds, and LEK is supported in part by NSF 2008720.

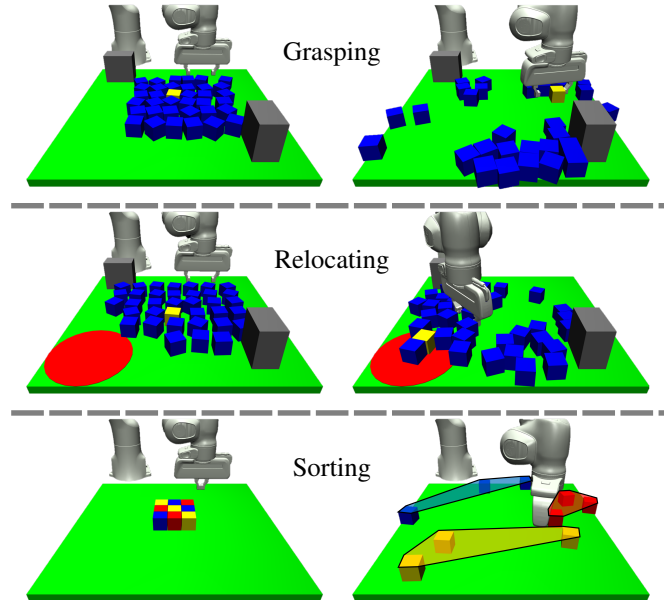


Fig. 1: Example rearrangement-based planning tasks. The left column shows the start states, and the right column shows the goal states achieved by our proposed planner. *Grasping*: The robot approaches a grasp for the target object (yellow) surrounded by other movable objects (blue) while avoiding collisions with the obstacles (gray); *Relocating*: The robot pushes the target object (yellow) to a goal region (red); *Sorting*: The robot rearranges to separate objects of different classes into clusters.

ulate multiple objects to achieve the task goal, has been broadly studied [5], [6]. A few example tasks are illustrated in Fig. 1. Nevertheless, it still remains a difficult class of problems because: 1) it has proven to be NP-hard [7]; and 2) as the problem space is composed of the robot configuration, robot control, and the configurations of all objects, the dimensionality of such problems is much higher than most other manipulation problems, rendering them computationally very challenging. Such challenges can be intuitively seen in the example grasping task, where the robot needs to rearrange the surrounding objects so that the gripper can reach a stable pre-grasp pose. A major difficulty of this task is that, while the surrounding objects are being rearranged, the target object is simultaneously moved by multiple object-object interactions.

Moreover, almost all existing nonprehensile rearrangement planners rely on either hand-crafted physics models or simulators to model the system state transitions. Such methods usually assume very precise models, e.g., the geometries and physical parameters of objects to be available during the planning time, which is in general infeasible. This fact brings

up a significant problem: the inherent discrepancies between the system models and the real world will accumulate errors throughout the entire execution of the planned actions. Thus, even if a planner has successfully generated a plan, the real execution will likely deviate the manipulation from the desired path and cause task failures. In other cases, if the initial perception of the system’s state is not reliable [8], or if the system’s state has been changed during execution due to human interruptions or environment changes, most open-loop approaches will merely continue executing without the capability to actively correct the errors.

In this work, inspired by dynamic window-based approaches for robot navigation in partially observable environments [9], we propose a kinodynamic planning framework for rearrangement-based manipulation with dynamic planning horizons. In brief, our approach is able to:

- 1) monitor the planning progress and dynamically determine the planning horizons, and direct the planning into more task-relevant subspaces to significantly improve the planning efficiency;
- 2) react to physical and perception uncertainties online, and work with imperfect system models, e.g., inaccurate object geometries, to progressively generate and execute rearrangement actions while correcting observed errors;
- 3) address various rearrangement problems, with and without explicitly defined goal configurations, to allow the robot to flexibly interact with all objects to facilitate the manipulation of the target objects.

## II. RELATED WORK

*Motion Planning:* In physical interaction-based motion planning problems where the robot-object-environment configurations are constantly changed, kinodynamic planning has been investigated to jointly model the robot configuration, the robot control, and the system transitions [10]. Among other approaches, sampling-based kinodynamic planning algorithms have been widely employed due to its great efficiency and generalizability. As compared with kinodynamic problems in collision-free environments [11], however, kinodynamic manipulation planning is much more complex due to the challenges of the dramatically increased problem dimensionality, highly nonlinear physics, and uncertainties in perception. Rearrangement-based manipulation is in particular a difficult set of problems for kinodynamic planning. In addition to the aforementioned challenges, as rearrangement is often about the relative reconfiguration between objects, it is infeasible for a planner to always explicitly define goal configurations. In this work, inspired by the work of robot navigation planning in partially observable or dynamically changing spaces [9], [12], we introduce progress control into kinodynamic manipulation planning. By dynamically adapting the planning horizon, our method is able to progressively plan the manipulation motions with significantly improved efficiency, and can handle problems without explicitly-defined goal configurations.

*Planning-based Rearrangement:* Kinodynamic RRT-based planning algorithms have shown promising potentials in rearrangement tasks. Using a problem-specific contact model under quasi-static assumptions, [13] analytically plans a diverse set of pushing motions but prohibits object-object interactions. Based on an efficient physics simulator, multi-object interactions are enabled, and dynamic motions of objects, e.g., rolling, can be incorporated [14]. Additionally by modeling the uncertainties in physics [15], or optimizing a continuous motion trajectory online [16], grasping in cluttered environments has been achieved by locally rearranging the occluding objects. Further, rearrangement tasks with relative goals, e.g., sorting, have been addressed by learning-based Monte Carlo Tree Search [17], and iterative local search to concurrently manipulate a large amount of objects [18]. However, the existing approaches either are not able to address the physical uncertainties during execution, or require very complex modeling of physics and sophisticated problem-specific heuristics, making them difficult to be easily generalized to various rearrangement-based manipulation problems. In contrast, based on any physical simulators, even without precise physical models, our proposed framework is able to react to physical uncertainties online, and generalize to complex tasks based on simple heuristics.

*Learning-based Rearrangement:* Recently, data-driven rearrangement planning has been extensively studied to tackle various tasks. In end-to-end settings, pushing-based relocation [19], multi-object rearrangement and singulation [20], [21], rearrangement-based grasping [22], etc., have been formulated as policy-learning problems to reactively generate robot actions online. Although such approaches have greatly simplified the system pipeline and allow for direct action generation purely based on the input images, as a common challenge, they in general require a large amount of training data for specific tasks, while the learned models are difficult to be transferred to achieve different tasks [23].

## III. PROBLEM STATEMENT

We formulate rearrangement-based manipulation planning as a kinodynamic motion planning problem. Given a bounded workspace  $\mathcal{W} \subset SE(2)$ , containing a robot manipulator,  $N$  movable objects to manipulate, and a set  $\mathcal{O}$  of static obstacles, we aim to find a sequence of robot motions, called a motion plan, such that the environment will be rearranged to reach a state satisfying the goal criterion.

### A. Terminology

1) *State Space:* Formally, we denote the robot state as  $q^R \in \mathcal{Q}^R \subset \mathbb{R}^r$ , where  $\mathcal{Q}^R$  is the robot configuration space and  $r \in \mathbb{R}$  is the robot’s degree of freedom. Let the state of a movable object be  $q^i \in \mathcal{Q}^i \subset \mathcal{W}$ , where  $\mathcal{Q}^i \subset SE(2)$ , and  $i \in \{1, \dots, N\}$ . The state space of the planning problem is then defined by the Cartesian product  $\mathcal{Q} = \mathcal{Q}^R \times \mathcal{Q}^1 \times \dots \times \mathcal{Q}^N$ , and a system state  $q \in \mathcal{Q}$  is denoted by a tuple  $q = (q^R, q^1, \dots, q^N)$ . A state  $q$  is valid only when the robot does not collide with itself or any static obstacle in  $\mathcal{O}$ , and all the movable objects are inside the workspace  $\mathcal{W}$ . All the

valid states compose the valid state space  $Q^{valid} \subseteq Q$ . Note that,  $Q^{valid}$  is different from the  $C^{free}$  space in traditional motion planning problems, as the contacts between movable objects and static obstacles, as well as between any pair of movable objects or the robot are allowed for a valid state.

2) *Control Space and Transition Function*: The control space  $U \subset \mathbb{R}^r$  is a sampleable continuous space consisting of all controls the robot is allowed to perform. We denote by a transition function,  $\Gamma : Q^{valid} \times U \mapsto Q$ , to represent the physics laws of the real world, which maps a state  $q_t \in Q^{valid}$  and a control action  $u_t \in U$  at time  $t$  to the state outcome at the next step  $q_{t+1} \in Q$ .

3) *Goal Criterion*: We define the goal criterion as a function  $g : Q^{valid} \mapsto \{0, 1\}$  specified by the manipulation task. Therefore, the goal region of the planning problem is defined by a set  $Q_G = \{q \in Q^{valid} \mid g(q) = 1\}$  of all states that satisfy the goal criterion.

## B. Problem Formulation

Given a start state  $q_{t_0} \in Q^{valid}$ , we aim to find a sequence of  $K$  robot controls  $\tau = \{u_{t_0}, \dots, u_{t_k}, \dots, u_{t_K}\}$  such that:

- The end state arrives at a configuration inside the goal region:  $\Gamma(q_{t_K}, u_{t_K}) \in Q_G$ ; and
- All the intermediate states along the plan are valid, i.e.,  $\forall k : q_{t_k} \in Q^{valid}$ .

However, due to the modeling inaccuracies, physical and perception uncertainties, the control sequence  $\tau$  will likely result in states different from what the planner predicted and cause task failures. As will be described below, in this work, we reformulate the problem by iteratively finding segments of controls  $\bar{\tau} \subset \tau$ , and interleave planning and robot execution between the control segments to close the manipulation loop, so that the system state is progressively transitioned towards the goal region  $Q_G$ .

## IV. KINODYNAMIC MANIPULATION PLANNING WITH DYNAMIC HORIZONS

To address the problem defined in Sec. III, we base our approach on the sampling-based kinodynamic RRT (kdRRT) framework [10], as it provides the ability to explore large high-dimensional state spaces, and can be easily integrated with any physics models or simulators to facilitate the generalization to various tasks.

Different from purely geometry-based motion planning algorithms, which explore the search space by sampling random state configurations and connect them to the search tree via linear interpolation, kinodynamic planning is much more complex due to the highly nonlinear system dynamics. In kinodynamic planning, given any sampled  $q_{rand}$ , one needs to find its nearest node,  $q_{near}$ , in the search tree, randomly sample a set of  $M$  controls at  $q_{near}$ ,  $\{u^1, \dots, u^M\}$ , and then selects one  $u^*$  to expand the search tree, such that the distance between  $\Gamma(q_{near}, u^*)$  and  $q_{rand}$  is minimized. This strategy of tree expansion can be computationally very expensive, especially for rearrangement problems involving multiple objects and many concurrent contacts, since adding

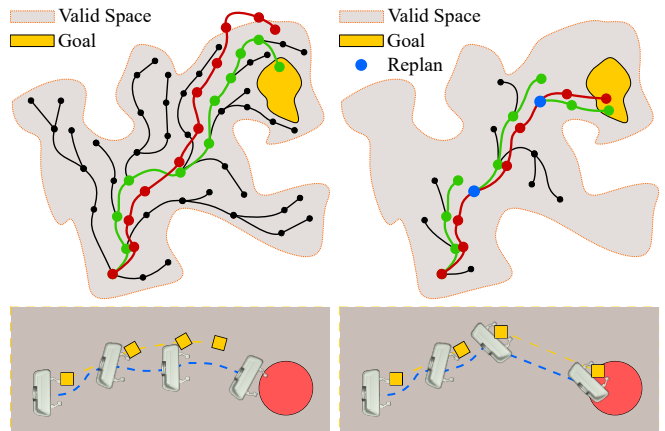


Fig. 2: A schematic plot of dhRRT (Right), detailed in Sec. IV-A, compared to kdRRT (Left). *Left*: The kdRRT extensively explores the entire valid space, and the execution (red) does not lead to the goal due to uncertainties, even if it finds a feasible plan (green). *Right*: The dhRRT progressively grows its tree towards the goal, with replanning along the way, and finally reaches the goal.

each node into the tree requires  $M$  times of physics calculation. Therefore, it could take up to several minutes to find a motion solution by expanding the tree very extensively in  $Q^{valid}$  [13]. In addition, as illustrated in Fig. 2, given a motion solution (green), the real-world execution (red) can deviate due to the physical uncertainties.

Another major challenge of such problems is the motion constraints. Although a robot arm can move in  $SE(3)$ , the motion of its end-effector has to be constrained within  $\mathcal{W}$  to rearrange the objects, although sometimes teleport can be achieved via free-space motions. Therefore, control sampling needs to be constrained in a task manifold. For this, specialized planners can be used [24], [25]. In this work, controls  $v \in se(2)$  are sampled for the end-effector only, and are subsequently converted by Jacobian-based local projections,  $JACOBIANPROJECTION(v)$ , to check if any  $u \in U$  can realize the sampled controls, as detailed in Sec. IV-B.

### A. Planning with Dynamic Horizons

To address the aforementioned challenges, we propose a kinodynamic RRT-based framework by incorporating Dynamic Horizon control in the planning process, and we term it as *dhRRT*.

For this, instead of explicitly defining goal configurations as in traditional settings, we require the task to be represented by a heuristic function:  $h : Q^{valid} \mapsto \mathbb{R}$ , such that when the state of the system moves closer to the task goal,  $h(\cdot)$  will decrease and will drive the system towards the goal region to eventually complete the task. In general, to guarantee that a search algorithm will be able to find an optimal solution for the task of  $h(\cdot)$ , the heuristic function has to be admissible and monotonic [26], meaning that  $h(\cdot)$  should never overestimate the cost and should monotonically decrease as the state moves closer to the goal. However, it is infeasible to always prove or guarantee these two criteria due to the complexity of problems in the real world. In practice, since most rearrangement-based planning problems

can be defined with simple cost-decreasing functions, non-optimal heuristics can also successfully drive the search to find solutions, although without optimality guarantees. As will be described in Sec. V and shown with experiments, our planning framework can easily integrate such functions to achieve various tasks.

While we grow the search tree similarly to kdRRT algorithms,  $h(\cdot)$  can be used to inform us about the planning progress. Given a search tree  $T$  rooted at the start state, every node added in the tree represents a state  $q_t$ , with its inward edge representing a control  $u_t$  that transitioned the state to  $q_t$ . During the tree expansion, we monitor the progress and will execute the current best control segment once good enough progress, determined by a threshold  $p \in \mathbb{R}$ , can be made by a leaf node.

---

### Algorithm 1 The dhRRT algorithm

---

**Input:** Start state  $q_{t_0}$ , goal region  $\mathcal{Q}_G(\cdot)$ , heuristic  $h(\cdot)$ , progress threshold  $p$ , tree limit  $D_{max}$   
**Output:** Control sequence  $\tau$

- 1:  $T \leftarrow \{nodes = \{(q_{t_0}, 0)\}, edges = \emptyset\}$
- 2:  $\tau \leftarrow \{\}, q^* \leftarrow Null$
- 3: **while** TIME.AVAILABLE() **do**
- 4:    $T \leftarrow \text{EXPANDTREE}(T)$  ▷ Alg. 2
- 5:    $\tau \leftarrow \text{EVALUATEPROGRESS}(T, h, p, D_{max})$  ▷ Alg. 3
- 6:   **if**  $\tau \neq \{\}$  **then**
- 7:      $q^* \leftarrow \text{EXECUTECONTROLS}(\tau)$  ▷ Observe Real State
- 8:     **if**  $q^* \in \mathcal{Q}_G$  **then** ▷ Task Complete
- 9:       **return**
- 10:    **end if**
- 11:     $T \leftarrow \{nodes = \{(q^*, 0)\}, edges = \emptyset\}$
- 12:     $\tau \leftarrow \{\}$ ,
- 13:     $q^* \leftarrow Null$
- 14:    **continue**
- 15:    **end if**
- 16: **end while**

---

As such, our planning is controlled by a *horizon* that dynamically changes in terms of the current state and motions. After each execution, our system observes the current state, which is likely to be different from the plan, and then repeats this procedure with different dynamically determined horizons, until the goal is reached. Note that, sometimes the robot can freely move around for a while without touching any object, hence making no positive progress. This is especially likely when the robot needs to relocate itself before making any rearrangement. To allow such motions without enforcing the robot to manipulate at every step, as well as facilitating random trap-escaping actions, the planning horizon is further limited by a threshold,  $D_{max}$ , of the maximum tree depth. If not enough progress can be made when  $D_{max}$  is reached, our algorithm will execute the best solution so far. The algorithm is summarized in Alg. 1.

As illustrated in Fig. 2, rather than planning and executing the entire control sequence, our approach progressively transitions the system state towards the goal region with control segments, while observing the state in the real world after every EXECUTECONTROLS( $\cdot$ ), making it possible to close the manipulation loop to deal with errors along the path. In practice, the dynamic horizon threshold  $p$  can be determined

in terms of the expected magnitude of physical uncertainties, as well as the granularity of the physics models. By setting  $p$  to a smaller value, the system will be more reactive, however, less efficient in finding solutions. Meanwhile, the tree depth limit  $D_{max}$  can be set to smaller values to avoid getting trapped in local optimum via more random local motions, while a larger  $D_{max}$  can allow more aggressive dynamic horizon control.

---

### Algorithm 2 ExpandTree( $\cdot$ )

---

**Input:** Current motion tree  $T$   
**Output:** Expanded tree  $T$

- 1:  $q_{rand} \leftarrow \text{SAMPLESTATE}()$
- 2:  $q_{near} \leftarrow \text{FINDNEAREST}(T, q_{rand})$
- 3: **for**  $i = 1, \dots, M$  **do**
- 4:    $v_i \leftarrow \text{SAMPLECONTROL}()$  ▷ In  $se(2)$
- 5:    $q_i \leftarrow \Gamma(q_{near}, v_i)$  ▷ State Transition
- 6: **end for**
- 7:  $(q_{new}, v^*) \leftarrow \arg \min_{(q_i, v_i)} \text{DISTANCE}(q_i, q_{rand})$
- 8:  $u^* \leftarrow \text{JACOBIANPROJECTION}(v^*)$  ▷ Sec. IV-B
- 9: **if**  $u^* \neq Null$  **then**
- 10:    $T.\text{ADDDNODE}(q_{new})$  ▷ Expansion
- 11:    $T.\text{ADDEEDGE}((q_{near}, q_{new}), u^*)$
- 12: **end if**
- 13: **return**  $T$

---



---

### Algorithm 3 EvaluateProgress( $\cdot$ )

---

**Input:** Current motion tree  $T$ , heuristic  $h(\cdot)$ , progress threshold  $p$ , tree limit  $D_{max}$   
**Output:** Control sequence  $\tau$

- 1:  $q_{new} \leftarrow T.\text{GETLATESTNODE}()$
- 2:  $\tau \leftarrow \{\}$
- 3: **if**  $q_{new} \in \mathcal{Q}_G$  **then** ▷ Goal Reached
- 4:    $\tau \leftarrow \text{EXTRACTCONTROLS}(T, q_{new})$
- 5: **else if**  $h(T.\text{GETROOT}()) - h(q_{new}) > p$  **then** ▷ Horizon
- 6:    $\tau \leftarrow \text{EXTRACTCONTROLS}(T, q_{new})$
- 7: **else if**  $T.\text{GETDEPTH}() = D_{max}$  **then** ▷ Depth Limit
- 8:    $q' \leftarrow \arg \min_{q \in T.\text{GETLEAVES}()} h(q)$
- 9:    $\tau \leftarrow \text{EXTRACTCONTROLS}(T, q')$
- 10: **end if**
- 11: **return**  $\tau$

---

## B. Jacobian-based Motion Projection

Since we sample robot controls in the end-effector's velocity space in  $se(2)$  to ensure the generated motions are constrained to the workspace, we need to project the controls to the robot's control space  $\mathcal{U}$  to enable real robot executions. As indicated by the function JACOBIANPROJECTION( $\cdot$ ) in Alg. 2, for every new state  $q_{new}$  to be added in the tree, we check whether the associated control  $v^*$  can be projected to a valid  $u^*$  to transition the state from  $q_{near}$  to  $q_{new}$ .

Note that, as Jacobian matrix can constantly change while a control is being applied over a duration  $[0, D]$ , the Jacobian-based projection needs to be conducted continuously throughout the transition from  $q_{near}$  to  $q_{new}$ , and a constant robot end-effector control  $v^*$  can be generally projected to a smooth trajectory in  $\mathcal{U}$ . In practice, we address this by sufficiently discretizing the control duration with a

small interval  $\Delta t$ , and then calculate the control  $u_i^*$  for each intermediate state  $q_i$ .

Given a state  $q_i$ , its Jacobian matrix  $J_i = \text{JACOBIAN}(q_i^R)$  is calculated based on the current robot configuration  $q_i^R \in \mathcal{Q}^R$ . The control  $u_i^*$  is then obtained by  $u_i = J_i^\dagger \cdot v^*$ . While we iterate over  $q_i$ , we can determine that a control  $u_i^*$  is invalid if: 1) the resulted robot configuration is invalid; or 2) the manipulability of the robot configuration, calculated by  $\sqrt{\det J_i J_i^T}$  [27], is smaller than a threshold, indicating that the robot is going to hit its singularity. If every intermediate projection is valid,  $\text{JACOBIANPROJECTION}(\cdot)$  will return by composing a control trajectory  $u^*$  based on all the intermediate  $u_i^*$ , and will otherwise return *Null*.

## V. EXAMPLE APPLICATIONS

To evaluate our framework, we task the robot with 3 different rearrangement-based manipulation tasks in clutter, as exemplified in Fig. 1.

1) *Grasping*: For grasping a target object in clutter, the robot needs to rearrange the surrounding objects so that the gripper can reach a stable pre-grasp pose. The major challenge of this task is that, while the surrounding objects are being rearranged, the target object is simultaneously moved by object-object interactions. The task goal is achieved when the center of the target object  $(x^o, y^o)$  is inside the area between the two fingers, denoted as  $\mathcal{G}^R$ , and the orientation of the gripper is roughly aligned with a feasible grasping angle. Formally, the goal criterion is:

$$(x^o, y^o) \in \mathcal{G}^R \wedge \min_{\alpha \in \mathcal{A}} |\theta^R - \alpha| \leq \epsilon_\alpha \quad (1)$$

where  $\theta^R$  is the orientation of the gripper,  $\mathcal{A}$  is the set of feasible grasping angles, and  $\epsilon_\alpha > 0$  is a threshold in radians for which we set to be 0.2 in all experiments.

The heuristic function used by our dhRRT planner for grasping encourages the gripper to approach the target object. Let us denote the state of the gripper as  $(x^R, y^R, \theta^R) \in SE(2)$ , we define the grasping heuristic function  $h_g$  to take the following simple form:

$$h_g(q) = w_d \cdot \sqrt{(x^R - x^o)^2 + (y^R - y^o)^2} + w_\alpha \cdot |\theta^R - \text{atan}(y^o - y^R, x^o - x^R)| \quad (2)$$

where  $w_d, w_\alpha$  are weighting factors and set to be  $w_d = 0.7, w_\alpha = 0.3$  in all experiments.

2) *Relocating*: The relocating task for the robot is to push the target object to a circular goal region  $\mathcal{G}$  centered at  $(x_G, y_G)$  with a radius of 0.1m. The goal criterion is:

$$(x^o, y^o) \in \mathcal{G} \quad (3)$$

This is a difficult task since the target object is not necessarily reachable by the gripper, and its motion is indirectly determined by all other objects. The heuristic function used by our dhRRT planner is simply defined by the summation of the distance between the target object and the gripper, and the distance between the target object and the goal region:

$$h_r(q) = \sqrt{(x^o - x^R)^2 + (y^o - y^R)^2} + \sqrt{(x^o - x_G)^2 + (y^o - y_G)^2} \quad (4)$$

3) *Sorting*: The sorting task is to rearrange all the movable objects to separate them into different classes, which are represented by different colors in our experiments. We denote by  $L$  the number of object classes, and by  $\mathbf{CH}_i(q) \subseteq \mathbb{R}^2, i \in \{1, \dots, L\}$ , the convex hull containing all the objects of the  $i$ -th class in state  $q \in \mathcal{Q}^{\text{valid}}$ , then the goal criterion of sorting is satisfied if all classes have at least a distance  $\epsilon_d > 0$  from each other. Formally,  $\forall i, j \in \{1, \dots, L\}$ :

$$\min_{i \neq j} \text{DISTANCE}(\mathbf{CH}_i(q), \mathbf{CH}_j(q)) > \epsilon_d \quad (5)$$

Our dhRRT planner uses a heuristic function similar to the reward signals in our previous work in [17] for sorting. Intuitively, a state where the objects are placed closer to each other for the same class and further apart for different classes will receive a lower cost value.

## VI. EXPERIMENTS

With the three tasks defined in Sec. V, we evaluate the proposed framework from three aspects relevant to real-world challenges. First, by increasing the size of the object clutter (number of objects), we test and report the success rate and planning efficiency of the planner. Second, we evaluate the robustness against inaccurate models with quantified model granularities. Finally, we challenge the planner by introducing nondeterministic physics to evaluate its reactivity. Our experiments were conducted both with a real Franka Emika Panda robot and in the MuJoCo simulator [28]. All objects were tracked via AprilTags [29].

In addition, we implemented two baseline algorithms to compare with the proposed dhRRT approach. First, we implemented a kdRRT algorithm modified from [13], and for a fair comparison, we replaced the physics model with the MuJoCo simulator, and we do not limit the number of concurrent contacts. Second, we enable kdRRT with replanning, termed as r-kdRRT, by observing the end states after every execution, and will trigger replanning if the goal is not reached. In all experiments, the controls were sampled in the robot gripper's velocity space. The linear velocity was bounded by  $[-0.2, 0.2]m/s$  in simulation, but by  $[-0.1, 0.1]m/s$  in the real world for better safety. The angular velocity was all bounded by  $[-1, 1]rad/s$ . The control duration was fixed to 0.2 seconds (grasping, relocation) and 0.4 seconds (sorting). In addition, all reported planning times were calculated from the successful runs only, and all the time budgets were set for planning only, excluding the execution time.

### A. Efficiency and Robustness

For this part of our evaluation, we conducted only real-world experiments as the planning efficiency will be similar to simulation-based experiments, but the system's robustness can be more realistically challenged in the real world. Example executions for the 3 tasks are shown in Fig. 3. For the grasping and relocating tasks, we used  $N = 10$  and  $N = 20$  objects, with one of them being the target object in each task. The sorting task used 6 objects and 2 classes: 3 blue objects and 3 red objects. For all experiments with our algorithm and the baseline algorithms, the time budget

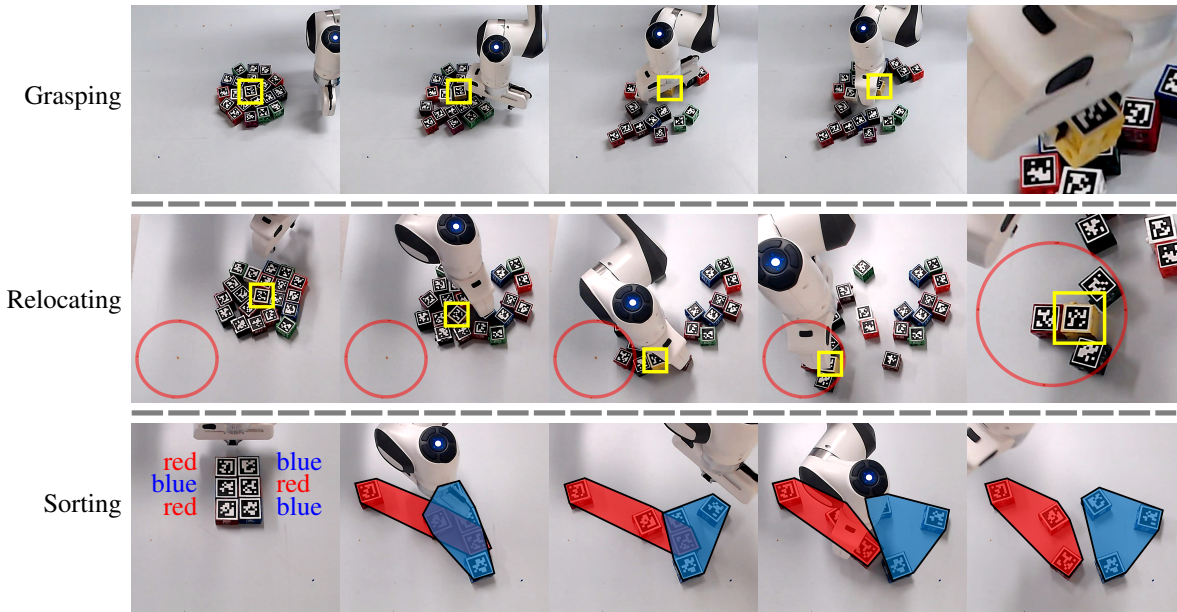


Fig. 3: Real-world experiments on 3 rearrangement-based manipulation tasks: grasping, relocating, and sorting. The target object is highlighted by yellow boxes for grasping and relocating, and the convex hulls for object classes are shown in blue and red for sorting.

(a) Grasping				
Scene	Metric	kdRRT	r-kdRRT	dhRRT
N = 10	Success Rate	1 / 10	6 / 10	<b>10 / 10</b>
	Time (seconds)	13.0 ± 0.0	22.5 ± 16.7	<b>9.8 ± 2.5</b>
N = 20	Success Rate	0 / 10	2 / 10	<b>8 / 10</b>
	Time (seconds)	-	20.1 ± 8.6	<b>11.0 ± 4.3</b>
(b) Relocating				
Scene	Metric	kdRRT	r-kdRRT	dhRRT
N = 10	Success Rate	3 / 10	4 / 10	<b>9 / 10</b>
	Time (seconds)	33.6 ± 16.6	40.1 ± 18.2	<b>15.2 ± 4.9</b>
N = 20	Success Rate	1 / 10	2 / 10	<b>9 / 10</b>
	Time (seconds)	34.9 ± 0.0	35.4 ± 0.5	<b>18.1 ± 11.1</b>
(c) Sorting				
Metric	kdRRT	r-kdRRT	dhRRT	
Success Rate	0 / 10	0 / 10	<b>4 / 10</b>	
Time (seconds)	-	-	<b>28.3 ± 16.4</b>	

Fig. 4: Experiment results of the 3 real-world tasks.

was set to 60 seconds, accumulated through the process if replanning was needed.

The experiment results are reported in Fig. 4. We can see that while both kdRRT and r-kdRRT could barely achieve 50% of success rates in grasping and relocating, our dhRRT has succeeded in more than 90% of tests, with only failure case happening due to one of the objects being pushed outside of the workspace due to physical uncertainties. It is evident that the dynamic horizon significantly improved the robustness by closing the manipulation loop, and enabled progressive manipulation while observing the real-world transitions. Moreover, we observe that dhRRT is much more efficient than the baseline algorithms. This is because: 1) dhRRT can more adaptively focus on task-relevant subspaces of the problem; and 2) it avoids making large trajectory deviations and requires less replanning effort compared with r-kdRRT.

For the sorting task, however, while none of the baseline algorithms could solve it at all, dhRRT achieved 40% success

rate. This is due to the much higher complexity of the sorting problem, which does not have a single target object, and the goal is achieved only when *all objects* are relatively reconfigured in certain ways. Under the time budget, and without training or carefully designing problem-specific heuristics, such as done in [17], [18], as well as not being able to teleport the gripper in between of actions, our dhRRT was not able to provide good sorting performance in the real physical world, although we achieved a success rate of 90% in simulation.

### B. Inaccurate Object Models

In real-world applications, due to the perception limitations, we do not always have access to perfect object models. For example, the object models can be incorrectly estimated, or the resolutions of the models are not good enough to reflect real-world contact physics. As such, we designed two test cases to evaluate the algorithms' robustness against perception uncertainties in grasping tasks by: 1) using incorrect object models in planning; and 2) iteratively reducing the resolution of object models in planning. For case 1), we conducted real-world experiments using 10 objects as exemplified in Fig. 5. We tested case 2) in simulation to be able to access perfect and resolution-reduced object models. The time budget was set to 60 seconds and 120 seconds for case 1) and case 2) respectively.

The experiment results of case 1) are summarized in Fig. 6. Due to the discrepancies between object models in planning and in the real world, kdRRT was never able to complete the task. Under this difficult setting, our dhRRT was able to succeed 9 out of 10 times, and the planning efficiency was almost not affected in comparison to the cases where accurate models were available. In addition, even if r-kdRRT was able to complete 3 out of 10 trials, the planning efficiency was,

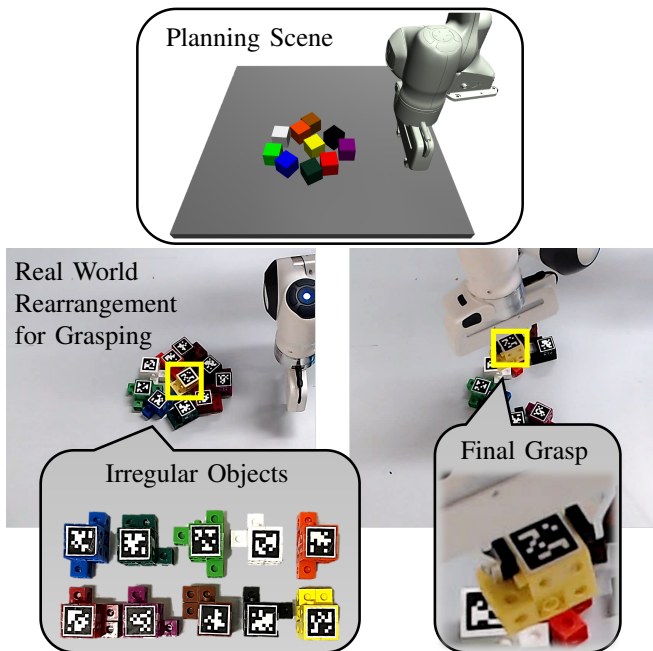


Fig. 5: The robot is tasked to grasp the yellow object in a clutter. Despite the fact that our planner incorrectly models perfect cube shapes for all objects, it can successfully rearrange the irregular object shapes in the real world and complete the task via dynamic planning horizons.

Metric	kdRRT	r-kdRRT	dhRRT
Success Rate	0 / 10	3 / 10	<b>9 / 10</b>
Time (seconds)	-	42.3 ± 9.9	<b>8.6 ± 3.4</b>

Fig. 6: Experiment results of real-world grasping tasks with inaccurate object models.

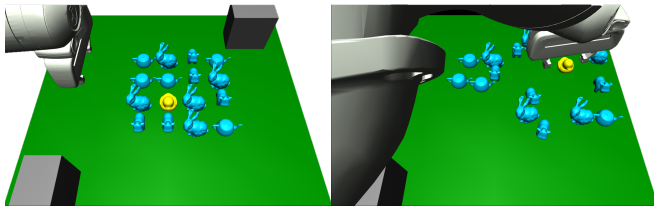


Fig. 7: Experiment setup for grasping with resolution-reduced object models. The target object is shown in yellow and static obstacles are shown in gray. *Left*: start configuration. *Right*: a goal state reached by our dhRRT.

again, evidently lower than our dhRRT, indicating that the proposed dynamic horizons provided essential reactivity to the system for both improved efficiency and robustness.

The experiment setup for case 2) is illustrated in Fig. 7. We selected 4 complex object shapes, shown in Fig. 8, and iteratively reduced their resolutions to 33%, 10%, 3.3%, and 1% to be used by planners. Note that, once a plan was generated for execution, the simulated execution used only the original object models. We can observe that, as reported in Fig. 9, although the model reduction significantly affected the baseline algorithms, dhRRT still could complete the task with high success rates. Also, the planning time for dhRRT was much lower than the baseline approaches. Notice that, the planning time for the baseline algorithms was occasionally faster than dhRRT due to their small number of successes and

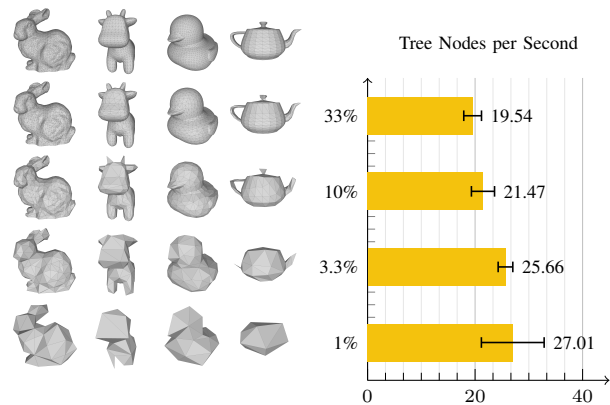


Fig. 8: Object models used in the experiment shown in Fig. 7. *Left*: The resolution-reduction level, determined by face reduction percentage, were 100% (original), 33%, 10%, 3.3%, and 1% from top to bottom. *Right*: Number of nodes added in the tree per second in terms of the reduction rates.

Red. Rate.	Success Rate			Time (seconds)		
	kdRRT	r-kdRRT	dhRRT	kdRRT	r-kdRRT	dhRRT
33 %	5/20	16/20	<b>20/20</b>	<b>11.9±4.8</b>	48.8±32.0	13.3±13.7
10 %	3/20	12/20	<b>17/20</b>	28.7±12.5	34.2±28.5	<b>17.6±10.4</b>
3.3 %	2/20	11/20	<b>15/20</b>	<b>2.2±0.7</b>	38.9±30.2	16.1±9.4
1 %	1/20	12/20	<b>17/20</b>	24.4±0.0	32.1±19.0	<b>22.6±18.1</b>

Fig. 9: Results of grasping tasks with resolution-reduced models.

$\Delta t(s)$	Success Rate			Time (seconds)		
	kdRRT	r-kdRRT	dhRRT	kdRRT	r-kdRRT	dhRRT
20	17/20	17/20	<b>20/20</b>	27.2±19.7	27.2±19.7	<b>20.1±17.7</b>
10	17/20	19/20	<b>20/20</b>	22.9±15.3	25.6±16.8	<b>15.1±11.0</b>
5	7/20	16/20	<b>19/20</b>	26.5±18.9	40.7±23.5	<b>19.5±22.8</b>
2	3/20	12/20	<b>17/20</b>	24.9±11.3	45.6±33.5	<b>14.9±11.1</b>
1	3/20	6/20	<b>15/20</b>	13.2±7.0	15.8±7.4	<b>11.6±8.0</b>
0.5	0/20	9/20	<b>17/20</b>	-	35.9±23.7	<b>13.3±7.4</b>

Fig. 10: Results of grasping tasks under nondeterministic physics.

randomness in statistics. Furthermore, Fig. 8 reports the tree expansion efficiency in terms of the number of added nodes per second. It is interesting to note that, although imperfect models enlarge the gaps between planning and the real world, simpler models can facilitate the planning efficiency. Therefore, in terms of the task requirements, in practice we can balance between the needed robustness and the modeling resolution of models to achieve higher efficiency.

### C. Nondeterministic Physics

Another factor that challenges manipulation planning in the real world is the nondeterministic physics. For example, the object's friction coefficient against the table surface is not a constant resulting in different dynamics over the execution, or that the objects can be moved by external perturbations. Therefore, we designed an experiment to introduce random local perturbations to the objects during the execution of manipulation plans. In this experiment, we used 16 cubes similar to the experiments shown in Fig. 1, and in every time interval  $\Delta t$ , we randomly select one object and assign it with a linear velocity of  $0.4m/s$ , and the goal is for the robot to grasp the target object. The time budget was set to 120 seconds.

The results are reported in Fig. 10. In this experiment, a shorter interval for applying random perturbations simulates a higher level of nondeterministic physics. Being consistent with other experiments, dhRRT outperformed both baseline planners in efficiency and robustness, which again shows that the proposed dhRRT planner can focus the planning procedure to more task-relevant subspaces, and facilitates a close-loop manipulation solution against physical uncertainties in executing motions plans.

## VII. CONCLUSION

We presented a kinodynamic manipulation planning framework for rearrangement-based manipulation problems. Based on efficient sampling-based planning, we proposed to monitor and dynamically adapt the kinodynamic planning horizons, and progressively transition the system states towards the goal region by interleaving planning and execution, which greatly enhanced the system’s robustness. Using simple heuristics, we showed that our approach is not only able to focus the planning on task-relevant subspaces to significantly improve the planning efficiency, but also enables implicit definitions of manipulation goals, in contrast to many traditional goals defined by explicit configurations.

With extensive experiments both in the real world and in simulation, we demonstrated the proposed approach with 3 challenging rearrangement-based manipulation tasks, and compared its performance against 2 baseline algorithms. In terms of efficiency and robustness, we showed that our approach is significantly faster and is able to complete tasks under perception uncertainties, local modeling errors, and nondeterministic physics in the real world.

In future work, we plan to incorporate motions that alternate the end-effector’s poses across subsequent action segments, enabling the robot to quickly switch between different problem subspaces without being constrained to move continuously in  $SE(2)$ , so as to improve the efficiency of rearrangement actions. In addition, we plan to study the adaptation of the planning horizon in terms of other task-relevant factors, e.g., the number and distribution of concurrent contacts, in order to more tightly close the manipulation loop. Besides, we will also investigate the optimization of the local trajectories to reduce the execution time.

## REFERENCES

- [1] A. Billard and D. Kragic, “Trends and challenges in robot manipulation,” *Science*, vol. 364, no. 6446, p. eaat8414, 2019.
- [2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis – a survey,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.
- [3] K. Hang, W. G. Birchler, A. S. Morgan, and A. M. Dollar, “Manipulation for self-identification, and self-identification for better manipulation,” *Science Robotics*, vol. 6, no. 54, 2021.
- [4] K. M. Lynch and M. T. Mason, “Dynamic nonprehensile manipulation: Controllability, planning, and experiments,” *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 64–92, 1999.
- [5] F. Ruggiero, V. Lippiello, and B. Siciliano, “Nonprehensile dynamic manipulation: A survey,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.
- [6] J. Z. Woodruff and K. M. Lynch, “Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks,” in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4066–4073.
- [7] G. Wilfong, “Motion planning in the presence of movable obstacles,” *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, pp. 131–150, 1991.
- [8] J. Bütepage, S. Cruciani, M. Kokic, M. Welle, and D. Kragic, “From visual understanding to complex object manipulation,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 161–179, 2019.
- [9] K. E. Bekris and L. E. Kavraki, “Greedy but safe replanning under kinodynamic constraints,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 704–710.
- [10] S. M. LaValle and J. J. Kuffner Jr, “Randomized kinodynamic planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [11] B. Lau, C. Sprunk, and W. Burgard, “Kinodynamic motion planning for mobile robots using splines,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 2427–2433.
- [12] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles,” *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.
- [13] J. E. King, J. A. Hausteine, S. S. Srinivasa, and T. Asfour, “Nonprehensile whole arm rearrangement planning on physics manifolds,” in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2508–2515.
- [14] J. A. Hausteine, J. King, S. S. Srinivasa, and T. Asfour, “Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states,” in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3075–3082.
- [15] M. Moll, L. Kavraki, J. Rosell *et al.*, “Randomized physics-based motion planning for grasping in cluttered and uncertain environments,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 712–719, 2017.
- [16] W. C. Agboh and M. R. Dogar, “Real-time online re-planning for grasping under clutter and uncertainty,” in *IEEE International Conference on Humanoid Robots (HUMANOIDS)*. IEEE, 2018, pp. 1–8.
- [17] H. Song, J. A. Hausteine, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, “Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9433–9440.
- [18] E. Huang, Z. Jia, and M. T. Mason, “Large-scale multi-object rearrangement,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 211–218.
- [19] W. Yuan, K. Hang, D. Kragic, M. Y. Wang, and J. A. Stork, “End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer,” *Robotics and Autonomous Systems*, vol. 119, pp. 119–134, 2019.
- [20] A. Eitel, N. Hauff, and W. Burgard, “Learning to singulate objects using a push proposal network,” in *Robotics research*. Springer, 2020, pp. 405–419.
- [21] T. Hermans, J. M. Rehg, and A. Bobick, “Guided pushing for object singulation,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4783–4790.
- [22] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, “Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations,” in *IEEE international conference on automation science and engineering*. IEEE, 2016, pp. 827–834.
- [23] L. P. Kaelbling, “The foundation of efficient robot learning,” *Science*, vol. 369, no. 6506, pp. 915–916, 2020.
- [24] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, “Manipulation planning on constraint manifolds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 625–632.
- [25] Z. Kingston, M. Moll, and L. E. Kavraki, “Exploring implicit spaces for constrained sampling-based planning,” *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.
- [26] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 2002.
- [27] T. Yoshikawa, “Manipulability of robotic mechanisms,” *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.
- [28] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 5026–5033.
- [29] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.