Task and Motion Planning for Mobile Manipulation

Jingren Xu

October 2022

Task and Motion Planning for Mobile Manipulation

A dissertation submitted to

THE GRADUATE SCHOOL OF ENGINEERING SCIENCE

OSAKA UNIVERSITY

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN ENGINEERING

BY

Jingren Xu

October 2022

Task and Motion Planning for Mobile Manipulation

Jingren Xu

Osaka University 2022

Abstract

Human workers are still highly engaged in assembly factories. In order to assemble a product, human workers have to pick up different assembly parts from a large storage area and transport them to the assembly line for assembly. Mobile manipulators are able to operate in a large workspace and have the potential to replace human workers in assembly factories. Especially, the current mobile manipulators are well-suited for part-supply tasks in a structured environment.

In this thesis, I consider the possible applications of mobile manipulators to the automation in an assembly factory. I assume the application scenario is that the mobile manipulators pick up the assembly parts from the storage area and transport them to the assembly line, and then these assembly parts are assembled by (mobile) manipulators. The involved robotic operations are (1) moving the mobile manipulator to the positions where the assembly parts are reachable for picking, (2) moving the manipulator to a picking configuration, and (3) grasping and assembling the assembly parts of various shapes and sizes. I approach this problem from a three-layer hierarchy, i.e., task-level planning, motion-level planning, and grasp-level planning, to improve the efficiency of performing the part-supply, grasping, and assembly tasks.

At the task level, I present a planner to plan a minimal sequence of robust positions to pick up assembly parts from different trays. The positioning uncertainty of the mobile base is considered to improve the robustness of the pick-and-place

tasks. Considering the practical implementation, I also discussed object placement styles and the update of the planned base positions, which further increases the robustness. At the motion level, I present an optimization-based motion planner for dynamic grasping, i.e., picking up the object during the whole-body motion of the manipulator and the mobile base. The simultaneous motion of the manipulator and the base further reduces the operation time and improves the overall efficiency. At the grasp layer, I consider the grasping and assembly of the assembly parts product. The robotic grippers have to robustly grasp assembly parts of various shapes and sizes. In addition, once the assembly parts are efficiently picked by the mobile manipulators, these assembly products are assembled by robotic manipulators. For high-mix low-volume assembly tasks, the assembly products frequently change. As a result, the grippers for the grasping and assembly tasks are required to change as well. I propose a method for efficient gripper selection and design to meet the demand for High-Mix Low-Volume assembly production.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

There is an increasing demand for robots that are able to flexibly perform tasks in the human environment [1] (such as opening doors [2] and fetching a cup of coffee [3]) and industrial production (e.g., inspection and sealant tasks in aerospace industry [4], robotic painting [5], robotic machining [6], and part pickup and transport operations in warehouses [7]). A mobile manipulator, combining a mobile base and a manipulator, is able to perform a variety of tasks in separate locations. Especially, the current mobile manipulators are well-suited for part-supply tasks in a structured environment (e.g., an assembly factory), where the robot picks up and transports objects to the desired location. However, these tasks are still heavily occupied by human workers. In this thesis, I expect to use mobile manipulators to replace human workers to perform the part-supply tasks in an assembly factory, where different categories of assembly parts are scattered in a large storage area (Fig. 1.1[1]). To perform the part-supply and assembly tasks, the mobile manipulators have to move to a sequence of positions to pick up the target assembly parts from multiple trays and then transport them to the assembly area for the final assembly. The involved operations are (1) moving the mobile manipulator to the positions where the assembly parts are reachable for picking, (2) moving the manipulator to a picking configuration, and (3) grasping and assembling the assembly parts.

To efficiently apply mobile manipulators to part-supply and assembly tasks, planners and methods are required to automatically determine the base positions,

[1]Image source: `https://www.bott-canada.com/images/betriebseinrichtung/avero/av_ref_framo_03_mit_person.jpg`, `https://myworkingspace.com/wp-content/uploads/2015/03/Production-Line3.jpg`

Figure 1.1: Assembly factories.

optimize the robot motion and design grippers for the grasping and assembly tasks. This is especially important in case of High-Mix Low-Volume (HMLV) manufacturing, which has increased significantly in the last couple of decades [8]. In HMLV manufacturing, the assembly line has to adapt to different assembly products rapidly. This poses many challenges for using mobile manipulators in an HMLV assembly factory. Firstly, assembly parts are placed in a large storage area, the mobile manipulator has to pick up assembly parts from trays in different locations. HMLV manufacturing frequently changes the assembly product, therefore the mobile manipulator has to visit different positions for different assembly products. Moreover, the mobile manipulator has to grasp and assemble parts of various shapes and sizes. In order to robustly grasp the assembly parts, the mobile manipulator has to use grippers with appropriate configuration and dimensions. Since the production cycle is short in HMLV manufacturing, it poses restrictions on time spent on designing the grippers for picking and assembly. I tackle the assembly factory automation problem from a three-layer hierarchy, as shown in Fig. 1.2. At the task level, I plan an efficient and robust sequence of base positions to pick up all the required assembly parts. At the motion level, I further improve the efficiency by optimizing the whole-body motion of the mobile manipulator to dynamic-grasp the object. At the grasp level, I consider the efficient design of

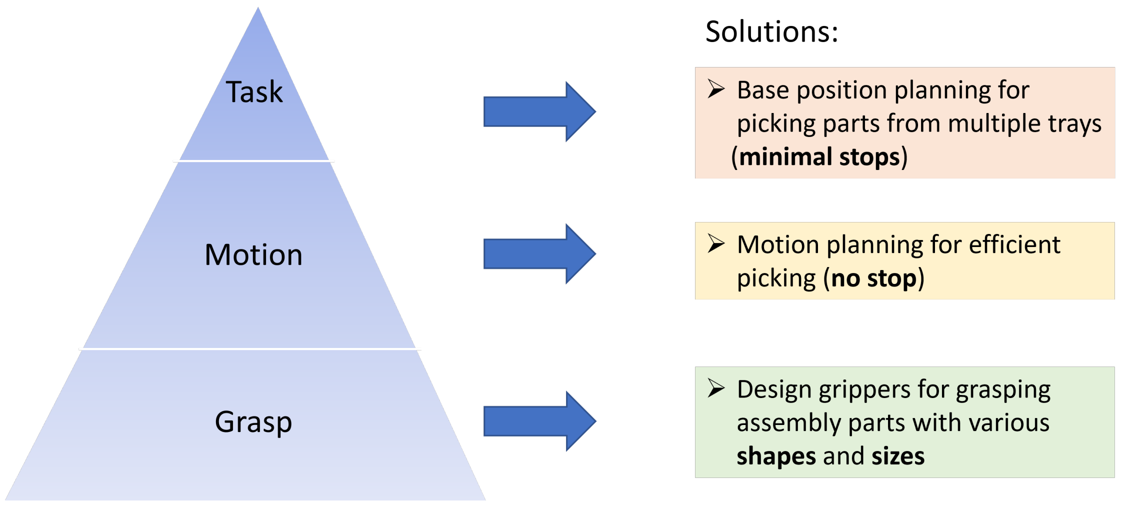

Task
Motion
Grasp
Solutions:
Base position planning for picking parts from multiple trays (minimal stops)
Motion planning for efficient picking (no stop)
Design grippers for grasping assembly parts with various shapes and sizes

Figure 1.2: Three-layer hierarchy approaches for automation in an assembly factory.

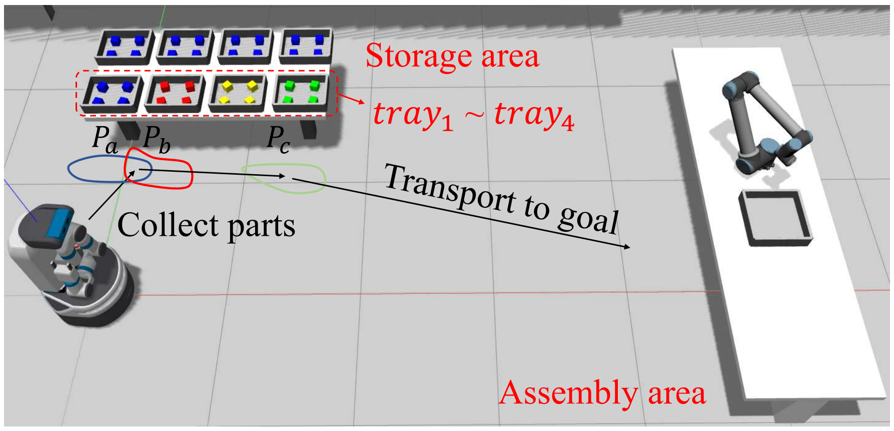


Figure 1.3: A schematic overview of the task. In an assembly factory, a mobile manipulator is used to pick up objects from multiple places and then transport them to the assembly line for further assembly.

grippers for grasping the assembly parts of various shapes and sizes, this meets the demand for high-mix low-volume production.

1.1 Task-level Planning

The task-level planning considers planning the minimal stops that enable the mobile manipulator to perform a sequence of pick-and-place tasks. Fig. 1.3 shows the schematic overview of the pick-and-place task to be performed by the mobile manipulator. The product P to be assembled is comprised of several types of parts: P_a, P_b and P_c, and these parts are categorized by their types and stored in different trays $tray_1$, $tray_2$ and $tray_4$, respectively. To supply the parts for the assembly tasks, the mobile manipulator may have to move to and stop at a sequence of positions to gradually pick up the required assembly parts from different trays. The picked parts are temporarily placed on the mobile base and carried with the mobile manipulator to the goal position. I assume that in each round of the pick-and-place task, the mobile manipulator picks at least one piece of part for every target type of assembly part. Our goal is to plan a minimal sequence of base positions, in which the mobile manipulator is able to grasp the required assembly parts from the trays without self-collision and collision with the environment. Notice that I am performing task-level planning considering kinematic feasibility, (i.e., planning the base positions) instead of low-level motion planning.

With the increased number of base positions (movements), the overall operation time increases significantly for the following reasons: (1) The mobile manipulator decelerates and accelerates before and after arriving at every base position of the base sequence, which lowers the overall base velocity and increases the total operation time. (2) Every time the mobile manipulator experiences a "stop and pick", there is a risk that the arrived position significantly deviates from the desired position. Then the mobile manipulator has to perform time-consuming repositioning. The risk increases with expanding base sequence, therefore, it is crucial to prune out unnecessary base movements to improve the overall efficiency.

To minimize the base sequence size for a sequence of pick-and-place tasks, it is preferable to move the mobile manipulator to the positions, where the mobile manipulator can pick up the assembly parts from multiple trays. As shown in Fig. 1.3, the mobile manipulator moves to the first position and can grasp the assembly parts in both $tray_1$ and $tray_2$, thus reducing the base sequence size by one. To obtain such positions, I first calculate the base region for every target tray in the given assembly task by using inverse kinematics. The base region is a set of base positions where there are collision-free inverse kinematics solutions for grasping the **target** assembly parts from that tray. Moreover, the base positioning uncertainty is taken into account by restricting the size of the applicable intersections, i.e., the intersections smaller than the base positioning uncertainty are discarded. Otherwise, the mobile manipulator is likely to move out of the intersections. Then, I solve the minimum number of intersections that visit all the base regions, by formulating it as a 0-1 knapsack problem. The centers of the planned intersections are robust base positions for performing the pick-and-place tasks. Finally, I search for the optimal sequence of visiting the base positions, which results in the shortest path connecting the start and goal positions via the base positions. Following the planned base sequence, the mobile manipulator can perform efficient and robust part-supply tasks in real-world applications.

After every round of pick-and-place tasks, the mobile manipulator returns to the start position and gets ready for the next round. As a result, the assembly parts in the trays are gradually picked away. The overall efficiency and robustness can be further improved by updating the base sequence according to the current status of parts. However, the situation is further complicated by part placement styles in real-world applications, where the parts are either randomly or regularly placed in the tray. I discussed and analyzed the feasible policies for different task specifications and parts placements.

Existing research usually optimizes the manipulator configuration or base position for each task, with respect to criteria such as manipulability and reachability [9, 10]. However, very few works consider a sequence of tasks. The most relevant ones I could find are [11, 12, 13, 14]. In [11, 12, 13], they assumed there is a mobile manipulator configuration corresponding to each task, and they optimized the commutation configurations for adjacent tasks or the distance between consecutive configurations, but they did not consider reducing the number of configurations or base positions. [14] considered optimizing the number of platform movements for reaching a set of poses in the workspace. Different from their work, our task is defined as grasping the objects in separate trays, and reaching a set of grasps is only one of the possible cases. Moreover, real-world issues like the object placement and update of the base sequence are discussed in our task, but they are not addressed in the existing works.

1.2 Motion-level Planning

The motion-level consider optimizing the non-stop mobile manipulator motion to further reduce the operation time. In the task-level planning, I assume the locomotion and manipulation are performed separately. In this case, the potential capabilities of mobile manipulators are not fully utilized. Most of the existing works also employ the decoupled use of the mobility and manipulation modes [15, 16], where the mobile manipulator firstly stops the mobile base before performing the manipulation tasks, and then moves to the next position or configuration. For example, in our previous work [15], I used the mobile manipulator in a decoupled manner to pick up objects from multiple trays. However, the decoupled motion of the mobile base and the manipulator is not efficient, the operation time can

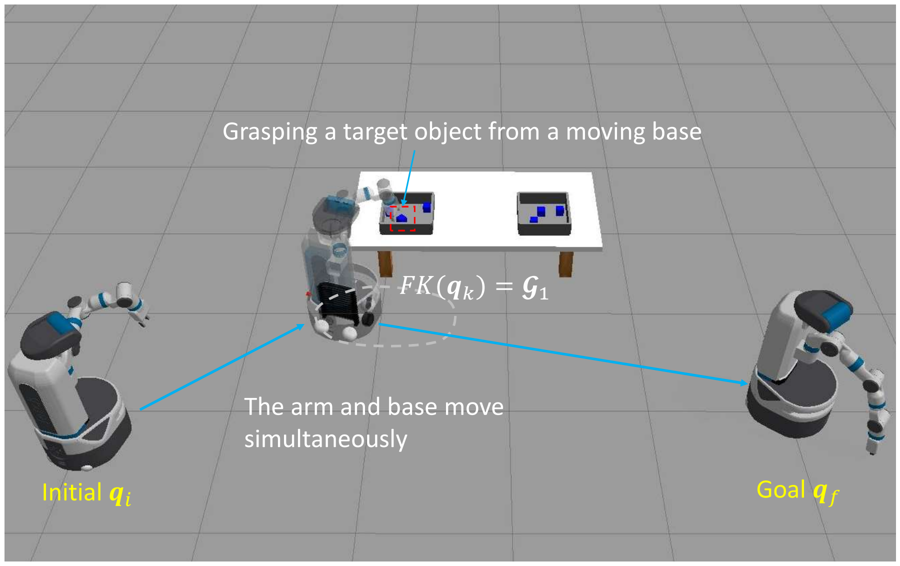

Grasping a target object from a moving base
$FK(q_k) = \mathcal{G}_1$
The arm and base move simultaneously
Initial q_i
Goal q_f

Figure 1.4: Overview of the motion planning problem with tasks during the motion. The mobile manipulator has to move from an initial configuration to a goal configuration, and it has to grasp some target objects during the simultaneous motion of the manipulator and the mobile base.

be further reduced by simultaneously moving the base and the manipulator when performing the tasks. There are a few attempts to pick up objects from a moving base [17, 18], but they firstly plan the path for the base and then plan the motion of the manipulator, i.e., the planning of the base and manipulator motion is separately performed. Moreover, [18] predefined zero gripper velocity with respect to the object, i.e., grasping the object with a stationary gripper from a moving base. Decoupled planning of base and manipulator motion and decoupled planning of geometric path and velocity profile may lead to the sub-optimality of the resultant trajectory. Moreover, the resultant trajectory may be infeasible since the constraints from another planning phase are not considered [19].

Therefore, I adopt the optimization-based method to plan the path and velocity profile simultaneously to obtain the time-optimal simultaneous motion of the mobile base and the manipulator. One of the difficulties in planning the optimal trajectory with tasks during the motion is the representation of the robot configuration at the moment of performing the task, since both the time and configuration are unknown. If I employ the commonly used trajectory optimization formulations (with fixed a time interval between waypoints), I have to set the timing for performing the task as an optimization variable. To obtain the robot configuration for performing the task, I have to first determine the interval (defined by two consecutive waypoints) within which the task is performed, and then interpolate these two ends of the interval to get the robot configuration. However, the interpolated configuration may jump among different intervals during the optimization, thus introducing nonsmoothness and discontinuity, and as a result, the NLP solver may not find a feasible solution. To address this issue, in the discretized representation of the trajectory, I propose to specify the waypoints for performing the task and set the time interval between the consecutive waypoints as optimization variables to continuously scale the trajectory. Therefore, the smoothness of this task constraint is guaranteed. The output of our planner is a locally time-optimal trajectory of the mobile manipulator, consisting of both the path and velocity profile.

There are mainly two research gaps I am addressing: (1) Firstly, there are very limited works on coupled motion planning with performing tasks (grasping) during the motion; (2) Secondly, very few works consider planning the simultaneous motion of the mobile base and the manipulator. The contribution of our work is that I proposed an optimization-based method to plan the optimal trajectory for a mobile manipulator to perform tasks during the motion.

Most of the existing research on (optimal) motion planning plans the motion between two points without performing tasks during the motion. If the entire motion is split into two parts and planned using the existing optimal motion planners, i.e., plan the motion from an initial configuration to the intermediate task configuration and the motion from the intermediate task configuration to the final configuration, then the resultant trajectory is not optimal. Moreover, mobile manipulators are usually redundant, there are infinite configurations satisfying the task constraint. In our formulation, I let the motion planner explore the robot configuration for the task instead of predefining one. These issues are addressed in our optimization-based whole-body motion planning.

1.3 Grasp-level Planning

The task-level and motion-level planning mainly address the high-level and low-level motion of the mobile manipulator. At the grasp level, the mobile manipulator has to grasp and assemble a set of assembly parts using robotic grippers. The gripper plays a pivotal role for the robot interacting with the object, the performance of the gripper grasping an assembly component is strongly influenced by how well the chosen gripper and its characteristics coincide with the characteristics needed for grasping a specific part [20]. Therefore, designing reliable grippers is one of the key issues for applying robots in grasping and assembly tasks.

However, robotic grippers are manually designed in most cases, the manual design process is time-consuming and requires a lot of experience and expertise, which makes it extremely challenging to design grippers, especially for an assembly task. In a general robotic assembly task, a set of specialized grippers are required to firmly grasp all the assembly components with different shapes and properties, in addition, the grippers have to satisfy the assembly constraints, such as avoiding collision with other subassemblies. Moreover, there is a trend in High-Mix Low-Volume production, which refers to producing a large variety of products in small

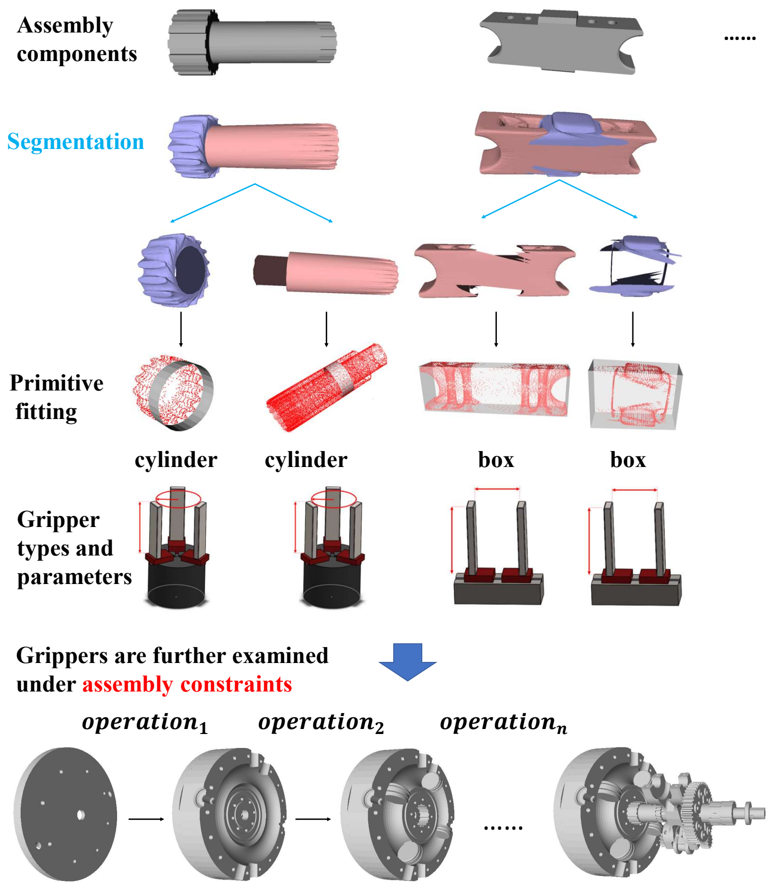


Figure 1.5: Overview of the proposed approach of selecting and designing grippers for an assembly task. In the first stage, suitable gripper types (2-finger or 3-finger gripper) and parameters (opening width) can be determined by mesh segmentation and primitive fitting. Then the segments and grippers of such configurations are further evaluated under the assembly constraints, such as affordance and collision avoidance. Finally, I optimize the number of grippers to cut down the total cost.

quantities, the fast-changing manufacturing routines propose great challenges for applying robots in such agile manufacturing. Therefore, in terms of the grippers used in the assembly tasks, a more efficient approach to designing grippers is highly demanded in order to quickly adapt to the frequently changing assembly tasks.

To efficiently design grippers satisfying the assembly constraints, I propose a structured approach of selecting and designing the grippers based on the shape analysis and assembly constraints, the overview of the method is illustrated in Fig. 1.5.

The insight is that industrial products are usually comprised of many regular shape primitives, such as cylinder and cuboid, each of the shape primitives can be firmly grasped by a suitable type of gripper. Therefore, I pre-define the rules for selecting suitable gripper types, which reduces the space for searching for possible gripper configurations and significantly accelerates the design process. Through mesh segmentation techniques, I can uncover the underlying shape primitives and assign predefined gripper types to them. The gripper parameters, such as the maximum and minimum opening widths, can be further extracted from the dimensions of the fitted primitives. These steps are automatically processed and provide reduced gripper configurations for further selection and evaluation. These gripper configurations work well in terms of grasping, however, robotic assembly is a much more complex task, where the grippers have to not only firmly grasp the assembly components but also avoid collision with the subassemblies. Furthermore, some segments are not suitable for grasping considering their affordance, and they are excluded from the selection of graspable segments. After the evaluation under assembly constraints, some of the remaining segments can be commonly grasped by the same gripper, therefore, the number of grippers can be optimized to reduce the total cost.

A few pieces of research were performed on the design of grippers for an assembly task. However, these researches are limited to designing the local shape of the fingertip [21] and general suggestions for designing the gripper systems [22]. There has been no attempt at the structured approach of selecting and designing grippers according to the assembly constraints, as well as minimizing the number of grippers, for an assembly task.

1.4 Contributions of This Thesis

In this thesis, I approach the assembly factory automation from a three-layer hierarchy, the contributions of this thesis are three-fold: (1) task-level planning, (2) motion-level planning, and (3) grasp-level planning. The core contributions are summarized as follows:

- A task-level planner that plans a minimal sequence of robust positions to pick up assembly parts from different trays. I consider both regularly and randomly placed objects and propose a method to estimate the base region for randomly placed objects. I discuss the possible policies for dynamically updating the base sequence, which further increases the robustness.
- A resolution complete method to approximate collision-free IK solutions. It returns a set of diverse IK solutions based on a precomputed reachability database, which is especially helpful for checking collisions in complex environments.
- An optimization-based motion planner for picking up the object during the whole-body motion of the manipulator and the mobile base. A push-grasp strategy is proposed to improve the robustness of dynamic grasping with respect to object pose uncertainty.
- A structured method for the efficient selection and design of gripper for grasping and assembly. The gripper types and parameters are automatically determined by mesh segmentation and primitive fitting. The assembly constraints are explicitly taken into account in the evaluation of the feasible gripper configurations. The number of grippers required for the assembly task is optimized to reduce the cost.

CHAPTER 2

RELATED WORK

In this chapter, according to the three-layer hierarchy toward assembly factory automation, I review the related work from three topics, i.e., task planning for mobile manipulation, motion planning for mobile manipulation, and gripper design for grasping and assembly.

2.1 Task Planning for Mobile Manipulation

In order to enable the robot to perform useful tasks in the real environment, the robot has to change the state of the world, by picking, moving, placing, and pushing the objects in the world. Task planning is to plan a sequence of states and transitions to change the initial state to a goal state [23]. In this thesis, the task planning plans a sequence of positions for a sequence of picking tasks.

In the task-level planning for mobile manipulators, the motion constraints from the picking tasks are considered, i.e., I plan the kinematically reachable base positions for picking parts from multiple trays. I do not plan the low-level motion of the mobile manipulator between these positions, instead, I assume that there exist low-level motion planners and controllers for moving the base between the planned positions and moving the manipulator to the grasping pose. This relates to the field of task and motion planning (TAMP), where task planning and motion planning are coupled. In this work, the connection to motion planning during the task planning is that I considered the kinematic feasibility/reachability, i.e., there should be collision-free inverse kinematics solutions in a feasible base position. Therefore, the task-level planning and motion planning are very weakly coupled in this work. The related research is two-fold: (1) Positioning the mobile manipulator for performing the tasks. (2) Mobile manipulators performing a sequence of tasks.

2.1.1 Positioning the Mobile Manipulator for Manipulation Tasks

Mobile manipulators are usually redundant due to the added mobility from the base. Positioning the mobile manipulator can be regarded as a line of research in the redundancy resolution of mobile manipulators [24, 25, 26]. There has been extensive research on exploring high-quality base positions for mobile manipulators to perform a variety of tasks, such as reaching/grasping a set of targets and maintaining velocity [27, 28, 29]. Manipulability [9, 10] is a widely used metric for evaluating the flexibility of manipulator configurations in a base position.

Yamamoto et al. [24] proposed a planning and control method to position the mobile manipulator at the preferred region to achieve high manipulability. Du et al. [30] used the manipulability index to determine a suitable base placement. Ren et al. [29] optimized the base positions for a mobile manipulator to reach a set of positions with required orientations and keep a stable velocity in local painting tasks. Berenson et al. [31] obtained the base placement and grasp for a mobile manipulator to move an object from one configuration to another, by optimizing a scoring function that combines the grasp quality, manipulability, and distance to the obstacle. OpenRAVE [32] provides an inverse reachability module, which clusters the reachability space for a base-placement sampling distribution that can be used to find out where the robot should stand in order to perform a manipulation task. Stulp et al. [33] proposed Action-Related Place to associate a base location with a probability of successfully performing a manipulation task, a capability map was used to determine if an object was theoretically reachable. Burget et al. [27] employed the inverse reachability map to select statically stable, collision-free stance configurations for a humanoid robot to reach a given grasping target. Zacharias et al. [34] took advantage of the reachability map to position a mobile manipulator to perform a linear trajectory in the workspace. Vahrenkamp et al. [35, 36] conducted a series of research on reachability analysis and its application, the base positions with a high probability of reaching a target pose can be efficiently found from the inverse reachability distribution. The reachability indicates the probability of finding an IK solution, while there is no guarantee of the completeness of obtained base positions. Some other works that used reachability and capability analysis are referred to [37, 38, 39].

In addition to the base placement for mobile manipulators, there is a line of research on the optimal placement for fixed-base manipulators [40, 41, 42, 43]. Feddema et al. [40] resolved the optimal position for a fixed-base manipulator to reach a set of points in the workspace, where no obstacle is assumed. Hsu et al. [41] considered the obstacles in the workspace, and a randomized path planner and a fast path optimization routine were combined to iteratively search for the best base location. Regardless of where the manipulator is mounted, a mobile base, or a fixed base, the optimization of base position shares many common criteria, such as manipulability and time-optimality of the trajectory.

However, these works only optimize the position or manipulator configuration for each task, they do not explore the base positions where the mobile manipulator can perform multiple tasks, to reduce the total number of base movements. Besides, in our task, the mobile manipulator is required to close the gripper to grasp the assembly parts, instead of following an end effector path or exerting force on the environment, therefore, optimal criteria such as the manipulability of the grasping configuration are not a critical issue, thus it is not considered in planning the base positions.

2.1.2 Mobile Manipulator Performing a Sequence of Tasks

To the best of our knowledge, [12] is one of the few works that considered the mobile manipulator configurations for a sequence of tasks. They planned the optimal commutation configurations for a sequence of tasks under constraints. The commutation configuration should be feasible for both current and next tasks. They discussed optimization criteria, such as optimum manipulability, the least torque norm, and minimization of the maximum actuator torque. But they did not address how to find the common base positions or configurations to reduce the number of base movements for a sequence of tasks, which is the major concern of this paper. Carriker et al. [13] considered optimizing the manipulator configurations for a sequence of tasks defined by desired positions, orientations, forces, and moments. The coordination of mobility and manipulation was formulated as a nonlinear optimization problem. A general cost function for point-to-point motion in Cartesian space was defined and minimized by simulated annealing. However, they implicitly assumed that there is a base position and a manipulator configuration corresponding to each task, and they did not optimize the number of manipulator configurations for the tasks. Vafadar et al. [14] studied the minimum platform movements to reach a set of poses in the workspace, which is a special case of our task definition. Moreover, I also discussed different placement styles of target objects and the update of the base sequence according to the remaining objects or the target objects to be picked, which are not covered in existing research.

From the above literature review, I find that (1) most of the existing works optimize the position or configuration for a single task, but they do not consider a sequence of tasks, and (2) most of the existing works do not optimize the number of base positions for a sequence of tasks, and (3) none of the existing works considers updating the base positions and different object placement styles. Although there has been extensive research on mobile manipulation, the problem of planning a minimum sequence of base positions for manipulation tasks in multiple locations has not been addressed.

2.2 Motion Planning for Mobile Manipulation

Motion planning is to find a collision-free trajectory for a robot to move from an initial configuration to the goal configuration while satisfying all the constraints. The two most widely used approaches are sampling-based planning and optimization-based planning. Current motion planners are able to plan the (optimal) path between two configurations in the C-space. However, they did not consider tasks during the motion. On the other hand, for planning the motion for mobile manipulation, there have been very few attempts to plan the simultaneous motion of the manipulator and the base. In this section, I review the literature from the perspective of motion planning and mobile manipulation. Before reviewing the motion planning literature, I review the fundamentals of the modeling and control of mobile manipulators.

2.2.1 Modeling and Control of Mobile Manipulators

A mobile manipulator consists of two subsystems, a manipulator and a mobile base. A comprehensive book covering the mathematics and modeling of mobile robots is referred to [44]. Chapter 13 of [45] introduces the modeling and control of several types of wheeled mobile robots, including omnidirectional wheeled robots, differential-drive robots, and car-like robots. And the challenges of controlling a mobile manipulator are three-fold: (1) the nonholonomic base is restricted on the possible velocities; (2) The manipulator and base have different dynamic characteristics [24], namely, the base usually has a slower dynamic response. (3) There is a dynamic interaction between the two subsystems.

Yamamoto et al. [24] studied the coordinated locomotion and manipulation of the mobile platform, where the mobile platform is controlled so that the manipulator is always positioned at the preferred region, but the mobile platform and manipulator are controlled separately. Later, the effect of dynamic interaction between the mobile base and the manipulator is studied in [46]. In [13], the coordination of mobility and manipulation is formulated as a nonlinear optimization problem. A general cost function for point-to-point motion in Cartesian space was defined and minimized using a simulated annealing method. Seraji [47, 48, 26] used the configuration control approach to control the mobile base and the manipulator in a unified manner, where the nonholonomic constraints and task constraints can be incorporated into the augmented Jacobian $J(q) = [J_b(q), J_m(q), J_c(q)]^T$, and the whole system kinematics and constraints can be written as:

$$\begin{bmatrix} J_b(q) \\ J_m(q) \\ J_c(q) \end{bmatrix} \dot{q} = \begin{bmatrix} 0 \\ \dot{X}_t \\ \dot{Z} \end{bmatrix} \tag{2.1}$$

where $J_m(q)$ represents the holonomic kinematic constraints of the entire mobile manipulator to achieve target end effector velocity $\dot{X}_t$, $J_b(q)$ denotes the non-holonomic constraints on the mobile base, and $J_c(q)$ represents Jacobian matrix associated with additional task constraints, it is derived by differentiating the task kinematic function $Z = g(q)$. Let $\dot{X}_d = [0, \dot{X}_d t, \dot{Z}_d]$ be the desired end effector velocity and task constraints, then

$$\dot{q} = J(q)^{-1}\dot{X}_d \tag{2.2}$$

is the solution of Eq. (1). And the following control law can be used to control the nonholonomic mobile manipulator in coordination and also correct the task-space trajectory error,

$$\dot{q} = J(q)^{-1}[\dot{X}_d + K(X_d - X)] \tag{2.3}$$

Since mobile manipulators are usually redundant, in addition to following an end-effector trajectory, they are able to achieve multiple tasks or criteria, such as manipulability. Bayle et al. [10] generalize the manipulability to mobile manipulator, built from a manipulator mounted on a wheeled base. The extended manipulability can be used to optimally position the manipulator and base, and plan the end-effector motion. De Luca et al. [49] extended the Projected Gradient (PG) and the Reduced Gradient (RG) optimization-based methods to nonholonomic wheeled mobile manipulators. White et al. [50] developed a dynamic-level redundancy resolution method for nonholonomic wheeled mobile manipulators and experimentally validated the dynamic end-effector interaction control.

To enable the mobile robot to autonomously perform challenging manipulation tasks in the natural environment, there has been a line of research on visual servo control of mobile manipulators. Wang et al. [51] developed a hybrid visual servo controller for robust grasping, the controller is a form of image-based visual servo

(IBVS) controller and guarantees asymptotic stability of the closed-loop system. To improve the robustness, they developed a discrete-event controller based on Q-learning to keep the visual features in the view of the camera.

2.2.2 Sampling-based Motion Planning With Constraints

Most of the existing motion planners plan the motion between two points in the configuration space [52, 53, 54]. Over the past two decades, sampling-based motion planners [52, 55, 56] have been extensively studied since they are effective in exploring high-dimensional space and probabilistically complete. Among the sampling-based planners, RRTs can easily incorporate a variety of constraints. Therefore, RRTs have been applied to motion planning for nonholonomic manipulators and mobile robots with nonholonomic constraints [57]. However, RRTs have not been used to plan the full-body motion of a mobile manipulator.

To plan the motion of the nonholonomic mobile robots, one common approach is to combine the sampling-based method [55] with a steering method function [58], which serves as the local path planner to extend an existing node closer to the sampled node [59]. The optimal steering curve between two configurations is studied for some wheeled vehicles in the absence of obstacles, such as Dubins curves [60] for a forward-only car and Reeds-shepp curves [61] for a car with a reverse gear. These curves are the shortest geometric path for the corresponding robot models, but for the time-optimal motion of the robot, the dynamics constraints should be considered.

For motion planning for the mobile manipulator, in addition to the nonholonomic constraints, other constraints may arise from the mobile manipulation tasks, such as grasping an object. For grasping an object, the end effector has to reach the grasping pose at the time of grasping. This pose constraint defines a lower-dimensional manifold embedded in the configuration space. Special techniques are needed to account for the constraints of sampling-based motion planning methods [62], including relaxation, projection, tangent space, atlas, and reparameterization.

2.2.3 Optimization-based Motion Planning

Motion planning can be formulated as an optimization problem. Compared to the sampling-based method, optimization-based motion planning can flexibly incorporate a variety of constraints and also optimize a certain objective. Commonly employed objectives for robot motion are operation time, trajectory length, input energy, jerk, smoothness and etc., and the constraints include collision avoidance, joint limits, and dynamics. A good tutorial on trajectory optimization methods is referred to [63].

Khatib [64] pioneered the potential field method for collision avoidance and motion planning, but it is sensitive to local minimum. Later, an analytical navigation function with a unique minimum is proposed [65, 66] to address the local minima problem, a navigation function should (1) be smooth, (2) has a unique minimum at goal configuration, and (3) be uniformly maximal on the boundary of the free space and (4) be Morse [67].

Kalakrishnan et al. [68] proposed stochastic trajectory optimization for motion planning (STOMP), it performs local optimization by adding Gaussian noise around an initial trajectory, and the cost function based on a combination of obstacles and smoothness is optimized in each iteration. STOMP uses gradient-free optimization, in contrast, CHOMP [54] uses functional gradient techniques to iteratively improve the quality of an initial trajectory, it optimizes a functional that trades off between smoothness and obstacle avoidance, and the trajectory optimization is invariant to parametrization. Schulman et al. [53] formulated motion planning as a sequential convex optimization procedure, where the collision is penalized with a hinge loss, and no-collision constraint for convex objects is efficiently computed based on signed distance. Ichnowski et al. [69, 70] combined the grasp selection and motion planning to explore the time-optimal trajectory for bin-picking, the collision avoidance and grasp pose constraint are linearized in the trust region, and the motion is computed by optimizing with sequential quadratic programming (SQP) and iteratively updating trust regions, however, the grasping action takes place at the beginning of the motion instead of during the motion.

However, sampling-based and optimization-based methods have rarely been applied to directly plan the whole-body motion of both the manipulator and the mobile base, besides, they do not consider the tasks during the motion. The most relevant work to ours is [18], they used a sampling-based method to plan the manipulator motion from a moving mobile base. But the motion of the mobile base is planned prior to planning the manipulator motion, which leads to sub-optimality. Besides, they consider zero gripper velocity for grasping, which restricts the overall efficiency of the task.

2.2.4 Mobile Manipulation

Mobile Manipulation refers to performing robotic tasks with the help of both locomotion and manipulation ability. In recent years, there have been more and more learning-based methods used in mobile manipulation due to their ability to adapt to changes in the problem settings. Two possible reasons are: (1) Mobile manipulators are great agents for exploring the environment; (2) Learning-based methods help address the positioning uncertainty. At the same time, MPC-based methods are also popular because they can be used as whole-body controllers for mobile manipulation. From these recent mobile manipulation research using either learning-based approaches or classic approaches, I notice that there has been a trend in the efficient use of mobile manipulators by simultaneously controlling the mobile base and the arm. [71, 72, 73, 74].

Classic Approaches to Mobile Manipulation

Most of the existing works in the mobile manipulation community use the mobility mode and manipulation mode separately [15]. Obviously, the decoupled motion of the manipulator and mobile platform is not efficient. In the early works in the 90s, there have been a few works on the coordinated control of the mobile manipulator. For example, Yamamoto et al. [24] proposed a method for controlling the mobile platform such that the manipulator is positioned at a configuration of high manipulability. Seraji [26] proposed a unified approach to control the manipulator and mobile base as a whole, using the augmented Jacobian matrix. These works usually assumed a given end-effector trajectory and the absence of obstacles.

Liao et al. [75] used the optimization-based method to plan the whole-body motion of a holonomic mobile manipulator (from $\mathbf{q}_i$ to $\mathbf{q}_f$ or from $\mathbf{q}_i$ to a target end-effector pose $\mathbf{P}_f$). Spahn et al. [74] also performed whole-body trajectory optimization for mobile manipulation, however, the mobile manipulator performs the picking and placing tasks at the beginning and the end of the trajectory, instead of during the

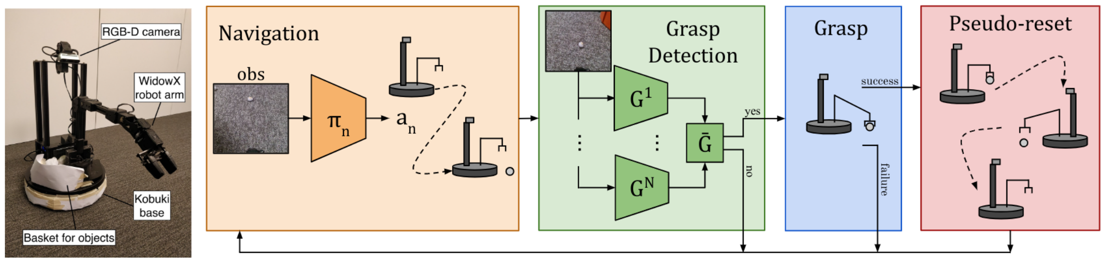


Figure 2.1: Overview of reinforcement learning for mobile manipulation.

motion. I notice that a recent work conducted aerial grasping during the motion [76], however, they have to predefine the grasp timing and velocity, therefore, basically, they plan two trajectories (1) from initial configuration to grasp and (2) from grasping to the final configuration.

There is another line of works on dynamic grasping [77, 78, 79], i.e., grasping a moving object. The common part is that there is relative motion between the robot and the object. However, optimal motion is not the major concern of these works, their pipeline usually involves motion prediction of the moving object, grasp planning, visual tracking, trajectory generation, and trajectory execution.

As reviewed above, although there are a few works on whole-body trajectory optimization for mobile manipulators [75, 74], none of the existing motion planners have addressed the problem of planning the optimal motion for a robot to **perform tasks during the motion**, which is our major contribution.

Learning-based Approaches to Mobile Manipulation

Sun et al. [80] proposed a system for a mobile manipulator to autonomously learn skills using a combination of navigation and manipulation. The overview of the system is shown in Fig. 2.1. This system learns a navigation policy and a grasping

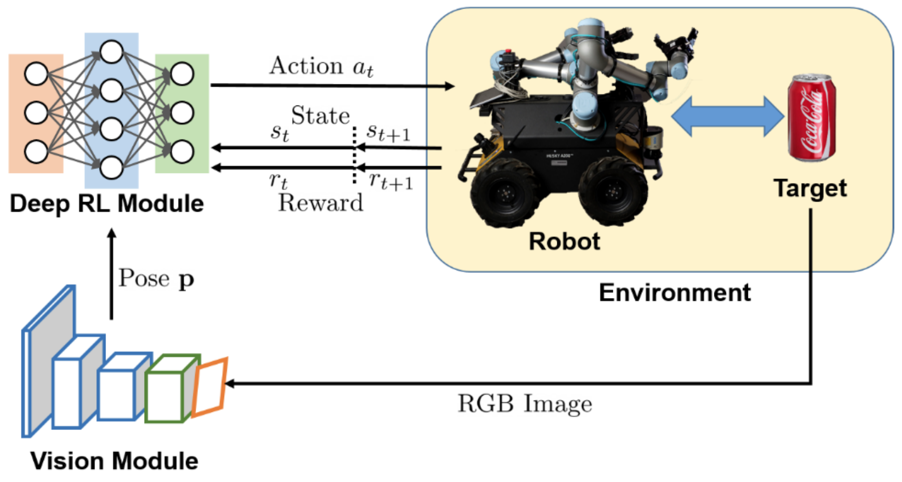


Figure 2.2: Overview of the deep reinforcement learning framework to control a mobile manipulator performing mobile picking tasks.

policy for the mobile manipulator to perform room cleaning tasks, i.e., picking up objects on the ground. The grasp policy predicts the probability of success for a grasp action from the RGB images observed by the robot's camera. The grasp action is simply represented as moving the gripper to a predicted (x, y) coordinates slightly above the ground for grasping, the orientation of the grasp is not considered since they consider grasping some simple objects.

Wong et al. [72] presented a teleoperation framework that allows simultaneous navigation and manipulation of mobile manipulators. They collected a large dataset in a simulated kitchen environment and proposed a learned error detection system to detect covariate shifts. They train imitation learning policies on the collected data and achieve 45% task success rate.

Honerkamp et al. [73] proposed a learning-based method to generate a kinematically feasible trajectory for the base of a mobile robot when its end-effector moves toward a certain goal pose. They formulate this problem as a goal-conditioned reinforcement learning problem. The reward function is composed of two parts: (1) The first part is an indicator function for kinematic feasibility evaluated by

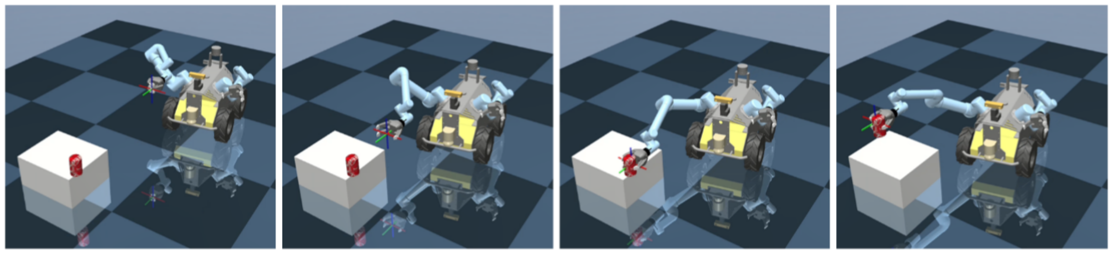


Figure 2.3: The mobile manipulator performs mobile picking using the proposed framework.

solving the inverse kinematics using an IK solver; (2) The second part penalize the unnecessary actions. This system learns a policy for generating the next base velocity command, using the current robot state, and the next end-effector goal pose. However, one of the limitations is that they did not consider collision avoidance with the environment.

Wang et al. [71] applied the deep reinforcement learning-based method to mobile manipulation tasks. The overview of the framework is shown in Fig. 2.2. For this deep reinforcement learning system, the state includes the position of the gripper w.r.t the robot base frame, the position of object w.r.t the gripper frame, the position of object w.r.t the robot base frame, the joint positions, and velocities of the arm, as well as the gripper state. The pose of the object is estimated from the camera mounted on the mobile platform. The action is defined as the gripper relative control action, the robot base relative control action, and the binary gripper control action. The reward function is composed of three terms, the first term is the control action reward, the second term measures the distance between the gripper and the object, and the third term is a sparse reward for a successful grasp.

2.3 Gripper Design for Grasping and Assembly

There has been a lot of research on gripper design [81, 82], however, very few of them design the grippers for an assembly task considering the assembly constraints. Another line of research that is related to our work is part/model/primitive-based grasp planning [83, 84, 85, 86, 87], in this sense, our work can be called shape primitive-based gripper design, considering assembly constraints and optimization of a number of grippers.

2.3.1 Gripper Design and Robotic Assembly

Generally, the grippers are specially designed according to the task to be performed [82, 81], in this case, the design process takes many iterations to obtain a satisfactory design. There have been very few attempts to design grippers for an assembly task and improve the design efficiency, to the best of our knowledge, the most relevant works to ours are [22, 88]. Pham et al. [22] surveyed the design methods to achieve versatile and cost-effective gripping and proposed a strategy for minimizing the number of grippers through part-family grouping, and later Pham et al. [88] proposed a system to determine the configuration of grippers for an assembly task. However, none of these works explicitly incorporate the assembly constraints into the gripper design, besides, the mesh segmentation and primitive fitting method used in our approach are able to handle models with more complex shapes.

In addition to the gripper configuration, the contact between the gripper finger and object plays an important role in grasp stability, therefore, the contact model has been studied extensively [89, 90, 91, 92]. Early research mainly used the point contact model [93], later on, the soft finger model was developed to model the contact in a more realistic way [90, 89]. Some researchers studied the finger design to change the contact characteristics and improve the performance of the gripper. Honarpardaz et al. [94, 21] proposed a generic optimized finger design (GOFD) to automate the finger design process, the fingertip shape was designed to mimic the local surface contour of the workpiece, thus the contact area was increased. Song et al. [95] noticed that most grasp contacts share a few local geometries, they proposed a uniform cost algorithm to cluster a set of example grasp contacts into several contact primitives and designed the fingertip shape to match the local geometry of the contact primitive in order to increase the contact area.

Rodriguez et al. [96] explored the effector form design for 1 DoF planar actuation, the mechanical function of a product is formulated as the product of the effector's shape and motion. Taylor et al. [97] investigated the role of shape and motion in the contact interaction and proposed a framework to optimize the shape and motion of a planar rigid body end-effector to achieve a manipulation task. Chavan-Dafle et al. [98] proposed a two-phase gripper to passively reorient the objects while picking them up. Birglen et al. [99] extensively reviewed the characteristics of industrial grippers, the stroke, weight, force, and weight, as well as performance, are investigated in detail. Hermann et al. [100] designed a gripper that can switch between two modes, including a grasping mode and a fully actuated precision mode.

For an assembly task, usually more than one gripper is required to grasp all the assembly components. Kramberger et al. [101] proposed a flexible and cost-effective grasping solution to quickly develop and test fingertips to handle multiple parts. Harada et al. [102] incorporated the tool changer into the assembly planner and proposed an assembly planner that is able to automatically select a suitable gripper to assemble parts. Nakayama et al. [103] designed grasping tools for an assembly task based on shape analysis of parts, however, the assembly constraints are not considered in the evaluation of graspable segments and suitable gripper configurations, and additionally I optimize the number of grippers for the assembly task.

2.3.2 Shape Approximation Based Grasping

Grasp planning is difficult due to a large number of possible gripper configurations, but grasping planning can be simplified if considering the shape of the object and the grasping strategy are closely related. Miller et al. modeled the object as a set of simple shape primitives [83], then the grasp location and preshape can be determined. Goldfeder et al. [104] used a decomposition tree of the object to prune the large space of possible grasps into a subspace that is likely to contain many good grasps. Huebner et al. [85] approximated the object by box primitives and selected grasps based on the approximated boxes. However, the error of approximation by primitives may result in low-quality grasps, to counteract this problem, Przybylski et al. [84] proposed the grid of spheres for grasp planning, which effectively reduces the search space for grasps without sacrificing potential high-quality grasps.

These researches **passively** plan grasps given the object model, but I can also **actively** design the gripper configurations according to the shape of the target object in order to easily obtain high-quality grasps. This idea is somewhat related to the taxonomy of grasps proposed in [105], where the grasps are classified based on task-related and geometric considerations, each type of grasps is corresponding to one category of tasks and object geometry. For grasping the assembly components, I select suitable grasping postures according to the shape of the assembly components, since I do not use a dexterous robot hand to realize these grasps, instead I abstract a simple gripper configuration from the grasping postures of a dexterous hand.

CHAPTER 3

BASE POSITION PLANNING FOR EFFICIENT PICKUP OF ASSEMBLY PARTS

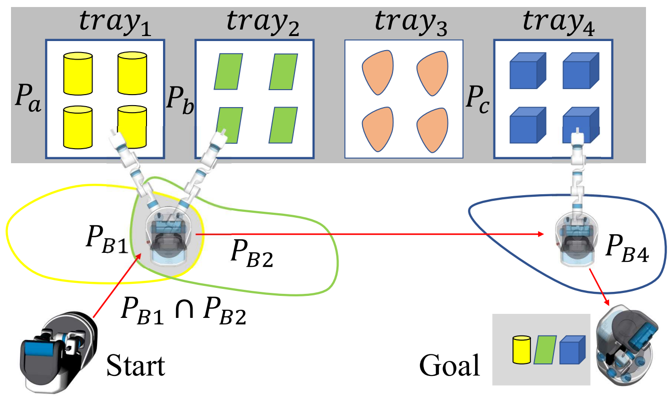
Mobile manipulators are able to operate in a large workspace, and have the potential to replace human workers to perform a sequence of pick-and-place tasks at separate locations. Many existing works optimize the base position or manipulator configuration for a single manipulation task, however, very few of them consider a sequence of tasks. In this chapter, I present a planner that plans a minimum sequence of base positions for a mobile manipulator to robustly collect objects stored in multiple trays. I use inverse kinematics to determine the base region where a mobile manipulator can grasp the target objects stored in a tray, and move the mobile manipulator to the intersections of base regions to reduce the operation time for moving the base. I ensure robustness by only considering the intersection whose radius of the inscribed circle is larger than the base positioning error. Then the minimization of the number of base positions is formulated as a 0-1 knapsack problem. Besides, considering different object placements in the tray, I analyze feasible policies for dynamically updating the base sequence based on either the remaining objects or the target objects to be picked. In the experiment, I examine our planner in various scenarios, including different object placements: (1) Regularly placed toy objects; (2) Randomly placed industrial parts; and different implementation policies: (1) Apply globally static base positions; (2) Dynamically update the base positions. The experiment results show that the time for moving the base decreases by 11.22 seconds (29.37%) to 17.26 seconds (36.77%) by reducing one base movement, and demonstrate the feasibility and potential of the proposed method.

3.1 Method Overview

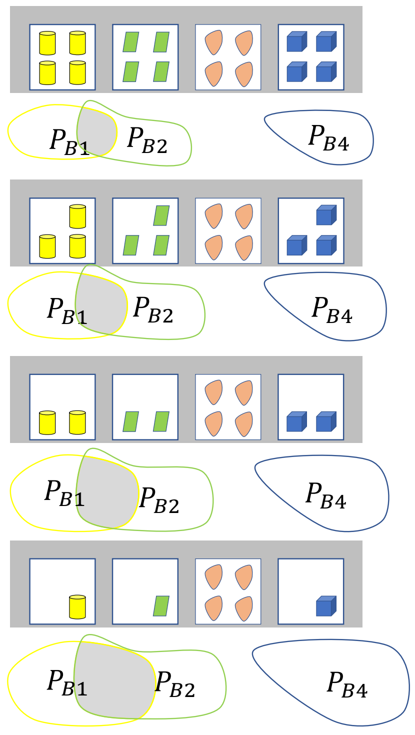
Fig. 3.1a illustrates the overview of the tasks by a simple example. The product P to be assembled consists of three types of assembly parts P_a, P_b and P_c (subscripts a, b and c are used to differentiate different types of assembly parts). Each type of assembly parts are placed in the same tray, thus they are classified and stored in three different trays $tray_1$, $tray_2$ and $tray_4$, respectively. Our goal is to plan the base sequence for collecting the target assembly parts from the containing trays. The following information is assumed to be known: (1) The types of parts to be collected and their associated trays. (2) The geometrical models of the trays and the potential obstacles in the environment. (3) The poses of the trays and obstacles. Since the target application scenario is in the manufacturing environment, the above information is readily available. The grasping poses for the objects can be obtained using either model-based or model-free methods, depending on whether the objects are regularly or randomly placed in the tray. In the first case, I need the geometric models of the objects, and in the latter case, the grasping poses can be estimated by the model-free method described in Section 3.7.2.

Algorithm 1 presents the overview of the planner. The algorithm mainly consists of three steps: (1) Solve the base regions for every target tray which contains the target assembly parts. (line 2 to line 4) (2) Determine the robust intersections of the base regions. (line 5 to line 10) (3) Plan a minimal sequence of base positions from the robust intersections to visit all the base regions (line 11).

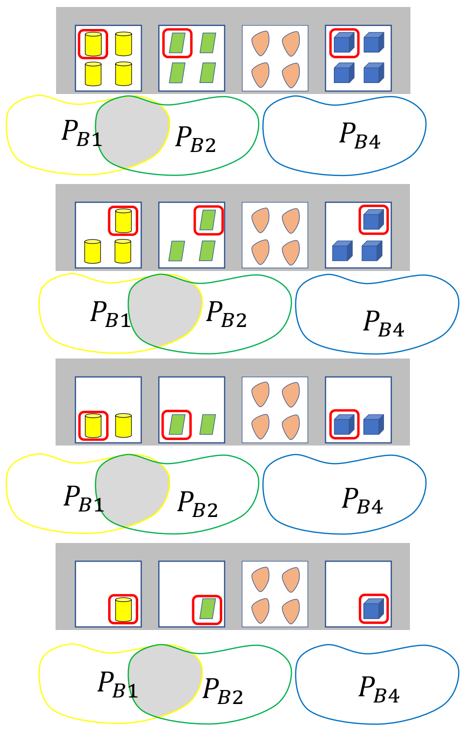
Fig. 3.1a, b, and c illustrate three policies that can be implemented in practical applications. They differ in the target objects used to plan the base positions. In Fig. 3.1a, all the objects in a tray are used to plan a globally static base sequence. Fig. 3.1b shows the update of base regions based on the remaining objects in the



(a)



(b)



(c)

Figure 3.1: Overview of the method (top view). Several types of parts are required in the task, and one or more parts of each type will be collected. The sub-figures show three types of policies respectively: (a) Global static base positions for collecting all the objects in the tray. (b) The base positions update based on the remaining objects in the tray. (c) The base positions update based on the objects to be picked, which are surrounded by red lines, in a round of pick-and-place tasks.

Algorithm 1: Overview of the algorithm

```
   Input: Grasping poses for objects in target trays
   Output: A sequence of base positions
1  $base\_regions \leftarrow \{\}$
2  for $tray \in target\_trays$ do
3      $base\_region \leftarrow$ SolveBaseRegion($tray$)
       // assume base region is robust
4      $base\_regions \leftarrow base\_regions \cup base\_region$
5  $robust\_intersections \leftarrow \{\}$
6  $intersections \leftarrow$ PossibleIntersections($base\_regions$)
7  for $intersection \in intersections$ do
       // radius of the inscribed circle
8      $r \leftarrow$ InscribedRadius($intersection$)
9      if $r \geq positioning\_uncertainty$ then
           // save robust intersections
10         $robust\_intersections \leftarrow robust\_intersections \cup intersection$
11 $base\_sequence \leftarrow$ PlanMinSequence($robust\_intersections$)
12 return $base\_sequence$
```

tray. After every round of pick-and-place, the remaining objects decrease and the area of the base region increases, which improves the overall robustness. Fig. 3.1c shows another policy for updating the base regions according to the objects to be picked. A good picking order may reduce the variance of the robustness in different rounds of pick-and-place tasks.

3.2 Inverse Kinematics

The inverse kinematics problem is to determine a set of joint angles that bring the end effector to the desired pose. In this study, I aim at obtaining a complete set of base positions where there exists at least one collision-free IK solution that reaches desired grasping poses. Since solving IK and collision check are performed separately, this leads to a planner that loses the ability to be probabilistically complete. In terms of collision check between the mobile manipulator and the environment, it helps to find a collision-free manipulator configuration by generating a variety of candidate IK solutions that cover all kinds of manipulator configurations, thus it is not likely to miss a feasible base position in which a collision-free IK solution can be found. For non-redundant manipulators, it is feasible to find out all the IK solutions and perform a collision check between the manipulator and the environment. However, generally there are an infinite number of IK solutions for redundant manipulators, a common IK solver returns only one or multiple but not necessarily representative IK solutions, in that case, the IK solver might only find IK solutions that consequently fail in collision check, even if there exist collision free solutions. Therefore, for redundant manipulators, it is important to find out representative IK solutions for further collision checks. Parametrized IK approaches for 7-DOF redundant manipulators [106, 107] are able to find all the feasible IK solutions, while this method is usually manipulator-specific. I propose a manipulator-independent method of obtaining approximated representative IK solutions, by querying the vicinity of a target pose in the reachability database, and it is proved to be a resolution complete method of solving IK.

3.2.1 Reachability Database

The reachability database is constructed by sampling the joint space of the manipulator, reachable poses are obtained by calculating forward kinematics (FK). However, this may introduce the preference for singular configurations, that large variations in joint angles only result in small differences in the grasping poses. To obtain a more uniform distribution of poses in the workspace, the manipulability measure [9] can be applied to relieve the congestion of configurations near the singular configurations. The downsampled resultant poses of the Fetch robot [108] are illustrated in Fig. 3.2 (Left). The reachability can also be represented by 3-dimensional voxels containing sampled grasping poses, as shown in Fig. 3.2 (Right), the Cartesian workspace is discretized into many 3 dimensional voxels according to the positions. The color of the voxels indicates the number of grasping poses that end up in the corresponding voxels.

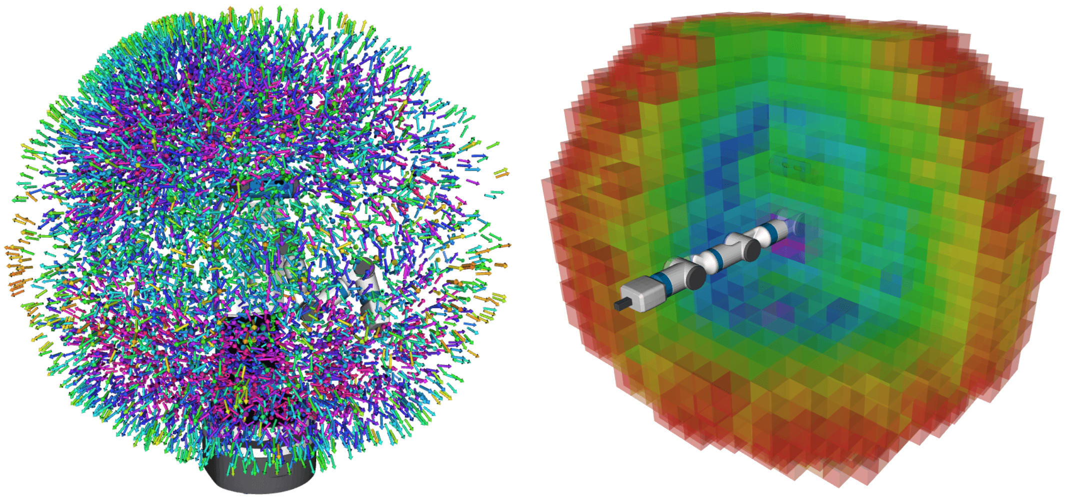


Figure 3.2: Representation of downsampled version of our reachability database. (Left) Reachable poses are represented by colored arrows, the color encodes the manipulability of the corresponding manipulator configuration. (Right) In the cross-sectional view of the voxelized 3d workspace of a Fetch robot, the color indicates the number of grasping poses contained in the 3d voxel. For further pose queries, the entire 92 million poses are distributed to 2 million 6d voxels, instead of 3d voxels.

For further query of poses comprised of both translation and rotation parts, 6-dimensional voxels are adopted instead, thus the workspace is discretized in both position (x, y, z) and orientation (represented by roll α, pitch β and yaw γ), the grid lengths are Δx, Δy, Δz, $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$, respectively. All the resultant poses calculated by forward kinematics, together with their joint angles, are stored in the corresponding 6-dimensional voxels according to their positions and orientations. For example, a grasping pose $\mathcal{G}_i = [x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i]^T$, should be stored in the voxel indexed by ($x_i/\Delta x$, $y_i/\Delta y$, $z_i/\Delta z$, $\alpha_i/\Delta\alpha$, $\beta_i/\Delta\beta$, $\gamma_i/\Delta\gamma$), within the voxel is a set of poses with similar position and orientation, thus the vicinity of a target pose can be quickly found by querying the pose from such a data structure.

3.2.2 IK Query

Instead of resolving the inverse kinematics by an IK solver, I obtain the IK solutions by querying the grasping pose in the database and accessing the corresponding joint angles. However, the reachability database is only a discrete representation of the continuously varying reachable poses. Probabilistically, I will fail to find an identical grasping pose in the database. Since solving IK and checking collision are treated separately, the IK solutions should be as diverse as possible in order to be resolution complete in finding collision-free IK solutions. Instead, I approximate the IK solutions of the target grasping pose $\mathcal{G}_t$ by querying a range of poses in the vicinity of $\mathcal{G}_t$, for $\mathcal{G}_i \in (\mathcal{G}_t - \Delta\mathcal{G}, \mathcal{G}_t + \Delta\mathcal{G})$, where $\Delta\mathcal{G} = [\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma]^T$, and the associated manipulator configurations of $\mathcal{G}_i$ are the approximated IK solutions of $\mathcal{G}_t$.

In the reachability database, every 6d voxel contains a small range of poses, firstly I find the voxel containing the target pose, and all the poses within the voxel are regarded as the vicinity of the target pose. For example, by querying pose $[0.6, 0, 0.8, 0, 0.5, 0.5]^T$ from the database, the indexed voxel is found to have 151 poses, and Fig. 3.3 shows a part of the manipulator configurations among them, these are the approximated IK solutions of the target pose[1]. As the database resolution goes to infinity, the approximation error approaches zero and the queried

[1]If exact IK solutions are desired, it is recommended to use the approximated IK solution as an initial seed in a numerical IK solver, then it can quickly converge to the exact solution after a few iterations

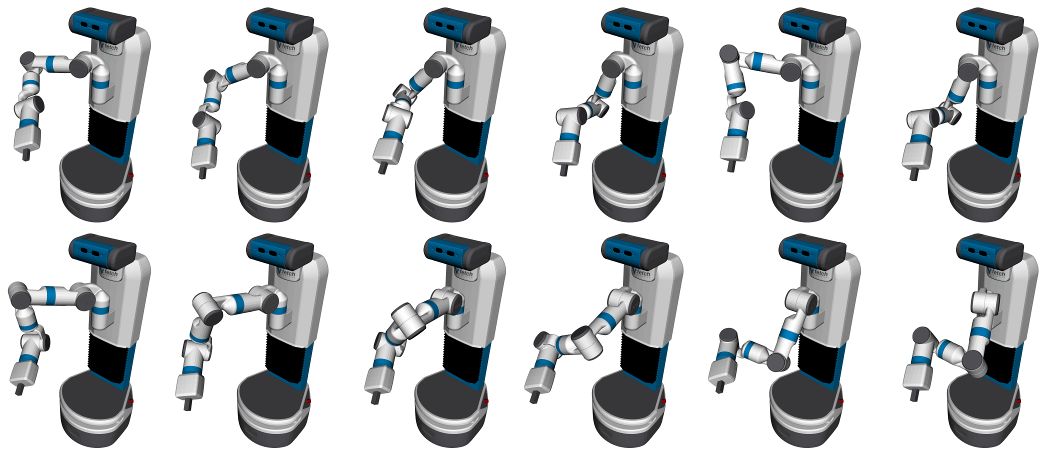


Figure 3.3: A part of obtained 151 representative manipulator configurations by querying the pose $[0.6, 0, 0.8, 0, 0.5, 0.5]^T$ from the reachability database, they are the approximation of the exact IK solutions of the target pose and cover different possible manipulator configurations.

manipulator configurations include complete IK solutions of the target pose, such that, collision-free IK solutions can be found if there exist, regardless of the location of the obstacles. The feasibility and completeness of approximating IK solutions of $\mathcal{G}_t$ by the associated manipulator configurations of $\mathcal{G}_i \in (\mathcal{G}_t - \Delta\mathcal{G}, \mathcal{G}_t + \Delta\mathcal{G})$, are proved by the following two lemmas, they are based on the differentiable mapping between configuration space and workspace, except for singular configurations. The first lemma is to prove that, as the sampling resolution goes to infinity, I can always find a manipulator configuration within the voxel, that approaches any one of the IK solutions of the target pose. The second lemma proves that, as the voxelization resolution goes to infinity, all the manipulator configurations within the voxel approach the IK solutions of the target pose. Note that the completeness of approximating all the IK solutions of $\mathcal{G}_t$ is already guaranteed by lemma 1, and lemma 2, together with Lemma 1, is to guarantee the completeness of obtained base positions in the next section.

Definition: Let $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$ be the manipulator configuration vector, $\Delta\theta$

and $\Delta\mathcal{G}$ be the sampling and voxelization resolution, $IK(\mathcal{G}_t)$ be the IK solution set of $\mathcal{G}_t$, $FK(\Theta)$ be the resultant pose calculated by forward kinematics of Θ, $IKRDB(\mathcal{G}_i)$ be the corresponding manipulator configuration of $\mathcal{G}_i$ in the reachability database, J^+ be the pseudo-inverse of Jacobian matrix J, $voxel(\mathcal{G}_t)$ be the voxel containing pose $\mathcal{G}_t$.

Lemma 1: $\forall \Theta_t = [\theta_1, \theta_2, \ldots, \theta_n]^T \in IK(\mathcal{G}_t)$, $\exists \mathcal{G}_i \in (\mathcal{G}_t - \Delta\mathcal{G}, \mathcal{G}_t + \Delta\mathcal{G})$, $\Theta_i = IKRDB(\mathcal{G}_i)$, such that $\lim\limits_{\Delta\theta \to 0} \|\Theta_i - \Theta_t\| = 0$.

Proof: Set $\{\Theta \mid FK(\Theta) = \mathcal{G}_t\}$ is equivalent to set $\{\Theta \mid \Theta = IK(\mathcal{G}_t)\}$, if $\Delta\theta \to 0$, then $\{\Theta \mid FK(\Theta) = \mathcal{G}_t\} \subset \{\Theta \mid FK(\Theta) = \mathcal{G}_i \in (\mathcal{G}_t - \Delta\mathcal{G}, \mathcal{G}_t + \Delta\mathcal{G}), \Delta\theta \to 0\}$, thus $\forall \Theta_t \in IK(\mathcal{G}_t)$, $\exists \mathcal{G}_i \in (\mathcal{G}_t - \Delta\mathcal{G}, \mathcal{G}_t + \Delta\mathcal{G})$, $\Theta_i = IKRDB(\mathcal{G}_i)$, such that $\lim\limits_{\Delta\theta \to 0} \|\Theta_i - \Theta_t\| = 0$.

Lemma 2: $\forall \mathcal{G}_i \in (\mathcal{G}_t - \Delta\mathcal{G}, \mathcal{G}_t + \Delta\mathcal{G})$, $\Theta_i = IKRDB(\mathcal{G}_i)$, $\exists \Theta_t \in IK(\mathcal{G}_t)$, that $\lim\limits_{\Delta\mathcal{G} \to 0} \|\Theta_i - \Theta_t\| = 0$.

Proof: $\dot{\mathcal{G}} = J(\Theta)\dot{\Theta}$, $\dot{\Theta} = J^+(\Theta)\dot{\mathcal{G}} + (I - J^+J)k$, where k is an arbitrary vector denoting redundancy, integrate two sides of the formula by a small time step, $\Delta\Theta = J^+(\Theta)\Delta\mathcal{G} + (I - J^+J)k\Delta t$, then $\forall \Delta\mathcal{G} \to 0$, $\exists k = 0$ such that $\Delta\Theta \to 0$. Because $|\mathcal{G}_i - \mathcal{G}_t| < \Delta\mathcal{G}$, $\Delta\mathcal{G} \to 0 \Rightarrow |\mathcal{G}_i - \mathcal{G}_t| \to 0$, thus $\lim\limits_{|\mathcal{G}_i - \mathcal{G}_t| \to 0} \|\Theta_i - \Theta_t\| = 0$. (replace J^+ with J^{-1} for non-redundant manipulators)

The above lemmas apply to both redundant and non-redundant manipulators, the only problem with this method is that the continuous mapping between joint space and workspace breaks down at singular configurations.

3.3 Base Region Calculation

The base region for a tray is a set of base positions where the mobile manipulator is able to reach all the targets in the tray, and avoid self-collision and the collision with the environment. Firstly, I prepare stable grasping poses with respect to the object for every object in the tray, using a grasp planner [109, 110, 111, 112]. Then sample base poses (x_i, y_i, ϕ) in front of the target tray, as illustrated in Fig. 3.6, here I assume that the orientation ϕ of the mobile manipulator is constant, because for many mobile manipulators, the joint connecting the manipulator and mobile base rotates around a vertical axis, thus counteracts the rotation of the mobile base and contributes almost nothing new. Then the set of stable grasping poses $\{\mathcal{G}_{t1}, \mathcal{G}_{t2}, \ldots, \mathcal{G}_{tn}\}_j$ with respect to the mobile base for object $\mathcal{O}_j$ is obtained for every object in the tray. Finally, if there exists a grasping pose $\mathcal{G}_{ti} \in \{\mathcal{G}_{t1}, \mathcal{G}_{t2}, \ldots, \mathcal{G}_{tn}\}_j$, such that I can find a collision-free manipulator configuration $IKRDB(\mathcal{G}_k)$ for a pose $\mathcal{G}_k \in voxel(\mathcal{G}_{ti})$, then $\mathcal{O}_j$ can be grasped from the base position, a base position for the tray is feasible if all the objects in the tray can be grasped. All the feasible positions formulate the base region for grasping objects from the tray.

I compare the base regions obtained by different IK approaches. The obtained base regions using IKFast plugin to find the IK solutions are shown in Fig. 3.4. Fig. 3.5 is the base regions calculated by the IK query approach proposed in this chapter, it is obvious the base region is larger. In the IK solver approach, IK solutions are not found in some base positions, and some of the found IK solutions fail in the collision check. For the IK query approach, the obtained base positions are not always feasible, but the base region is complete when the resolution of the reachability database goes to infinity.

Algorithm 2: Calculate the base region for a tray

```
   Input: Target objects in the tray
   Output: Base region of the tray
1  Function SolveBaseRegion(tray)
       // plan grasps for every target object
2      $grasps\_all\_objs \leftarrow$ PlanGrasps($objs\_in\_tray$)
3      $sampled\_positions \leftarrow$ SampleBasePositions($tray$)
4      $base\_region \leftarrow \{\}$
5      for $position \in sampled\_positions$ do
           // All target objects should be graspable
6          $all\_objs\_graspable \leftarrow true$
7          for $grasps\_obj_i \in grasps\_all\_objs$ do
               // At least one grasp should has collision-free IK
8              for $grasp \in grasps\_obj_i$ do
9                  $IKs \leftarrow$ SolveIK($grasp, position$)
10                 if ExistCollisionFreeIK($IKs$) then
11                     break
12                 else if $IsLastGrasp$ then
13                     $all\_objs\_graspable \leftarrow false$
14                 else
15                     continue
               // if not all the objects are graspable, check the
                  next position
16             if Not $all\_objs\_graspable$ then
17                 break
18         if $all\_objs\_graspable$ then
19             $base\_region \leftarrow base\_region \cup position$
20     return $base\_region$
```

3.4 Base Sequence Planning

Although the ”target” objects vary in different policies (Section 3.5), they share the same algorithm for planning the base sequence given the grasping poses for the ”target” objects, as introduced in this section.

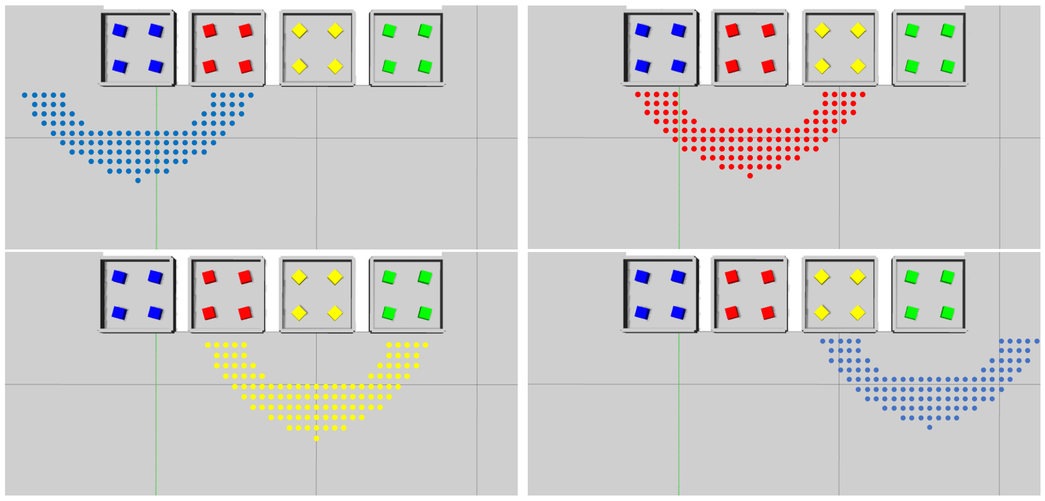


Figure 3.4: The base regions obtained by using the IKFast solver.

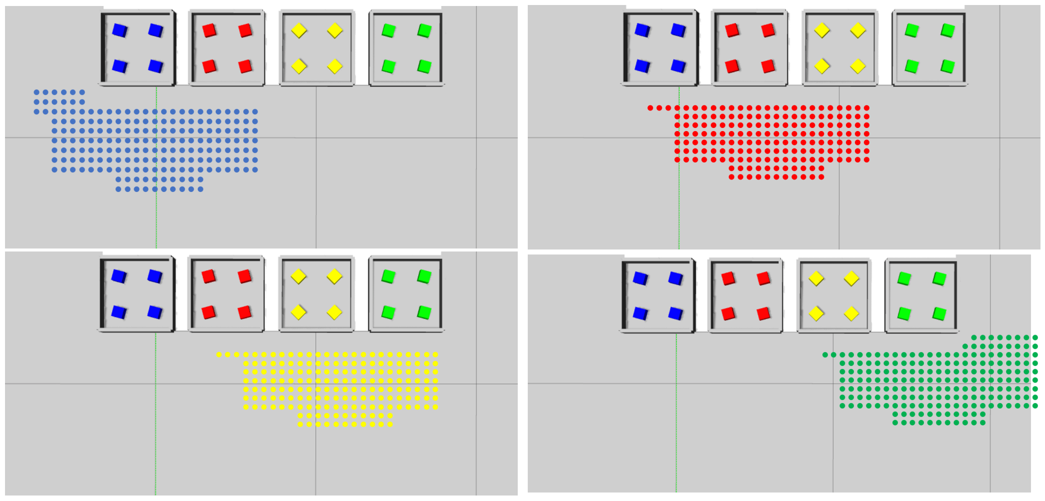


Figure 3.5: The base regions obtained by the proposed IK query method.

3.4.1 Task Defined as Reaching the Grasping Poses

For grasping the target objects in the target trays, a sequence of tasks is defined as the grasping poses to be reached when the mobile manipulator visits the base positions. The mobile manipulator should be able to grasp every target object with at least one grasping pose. The grasp planning method varies for different object placements, and the grasp planning method considered in this chapter is shown in Algorithm 3. For objects regularly placed in the tray, a model-based grasp planner [109, 113] can be used to prepare a set of grasps for the target object $\mathcal{O}_j$ in the tray in the offline phase (lines 2 and 3). For objects randomly placed in the tray, I treat the objects in the tray as a whole and use a model-free grasp planning method [114] to estimate a set of grasps from depth images (line 5), the details are described in Section 3.7.2.

Algorithm 3: Planning grasps

```
  Input: Mesh models of the objects or rendered depth images
  Output: Planned grasps
1 Function PlanGrasps()
2     if RegularlyPlaced then
3         grasps ← ModelBasedGraspPlanning(mesh)
4     else
5         grasps ← EstimateFromDepthImages(imgs)
6     return grasps
```

Algorithm 2 explains the procedure of calculating the base region for a tray. The base region for a tray is a set of base positions where the mobile manipulator is able to reach all the targets in the tray, without self-collision and the collision with the environment. Firstly, I prepare stable grasping poses with respect to the object for every target object in the tray (line 2), using a grasp planner. Then I uniformly sample base poses (x_i, y_i, ϕ) in front of the target tray (line 3), as illustrated in Fig. 3.6. Here, I assume that the orientation ϕ of the mobile manipulator is constant and keep the mobile manipulator facing the tray, which is the positive direction of the y-axis as illustrated in Fig. 3.6. This assumption is based on the observation that in many mobile manipulators, the joint connecting the manipulator and mobile base rotates around a vertical axis, thus having an equivalent effect as rotating the mobile base. Then the set of stable grasping poses $\{\mathcal{G}_{t1}, \mathcal{G}_{t2}, \ldots, \mathcal{G}_{tn}\}_j$ with respect to the mobile base for object $\mathcal{O}_j$ are obtained for every target object in the tray.

Algorithm 4: Solve inverse kinematics

```
    Input: Planned grasps specified in local frame (tray or object) and the
           position of mobile base
    Output: Approximated inverse kinematics solutions
1   Function SolveIK(grasp, position)
        // get grasping pose w.r.t the mobile base
2       $EE\_pose \leftarrow$ Transform($grasp, position$)
3       $voxel \leftarrow$ QueryDB($EE\_pose$)
4       $IKs \approx$ configs_in_voxel($voxel$)
5       return $IKs$
```

In order to determine the feasibility of a grasping pose, firstly I solve inverse kinematics (IK) and then check if the IK solutions are collision-free. I use the method presented in section 3.2 to solve inverse kinematics approximately. The advantage of this method is that it returns a set of diverse IK solutions, which are helpful for further collision checks, since solving IK and checking collisions are performed separately. The IK solutions are obtained by querying the grasping pose in a pre-computed database and then accessing the corresponding joint angles. The workflow is briefly explained in Algorithm 4. Firstly I transform the end-effector pose (EE_pose) in the frame of the mobile manipulator (line 2), then I locate the corresponding voxel in the reachability database (line 3), finally, the manipulator configurations stored in the voxel are the approximated IKs.

If there exists at least one grasping pose $\mathcal{G}_{ti} \in \{\mathcal{G}_{t1}, \mathcal{G}_{t2}, \ldots, \mathcal{G}_{tn}\}_j$, such that I can find a collision-free IK solution, then object $\mathcal{O}_j$ can be grasped from the base position. A base position for the tray is feasible if all the target objects in the tray are graspable. The corresponding pseudocode is line 6 to line 19 in Algorithm 2. All the feasible positions constitute the base region for grasping target objects from the tray. Considering the base region for a tray may update due to the change of target objects, it is preferable to calculate the base region for every object in the

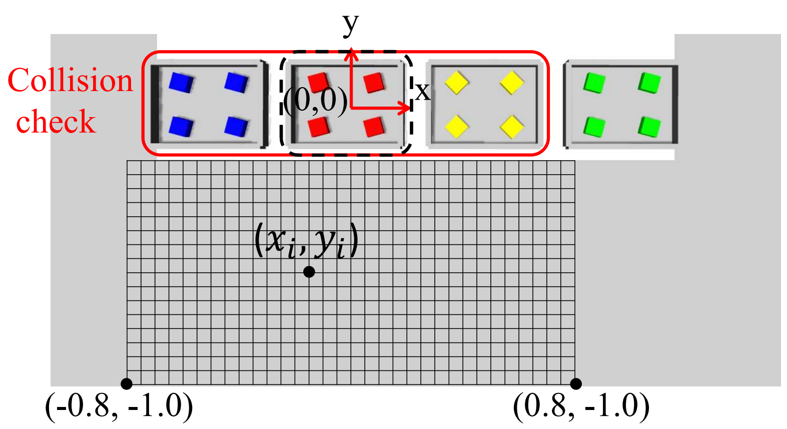


Figure 3.6: The sampled base positions for one of the trays, which is circled by a black dashed line. The range of sampling is determined by referring to the reachable workspace of the robot. Collision check is performed between the mobile manipulator and the target tray, its neighboring trays, and other obstacles in the environment.

tray and save such information for further access. Then the base region for a tray, which is the intersection of the base regions of the target objects, can be quickly solved.

3.4.2 Robust Intersections of Base Regions

To reduce the number of base movements, the mobile manipulator had better move to the intersections where the mobile manipulator is able to pick up the objects in multiple trays. For the IK solver approach, there are 5 intersections for the obtained 4 base regions of 4 trays, all of them are the intersections of two base regions, while for the IK query approach, there are the intersections of two base regions and even the intersections of 3 base regions. The intersections and their associated trays are labeled with numbers in Fig. 3.7 and Fig. 3.8.

Given a part-supply task to collect the objects in trays $\{tray_1, tray_2, \ldots, tray_n\}$, the corresponding base regions $\{P_{B1}, P_{B2}, \ldots, P_{Bn}\}$ and their intersections can be

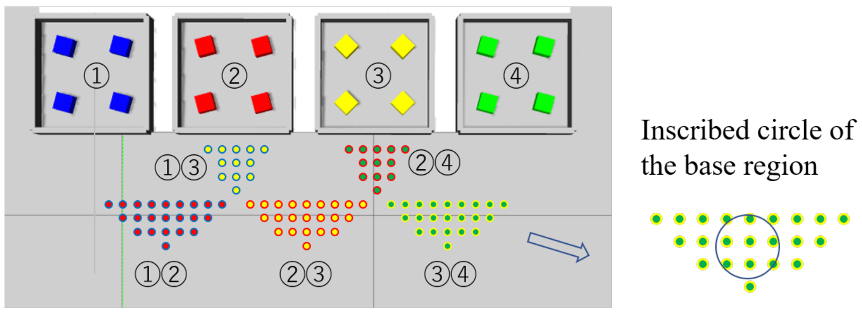
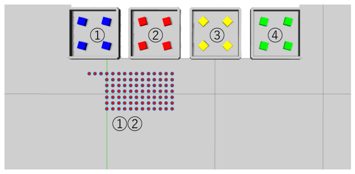
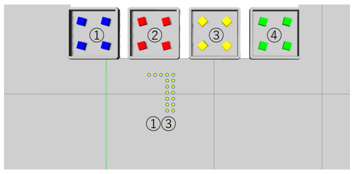


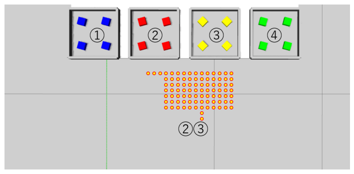
Figure 3.7: The intersections of the base regions in Fig. 3.4, the centers of their inscribed circles are the most robust base positions.



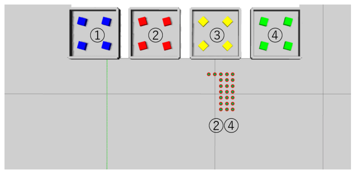
(a) $P_{B1} \cap P_{B2}$



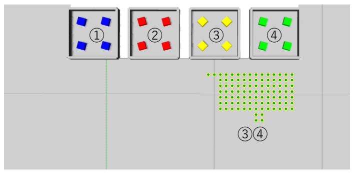
(b) $P_{B1} \cap P_{B3}$



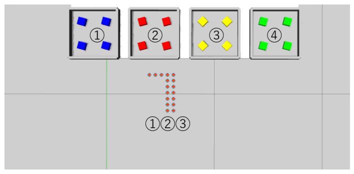
(c) $P_{B2} \cap P_{B3}$



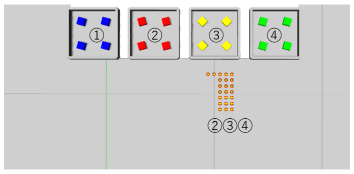
(d) $P_{B2} \cap P_{B4}$



(e) $P_{B3} \cap P_{B4}$



(f) $P_{B1} \cap P_{B2} \cap P_{B3}$



(g) $P_{B2} \cap P_{B3} \cap P_{B4}$

Figure 3.8: All the intersections of the base regions in Fig. 3.5. (a)$\sim$(e) are all the intersections of two base regions, (f)$\sim$(g) are all the intersections of three base regions.

obtained following the proposed method. As described in line 6 of Algorithm 1, firstly I obtain the possible intersections between all the base regions. Let $\cap_i^k P_{Bi}$, $(1 \leq i \leq n)$, denote all the k-th order intersections, which are the intersections of k base regions (base regions themselves are regarded as first order intersections), and λ be the largest k, then $\{\cap_i^1 P_{Bi}, \cap_i^2 P_{Bi}, \ldots, \cap_i^\lambda P_{Bi}\}$ represents the set of all the possible intersections. However, practically the mobile manipulator is not able to accurately arrive at the planned positions. The positioning error is the result of numerous influencing factors, including map accuracy, sensor accuracy, environmental complexity, difficulties of controlling the nonholonomic base, and the performance of the mechanical system. Therefore, the positioning error is assumed to be random and homogeneous in different directions. Let the average base positioning error be $\bar{\sigma}$(m), the mobile manipulator is most likely to arrive at a position $\bar{\sigma}$(m) away from the planned position. In some base positions close to the boundary, the mobile manipulator may fail to reach all the target objects in the tray when positioning error is imposed. The robustness with respect to the base positioning uncertainty increases with the distance to the boundary of the base region of the intersection. As a result, the most robust base position is specified by the center of the inscribed circle of the intersection. If the radius of the inscribed circle of an intersection is smaller than the base positioning uncertainty, then it is regarded as not robust and removed from the set of possible intersections. This is corresponding to line 7 to line 10 in Algorithm 1. Notice that, the overall operation time is hardly influenced by choosing different positions within the base region or intersection, due to their limited area, thus the robustness is given much higher priority in this stage without sacrificing much performance.

The uncertainty of the orientation is not considered here, because it depends on the model of the mobile manipulator. For some of the mobile manipulators, the uncertainty of the orientation does have an influence on the result. For example, in the Fetch robot [108] used in our experiment, the joint connecting the torso and the manipulator is constrained by the existence of the torso, and the uncertainty of the orientation will affect the reachable space of the manipulator. However, as stated in Section 3.4-B, in some mobile manipulators, the joint connecting the manipulator and mobile base rotates around a vertical axis. The joint motion can

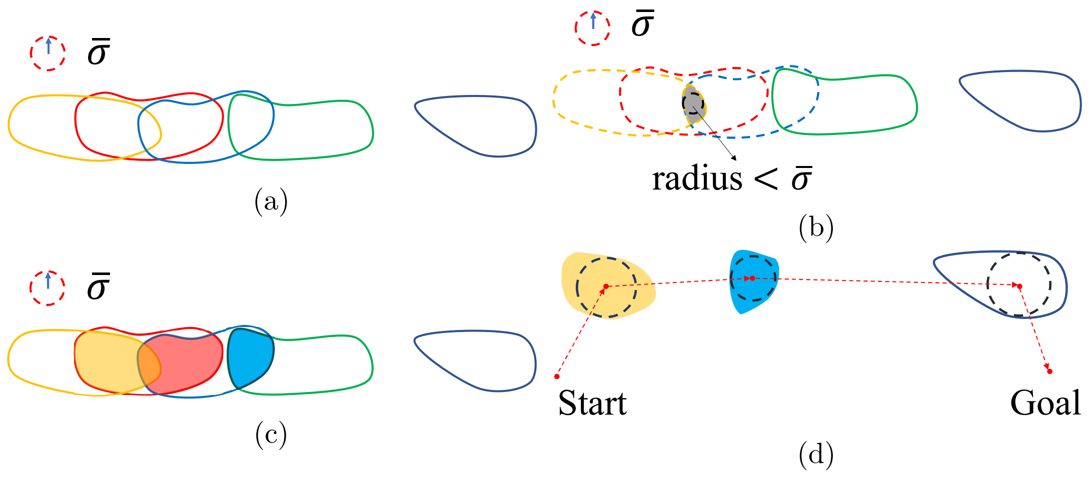


Figure 3.9: The procedure of path planning. Every circle in the subfigures represents a base region for a tray. (a) From left to right are five base regions $P_{B1} \sim P_{B5}$ for 5 trays. (b) $InscribedRadius(P_{B1} \cap P_{B2} \cap P_{B3}) < \bar{\sigma}$, thus discarded. (c) Three feasible second-order intersections are filled with different colors, and five base regions themselves are first-order intersections. (d) The planned intersections $P_{B1} \cap P_{B2}$, $P_{B3} \cap P_{B4}$ and P_{B5} are connected by the shortest path.

completely offset the rotation of the base, so the uncertainty of the orientation does not change the reachable space of the manipulator. For the case where the orientation does have an influence on the result, I can simply rotate the base in place to correct the orientation error, which can be obtained by detecting the marker attached in the environment. Rotation in place is very easy to accomplish for a differential-drive mobile robot.

3.4.3 Path Planning

The function $PlanMinSequence(robust\_intersections)$ in line 11 of Algorithm 1 takes N robust intersections as the input, and plans the minimal subset of the input with size m that visit all the target trays. This is equivalent to the problem of assigning $(N - m)$ zeros and m ones to a base sequence vector $[x_1, x_2, \ldots, x_N]^T$ and minimizing the sum of its elements, which is formulated as:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{N} x_i \\ \text{subject to} \quad & x_i = \{0,1\} \\ & \textstyle\sum_{i=1}^{N} a_{1i} x_i \geq 1 \\ & \textstyle\sum_{i=1}^{N} a_{2i} x_i \geq 1 \\ & \dots \\ & \textstyle\sum_{i=1}^{N} a_{ni} x_i \geq 1 \end{aligned} \tag{3.1}$$

Here, $a_{si}, s = \{1, 2, \dots, n\}$, is 1 if P_{Bs} is reached by the intersection, and $x_i = 1$ if the robot moves to the corresponding intersection. This is the 0-1 knapsack problem, which can be solved by the branch-and-bound method [115]. Finally, I search for the shortest path that connects the start and goal positions, via the centers of the obtained m intersections.

For the purpose of illustrating the intersections of two or more base regions, I use a simple example where there are 5 base regions of 5 trays, as shown in Fig. 3.9. One of them is a third-order intersection and four of them are second-order intersections. Notice that, as shown in Fig. 3.9b, this third-order intersection is also a second-order intersection, i.e., $P_{B1} \cap P_{B2} \cap P_{B3} = P_{B1} \cap P_{B3}$. However, the radius of the inscribed circle of the third-order intersection is smaller than the base positioning uncertainty, i.e., $\mathit{InscribedRadius}(P_{B1} \cap P_{B2} \cap P_{B3}) < \bar{\sigma}$, thus it is removed from the total set of intersections. From the remaining 8 intersections (3 second-order intersections and 5 first-order intersections), three of them are planned to reach all the trays. Then I perform a brute-force search for the shortest path. If the sequence size becomes too large for searching, the shortest path can be approximated by the SA method [116].

3.5 Dynamically Update the Base Positions

The algorithms presented in Section 3.4, explained planning a sequence of base positions for a given task, defined by the grasps for the target objects. In this section, I dive deeper into the tasks in practical application scenarios. The most straightforward task is grasping all the objects in the tray, then the planned base positions are feasible for the mobile manipulator to grasp **all** the objects in the tray. Even though the number of objects decreases as the pick-and-place tasks proceed, the base positions remain valid no matter how many objects are left in the tray. This assumption is not necessarily appropriate as the objects are picked away gradually. It is possible to dynamically update the base positions according to the remaining objects, such that the base region becomes larger with the decreasing remaining objects, and the robustness with respect to base positioning uncertainty is improved. Therefore, I investigate the feasibility and performance of dynamically updating the base regions, which depends on whether the update can be performed in the online phase. The feasibility of updating the base regions online is influenced by the object placement in the trays. Therefore, I have to consider the object placement styles in the tray, including the following two situations: (1) The objects are regularly placed in the trays, where the poses of objects with respect to the tray are known; (2) The objects are randomly placed in the trays, where the poses of objects with respect to the tray are random and unknown. Both of these situations are common in the manufacturing environment.

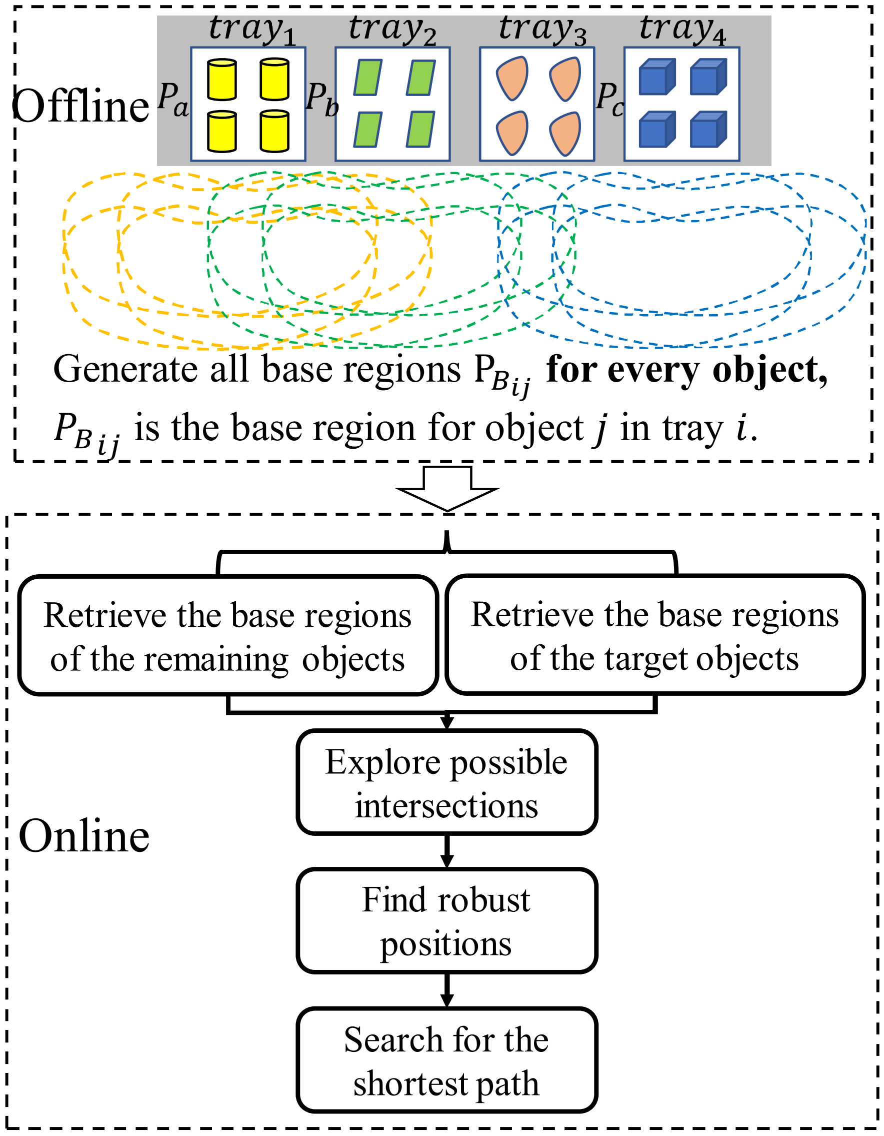

$tray_1$
$tray_2$
$tray_3$
$tray_4$
Offline
P_a
P_b
P_c
Generate all base regions $P_{B_{ij}}$ **for every object,**
$P_{B_{ij}}$ is the base region for object j in tray i.
Online
Retrieve the base regions of the remaining objects
Retrieve the base regions of the target objects
Explore possible intersections
Find robust positions
Search for the shortest path

Figure 3.10: Workflow of two policies for updating the base positions for picking regularly placed objects in the trays, two policies differ in the base regions to be retrieved in the online phase.

3.5.1 Objects Regularly Placed in the Trays

One policy for updating the base positions is based on the remaining objects. The update is performed after every round of pick-and-place tasks, using the current remaining objects in the tray as the target. Intuitively, the size of the base region increases as the task proceeds. The workflow is described in Fig. 3.10. Since the objects are regularly placed in the tray with known poses, the base region P_{Bij} for object $\mathcal{O}_j$ in $tray_i$ can be calculated in the offline phase for all the objects. In the online phase, firstly I determine the remaining objects in the tray, by either remembering which object has been picked or using a camera to extract the configuration of the remaining objects. Then the base regions for the remaining objects can be retrieved from the offline database. The retrieved base regions are further processed to explore the possible intersections, iteratively find robust positions, as well as searching for the shortest path.

Another policy for updating the base positions is based on the target objects to be picked. This is motivated by the scenario where the mobile manipulator is requested to pick up a certain amount of objects in every round of pick-and-place task. Therefore, the base region can be calculated from the target objects to be picked, instead of all the objects or all the remaining objects. For example, if there are m_a objects remaining in a tray with a specific configuration, in a round of pick-and-place task, m_b objects, where $(m_b < m_a)$, should be picked from the tray. I can either exhaustively search for an optimal order of picking objects that achieves an efficient and robust sequence of base positions, or heuristically specify the order of picking objects from the tray. Fig. 3.11 illustrates a simple heuristic. Firstly, I pair the objects in neighboring trays, such that the distance between objects in every pair is nearly constant. Then these pairs of objects take precedence to be

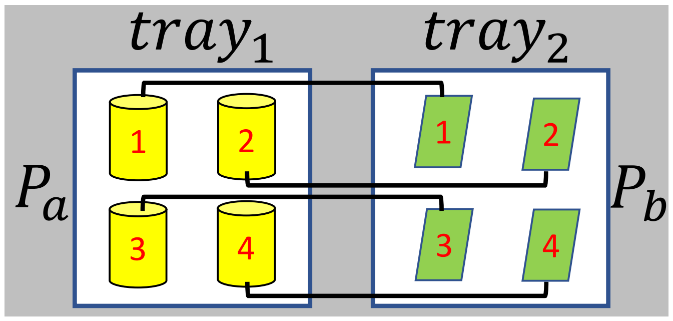


Figure 3.11: Pair the objects in neighboring trays and keep the distance between the objects in a pair nearly constant, a pair of objects are labeled with the same number and connected by a black line. Picking objects following such pairing reduces the overall variance of the robustness in different rounds.

picked when the mobile manipulator has to pick up objects from two neighboring trays. By doing so, the robustness is more consistent in different rounds of pick-and-place tasks. Because the distance between objects does not change much, so does the size of the intersection of their base regions.

Through dynamically re-planning the base sequence for the remaining objects or the target objects to be picked, the robustness with respect to base positioning uncertainties can be improved. The overall efficiency is also likely to be improved, since the base region becomes larger, which may result in more intersections or higher-order intersections among the base regions.

3.5.2 Objects Randomly Placed in the Trays

If the objects are randomly placed in the tray, it becomes infeasible to obtain the base region for grasping an individual object. However, I can treat the objects in the tray as a whole and estimate the total grasps using the method described in Section 3.7.2, and then the base region of the tray can be planned using the estimated grasps. Similarly, the grasps for a tray where there are different amounts

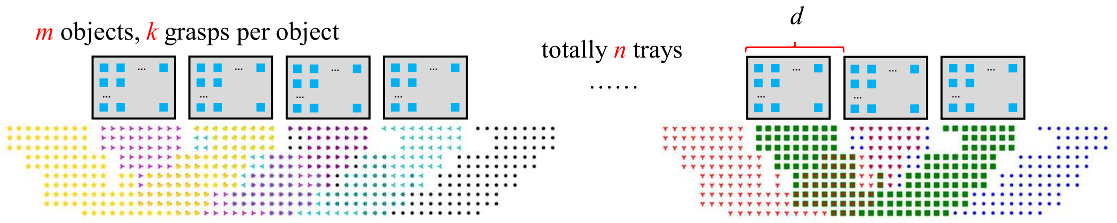


Figure 3.12: Setup of numerical analysis. There are totally n trays in a row, m objects in each tray, and every object is provided with k grasps.

of remaining objects can be estimated. For instance, in the offline phase, I can estimate the base regions for the trays where there are 100%, 75%, 50%, 25% of the objects remaining. During the online execution, the amount of remaining objects can be measured by a weighing device, then the base region with a similar amount of remaining objects can be retrieved from the offline database.

If the base sequence updates according to the target objects to be picked, since the base region for an individual object is not available in advance, I have to select the target objects to be grasped and plan the base positions online, which is time-consuming and impractical.

3.6 Numerical Results and Analysis

In this section, I perform numerical analysis on the base sequence planner. I use the Fetch robot [108] with a 7-DOF manipulator mounted on a differential drive mobile base.

Fig. 3.12 illustrates the setup of the trays and objects for the numerical analysis. The numerical analysis is based on regularly placed objects, but the results also apply to randomly placed objects. I assume there are totally n consecutive trays aligned in a row, every tray contains m objects regularly placed at discrete grid points, and every object is provided with k candidate grasps. In Fig. 3.12, the plotted base regions are the results of parameters: $m = 12$ (3×4), $k = 1$, the tray size is 30cm $\times$ 40cm. In all the planning, I use uniformly sampled base positions as shown in Fig. 3.6, and the distance between the discretized base positions is 5cm. Different base regions are colored differently. Since some of the qualitative results of the base region have been presented in [15], here I focus on the quantitative analysis of the planner.

3.6.1 Base Regions and Intersections

Here I assume the task is to grasp all of the objects in the tray. The change of base region with respect to the number of objects in the tray (assume one grasp per object) is shown in Fig. 3.13 and Table 3.1. The number of base positions decreases drastically in the beginning but does not further decrease as the number of objects increases. This is because the size of the base region is mostly determined by the boundary of the grasp poses in the workspace. When the number of objects in the tray is more than 4, the 4 corners of the tray are filled with objects such that the positional boundary of all the possible objects in the tray is defined. Therefore, the base region changes less significantly as the number of objects further increases.

Fig. 3.14 shows the relation between the radius of the inscribed circle of the intersection of two base regions and the distance between two trays (there are 9 $\times$ 12 objects in the tray). From this figure I can determine the maximum distance between the trays such that their intersection is robust. This example shows that when the distance between two trays is less than 0.8m, the intersection is robust (the radius of the intersection is larger than 0.1m, and 0.1m is the position uncertainty of the mobile manipulator which will be introduced later).

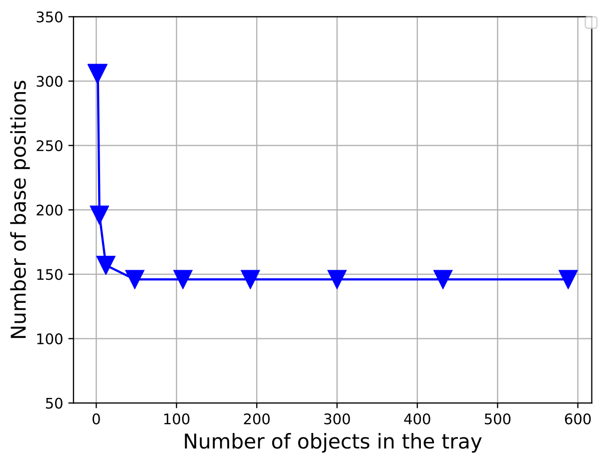


Figure 3.13: The number of planned base positions with respect to the number of objects in the tray. The number of base positions decreases drastically in the beginning but does not further decrease as the number of objects increases.

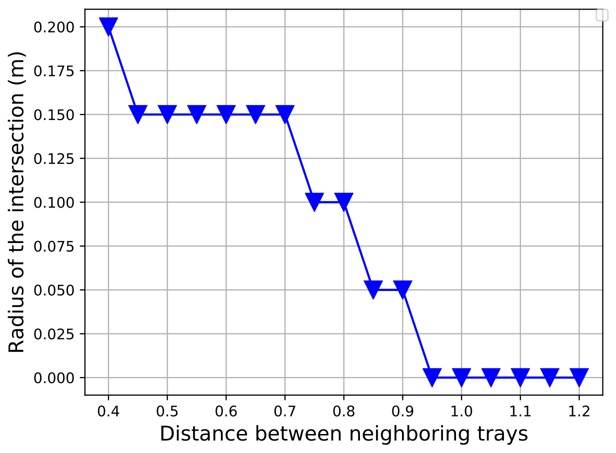


Figure 3.14: The change of radius of the intersection with respect to the distance between the trays. This example shows that when the distance between two trays is less than 0.8m, the intersection is robust. Because when the distance is less than 0.8m, the radius of the intersection is larger than 0.1m, and 0.1m is the position uncertainty of the mobile manipulator which will be introduced later.

Number of objects	Number of base positions	Calculation time (s)
objects 1x1	306	0.22
objects 1x2	306	0.26
objects 2x2	196	0.28
objects 3x4	157	0.51
objects 6x8	146	1.64
objects 9x12	146	3.01
objects 12x16	146	5.10
objects 15x20	146	7.84
objects 18x24	146	11.14
objects 21x28	146	15.41

Table 3.1: Numerical result for a single tray.

3.6.2 Calculation Time

The calculation time for the base sequence planning is mainly spent on 4 parts: (1) generating the base regions for the target trays, (2) exploring the intersections between the base region, (3) determining the robust base positions, and (4) searching for the shortest path. The calculation is implemented in C++ and runs on a laptop with Intel 2.5GHz processors and 16GB of RAM. Typical calculation time for the sub-tasks is listed in Table 3.2.

Table 3.1 shows the result of the calculation time for planning the base region for a single tray. Fig. 3.15 indicates that the calculation time for planning the base region of a tray grows linearly with respect to the number of objects in the tray. Table 3.3 presents the total calculation time of the full planning with respect to the number of trays. As shown in Fig. 3.16, the calculation grows nearly linearly when the number of trays is smaller than 18, then it grows explosively due to the exponential complexity of path searching and exploring the minimal number of base positions.

Sub-tasks	Calculation time (s)
Generate base regions for all objects	30.58
Explore intersections	0.029
Find robust positions	0.226
Search for the shortest path	0.0

Table 3.2: Typical calculation time for different sub-tasks. The result is conducted on 10 neighboring trays, where there are 108 (9×12) objects in every tray, every object is provided with one grasp. Except for the calculation of base regions, all the other tasks can be calculated online.

Number of trays	Sequence size	Calculation time (s)
2	1	6.17
4	2	12.01
6	3	18.40
8	4	24.00
10	5	31.07
12	6	36.57
14	7	42.55
16	8	48.73
18	9	54.94
20	10	67.93
22	11	139.25

Table 3.3: Numerical result for multiple trays (full planning).

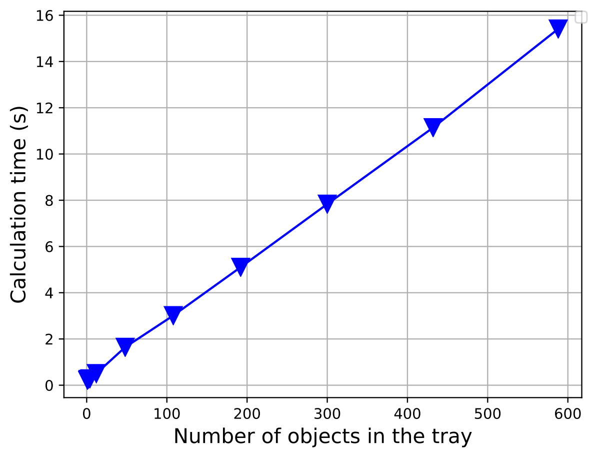


Figure 3.15: Calculation time with respect to the number of objects in a tray. The calculation time for planning the base region of a tray grows linearly with respect to the number of objects in the tray.

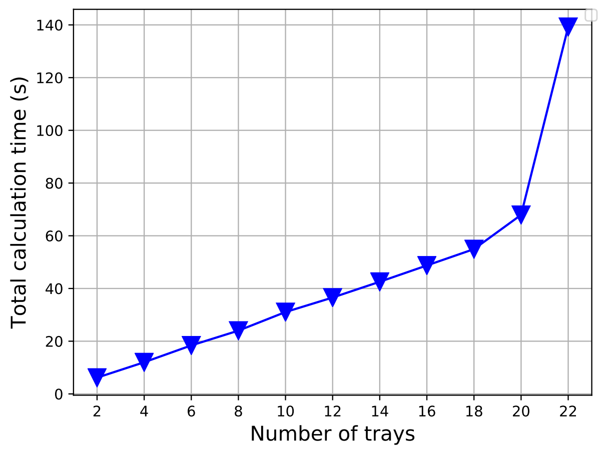


Figure 3.16: Calculation time with respect to the number of trays. The calculation grows nearly linearly when the number of trays is smaller than 18, then it grows explosively due to the exponential complexity of path searching and exploring the minimal number of base positions.

3.6.3 Analysis of Different Policies

I use some numerical examples to evaluate the performance of different policies. As shown in Table 3.2, all the sub-tasks except the calculation of base regions are feasible for online execution. Notice that the complexity of brute-force search for the shortest path is $O(n!)$, where n is the number of planned base positions. However, the calculation can be done within one second as long as the base sequence size is less than 10 (excluding the start and goal positions).

For regularly placed objects, I consider the case where there are 9 consecutive trays in a row, every tray contains 12 objects regularly placed at a 3 by 4 grid, and every object is provided with one candidate grasp. The first policy plans the base positions using the grasps of all the objects. For the other two policies which update the base positions, I assume only one object is picked from a tray in every round of pick-and-place task, and the base positions will be updated after every round of task. The calculated base positions for all the three policies are reliable

Policies	Base positions	Average robustness	Online	Feasibility
Reg/ALL	Reliable	0.1 mm	OK	Yes
Reg/updateR	Reliable	0.155 mm	OK	Yes
Reg/updateT	Reliable	0.186 mm	OK	Yes
Rand/ALL	Reliable	0.1 mm	OK	Yes
Rand/updateR	Unreliable	0.143 mm	OK	No
Rand/updateT	Reliable	0.229 mm	No	No

Table 3.4: Comparison of different policies. In the column of policies, Reg and Rand are the abbreviations of regular and random placement, respectively. updateR and updateT represent updating the base regions based on remaining objects and target objects to be picked, respectively. ALL represents all the objects that fill up a tray, respectively.

since the grasps are identical to the grasps used in the offline calculation. They are also feasible for online execution since base regions are readily available from the offline database. The robustness of different policies are evaluated by the average robustness of all the rounds of the tasks,

$$average\_robustness = \sum_{i=1}^{m} robustness_i/m \tag{3.2}$$

where $robustness_i$ denotes the average robustness of base positions in i-th round and m is the total number of rounds. The robustness of a base position is computed as the radius of the inscribed circle of the corresponding intersection. As a result, the policy that updates the base positions according to the target objects to be picked has the highest average robustness score.

For randomly placed objects, it is possible to estimate all the grasps of all the objects that fill up a tray, and perform the offline calculation of the globally static base positions, which are valid for different rounds of pick-and-place tasks. Then the mobile manipulator can pick any object from the tray in the online phase. However, updating the base positions for randomly placed objects is difficult. In the case of updating based on the remaining objects, the base region with a similar amount of remaining objects is retrieved from the offline database, but the retrieved base region is not reliable. Because the base regions calculated offline assume the randomness of the poses of the placed objects, but the actual picking is usually not performed randomly in terms of the poses of the remaining objects. Instead, the robot picks according to some metrics, such as grasp quality metrics, which may favor specific poses. Therefore, the poses of the remaining objects are not guaranteed to be random. In another word, there are discrepancies between the actual base regions and the offline-generated base regions. Furthermore, if the remaining objects are assumed to be randomly distributed in the tray, then the base regions for the tray with different amounts of remaining objects are theoretically the same. From this perspective, updating the base region is not necessary for randomly placed objects.

On the other hand, updating the base region for randomly placed objects according to the target objects cannot be implemented online. Because the offline-generated base regions are estimated by treating the objects as a whole, while the base region for an individual object is not available, thus the base regions for the target objects have to be calculated online. However, Table 3.2 shows that the calculation of the base region is the most time-consuming sub-task, which involves many IK queries and collision checks. The total calculation time grows linearly with respect to the number of objects. Therefore, dynamically updating the base region for a large number of target objects to be picked may not be practical for randomly placed objects.

Table 3.4 summarizes all the 6 policies. From the above analysis, 4 of them are feasible for practical application. For randomly placed objects, I conclude that a globally static sequence of base positions should be used, without further update. For regularly placed objects, both static and dynamically updated base sequences are applicable. Updating the base positions improves the overall robustness, and the update policy based on the target objects to be picked has the highest average robustness score. However, one of the disadvantages is that, if the picking fails, the robot may have to re-plan and move to another position to try the picking once again. Furthermore, updating the base positions cannot be completed until the mobile manipulator finishes one round of the task, this obstructs the efficient use of multiple mobile manipulators. One has to wait until another mobile manipulator finishes a round of pick-and-place tasks, and then update the base positions and perform the pick-and-place using the updated base positions. But for the globally static base sequence calculated from all the objects, multiple mobile manipulators can cooperate in the tasks efficiently. For instance, when a mobile manipulator finishes picking objects from $tray_1$ and $tray_2$ and is ready to move to the next base position, another mobile manipulator can immediately join the task and move to pick objects from $tray_1$ and $tray_2$. Therefore, despite the overall robustness being outperformed by the policies that update the base positions, it still makes sense to use the offline planning policies without further updates.

3.7 Experiments

I present three sets of experiments to demonstrate the 4 feasible policies, which cover different object placement styles and whether the base positions update or not. The Fetch robot [108], a single arm mobile manipulator equipped with a parallel-jaw gripper, is used to pick objects from multiple trays. There is a Primesense Carmine 1.09 short-range RGBD sensor mounted on the head of the Fetch

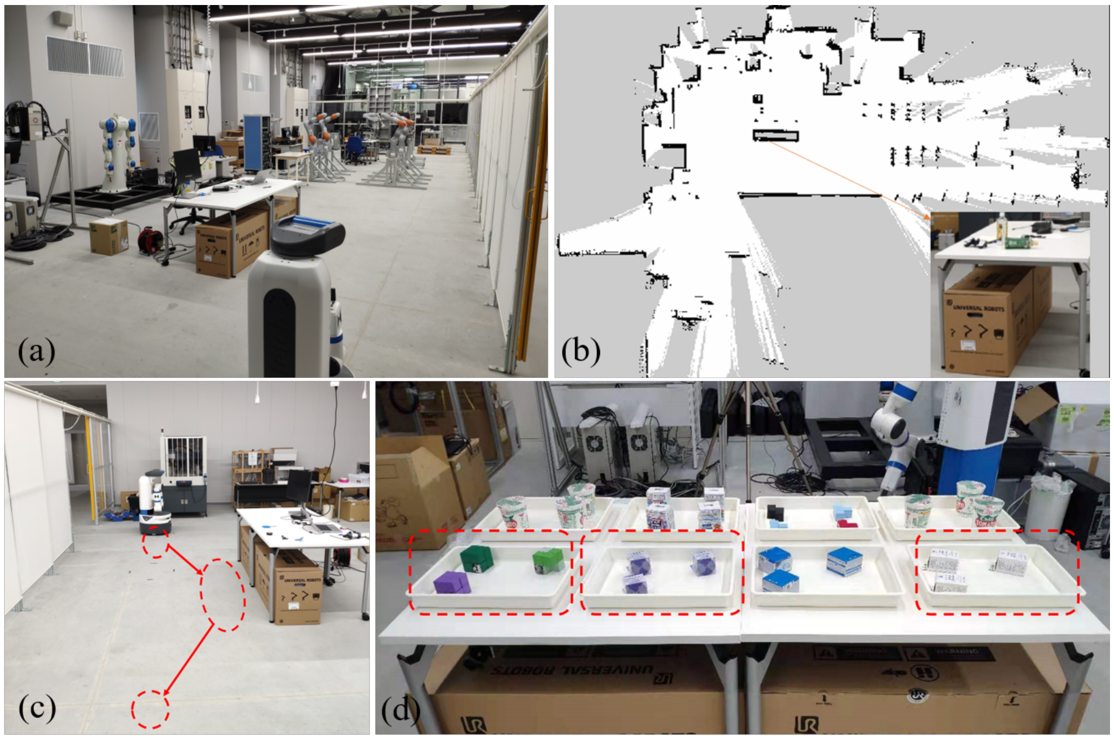


Figure 3.17: Experiment setup: (a) Indoor experimental environment. (b) A 2D map built by the laser scanner. (c) Task overview. (d) The trays containing target objects are marked with red dashed lines.

robot. ROS [117] navigation packages and Moveit! are used to plan and control the motion of the robot. The size of the tray used to store objects is 0.4m × 0.3m × 0.1m. The recorded videos of all the experiments are available in the supplementary material and following link: `https://www.youtube.com/watch?v=JNww18l13dI`. Section 3.7- A and B demonstrate picking regularly and randomly placed objects without updating the base positions, respectively. Section 3.7-C demonstrates picking regularly placed objects, where the base positions update based on the remaining objects or the target objects, after every round of pick-and-place tasks.

3.7.1 Regularly Placed, Globally Static Base Sequence

The mobile manipulator navigates in an indoor environment as shown in Fig. 3.17. It starts from a predefined position in the environment and moves to pick up 3 objects stored in 3 different trays (as circled by the red dashed lines), whose locations in the environment are known. Then the mobile manipulator carries the collected objects to the goal position. The base regions and intersections are calculated by the proposed method. In order to obtain a robust base sequence, the base positioning uncertainty and repeatability are tested by looping the mobile manipulator between two fixed positions. The actually arrived positions are observed to deviate about 10cm in average from the planned positions. Since the base positioning error is the result of map accuracy, sensor accuracy, environmental complexity, difficulties of controlling the nonholonomic base and the performance of the mechanical system, with so many factors involved, it is assumed to be random and homogeneous in all directions. Therefore, the base positioning uncertainty level $\bar{\sigma}$ is set as 10cm. Then a sequence of base positions can be planned by the algorithm described in Section 3.4.3. As a result, the mobile manipulator should successively move to the center of $P_{B1} \cap P_{B2}$ and P_{B4} to collect all the required parts.

When the Fetch robot moves to the calculated position, its head-mounted camera points to the center of the target tray to obtain the point cloud of the objects. I remove the point cloud segments of the table and tray to get the objects' point cloud. For the simple box-shaped objects used in this experiment, the remaining point cloud is fitted with cuboids, such that the object pose can be determined, then the grasping poses are retrieved from the offline planned grasps. A more straightforward way which is used in Section 3.7.3 is to attach a marker in front of the tray, then the object poses, as well as the grasps, with respect to the robot

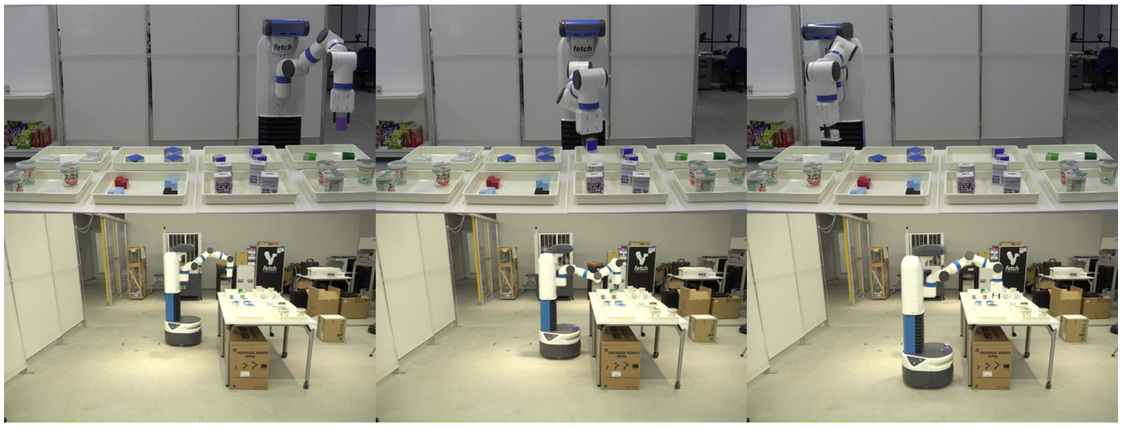


Figure 3.18: (Left) Move to P_{B1} to pick up a part from $tray_1$. (Middle) Move to P_{B2} to pick up a part from $tray_2$. (Right) Move to P_{B4} to pick up a part from $tray_4$.

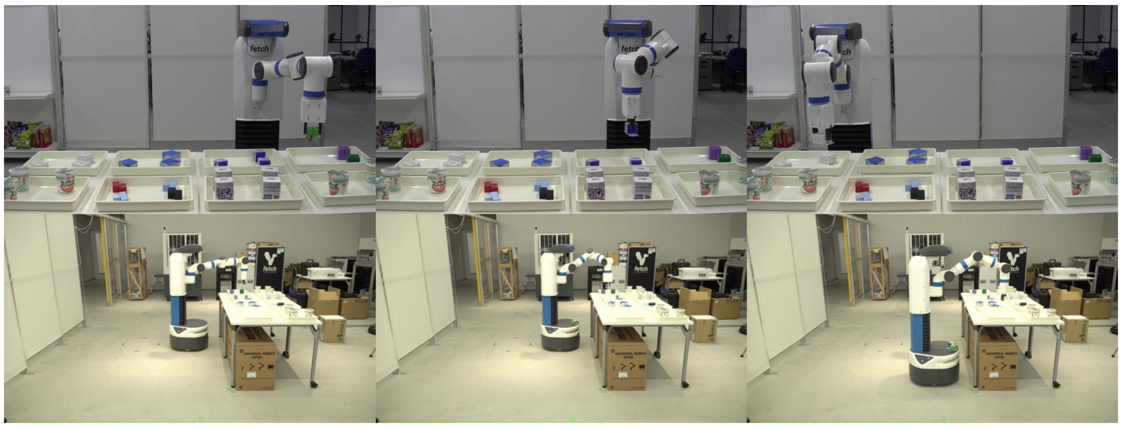


Figure 3.19: (Left and Middle) Move to $P_{B1} \cap P_{B2}$ to pick up parts from $tray_1$ and $tray_2$. (Right) Move to P_{B4} to pick up a part from $tray_4$.

are easily obtained.

For comparison, I move the robot to the center of P_{B1}, P_{B2} and P_{B4} to collect the parts from three trays, respectively. As shown in Fig. 3.18, in each base position, the mobile manipulator picks up one object from the associated tray. Fig. 3.19 shows the robot motion following the planned base sequence, the mobile manipulator moves to the center of $P_{B1} \cap P_{B2}$ to pick up parts from $tray_1$ and $tray_2$, then it moves to the center of P_{B4} to pick up part from $tray_4$. In this experiment, the total operation time is reduced by 17 seconds due to reduced one base movement, and the efficiency of the proposed method becomes more significant when there are a large number of target trays.

3.7.2 Randomly Placed, Globally Static Base Sequence

In the manufacturing environment, the mechanical components are often randomly placed in the tray. In this experiment, I use a mobile manipulator to pick up multiple mechanical components randomly placed in different trays. The overall experiment setup is shown in Fig. 3.20. Compared to the first experiment, a different grasp planning method is used for the mechanical components with complex shapes and reflective surfaces. The head-mounted camera on the Fetch robot can only capture a sparse and incomplete point cloud of the object, therefore I use fixed PhoXi 3D scanners to scan the mechanical components to obtain depth images. Fast graspability evaluation [114] is used to plan grasps for randomly placed objects from a single depth image. The gripper is represented by two mask images as shown in Fig. 3.21a and b, Fig. 3.21a represents the contact region where the gripper should contact the object and Fig. 3.21b represents the collision region where the gripper should avoid collision with the environment. The mask image of the contact region is used to convolve with the object's mask image to find the centroids of grasps, and the mask image of the collision region is applied to find collision-free orientations around the centroid normal.

In order to obtain the base region where the mobile manipulator is able to grasp all the randomly placed objects, I have to find a set of object poses that approximate the possible poses and plan grasps for these poses. Therefore, I randomly place objects in the tray and scan the objects in the tray. Repeat this process a couple of times, then it is assumed that these recorded object poses nearly represent all

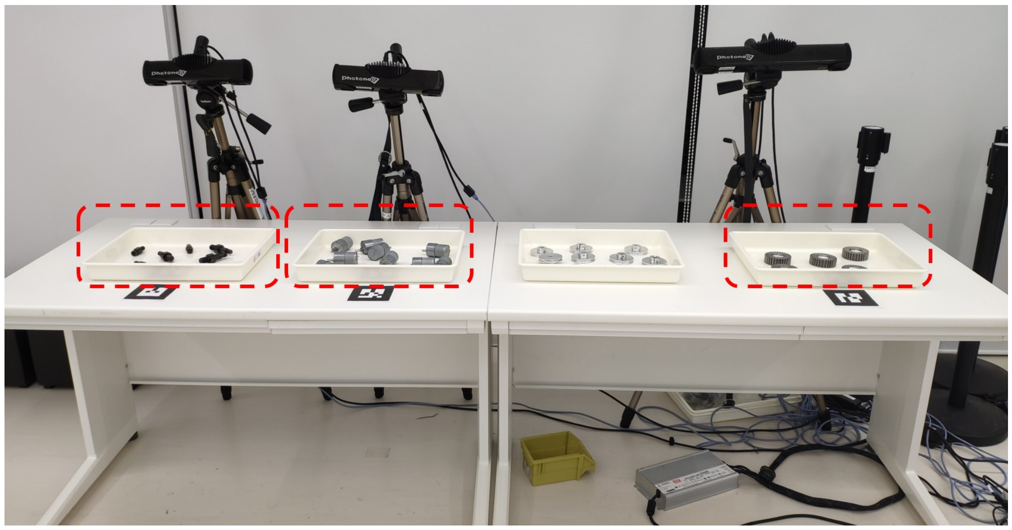


Figure 3.20: Experiment setup: The mechanical components to be grasped are marked with red dashed lines, PhoXi scanners are configured above the target components, and the markers in front of the tray are for obtaining the transform from the tray to the mobile manipulator.

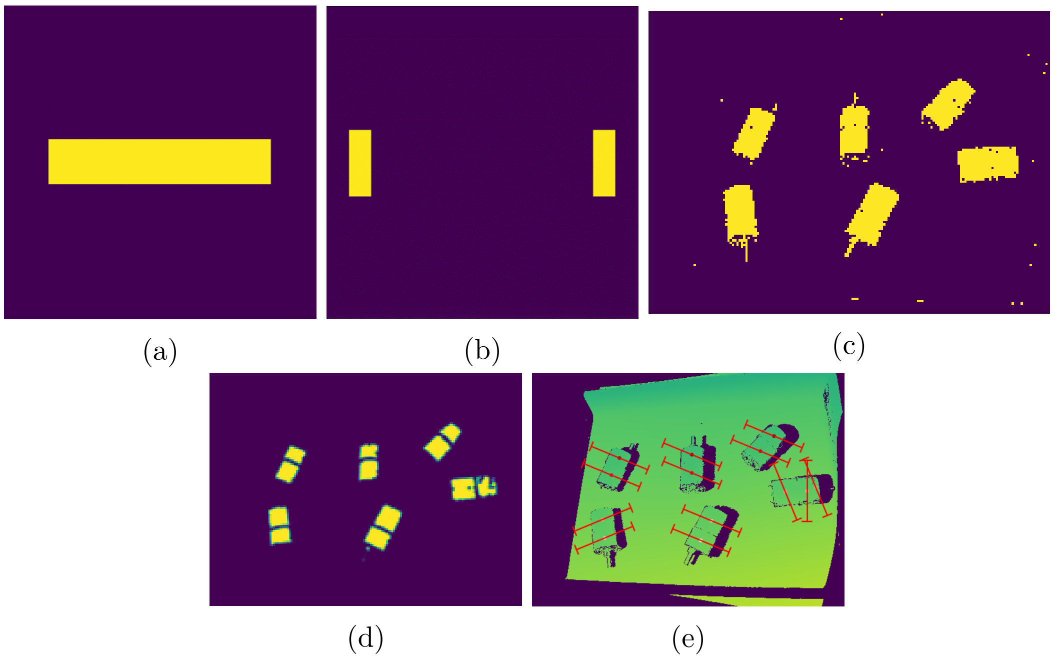
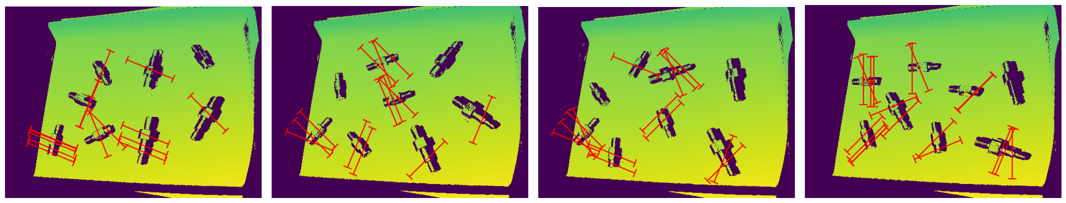


Figure 3.21: Grasp planning method for randomly placed objects. (a) Contact region of the gripper model. (b) Collision region of the gripper model. (c) Mask image of the object. (d) Graspability map. (e) Planned grasps are represented by red lines and plotted on the depth image. Different shades of green represent different depths.



(a) (b) (c) (d)

Figure 3.22: Planned grasps at different object configurations. The union of these grasps is assumed to approximately represent all the possible grasps for grasping all the randomly placed objects, and they are used to estimate the base region.

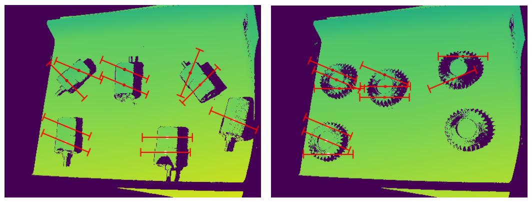


Figure 3.23: Planned grasps for different objects.

the possible object poses. Fig. 3.22 shows 4 different object placements, from the corresponding depth images grasp planning is performed to obtain grasps $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$ and $\mathcal{G}_4$ for each of the object placements, respectively. Actually, for grasping one object, I only need to guarantee that at least one of the grasp candidates is reachable, this is what I did for the case of regularly placed objects. However, for the randomly placed objects, I simply used all the planned grasp candidates to approximate the possible grasp distribution in the tray. The union of the grasps for each object placement $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \cup \mathcal{G}_4$ roughly represents the required grasps for grasping all the objects randomly placed in a tray. Then I use $\mathcal{G}$ to calculate the base region of the tray following the method in Section 3.4. The grasp planning examples for other objects are shown in Fig. 3.23.

During the online execution, the mobile manipulator moves to the calculated positions in the global map, the planned base positions are the black dots in Fig. 3.24. Once the mobile manipulator arrives at the calculated position, its head-mounted camera detects the marker in front of the tray to get the accurate pose of the tray

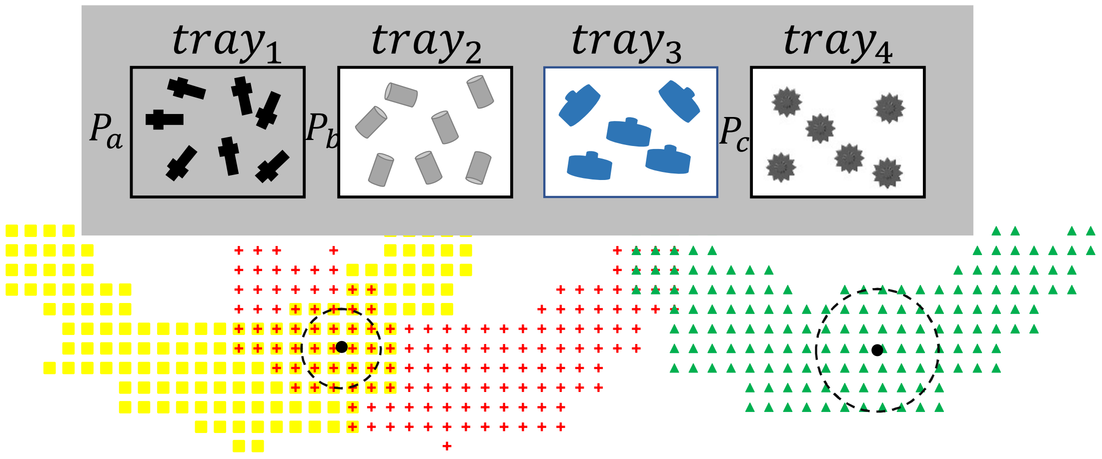


Figure 3.24: Schematic diagram of randomly placed objects in the trays (view from top), the dots of one color represent the base regions of the corresponding tray. The robot moves to the center of the first intersection to pick up the objects in $tray_1$ and $tray_2$, then it moves to the center of the third base region to pick up objects in $tray_4$.

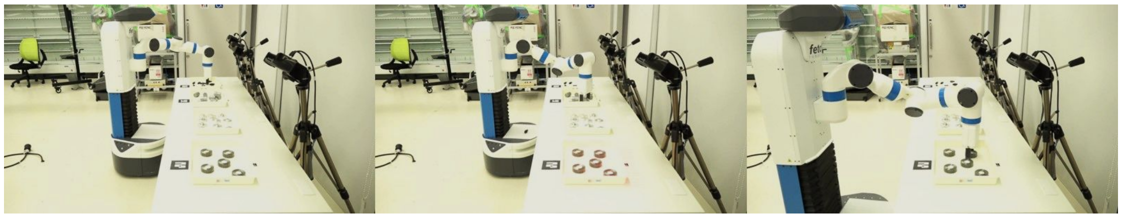


Figure 3.25: Pick up randomly placed objects following the planned globally static positions in Fig. 3.24.

relative to the robot. The online grasp planning functionality is implemented as a ROS service, once it is requested, it returns a set of grasps in the frame of the PhoXi scanner. Since the pose of the PhoXi scanner is also calibrated with respect to the tray, the poses of planned grasps in the robot's base frame can be derived. Among all the returned grasps, the grasp with the highest graspability score will be executed. Fig. 3.25 shows the grasping of randomly placed objects during the experiment.

3.7.3 Regularly Placed, Dynamically Update the Base Sequence

For regularly placed objects in the tray, it is feasible to update the base positions according to either the remaining objects or the target objects to be picked. Both two cases are demonstrated by the experiment in this section. The experimental configuration is similar to that of the first experiment, the difference is that the base positions are updated after every round of pick-and-place tasks. I assume one object is picked from every target tray in every round of the task, then there are totally 4 rounds since every tray contains 4 objects. For randomly placed objects, it is impractical to determine the picking order of objects in advance, but for regularly placed objects, I can either determine which object to pick in the online phase or specify the picking order in the offline phase. In this experiment, I choose to specify the picking order before the pick-and-place tasks begin, such that all the base positions in different rounds of tasks can also be calculated in advance. In the online execution, the base positions are updated after every round of the task.

Fig. 3.26 shows the experiment on updating the base positions based on the target objects to be picked. The picking order is specified using a simple heuristic, as shown in Fig. 3.11. Referring to the same object index as Fig. 3.11, the mobile manipulator picks up object $\mathcal{O}_i$ in $tray_1$ and $tray_2$ and $tray_4$ in the i-th round of the task. From the snapshots of the experiment, though not very obvious, the change of base positions in different rounds of tasks can be observed. Following the dynamically updated base positions, the mobile manipulator can robustly collect the target objects in different rounds of the task. I also conducted the experiment on updating the base positions based on the remaining objects, which can be seen

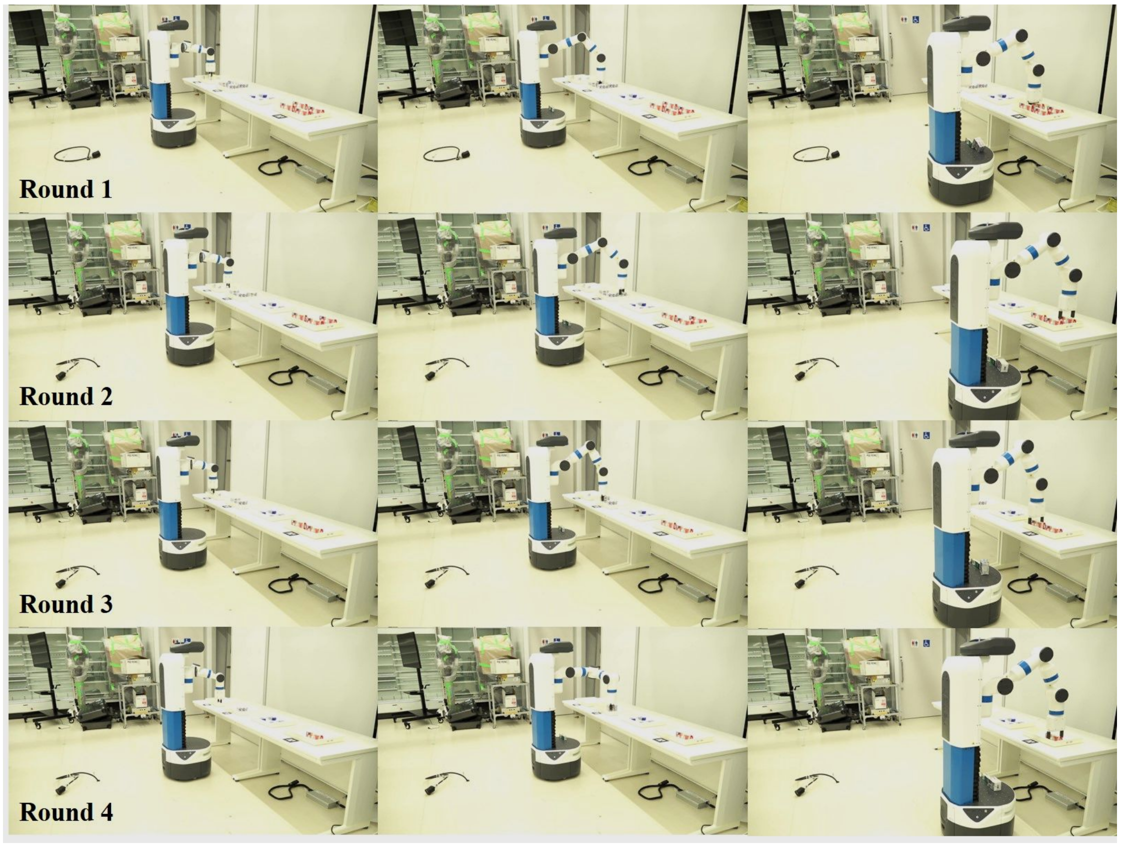


Figure 3.26: The base positions update based on the target objects to be picked, the update is performed after every round of pick-and-place tasks.

in the supplementary video.

3.7.4 Discussion of the Experiment Results

Through the experiments, I have demonstrated the proposed planner with different policies for different application scenarios. To evaluate the efficiency of the proposed method, I use the time for moving the base as the evaluation metric. Because one of the major goals of this work is to reduce the operation time for a sequence of picking tasks at multiple places. In this work, I consider reducing the number of base positions to visit to reduce the operation time for moving the

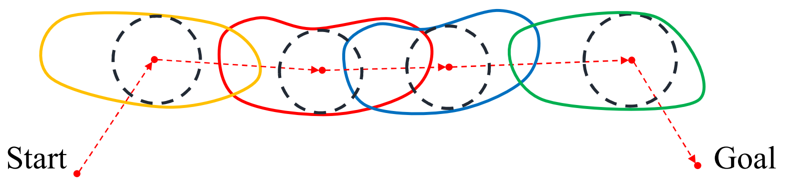
Experiment	Baseline	Planned sequence	Improvement
ExperA	46.94s	29.68s	-17.26 (-36.77%)
ExperB	38.20s	26.98s	-11.22 (-29.37%)

Table 3.5: Average time for moving the base in our experiments. ExperA and ExperB refer to the experiment setup in Section 3.7.1 and Section 3.7.2, respectively.

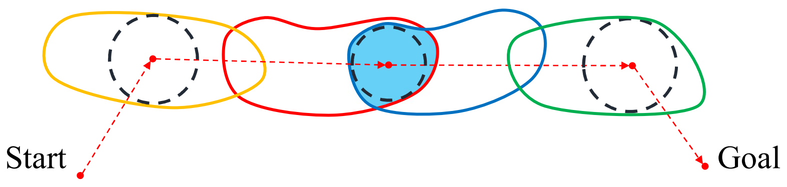
base. Therefore, the time for moving the base is the most appropriate metric for evaluating our planner.

I measure the time for moving the base following the planned base sequence and compare it to the baseline. Since there is no research on planning a minimum sequence of base positions for picking objects stored in separate trays, there are no directly comparable state-of-the-art methods. The existing works usually assume there is a base position or manipulator configuration corresponding to each task, therefore, the baseline for comparison can be set as moving the mobile manipulator to the center of every base region of the target tray. I use the experiment setup in Section 3.7.1 and Section 3.7.2 since their base positions do not change. I run the experiment 10 times and record the time for moving the base. The result is summarized in Table 3.5, the average time for moving the base is reduced by 17.26 seconds and 11.22 seconds in these two experiment setups, respectively. Therefore, the total operation time for the part-supply tasks is reduced and the overall efficiency is improved using the proposed method.

Notice that the length of the path connecting the planned base positions is not an appropriate metric for evaluation. Because the path length does not necessarily decrease with the decrease of the number of base positions. Fig. 3.27 shows a simple example where the path lengths are almost the same, but the number of base positions is different. From another perspective, instead of reducing the path length, our approach is to reduce the number of base positions to visit, which



(a)



(b)

Figure 3.27: (a) The baseline base sequence and path. (b) The planned base sequence and path using our planner. In this example, the path length is almost the same even with the reduced base positions. Therefore, the path length is not an appropriate metric for evaluating the planner.

increases the average base velocity, thus reducing the operation time.

The current limitations are observed through the physical experiments. I acknowledge that the success rate of the experiment is not high (about 50% for a single picking task). But this is the collective result of several factors beyond our planner, such as (1) the perception error of the collision environment, (2) motion planning timeout during the online execution, (3) pose estimation error for the marker and objects, and (4) positioning uncertainty. These extra factors lead to common failures in the experiments: (1) The manipulator collides with the trays or table; (2) The robot fails to grasp the objects; (3) Motion planning for the manipulator to move to the grasping pose times out. Therefore, the success rate may not be an appropriate metric for the evaluation of the proposed planner. In addition, the motion of the manipulator is not optimized. In the future, I consider using optimization-based method motion planning method to further reduce the operation time for moving the manipulator.

CHAPTER 4

WHOLE-BODY MOTION PLANNING FOR DYNAMIC GRASPING

In chapter 3, a minimal sequence of positions is planned for the mobile manipulator to pick up assembly parts from multiple trays. However, I used the mobile manipulator in a decoupled manner, where the mobile manipulator firstly stops the mobile base before performing the manipulation tasks and then moves to the next position or configuration. Obviously, the decoupled motion of the mobile base and the manipulator is not efficient. The operation time can be further reduced by simultaneously moving the base and the manipulator when performing the tasks.

In this chapter, I present an optimization-based motion planner to plan a locally time-optimal whole-body motion of a nonholonomic mobile manipulator, to pick up objects while simultaneously moving the manipulator and the base. The simultaneous motion further reduces the operation time of the picking tasks. What distinguishes our planner from the common motion planners, which plan the motion between two configurations, is that our planner considers performing tasks, such as grasping an object, during the motion. I formulate the time-optimal motion planning as an optimization problem. One of the major difficulties is finding an appropriate representation of the constraints for the tasks during the motion, since the time and configuration of the robot at the moment of performing the task are unknown. To address this issue, I propose a novel formulation of the optimization variables such that constraints arising from the tasks are smooth and differentiable, which is essential for obtaining a feasible solution using a nonlinear optimization solver. I present numerical results of the proposed planner to show that our planner can obtain a feasible trajectory that satisfies all the constraints.

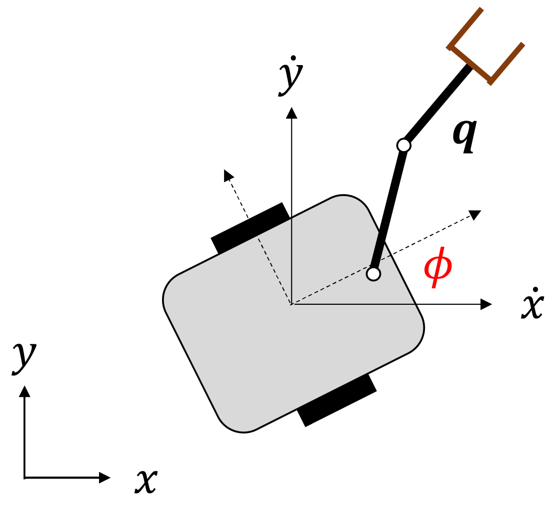


Figure 4.1: The kinematic model of the nonholonomic mobile manipulator. In the numerical calculation, I use the kinematic model of the Fetch robot, which has a 7 DOF manipulator mounted on a diff-drive mobile base.

For real-world implementation, I discussed a robust grasp strategy (push-grasp) for dynamic grasping with whole-body motion. I also demonstrated the planned trajectory in both physical simulation and real-world experiments.

4.1 Method

The goal is to plan a locally **time-optimal** trajectory of a nonholonomic mobile manipulator to reach a grasping pose $\mathcal{G}$ during the motion from an initial configuration $\mathbf{q_i}$ to a final configuration $\mathbf{q_f}$, w.r.t the kinematics and dynamics constraints.

4.1.1 Problem Formulation

This problem can be formulated as the following optimization problem:

$$\min_{\mathbf{x},\Delta T_1,\ldots,\Delta T_{m+1}} \quad \sum_{i=1}^{m} \Delta T_i N_i \tag{4.1}$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X} \tag{4.2}$$

$$\Delta T_j > 0, j \in \{1, 2, ..., m+1\} \tag{4.3}$$

$$F(\mathbf{q}_k) = 0 \tag{4.4}$$

$$Nonholonomic(\mathbf{q}_i, \dot{\mathbf{q}}_i) = 0 \tag{4.5}$$

$$\mathbf{q}_i + (\dot{\mathbf{q}}_i + \dot{\mathbf{q}}_{i+1})\Delta T/2 = \mathbf{q}_{i+1} \tag{4.6}$$

$$\mathbf{a}_l \Delta T \leq \dot{\mathbf{q}}_{i+1} - \dot{\mathbf{q}}_i \leq \mathbf{a}_u \Delta T \tag{4.7}$$

$$\mathbf{q}_1 = \mathbf{q}_{initial}, \dot{\mathbf{q}}_1 = \dot{\mathbf{q}}_{initial} \tag{4.8}$$

$$\mathbf{q}_N = \mathbf{q}_{final}, \dot{\mathbf{q}}_N = \dot{\mathbf{q}}_{final} \tag{4.9}$$

$$\mathbf{q}_i \in \mathcal{C}_{free} \tag{4.10}$$

The trajectory is discretized in the state space, i.e., both the joint configuration and velocity are the optimization variables. Fig. 4.1 illustrates the mobile manipulator considered in this chapter, and its configuration includes the position and orientation of the diff-drive base and the joint position of the 7 DOF manipulator, i.e., $\mathbf{q} = [x, y, \phi, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7]^T$. I use $\mathbf{x}$ to denote the vector of the discretized states, i.e., $\mathbf{x} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N, \dot{\mathbf{q}}_1, \dot{\mathbf{q}}_2, \ldots, \dot{\mathbf{q}}_N]^T$. Note that, I do not use the control inputs of the mobile base (left and right wheel velocities) as optimization variables, instead, I derive the control inputs (u_r, u_l) from the discretized states $(\dot{x}, \dot{y}, \dot{\phi}, \phi)$ of the mobile base according to the following first-order dynamics of the mobile manipulator (equation (11)), and then impose bound constraints on the derived control inputs.

$$\begin{bmatrix} \dot{\mathbf{q}}_{arm} \\ \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}_{arm} \\ \frac{1}{2}(u_r + u_l) r_w cos(\phi) \\ \frac{1}{2}(u_r + u_l) r_w sin(\phi) \\ \frac{(u_r - u_l) r_w}{R_{base}} \end{bmatrix} \tag{4.11}$$

Assume the mobile manipulator has to grasp m objects during the motion, then the trajectory is split into $m + 1$ segments by the tasks, as shown in Fig. 4.2. I heuristically predefine the correspondence between m waypoints and these m tasks. However, I do not restrict the time for performing the tasks and leave it for the optimizer to explore. The time intervals ΔT_i (of i-th segment) between the consecutive waypoints are set as optimization variables, and totally I have $m + 1$ time intervals for $m + 1$ trajectory segments. Therefore, the optimization variables in our formulation includes the time intervals and the discretized state: $[\Delta T_1, \ldots, \Delta T_{m+1}, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N, \dot{\mathbf{q}}_1, \dot{\mathbf{q}}_2, \ldots, \dot{\mathbf{q}}_N]^T$, where $\mathbf{q}_i = [x_i, y_i, \phi_i, \theta_{1i}, \theta_{2i}, \theta_{3i}, \theta_{4i}, \theta_{5i}, \theta_{6i}, \theta_{7i}]^T$ is the configuration of the i-th waypoint.

Since I seek to obtain the time-optimal trajectory, the objective function (equation (1)) is simply the total operation time, which is the sum of the time intervals among all the waypoints, i.e., $J = \sum_{i=1}^{m} \Delta T_i N_i$. Equation (2) is the constraint on the state. There are limits on the manipulator's joint position and velocity. But there are no bounds on the position and orientation of the mobile base. The velocity of the base is limited by the wheel velocity. Therefore, I calculate the corresponding left and right wheel velocities according to $[\dot{x}, \dot{y}, \dot{\phi}, \phi]$, and then impose the bounds on the converted form. Equation (3) ensures the time intervals are positive. Equation (4) denotes the task constraints $F(\mathbf{q}_k) = 0$ on the k-th waypoint, specifically for a grasping task, I transform the robot configuration at k-th waypoint to a grasping pose $\mathcal{G}_i$: $FK(\mathbf{q}_k) = \mathcal{G}_i$. If there are m tasks during the motion,

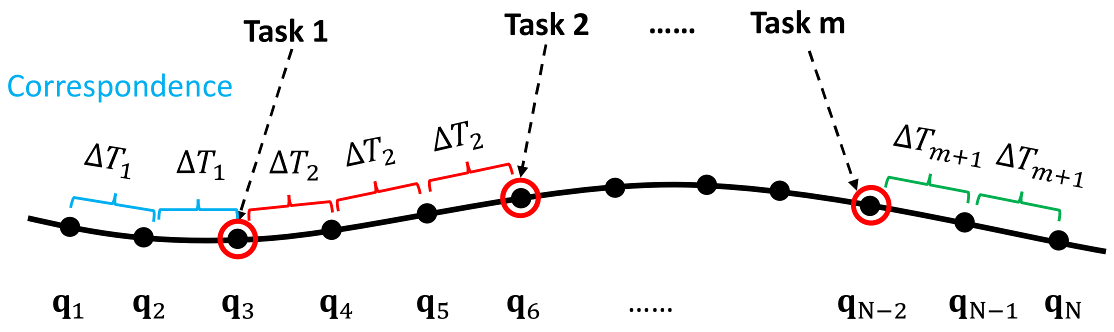


Figure 4.2: Illustration of the correspondence of the states and tasks. Assume the mobile manipulator has to grasp m objects during the motion, then the trajectory is split into $m+1$ segments by the tasks. I heuristically predefine the correspondence between m waypoints and these m tasks, however, I do not restrict the time for performing the tasks and leave it for the optimizer to explore. The time intervals ΔT_i (of i-th segment) between the consecutive waypoints are set as optimization variables, and totally I have $m+1$ time intervals as optimization variables for $m+1$ trajectory segments.

I should heuristically select m waypoints and impose the task constraints on these waypoints. Equations (5) to (10) impose constraints on the waypoints. Equation (5) describes the nonholonomic constraints at all the waypoints, specifically for the diff-drive mobile base as shown in Fig. 4.1, the nonholonomic constraint for i-th waypoint is written as $\dot{x}_i sin(\phi_i) - \dot{y}_i cos(\phi_i) = 0$. Equation (6) also enforces the consecutive waypoints to follow the constraint implied by velocity, here I use the average velocities of i-th and (i+1)-th waypoints. Equation (7) limits the maximum and minimum accelerations[1]. In equation (7), ΔT_j of the time interval depends on which segment of the trajectory it belongs to. For example, for the 4-th and 5-th waypoint shown in Fig. 4.2, the ΔT_j is ΔT_2. Equations (8) and (9) specify the initial and final positions and velocities of the motion planning problem. Finally, equation (10) enforces that the mobile manipulator does not collide with itself and the obstacles. I calculate the distances between the robot links and the obstacles and the distances between robot links, and then enforce the distances are positive.

[1]The official documentation of the Fetch robot only provides the torque limits of the manipulator's joints. To simply the problem, I manually specify the acceleration bounds.

4.2 Numerical Results

In this section, I present some numerical results of the proposed motion planner. I consider planning the time-optimal motion for grasping one object during the motion. For the initialization of the variables, I focus on obtaining a good estimation of the time intervals. Firstly I solve the problem with a small number of waypoints, after several trials with different initial seeds and a different number of waypoints, the rough range of optimal operation time is obtained. Once I have a reasonable guess of the total operation time, the time intervals of different trajectory segments are calculated accordingly. For the initialization of the state of the waypoints, I linearly interpolate the positions between the initial and final configurations to get the positions of the waypoints, and then use the time intervals to initialize the velocities.

I implemented the optimization problem using the nonlinear optimization package NLopt [118], the optimizer that I use the SLSQP (Sequential Least SQuares Programming) solver, a local derivative-based method. Fig. 4.3 shows the time-optimal trajectory to grasp an object with $\mathcal{G} = [5.5, -2, 0.5, 1, 1, 0]^T$ during the motion from the initial configuration $\mathbf{q}_1 = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$ to the goal configuration $\mathbf{q}_N = [10, 0, -1, 1.13, 0.3, 1.9, 1.1, 2.1, 0.5, 2.5]^T$. In this example, I heuristically set the waypoint in the middle of the trajectory to correspond to the task. The trajectory is smooth and satisfies all the constraints. The violation of nonholonomic constraints is plotted in Fig. 4.4, the violation is in the order of 10^{-5} and is negligible, and it can be further reduced by setting a lower tolerance for the constraints.

By solving the optimization problem with a different number of waypoints, I verify the independence of the solutions in the resolution of the discretization. The code

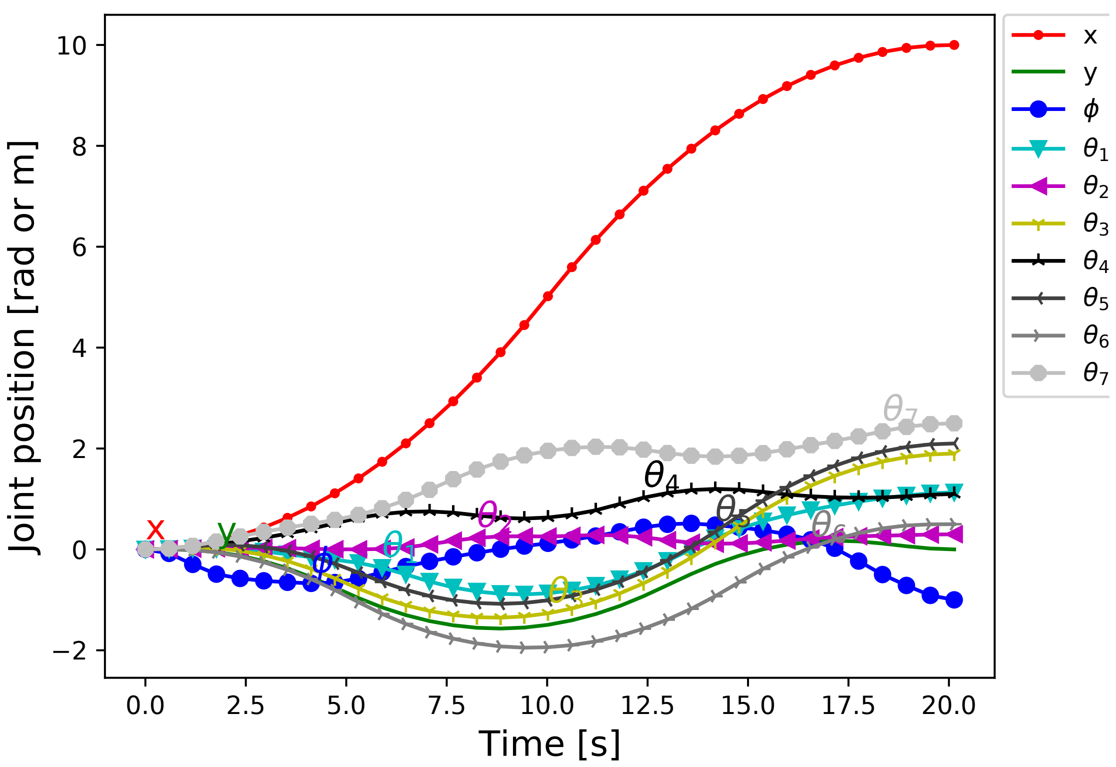


Figure 4.3: The time-optimal trajectory of all the 10 joints. This solution is the result of discretizing the trajectory into 35 waypoints. $\Delta T_1 = 0.589s, \Delta T_2 = 0.595s$, and the total operation time is 20.13s.

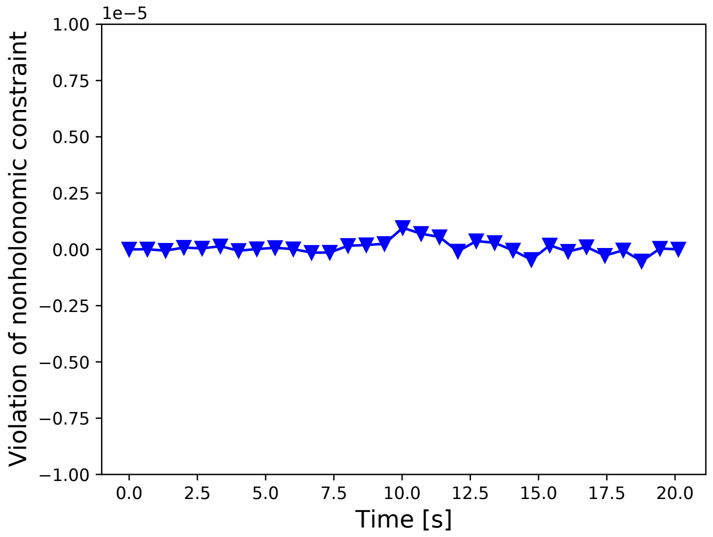


Figure 4.4: The violation of nonholonomic constraints along the trajectory. This figure shows that the nonholonomic constraint is well satisfied.

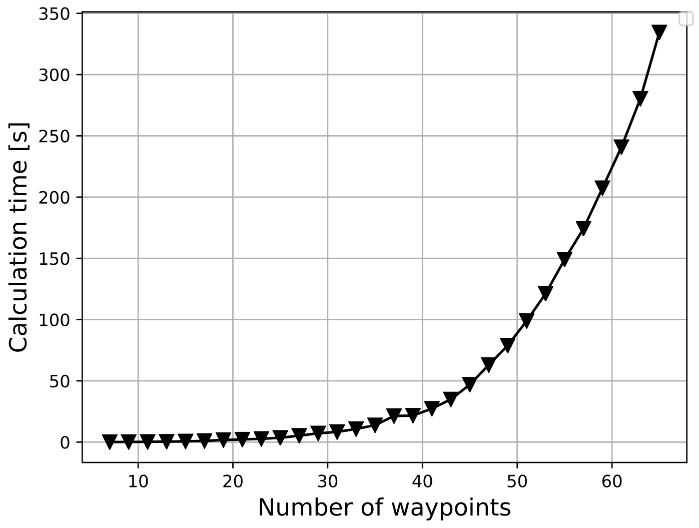


Figure 4.5: Calculation time with respect to the number of discretized waypoints.

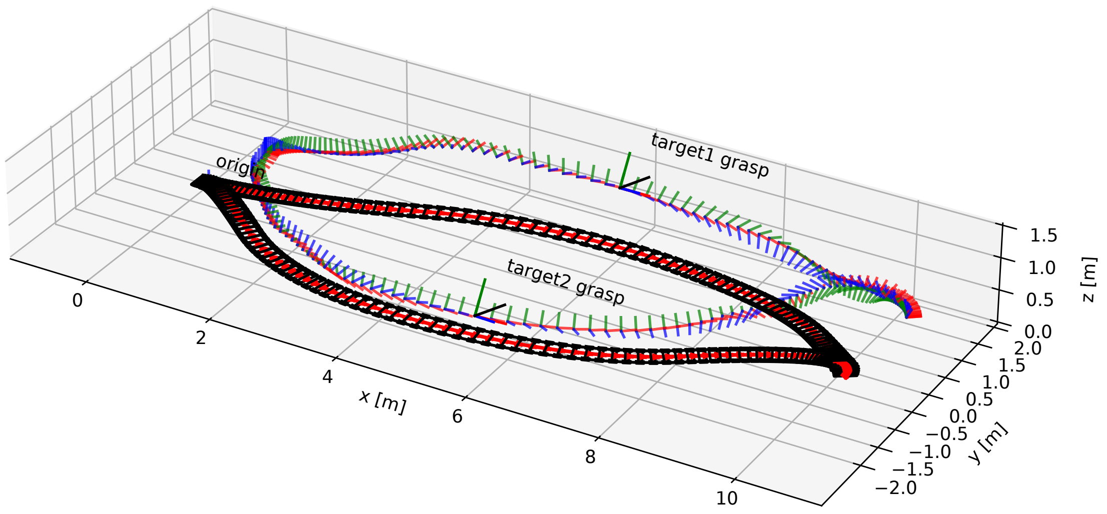


Figure 4.6: Visualization of the end-effector and the base trajectory with two different grasping tasks, $\mathcal{G}_1 = [5.5, 2, 0.5, 1, 1, 0]^T, \mathcal{G}_2 = [5.5, -2, 0.5, 1, 1, 0]^T$.

is programmed in C++ and runs on a laptop with Intel 2.5GHz processors and 16GB of RAM. Fig. 4.6 visualizes two trajectories of the end effector and the base with two different grasping tasks. The end effector passes through the target grasping poses during the motion.

4.3 Strategy for Robust Grasping

Till now, I have introduced planning the time-optimal trajectory for the mobile manipulator to reach a target grasping pose during the motion. However, grasping an object during the motion is fragile. A more robust grasping strategy is required for such dynamic grasping.

Thakar et al. [17] studied dynamic grasping with pose estimation uncertainty, but they employed zero gripper velocity with respect to the object [18], because lower speed was considered to be more robust for grasping with uncertainty in pose estimation of the object. However, zero gripper velocity restricts the overall efficiency of mobile manipulation. Since one of the major purposes is to reduce the operation time, I do not assume zero gripper velocity relative to the target object. In this section, I propose to use push-grasp for the dynamic grasping, which improves the robustness with respect to the positioning uncertainty of the object or the mobile manipulator, and at the same time, it is more efficient compared to the zero-velocity grasp since the gripper can move with a nonzero velocity.

I assume the gripper closes with a limited velocity. As shown in Fig. 4.7(b), if not taking the push-grasp strategy, in order to contact the object with both fingers of the gripper at the planned timing for grasping, the gripper has to start to close in advance. Since the gripper closing velocity is usually low, there is a very narrow space between the fingers and the object when the object starts to enter the center of the two fingers. In contrast, push-grasp is able to grasp the object more robustly as shown in Fig. 4.7(a). The gripper does not close until the object is within the coverage of two fingers of the gripper, which increase the free space between the fingers and the target object. Once the object is within the coverage of the two fingers, the object will slide on the table as the gripper moves and closes, and

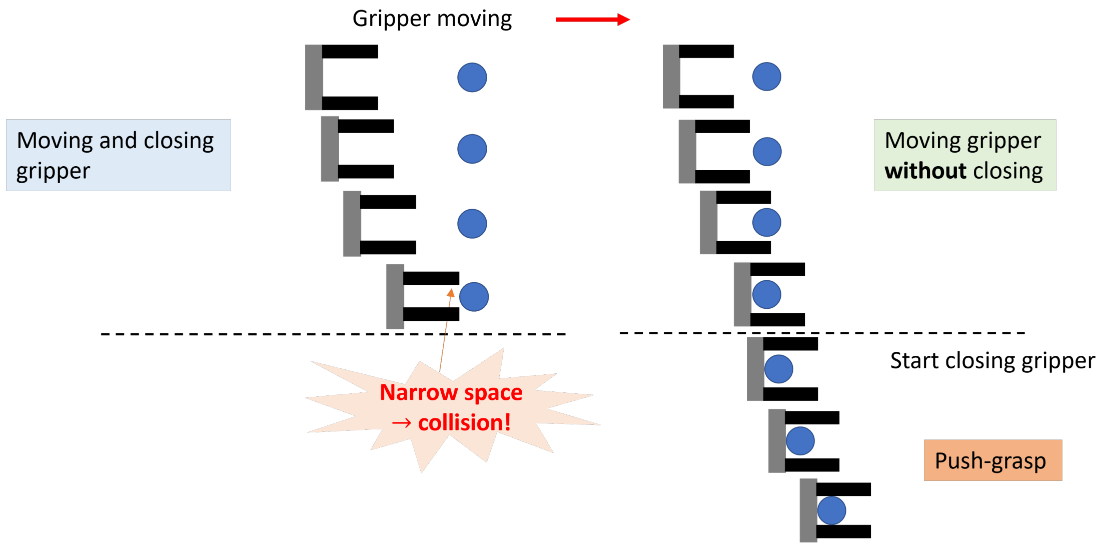

Gripper moving
Moving and closing gripper
Moving gripper **without** closing
Narrow space → collision!
Start closing gripper
Push-grasp

Figure 4.7: (Left) Closing the fingers and moving the gripper at the same time. This will lead to narrow space between the fingers and the object, and as a result, the gripper is prone to collide with the object. (Right) Push-grasp. The fingers do not close until the object contacts the palm of the gripper, or the object is within the coverage of the fingers.

finally the object will be grasped as the two fingers contact the object.

Note that I assume that once the object contacts the palm of the gripper, it will remain in contact with the gripper. However, if the gripper moves at a high speed and hits the object, the object will detach the gripper after the impact. I may impose an additional velocity limit on the gripper when reaching the target grasping pose to avoid this issue.

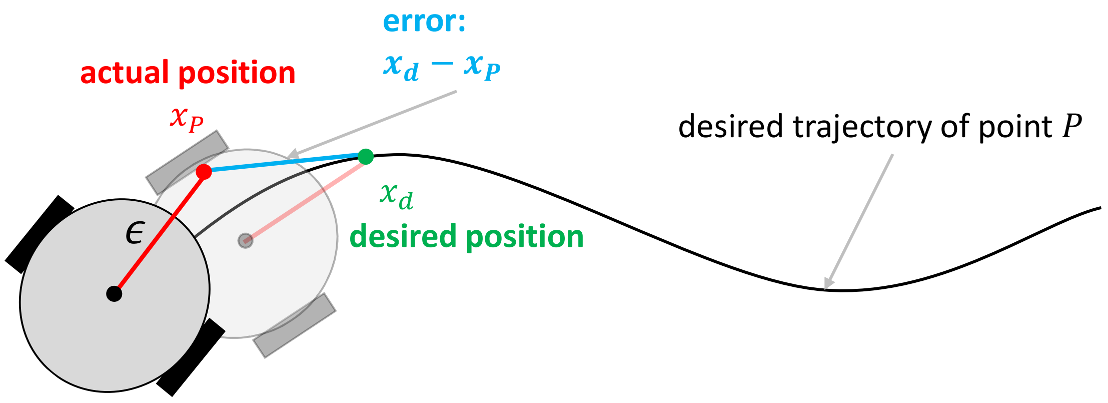


Figure 4.8: Illustration of the base trajectory tracking method. A PID controller is used to correct the base trajectory error. Since the mobile base is subject to differential constraints, the PID controller is used to track a holonomic point P relative to the mobile base.

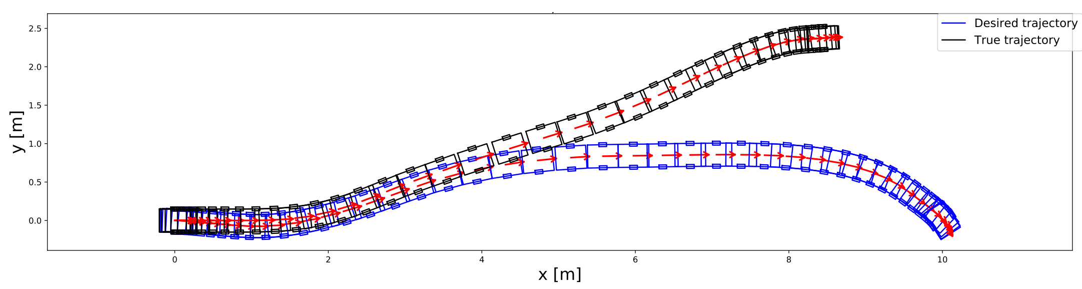
4.4 Experiment

4.4.1 Experiment Setup

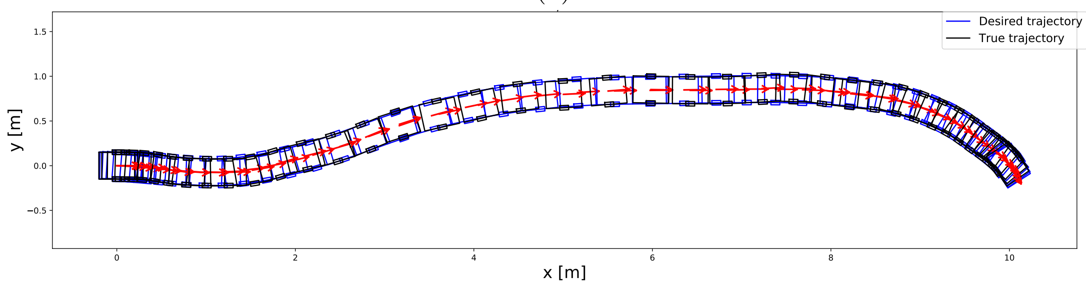
I use a Fetch mobile manipulator with a 7DOF manipulator mounted on a diff-drive mobile base, for both physical simulation and real-world experiments. The equipped gripper is a parallel two-finger electric gripper with a constant closing speed of 0.1m/s. The mobile manipulator accepts joint velocity input for the manipulator and the twist for the mobile base. The task is to grasp an object from a table using push-grasp, as shown in Fig. 4.11.

4.4.2 Physical Simulation

In the implementation of the planned trajectory, the manipulator trajectory tracking is relatively accurate, and the error mainly comes from the mobile base. I use



(a)



(b)

Figure 4.9: The base trajectory tracking performance. (a) Open-loop execution without feedback control. (b) With a PID controller, the control parameters are $K_p = 2.0, K_i = 0.1, K_d = 0.3$. Using a PID controller greatly reduces the base trajectory tracking error.

a simple PID controller to track the base trajectory. The mobile base is subject to nonholonomic constraints, which restrict the feasible velocity. To track the calculated trajectory, I use the method described in [45] and track a holonomic point P relative to the mobile base (as illustrated in Fig. 4.8). The relation between the position of P and the position of the mobile base is as follows:

$$x_P = x + \epsilon cos\phi \tag{4.12}$$

$$y_P = y + \epsilon sin\phi \tag{4.13}$$

$$\dot{\mathbf{x}}_P(t) = K_P\mathbf{e}(t) + K_i\int_0^t \mathbf{e}(\tau)d\tau + K_d\frac{d\mathbf{e}(t)}{dt} \tag{4.14}$$

where $[x, y]^T$ is the position of the mobile base, $\mathbf{x}_P = [x_p, y_p]^T$ is the position of the holonomic point P to be tracked by the PID controller, ϵ is the length of the imaginary rod connecting the mobile base and P, and the tracking error $\mathbf{e}(t)$ is defined as the difference between the desired position $\mathbf{x_d}$ and actual position $\mathbf{x_P}$:

$$\mathbf{e}(t) = \mathbf{x_d} - \mathbf{x_P} \tag{4.15}$$

The desired position $\mathbf{x_d}$ is calculated according to the planning base trajectory using equations (12) and (13).

Fig. 4.9(a) shows the tracking performance of the base trajectory using the P controller without feedback. The deviation between the actual base trajectory and the desired trajectory accumulates as the robot moves. By using the tuned PID controller, the base trajectory tracking performance is greatly improved, as shown in Fig. 4.9(b). Fig. 4.10 shows the executed trajectory in the Gazebo simulation. In Fig. 4.10-(5), -(6) and -(7), the gripper pushes the object while closing fingers, and the object slides on the table under the push-grasp.

4.4.3 Real-world Experiment

I implemented the trajectory on a Fetch mobile manipulator. The task is to use push-grasp to pick up a bottle from a table, as shown in Fig. 4.11. Since the onboard perception during the mobile manipulation is a challenging problem itself,

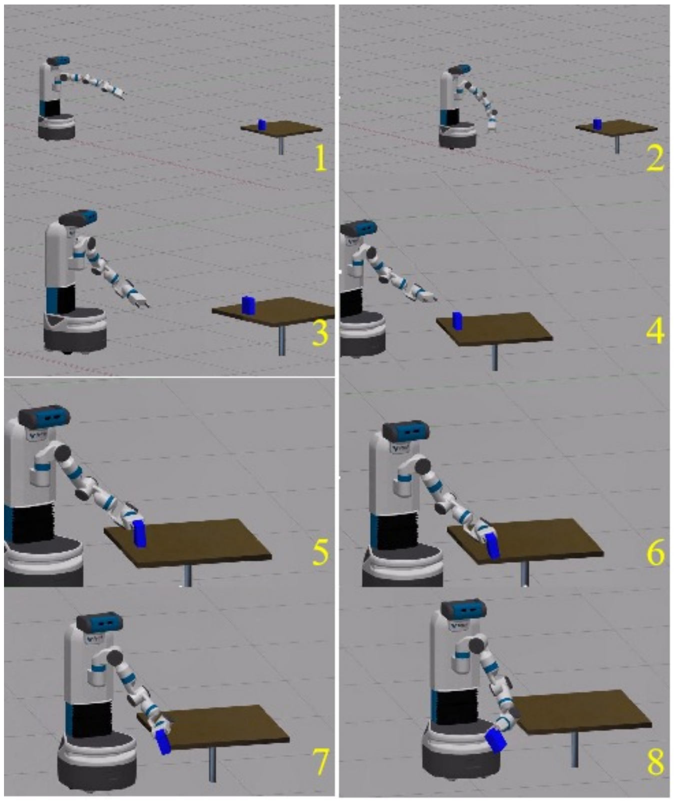


Figure 4.10: The simulation snapshots of the trajectory for picking up an object during the whole-body motion.

here I only demonstrate the open-loop execution of the trajectory. I have shortened the total trajectory to reduce the accumulated position error from the mobile base. For safe execution, I also scale down the overall velocity of the mobile manipulator to 1/5 of the planned velocity. Fig. 4.12 shows three executed trajectories. Once the gripper contacts the object, the object slides on the table under the push-grasp, and the object is finally grasped as the gripper closes its fingers.

4.5 Discussion

If I use the common discretization scheme for the trajectory and do not predefine the correspondence between the waypoints and tasks, discontinuity is introduced

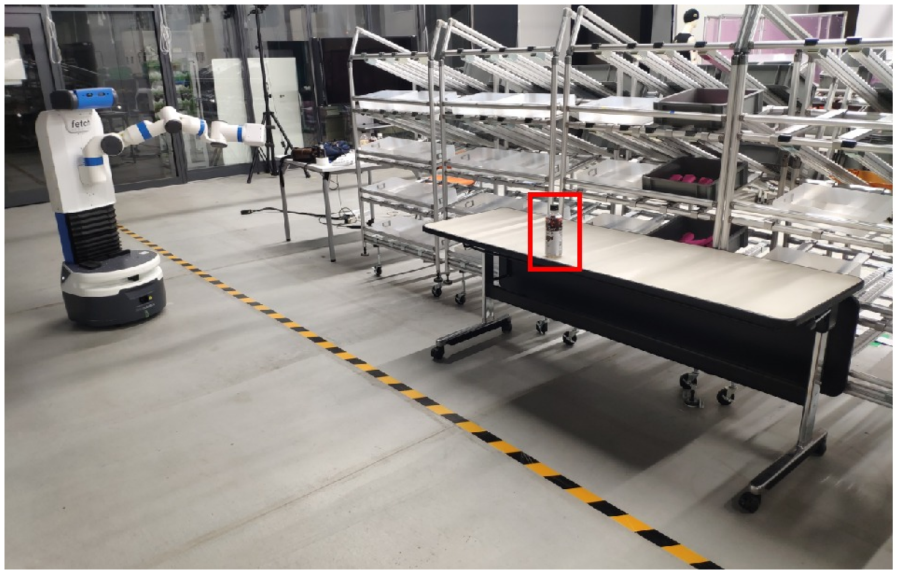
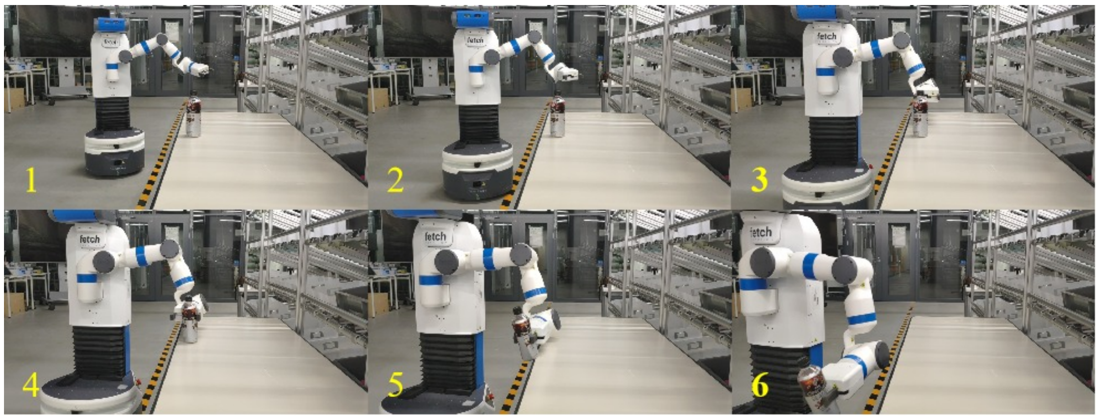


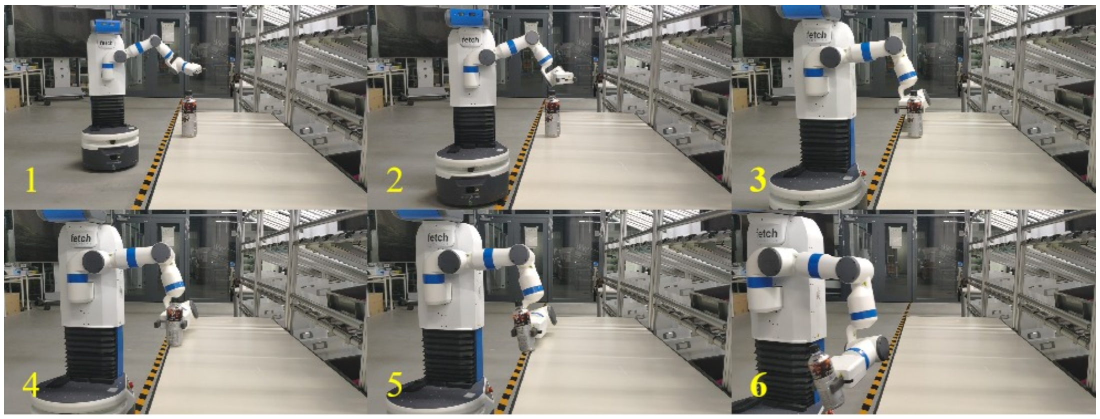
Figure 4.11: The overview of the experiment. The task is to use push-grasp to pick up a bottle from a table.

by the task constraints. Since I need to first locate the interval, within which grasping happens, and then linearly interpolate the two ends of the interval to get the robot configurations for the tasks. For example, as shown in Fig. 4.13a, in i-th iteration of the optimization, the interpolated grasping state is within the interval between the 3rd and 4th waypoints. However, in the next iteration of the optimization, the state may jump to another interval. Fig. 4.13b shows the grasping state jumps to the interval between the 5th and 6th waypoints, then the grasping state has to linearly interpolate the 5th and 6th waypoints. The jumping among different intervals results in different expressions for calculating the grasping state, and introduces discontinuity in the grasp constraint. It may lead to slow convergence or even divergence [119], as a result, I may fail to obtain a feasible trajectory.

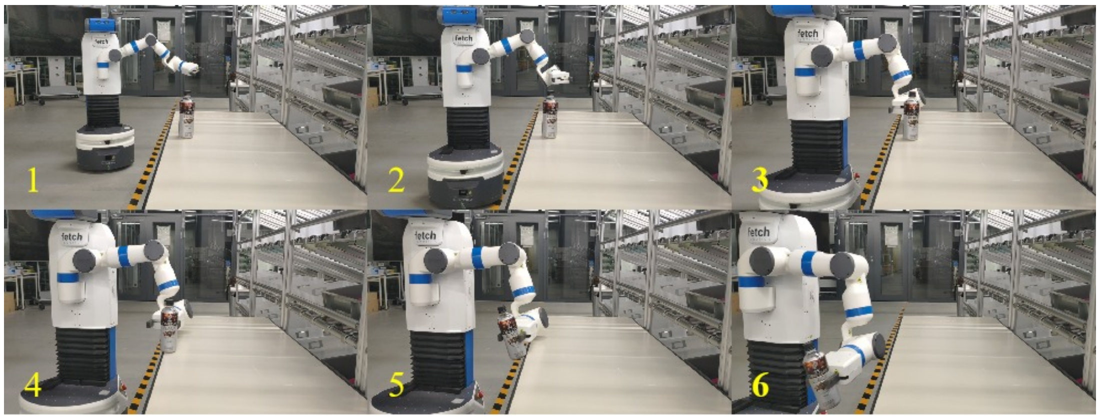
Our formulation of the optimization problem eliminates the discontinuity and non-smoothness. However, for the task constraints, I need to predefine which waypoints



(a)

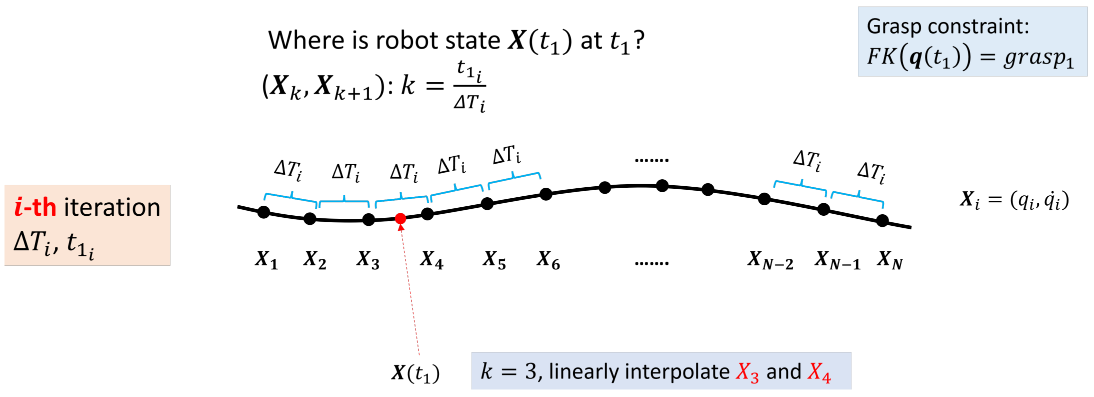


(b)



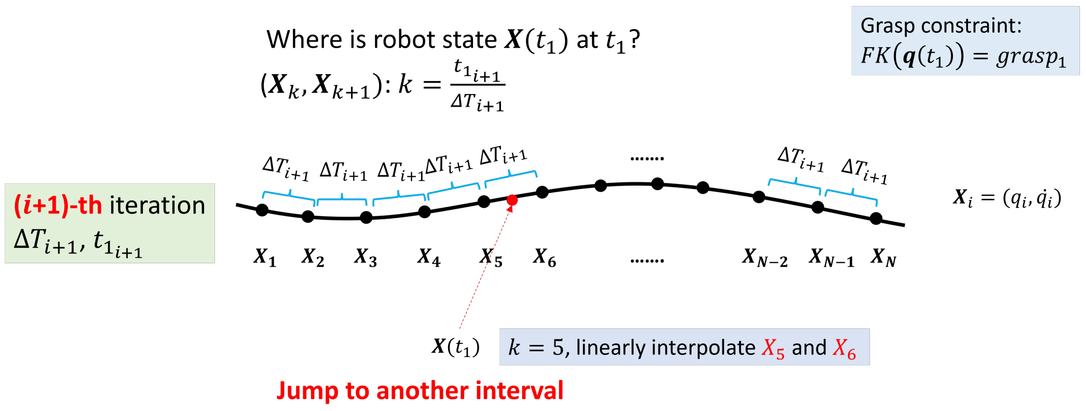
(c)

Figure 4.12: The executed motion of picking up a bottle from a table using push-grasp during the whole-body motion.



Where is robot state $X(t_1)$ at t_1?
(X_k, X_{k+1}): $k = \frac{t_{1_i}}{\Delta T_i}$
Grasp constraint: $FK(q(t_1)) = grasp_1$
i-th iteration
ΔT_i, t_{1_i}
$X_i = (q_i, \dot{q}_i)$
$X(t_1)$
$k = 3$, linearly interpolate X_3 and X_4

(a) i-th iteration of the optimization.



Where is robot state $X(t_1)$ at t_1?
(X_k, X_{k+1}): $k = \frac{t_{1_{i+1}}}{\Delta T_{i+1}}$
Grasp constraint: $FK(q(t_1)) = grasp_1$
(*i*+1)-th iteration
ΔT_{i+1}, $t_{1_{i+1}}$
$X_i = (q_i, \dot{q}_i)$
$X(t_1)$
$k = 5$, linearly interpolate X_5 and X_6
Jump to another interval

(b) (i+1)-th iteration of the optimization.

Figure 4.13: The executed motion of picking up a bottle from a table using push-grasp during the whole-body motion.

correspond to the tasks. The indexes of the waypoints are heuristically determined by the relative position of the grasping tasks compared to the whole trajectory. In practice, this heuristic is not critical to the result, as long as there are enough waypoints in the trajectory segments split by the tasks.

The calculation time seems to scale poorly with respect to the number of waypoints, as shown in Fig. 4.5. One of the reasons is that the SLSQP implemented in NLopt uses dense-matrix methods, which ignore the sparsity of the problem. For large-scale problems with thousands of parameters, maybe I should use other packages such as Ipopt [120].

Since the current method is based on nonlinear optimization and the dimension of the state space is large, it is not suitable for real-time trajectory generation. It can be used to generate locally optimal trajectories offline, and the generated trajectories can be used as reference trajectories for online tracking. Besides, there are still some issues:

- In the physical experiment and simulation: The trajectory is disturbed by the contact forces when the end-effector hits the object. Moreover, as the object slides on the table under the push-grasp, the friction force acts as a disturbance force. Especially when the friction between the object and table is large, the trajectory is disturbed out of control. I can improve this by reducing the friction, but this is an inherent problem of our formulation since I consider kinematic planning, without considering the dynamics.
- Due to limited gripper closing speed, the timing for closing the gripper is delicate with respect to position uncertainty. The robustness can be increased using industrial grippers, which can close instantaneously.
- For some grasping poses, the SLSQP solver is not able to converge to a feasible solution or even fails to obtain a solution due to a segmentation fault encountered during the optimization. Further investigation is required to find out the fundamental reasons.
- As the object slides on the table under the push-grasp, I implicitly assume that there are no other (or very few) obstacles near the target object. Otherwise, the optimization solver may not find a collision-free trajectory for push-grasp. In another word, our planner can not apply to dynamic grasping in a cluttered environment.

CHAPTER 5
GRIPPER SELECTION AND DESIGN FOR PICKING AND ASSEMBLY

Chapter 3 and chapter 4 introduce the task and motion planning for efficiently picking the assembly parts. The assembly parts vary in shape and dimension. To firmly grasp the assembly parts and perform the assembly, the mobile manipulator may have to change different grippers for different assembly parts. However, manually designing the grippers is time-consuming. In this chapter, to improve the efficiency of gripper selection and design, I present a structured approach to selecting and designing a set of grippers for an assembly task. Compared to the current experience-based gripper design method, our approach accelerates the design process by automatically generating a set of initial design options for gripper types and parameters according to the CAD models of assembly components. I use mesh segmentation techniques to segment the assembly components and fit the segmented parts with shape primitives, according to the predefined correspondence between shape primitive and gripper type, suitable gripper types and parameters can be selected and extracted from the fitted shape primitives. Moreover, I incorporate the assembly constraints in the further evaluation of the initially obtained gripper types and parameters. Considering the affordance of the segmented parts and the collision avoidance between the gripper and the subassemblies, applicable gripper types and parameters can be filtered from the initial options. Among the applicable gripper configurations, I further optimize the number of grippers for performing the assembly task, by exploring the gripper that is able to handle multiple assembly components during the assembly. Finally, the feasibility of the designed grippers is experimentally verified by assembling a part of an industrial product.

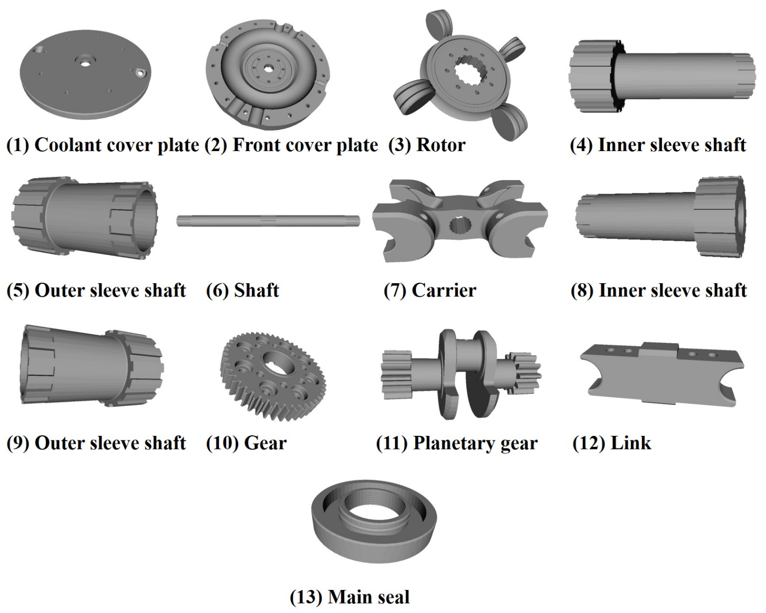


Figure 5.1: Models of all the assembly components before processing, are displayed in the order of the assembly sequence.

The rest of the chapter is organized as follows: Section 5.1 introduces the segmentation of the assembly components. In Section 5.2, the initial set of gripper configurations is extracted by primitive fitting. In Section 5.3, I evaluate these gripper configurations under the assembly constraints and optimize the number of grippers. The feasibility of the designed grippers is confirmed by an assembly experiment in Section 5.4.

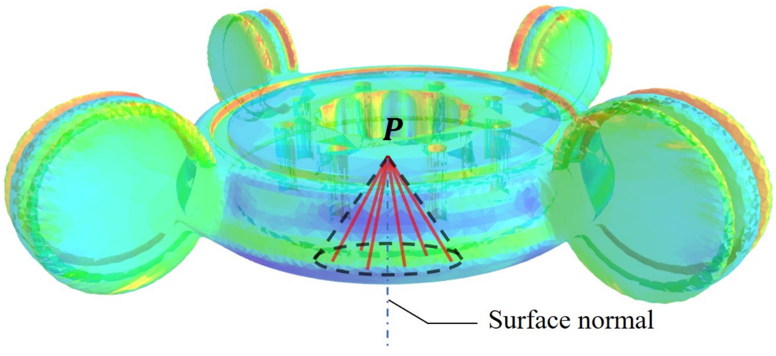
5.1 Mesh Segmentation

Mechanical products are usually comprised of many regular shapes, such as cylinders and cuboids, which makes the proposed method feasible and promising in industrial applications. I use mesh segmentation to find the underlying shape primitives of an assembly component, then suitable gripper types are determined according to the predefined rules. The mesh models of the assembly components

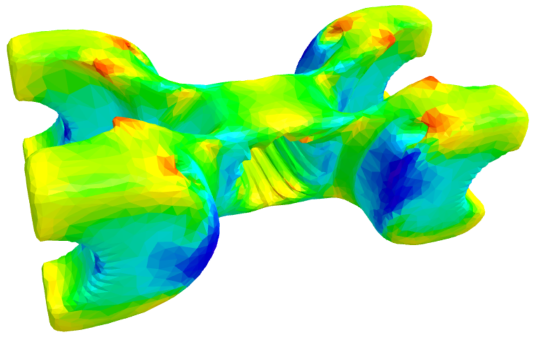
(Fig. 5.1) are segmented based on the Shape Diameter Function (SDF) [121], which is a scalar function measuring the neighborhood diameter of an object at each point on the surface. To obtain the SDF value at a point P on the surface, I construct a cone centered around the inward-normal direction of P, as sketched in black dashed lines in Fig. 5.2 (a), from P I shoot a set of rays (red lines) inside the cone and stop at the intersections on another side of the mesh. The SDF value is calculated as the weighted average length of the rays. In our implementation, I shoot 30 rays per point and set the cone angle to 120○, as a result, Fig. 5.2 (b) and (c) show two examples of SDF distribution on the model. The mesh segmentation process is composed of soft clustering and hard clustering. Soft clustering is a Gaussian mixture model that fits a set of Gaussian distributions to the distribution of the SDF values, this step outputs the probability matrix for each face to belong to each cluster, note that a cluster may contain multiple segments. Hard clustering yields the final segmentation of the mesh by minimizing an energy function by combining the probability matrix and geometric surface features [121, 122]. Readers are referred to [123] for other mesh segmentation methods.

Before mesh segmentation, smoothing is applied to the mesh to eliminate the sharp edges of the screw thread, otherwise, it may result in undesirable segments [124]. Fig. 5.3 shows the mesh after smoothing is applied. Then all the assembly components are segmented based on SDF values. The segmentation results are visualized in Fig. 5.4, different segments are colored differently, and each of the segments is regarded as a candidate for grasping[1].

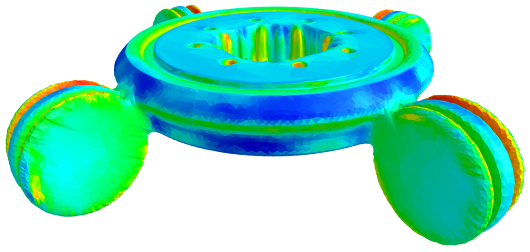
[1]Simultaneously grasping multiple segments is not considered in this chapter.



(a)



(b)



(c)

Figure 5.2: (a) The Shape Diameter Function (SDF) is the weighted average length of the rays (red lines). (b) & (c) SDF distribution of the carrier and the rotor.

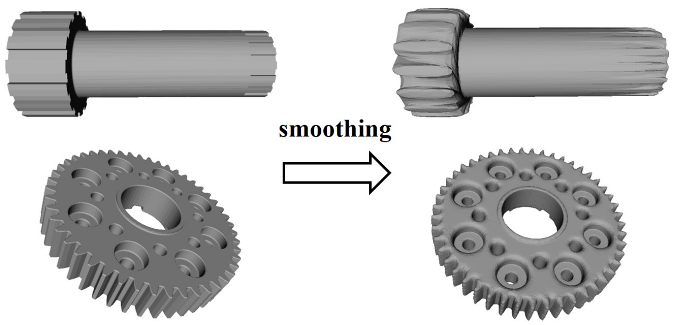


Figure 5.3: Two examples of models before and after smoothing.

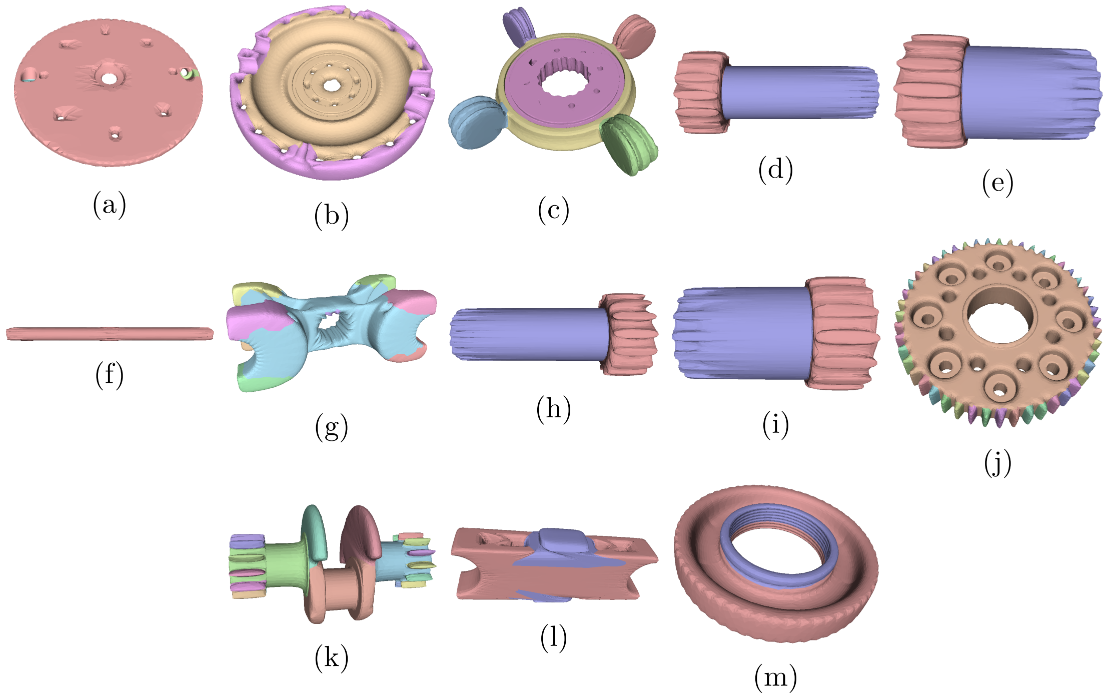


Figure 5.4: After mesh segmentation, an assembly component is decomposed into several segments, different segments are rendered with different colors. The original component with a complex shape is decomposed into segments with simpler shapes, which are suitable for further primitive fitting.

5.2 Gripper Selection and Dimensioning

Through mesh segmentation, an assembly component with a complex shape is decomposed into segments with simpler shapes. Obviously, some shape primitives can be easily grasped by some common types of grippers, e.g., cylinders can be easily grasped by the 3-finger centric gripper. Therefore, I attempt to fit the segments with shape primitives and then determine the suitable gripper types according to the predefined rules. In this section, I obtain the initial decision on gripper types and parameters based on previous segmentation results.

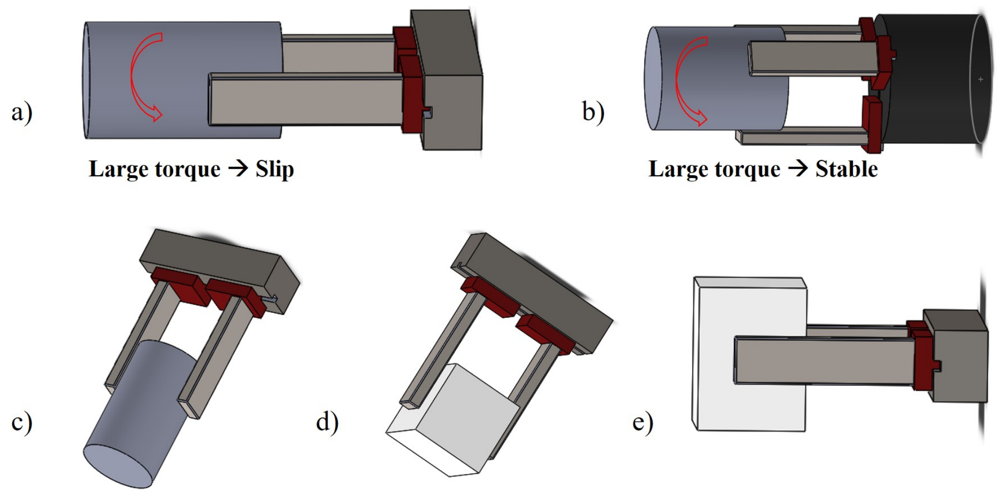


Figure 5.5: (a) & (c) Grasping a cylinder with a 2-finger gripper is not stable against large external torques in the cylinder's radial direction, the object may slip around the contact normal. (b) Grasping a cylinder by a 3-finger gripper is stable against large external torques in the cylinder's radial direction. (d) & (e) Grasping a box shape with a 2-finger gripper is appropriate.

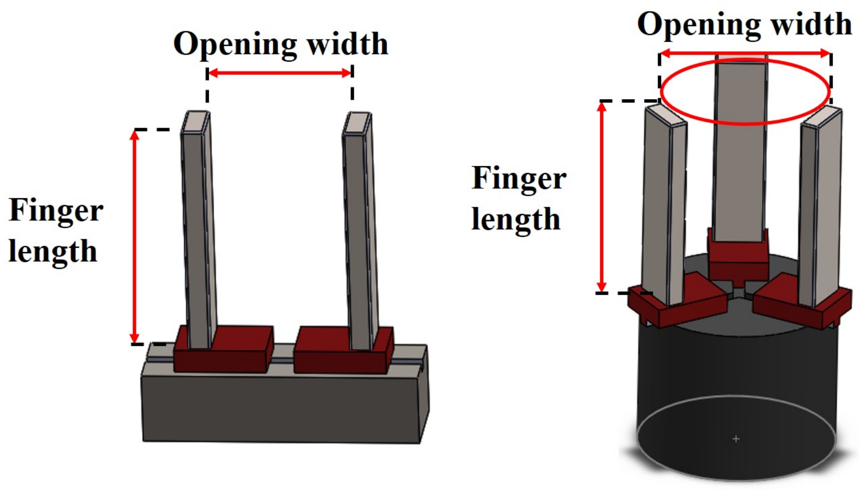


Figure 5.6: The opening width and finger length of the 2-finger and 3-finger grippers.

5.2.1 Rules for Gripper Type Selection

In this chapter, I consider using two common types of grippers[2]: 2-finger parallel jaw grippers and 3-finger centric grippers as shown in Fig. 5.5 (a) and (b). 2-finger grippers are suitable for grasping parts with (nearly) parallel surfaces, Fig. 5.5 (d) and (e) show a 2-finger gripper grasping a box with parallel surfaces, the gripper fingers coincide well with the object surfaces, and they have a large contact area, thus the grasp is stable. However, it may not be suitable to grasp a cylinder using a 2-finger gripper, as shown in Fig. 5.5 (a), small external torques in a cylinder's radial direction can be balanced by assuming soft-finger contact[3], but in some assembly operations, the gripper may have to exert large forces/torques on the assembly components, which may lead to slip around the contact normal. Therefore, I favor 3-finger centric grippers over 2-finger grippers for grasping cylindrical objects, which are guaranteed to be stable against the torque in the radial direction, as shown in Fig. 5.5 (b). Another merit of grasping cylindrical objects using 3-finger grippers is that the grasp stability is independent of the radius of the cylinder, however, the stability of grasping a cylindrical object using a 2-finger gripper depends on the relative curvature of the finger and object surface, that is, grasping a cylinder with larger radius is more stable since the contact area is larger[4].

[2]More gripper types and shape primitives can be used to cope with more complex shapes.

[3]The object and the main body of the finger are assumed to be rigid, but the soft pad can be attached to the fingertip.

[4]Assume soft finger contact and constant external forces.

5.2.2 Gripper Type

Each segment of an assembly component shown in Fig. 5.4 is a candidate for grasping, in order to determine a suitable gripper type for grasping the segment, I fit every segment with a cylinder and a bounding box. If the volume of a cylinder is closer to the volume of the segment, then a 3-finger centric gripper is selected for this segment, otherwise, the 2-finger jaw gripper is used. Since the segments of a surface mesh are usually not closed surfaces, the volume of such segments is obtained by calculating the volume of their convex hulls.

Fig. 5.7 shows two examples of fitting the segmented parts with primitives. The rotor in Fig. 5.7 (b) is segmented into 6 parts, and I fit all of them with cylinders and bounding boxes, by comparing the volume of the segmented part and the fitted primitives, the appropriate fitting for every segment can be determined. Note that the criteria for determining a better fitting are by comparing the volume, but the precondition is that surface of the fitted primitive is non-empty, such that the gripper can grasp the fitted primitives. And this is why I prefer the RANSAC model fitting provided in PCL [125], it guarantees that the fitted primitive has a non-empty surface for grasping. For example, if I fit the point cloud of the third segment in Fig. 5.7 (b) with cylinder models, the best fitting I can find may be the cylinder that aligns with its inner hole surface, but the volume difference with the segment is larger comparing to the fitted bounding box, therefore, the better fitting is actually the bounding box. However, there is no box fitting provided in PCL, so I use the bounding box to fit the segment, in this case, I have to check which faces of the bounding box are empty and which are not, and then select the non-empty faces to be in contact with the gripper. As a result, five of them can be closely fitted by cylinders and the other one is fitted by its bounding

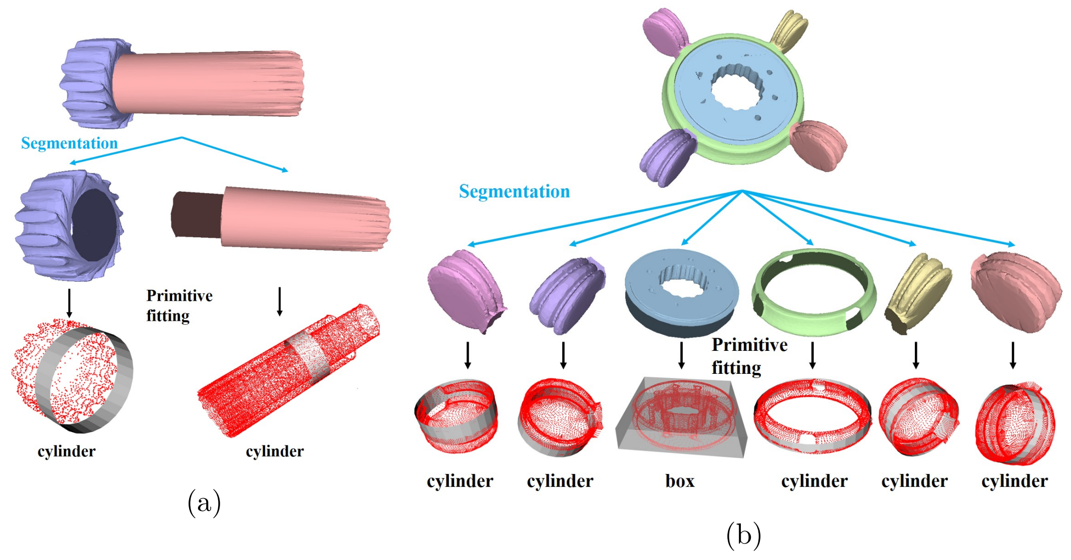


Figure 5.7: Two examples of fitting the segmented parts with shape primitives. (a) The inner sleeve shaft is segmented into two parts, and both the two parts are fitted appropriately by cylinders. (b) Five segments of the rotor are more closely fitted by cylinders and the other one is fitted with its bounding box, notice that the third segment looks cylindrical but it is empty on the cylindrical surface.

box. The fitted cylinders are represented by gray belts, the height of the cylinder is manually set to 1 cm for visualization, but it can also be calculated from the maximum distance along the cylinder's axis between the points on the segment. Then the corresponding gripper type can be selected for every segment based on the predefined rules. In order to grasp a mechanical component, at least one of its segmented parts should be graspable by the designed grippers, e.g. the gripper for grasping the rotor should be capable of grasping at least one of the 6 segments in Fig. 5.7 (b).

5.2.3 Gripper Parameters

The maximum and minimum opening widths and finger length are important parameters of the grippers. In order to grasp a segment, the characteristic length of the shape primitive, which is the diameter of the fitted cylinder or the distance between the opposite faces (in touch with the gripper) of its bounding box, must be within the stroke of the gripper. The capability of grasping a segment does not directly impose constraints on the finger length, however, the finger length has to fulfill some requirements in order to satisfy the assembly constraints, for example, the finger should be long enough to avoid collision with the shaft when inserting the shaft sleeve to the shaft. And the constraints on the finger length are described in section 5.2. In addition, the gripper approach direction can be extracted from the fitted primitives. The 3-finger centric gripper should approach the part along the axial direction of the fitted cylinder, and the 2-finger gripper can approach the part as long as the finger surfaces are parallel to the non-empty surfaces of the bounding box.

5.3 Evaluation Under Assembly Constraints

Through mesh segmentation and shape primitive fitting, I have obtained the initial candidate gripper types and parameters for all the segmented parts of the assembly components, however, some of them are not applicable considering the assembly constraints. In this section, I take into account the assembly constraints and finalize the minimum number of grippers for the given assembly task.

5.3.1 Assembly Task Specification

Referring to the assembly task decomposition method proposed by Mosemann et al. [126], an assembly task can be represented as a sequence of assembly operations,

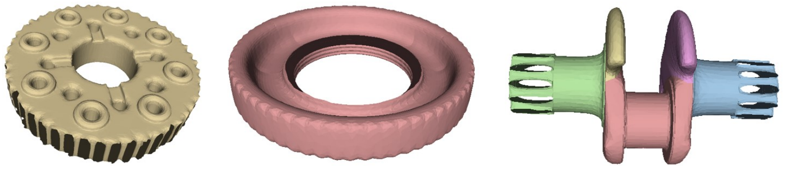


Figure 5.8: Gear teeth and screw thread do not afford to grasp, thus removed from the candidate graspable segments.

in each assembly $operation_i$, a new assembly component is added to the existing subassembly. I assume the assembly sequence is already given, then the assembly task is denoted as $Assembly = \{operation_1, operation_2, \ldots, operation_n\}$. Each assembly operation can be represented as $\langle c_a, c_p, {}^aT_p, {}^{a'}T_p \rangle$, where c_a and c_p are the active and passive subassemblies to be manipulated in the operation, and ${}^aT_p$ and ${}^{a'}T_p$ are spatial transformations between active and passive subassemblies before and after the assembly operation, respectively. The active subassembly is the subassembly grasped by the gripper during the assembly operation, and it moves with the gripper until the assembly operation finishes. The passive subassembly is usually fixed in the environment, and it serves as an environmental obstacle that should not collide with the gripper. In Fig. 5.9, the gripper should grasp the active subassembly c_a to assemble it to the passive subassembly c_p.

5.3.2 Assembly Constraints

In an assembly operation $operation_i$, the gripper has to grasp one segment of c_a and change the spatial relationship from ${}^aT_p$ to ${}^{a'}T_p$. When grasping c_a, not every segment of c_a is suitable for grasping, the affordance of different segments should be taken into account in selecting graspable segments. Moreover, the gripper must avoid collision with the subassemblies during the assembly.

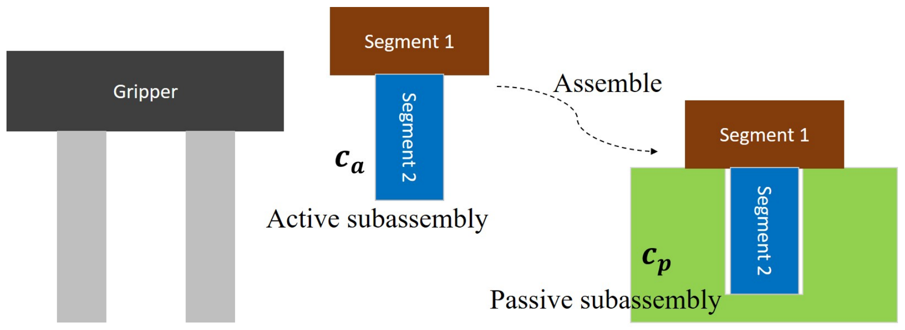


Figure 5.9: The gripper has to avoid collision with the subassemblies in the assembly task.

Affordance

Affordance is defined as the possible action on an object or environment [127]. In an assembly operation, not all the segments of an assembly component afford to grasp. For example, screw thread and gear teeth are mainly used for fastening and transmission, they may be damaged and lose their main affordance if they are directly grasped by the gripper. As illustrated in Fig. 5.8, some segments are manually removed from the candidate segments for grasping, considering their major affordance.

Collision Avoidance

The gripper has to avoid collision with the subassemblies during the assembly, the example illustrated in Fig. 5.9 shows that the gripper will collide with the subassembly if segment 2 is grasped in this assembly operation, thus segment 1 should be selected as the graspable segment. A segment is graspable only if there exists a collision-free grasping pose for the gripper to assemble c_a to c_p. To get the graspable segments satisfying the collision avoidance constraint, I plan a set of grasps for each segment and check the collision between the gripper and the subassemblies, the segment is graspable if there is at least one collision-free grasp.

5.3.3 Grasp Planning

After removing ungraspable segments according to their affordance, grasp planning is performed on the remaining segments to determine if there are collision-free grasps for the segments. For the segments to be grasped by 2-finger parallel grippers, I first use planar clustering [113] to cluster the mesh into a set of nearly planar facets, and then search for nearly parallel facets to be in contact with the fingers of the gripper, and rotate the gripper around the contact normal to obtain more grasps. Fig. 5.10 shows some examples of planar clustering, different clustered facets are rendered with different colors. By searching nearly parallel facets from the clustered model, pairs of facets and contact points for grasping are obtained, as shown in Fig. 5.11. In terms of the segments to be grasped by 3-finger grippers, the grasp can be easily extracted from the fitted cylinder, the axis of the gripper should align with the axis of the cylinder.

The planned grasps are then examined by checking the collision between the gripper and the subassemblies. In Fig. 5.12, I explain how grasp planning is used to determine the graspable segment with respect to collision avoidance constraints, and in this example, the sub-task is to assemble the rotor to the subassemblies (including the coolant cover plate and the front cover plate shown in Fig. 5.1). The rotor, as presented in Fig. 5.7 (b), is composed of 6 segments. Its third segment should be grasped by a 2-finger parallel jaw gripper, and the planned grasps are shown in Fig. 5.12 (a), (b), and (c), however, as seen in Fig. 5.12 (c), all the grasps are in collision with the subassemblies, therefore, the third segment is not graspable. The first, second, fifth, and sixth segments of the rotor should be

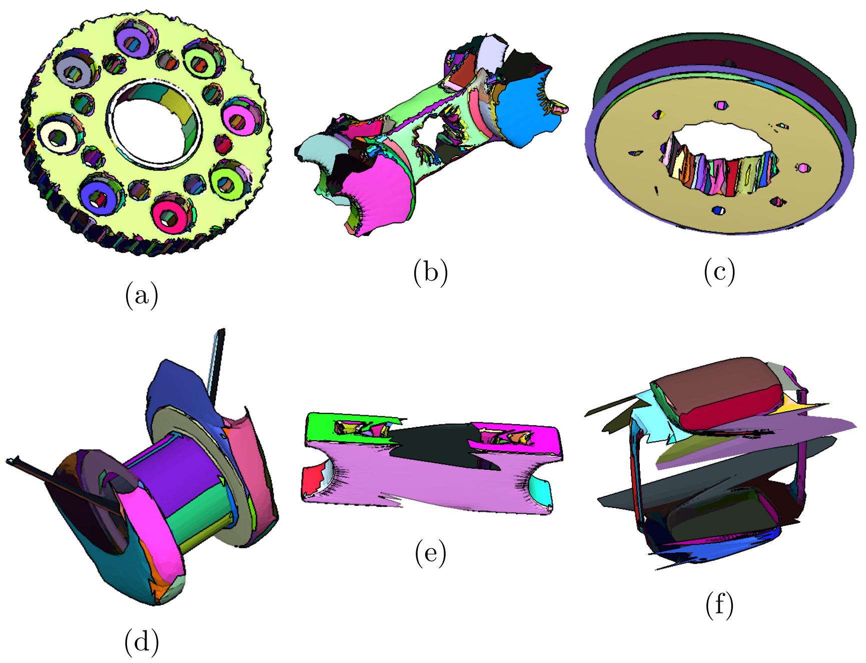


Figure 5.10: Planar clustering of the segments to be grasped by 2-finger grippers, different colors render different clustered facets.

grasped by a 3-finger centric gripper, but they are also not graspable since the planned grasps are in collision with the subassemblies (Fig. 5.12 (d) & (e)). The fourth segment of the rotor should be grasped by the 3-finger centric gripper, and it is the only graspable segment, the collision-free grasp is shown in Fig. 5.12 (f). After the evaluation under the assembly constraints following such process, the remaining graspable segments of every assembly component are listed in Fig. 5.13, alongside the graspable segments, and there are the constraints on gripper types and parameters for grasping the segments. Among these constraints, the opening width is set to be the characteristic length of the segment, which is the diameter of the fitted cylinder or the distance between the opposite faces (in touch with the gripper) of its bounding box, and the opening width should be within the stroke of the gripper. The finger length should be set such that the finger can contact the segment and also avoid the collision between the gripper and other segments and subassemblies. For example, consider assembling assembly component No. 5

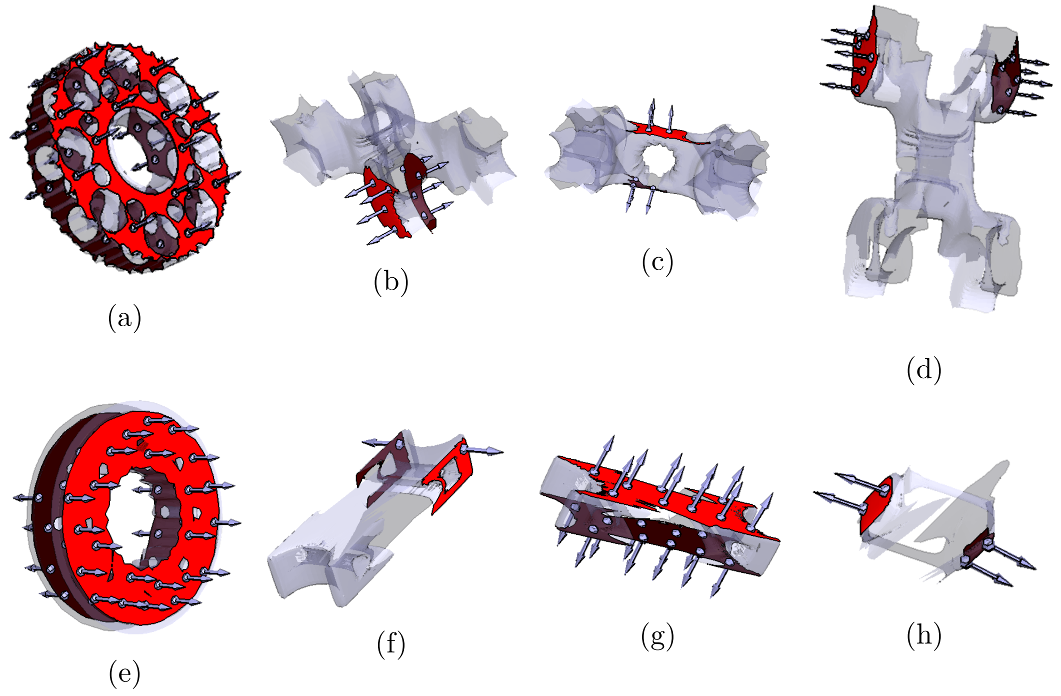


Figure 5.11: Pairs of facets and contact points for grasping by 2-finger grippers, the origins of the arrows are the contact points, and the arrows point to the surface normal directions at the contact points.

to No. 4 (see Fig. 5.16 (b1)), if I grasp the second segment of No. 5, then the finger length has to be long enough to avoid colliding with assembly component No. 4 and also the first segment of No. 5, and similarly, I obtain the finger length constraint for the second segment of assembly component No. 9 (see Fig. 5.16 (c2)). For assembly component No. 10, the finger should be longer than the depth of the gear teeth to contact the target segment (see Fig. 5.16 (c4)). For assembly components No. 8 and No. 13, the finger length should be above the threshold to avoid colliding with the shaft (No. 6) (see Fig. 5.16 (d1) and (d5)).

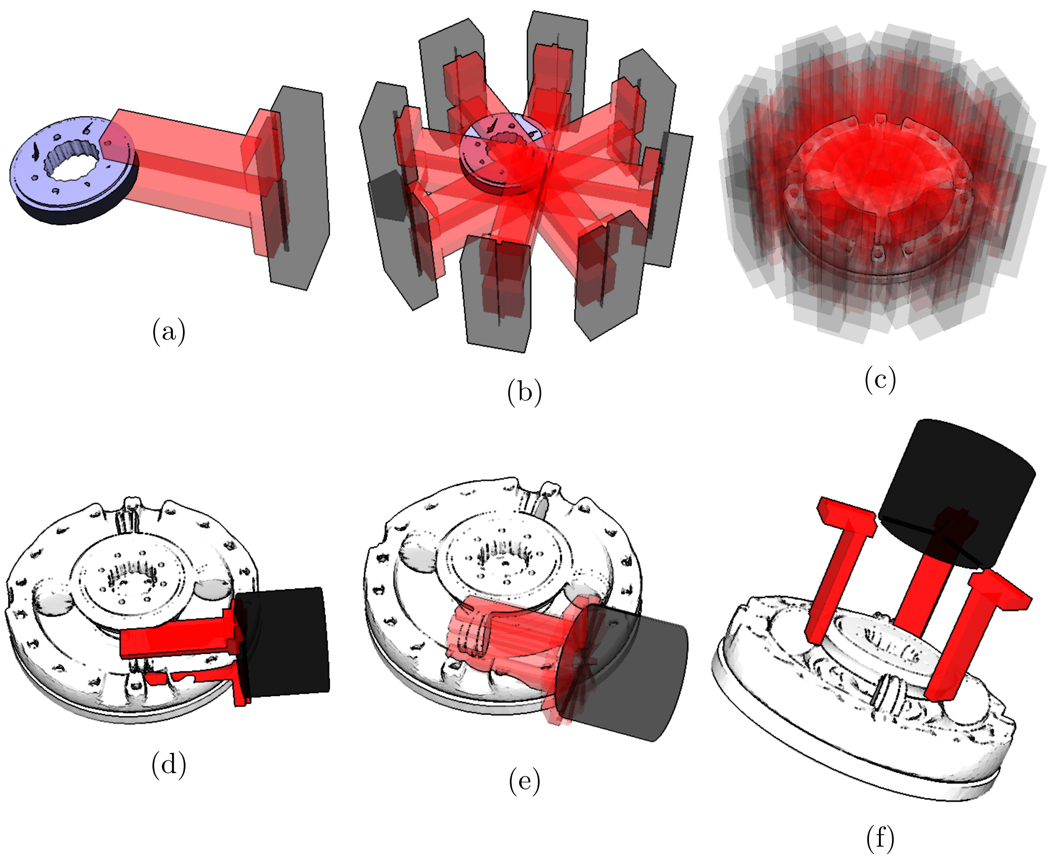


Figure 5.12: Mesh segmentation and primitive fitted have determined the gripper types for the segments, then based on the gripper type, grasp planning is performed on the segmented parts to determine if the segment is graspable, by checking if there is at least one collision-free grasp. (a) A planned 2-finger grasp at a pair of contact points. (b) A set of planned 2-finger grasps are obtained by rotating the gripper around the contact normal. (c) Check the collision between the gripper and the subassemblies. (d) A planned 3-finger grasp for the segment. (e) All the planned 3-finger grasps are in collision with the subassemblies. (f) Collision-free grasps can be found for this segment, and it is the only graspable segment for grasping the rotor to perform this assembly operation.

5.3.4 Minimize the Number of Grippers

Some assembly components can be commonly grasped by the same gripper, thus the total cost of grippers can be cut down by reducing the number of grippers for the assembly task. From the previous analysis, I have obtained the graspable segments from all the assembly components, and every assembly component c_i

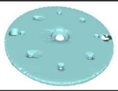
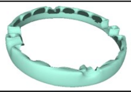
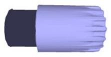
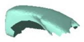
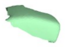
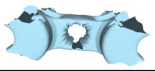
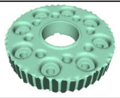
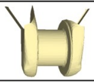
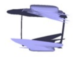
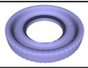
No.	Graspable segments	Gripper type	Opening width	Finger length
1		3-finger	227 mm	> 0 mm
2		3-finger	227 mm	> 0 mm
3		3-finger	110 mm	> 0 mm
4		3-finger	25 mm	> 0 mm
5		3-finger	32 mm	> 0 mm
		3-finger	40 mm	> 41 mm
6		3-finger	18 mm	> 0 mm
7		2-finger	16.7 mm	> 0 mm
		2-finger	19.1 mm	> 0 mm
		2-finger	17.2 mm	> 0 mm
		2-finger	15.9 mm	> 0 mm
		2-finger	16.5 mm	> 0 mm
		2-finger	15.8 mm	> 0 mm
		2-finger	16.9 mm	> 0 mm
		2-finger	18.5 mm	> 0 mm
		2-finger	60 mm	> 0 mm
8		3-finger	40 mm	> 50 mm
9		3-finger	32 mm	> 0 mm
		3-finger	40 mm	> 41 mm
10		2-finger	17 mm	> 4.5 mm
11		3-finger	20 mm	> 0 mm
12		2-finger	15 mm	> 0 mm
		2-finger	32.8 mm	> 0 mm
13		3-finger	37.5 mm	> 22.7 mm

Figure 5.13: The remaining graspable segments after checking the assembly constraints.

imposes a set of constraints on the gripper, such as the number of fingers F_i, opening width W_i, minimum finger length L_i^- and maximum finger length L_i^+. I denote the constraints for assembly component c_i as $\mathcal{C}_i = \{F_i, W_i, L_i^-, L_i^+\}$, $i = 1, 2, \ldots, M$, M is the number of assembly components, if an assembly component c_i has m graspable segments $\{c_{i1}, c_{i2}, \ldots, c_{im}\}$, then $\mathcal{C}_i = \mathcal{C}_{i1} \cup \mathcal{C}_{i2} \cup \cdots \cup \mathcal{C}_{im}$, where $\mathcal{C}_{ij}$ is the constraints imposed by segment c_{ij} of c_i. I generate N gripper parameters $\{P_1, P_2, \ldots, P_N\}, P_j = \{F_j, W_j^-, W_j^+, L_j\}$, covering the minimum and maximum gripper parameters, then determine the minimal subset out of the N gripper parameters that can grasp all the assembly components, and the problem is formulated as follows,

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{j=1}^N x_j \\ \text{subject to} \quad & x_j = \{0,1\} \\ & \textstyle\sum_{j=1}^N a_{1j} x_j \geq 1 \\ & \textstyle\sum_{j=1}^N a_{2j} x_j \geq 1 \\ & \vdots \\ & \textstyle\sum_{j=1}^N a_{Mj} x_j \geq 1 \end{aligned} \tag{5.1}$$

$$\begin{pmatrix} F_j = F_i \\ L_i^- < L_j < L_i^+ \\ W_j^- < W_i < W_j^+ \end{pmatrix} \tag{5.2}$$

where $i \in \{1, 2, \ldots, M\}, j \in \{1, 2, \ldots, N\}$, M is the number of assembly components, N is the number of gripper parameter samples, a_{ij} is 1 if Eq. (2) is satisfied, otherwise $a_{ij} = 0$, W_j^- and W_j^+ are the minimum and maximum opening widths of a gripper, respectively, L_j is the finger length, and x_j is 1 if j-th gripper parameter sample P_j is used and is 0 otherwise. Eq. (2) lists the conditions for the gripper with parameter P_j to grasp the assembly component c_i, and the inequalities in Eq. (1) ensure that every assembly component c_i can be grasped by at least one gripper.

This is a multidimensional 0–1 knapsack problem [128], to solve it, I have to, (1) obtain N gripper parameters and then determine a minimal subset out of N subject to these constraints, and (2) figure out all the coefficients a_{ij}. For generating N samples of gripper parameters, I separate the samples into two groups, the first group is for 2-finger grippers ($F_j = 2$), and the second group is for 3-finger grippers ($F_j = 3$). For each group, I set the upper and lower bounds of the maximum opening width W^+_{upper}, W^+_{lower} to be just enough to cover the maximum and minimum values of opening width W_i, that is $W^+_{upper} = \{W_i\}_{max}, W^+_{lower} - d = \{W_i\}_{min}$, where d is the stroke of the gripper, if $\{W_i\}_{max} - \{W_i\}_{min} \leq d$, then I set $W^+_{upper} = W^+_{lower} = \{W_i\}_{max}$. Notice that, I do not assume the gripper can be fully closed, instead I assume the gripper has a certain stroke d, or more specifically, $d_{2finger}$ for the 2-finger gripper and $d_{3finger}$ for the 3-finger gripper, following such assumption, the minimum opening width can be directly derived as: $W^-_j = W^+_j - d$, and the strokes of the grippers that I use are introduced in Section 6. The upper and lower bounds of finger length L_{upper}, L_{lower} are set to be the maximum and minimum finger lengths among these constraints. So far, I obtain the bounds for the opening width and the finger length for each group, then for each group, I uniformly generate n values of W^+_j from range $[W^+_{lower}, W^+_{upper}]$ and m values of L_j from range $[L_{lower}, L_{upper}]$. Specifically, I obtain n_1 values of W^+_j and m_1 values of L_j for the first group, and n_2 values of W^+_j and m_2 values of L_j for the second group. Therefore, I totally have $N = n_1 \times m_1 + n_2 \times m_2$ sets of gripper parameters for the two parameter groups, and the coefficients a_{ij} are obtained by checking if Eq. (2) is satisfied for assembly constraint $\mathcal{C}_i$ and gripper parameter P_j, then this 0-1 knapsack problem can be solved by using the branch-and-bound method [129]. Since there is no upper bound for the finger length in our case, the upper bound can be flexibly set to a value as long as it is larger than the maximum lower limit of these finger length constraints, the obtained solutions for our case is presented in Section 5.4.

5.3.5 Discussion and Limitation

In this research, I assume that the target assembly components can be well decomposed into boxes and cylinders, and I only use two types of grippers, which are 2-finger parallel jaw grippers and 3-finger centric grippers. To cope with more complex shapes, I have to use more shape primitives, such as cones and pyramids. In addition to affordance and collision avoidance described above, there are other aspects to be considered for further improvement.

Stability of Grasping Different Segments

In an assembly operation, grasping different segments may result in different force/torque distributions. Consider assembling the carrier to the shaft (Fig. 5.1), if the grasping contact positions are not symmetric about the shaft, Fig. 5.16 (b5) shows an example of such a situation, it will lead to uneven normal force between shaft and hole, which may result in insertion failure, or even damage the assembly components. Therefore, it is necessary to analyze the contact force distribution when selecting a suitable segment for grasping during the assembly.

Finer Finger Design

The assembly components must be stably grasped without slipping during the assembly, in which the external forces include gravity, assembly force, etc. It is necessary to fine-tune the shape of the fingertip surface to increase the contact area with the object, especially when the object surface is curved. Assuming the soft-finger contact model, I can calculate the contact area from the relative curvature between fingertip surface and object surface [89], then an appropriate fingertip surface curvature that ensures the grasp stability can be determined. Fig. 5.5 (a) illustrates the situation of the maximum torque caused by gravity, and it should be balanced by the torsional friction exerted by the soft finger contact.

5.4 Experiment

In this section, the effectiveness and feasibility of our approach are verified by assembling a part of an industrial product using the designed grippers. Considering the limit of our 3D printer, the product is scaled to 55% of its original size and printed out as shown in Fig. 5.14 (Left). The grippers are constructed by attaching printed fingers to air chucks, the 2-finger gripper is constructed by attaching 2 fingers to an SMC MHF2-12D2 air chuck (stroke: 48mm, 0mm to 48 mm), and the 3-finger gripper is constructed by attaching 3 fingers on an SMC MHSL3-32D air chuck (stroke: 8mm, 34 mm to 42 mm), the stroke of an air chuck determines the difference between the maximum and minimum opening widths ($W_j^+ - W_j^-$). According to the strokes of air chucks I use and the scaled dimensions of the assembly constraints, the feasible solutions for the gripper parameters are $\{2, 0, 33, 30\}, \{3, 14, 22, 30\}, \{3, 52.5, 60.5, 30\}, \{3, 116.9, 124.9, 30\}$, that is one

2-finger gripper with a maximum opening width of 33 mm, and three 3-finger grippers with maximum opening widths of 22 mm, 60.5 mm and 124.9 mm are required. I model and print out the fingers and attach them to the air chucks, and I calculate the position where the fingers should be fixed on the air chucks such that the maximum and minimum opening widths correspond to the solutions. For the 2-finger gripper, since the stroke is 48 mm, which is larger than the difference between the maximum and minimum opening widths in solution $\{2, 0, 33, 30\}$, so I simply design the 2-finger gripper with the opening width ranging from 0 mm to 48 mm, which covers the range of opening width in the solution and works equally well. The actual maximum opening widths for the other three 3-finger grippers are slightly larger than the calculation results to account for the uncertainties. Besides, there is no upper bound on finger length, it is free to set finger length above 30 mm, and all the gripper parameters of the actual gripper used in our experiment are shown in Fig. 5.14 (Right).

I performed the assembly experiment on a NEXTAGE robot from Kawada Robotics Inc., as shown in Fig. 5.15, all the 13 assembly components can be firmly grasped by using the designed 4 grippers. I assume the assembly sequence is known, the target segment of an assembly component for grasping can be obtained from the previous analysis, then the robot is able to successfully complete the task without collision with the subassemblies during the assembly, as shown in Fig. 5.16.

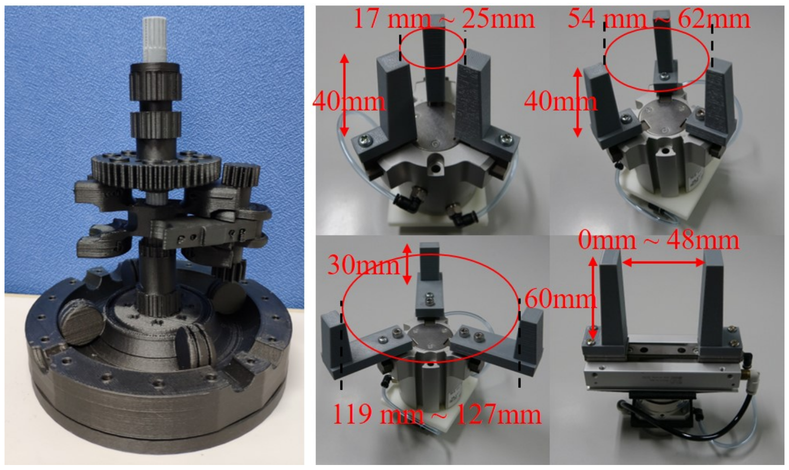
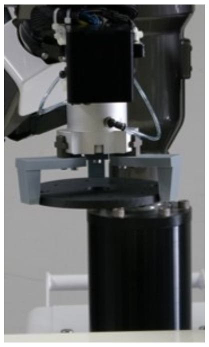
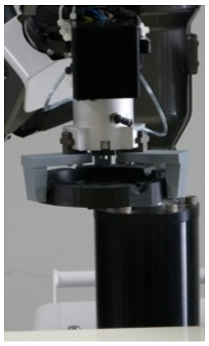
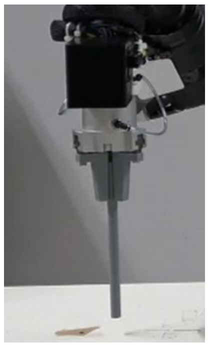


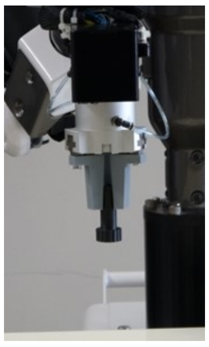
Figure 5.14: (Left): The product to be assembled. (Right): The designed 4 grippers in their maximum opening state, the strokes are 8mm for the 3-finger air chuck and 48mm for the 2-finger air chuck, respectively.


(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

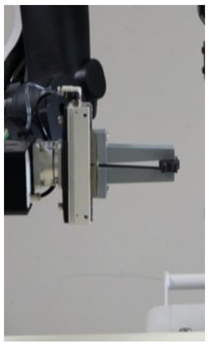
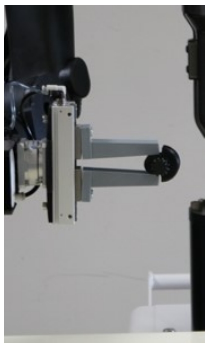
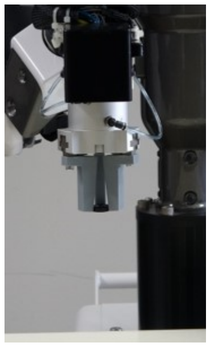
(j)

(k)

(l)

(m)

Figure 5.15: Designed 4 grippers are able to firmly grasp all the 13 assembly components.

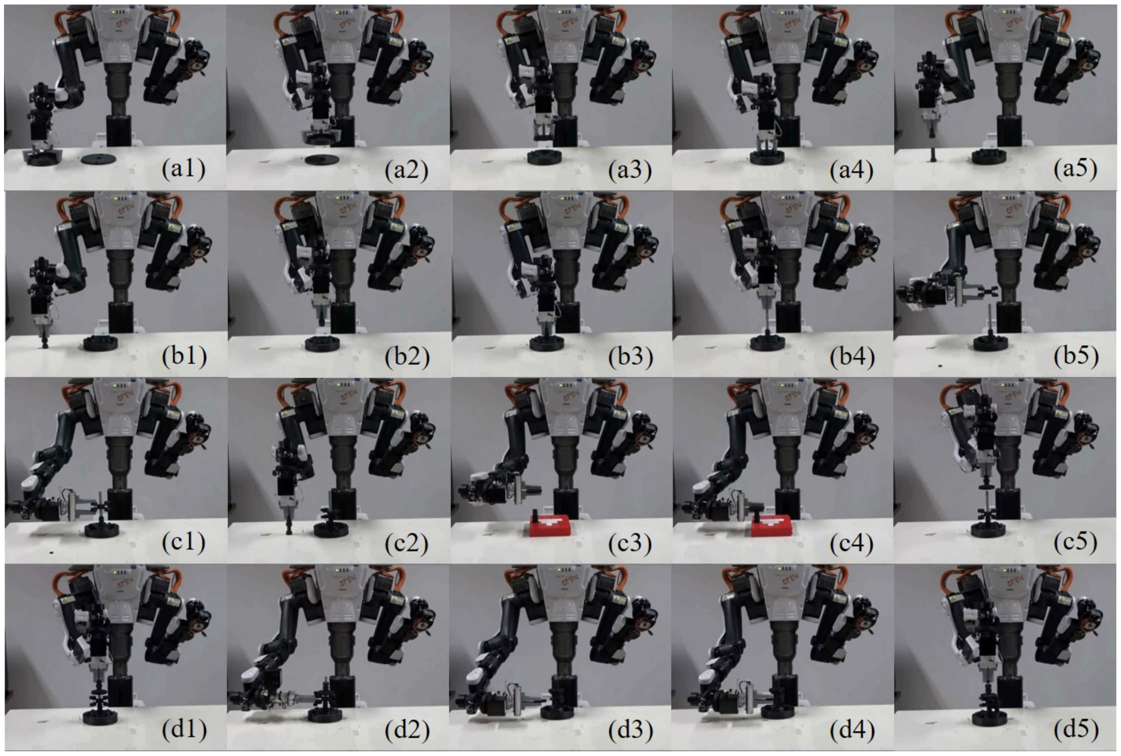


Figure 5.16: The robot can successfully assemble the product, and there is no collision between the gripper and the subassembly, the red object appearing in (c3) and (c4) is used to support the subassembly.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This thesis addresses the automation in an assembly factory from a three-layer hierarchy, i.e., task-level planning, motion-level planning, and grasp-level planning, to improve the efficiency of performing the part-supply, grasping, and assembly tasks.

At the task level, I presented a planner that plans a minimum sequence of base positions for a mobile manipulator to perform a sequence of tasks in multiple locations. Our work contributed to the limited work on mobile manipulators performing a sequence of tasks. Specifically, I considered multiple picking tasks in multiple locations, but our planner also applies to other manipulation tasks. In that case, I should consider constraints from the new tasks and plan the base region satisfying the new task constraints. Our planner can be applied to the pick-and-place tasks involved in the part-supply tasks in the assembly factories and warehouses. By reducing the number of base positions to visit for a sequence of tasks, the overall efficiency of the tasks is improved, thus improving the efficiency of the production. I also discussed the object placement styles and the update of the base sequence, which are critical for practical applications.

I performed numerical analysis on the planner. The numerical analysis shows the calculation time for the base region is the most time-consuming part of the planning, and this part should be performed offline. The calculation time grows nearly linearly with respect to the number of objects in the tray and the number of trays (when the number of trays is less than 18). The numerical result on the size of base regions and the intersection also provides a reference for configuring the objects and trays for practical settings. Combined with the numerical analysis result, I found that for regularly placed objects, both globally static and dynamically updated base positions are feasible, but for randomly placed objects, it is impractical and unnecessary to update the base positions. Extended experiments are conducted to demonstrate the feasible policies.

The efficiency of our planner is demonstrated by comparing the operation time for moving the base with a naive base sequence. Our experiments show that the average operation time is reduced by 11.22s to 17.26s by reducing one base position. However, some limitations are observed in the current experiments. The success rate is not high, mainly due to perception errors and motion planning timeout, and the motion of the manipulator is not optimized. These issues will be considered in our future work.

At the motion level, I aim to further reduce the motion time. I have presented an optimization-based motion planner to plan the optimal motion for a nonholonomic mobile manipulator to perform tasks during the motion. Our major contribution is a novel formulation of the optimization, such that the task constraints are guaranteed to be smooth. The numerical results show that using our formulation, I can find feasible solutions satisfying all the constraints. The proposed motion planner can be used to generate locally optimal trajectories offline, the generated trajectories can be taken as the reference trajectories for online tracking. I also proposed to use the push-grasp strategy to robustly grasp the target object during the motion. I have demonstrated the planned trajectories in a simulator and the physical experiment.

At the grasp level, I tackle the challenges of designing grippers for grasping and assembly. I presented a structured approach to selecting and designing the grippers. The input for our approach is the assembly specification and the geometrical models of the assembly components. In the first phase, the assembly components with complex shapes are segmented into simpler parts, then the segmented parts are fitted with shape primitives. By defining the correspondence between simple shape primitives and gripper types, suitable gripper types and parameters can be determined from the results of mesh segmentation and primitive fitting. In the second phase, the results in the first phase are examined under the assembly constraints, afterwards, the number of grippers is optimized by finding a minimal set of gripper parameters that satisfy the constraints imposed by all the assembly components. Finally, the effectiveness of the designed grippers is confirmed by the assembly experiment. The current work can be improved in several aspects: (1) More powerful mesh segmentation methods [130] can be considered to decompose the assembly components, and the affordance of the assembly component will be taken into account in the segmentation. (2) More shape primitives such as cones and pyramids can be used to improve the ability to fit more complex shapes. (3) The representation of the assembly task and constraints can be refined, and classifying the basic assembly operations (such as peg-in-hole) can further automate the design process. (4) The shape of the fingertip surface can be fine-tuned to increase the contact area, thus the grasp stability is improved.

In the future, these three parts of work can be combined into a whole mobile manipulation system. I envisage that mobile manipulators pick up the assembly parts from the storage area and transport them to the assembly line, and then these assembly parts are assembled by mobile manipulators. To grasp the assembly parts from different trays, it is feasible to use a two-finger gripper to grasp all the assembly parts. However, in case of the assembly tasks, the mobile manipulator may have to change grippers to firmly assemble different assembly parts. To perform assembly tasks by using mobile manipulators, two issues have to be addressed: (1)

the base positioning error of the mobile manipulator, and (2) force control for the contact-rich assembly tasks. The positioning error of the mobile manipulator can be reduced by using extra sensors and repositioning itself. The force control in the assembly tasks is very challenging, especially for mobile manipulators, due to the low rigidity of the mobile manipulator. The learning-based method may be a good choice to solve this challenging contact-rich mobile assembly problem, but more investigation is required to improve this system to be applicable in real industrial environment.

ACKNOWLEDGMENTS

It has been a tough journey, without the help of many people, I could not have been here. Firstly, I would like to thank my advisor Prof. Harada, who gave me a chance to do research in the field of robotics, which I still believe is one of the most important and right decisions for my future development. Prof. Harada has always been supportive for my academic research and considerate for my daily life in Japan. I am very grateful to Prof. Harada for providing the opportunity to work as a research assistant in AIST. He has always been willing to support me whenever I encounter problems, and I appreciate his kind heart. Besides, Prof. Wan has been inspiring me with his passion and diligence in work. I thank him for these inspirations and suggestions on my research. I also thank Prof. Kiyokawa and Prof. Koyama for their valuable comments on my research during the midterm presentations. Moreover, I am very grateful for the excellent research environment created by all the professors and students. When I first came to Harada laboratory. I almost knew nothing about robotics research. It was through the students' midterm presentations, I quickly got the big picture of robotic manipulation research, so I thank my colleagues for their effort.

During my PhD studies, I had a wonderful time working as a research assistant with Domae-san and Ueshiba-san in AIST. Domae-san proposed many valuable suggestions to me during the weekly research meeting, and he has always been supportive for my research. Whenever I need something for help, I know that Domae-san and Ueshiba-san will get them ready for me.

I would like to thank my parents for their unconditional support and encouragement, without which I cannot imagine how I can get through these difficult times. Last but not least, I would like to thank my wife for the company throughout the

journey, I am looking forward to the shared future with you.

BIBLIOGRAPHY

[1] Oussama Khatib, Kazu Yokoi, Oliver Brock, K Chang, and Arancha Casal. Robots in human environments: Basic autonomous capabilities. *The International Journal of Robotics Research*, 18(7):684–696, 1999.

[2] Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. Planning for autonomous door opening with a mobile manipulator. In *2010 IEEE International Conference on Robotics and Automation*, pages 1799–1806. IEEE, 2010.

[3] Anthony Pratkanis, Adam Eric Leeper, and Kenneth Salisbury. Replacing the office intern: An autonomous coffee run with a mobile manipulator. In *2013 IEEE International Conference on Robotics and Automation*, pages 1248–1253. IEEE, 2013.

[4] Kai Zhou, Gerhard Ebenhofer, Christian Eitzinger, Uwe Zimmermann, Christoph Walter, José Saenz, Luis Pérez Castaño, Manuel Alejandro Fernández Hernández, and José Navarro Oriol. Mobile manipulator is coming to aerospace manufacturing industry. In *2014 IEEE International Symposium on Robotic and Sensors Environments (ROSE) Proceedings*, pages 94–99. IEEE, 2014.

[5] Qiankun Yu, Guolei Wang, Xiaotong Hua, Simin Zhang, Libin Song, Jiwen Zhang, and Ken Chen. Base position optimization for mobile painting robot manipulators with multiple constraints. *Robotics and Computer-Integrated Manufacturing*, 54:56–64, 2018.

[6] Yuhao Zhang, Xingwei Zhao, Bo Tao, and Han Ding. Point stabilization of nonholonomic mobile robot by bzier smooth subline constraint nonlinear model predictive control. *IEEE/ASME Transactions on Mechatronics*, 2020.

[7] S. Thakar, P. Rajendran, A. M. Kabir, and S. K. Gupta. Manipulator motion planning for part pickup and transport operations from a moving base. *IEEE Transactions on Automation Science and Engineering*, pages 1–16, 2020.

[8] *HIGH-MIX, LOW-VOLUME ASSEMBLY, https://blog.unex.com/high-mix-low-volume-assembly.* accessed July, 2022.

[9] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[10] Bernard Bayle, J-Y Fourquet, and Marc Renaud. Manipulability of wheeled mobile manipulators: Application to motion generation. *The International Journal of Robotics Research*, 22(7-8):565–581, 2003.

[11] Francois G Pin and J-C Culioli. Multi-criteria position and configuration optimization for redundant platform/manipulator systems. In *IEEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pages 103–107. IEEE, 1990.

[12] Francois G Pin and Jean-Christophe Culioli. Optimal positioning of combined mobile platform-manipulator systems for material handling tasks. *Journal of intelligent and Robotic Systems*, 6(2-3):165–182, 1992.

[13] Wayne F Carriker, Pradeep K Khosla, and Bruce H Krogh. Path planning for mobile manipulators for multiple task execution. *IEEE Transactions on Robotics and Automation*, 7(3):403–408, 1991.

[14] Saman Vafadar, Adel Olabi, and Masoud Shariat Panahi. Optimal motion planning of mobile manipulators with minimum number of platform movements. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pages 262–267. IEEE, 2018.

[15] Jingren Xu, Kensuke Harada, Weiwei Wan, Toshio Ueshiba, and Yukiyasu Domae. Planning an efficient and robust base sequence for a mobile manipulator performing multiple pick-and-place tasks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11018–11024. IEEE, 2020.

[16] Fei Chen, Mario Selvaggio, and Darwin G Caldwell. Dexterous grasping by manipulability selection for mobile manipulator with visual guidance. *IEEE Transactions on Industrial Informatics*, 15(2):1202–1210, 2018.

[17] Shantanu Thakar, Pradeep Rajendran, Vivek Annem, Ariyan Kabir, and Satyandra Gupta. Accounting for part pose estimation uncertainties during trajectory generation for part pick-up using mobile manipulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1329–1336. IEEE, 2019.

[18] Shantanu Thakar, Pradeep Rajendran, Ariyan M. Kabir, and Satyandra K. Gupta. Manipulator motion planning for part pickup and transport operations from a moving base. *IEEE Transactions on Automation Science and Engineering*, pages 1–16, 2020.

[19] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, 26(6):2182–2189, 2017.

[20] Johannes Schmalz and Gunther Reinhart. Automated selection and dimensioning of gripper systems. *Procedia CIRP*, 23:212–216, 2014.

[21] Mohammadali Honarpardaz, Johan Ölvander, and Mehdi Tarkian. Fast finger design automation for industrial robots. *Robotics and Autonomous Systems*, 113:120–131, 2019.

[22] DT Pham and SH Yeo. Strategies for gripper design and selection in robotic assembly. *The International Journal of Production Research*, 29(2):303–316, 1991.

[23] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, & Autonomous Systems*, 4:265–293, 2021.

[24] Yoshio Yamamoto and Xiaoping Yun. Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Transactions on Automatic Control*, 39(6):1326–1332, 1994.

[25] Homayoun Seraji. An on-line approach to coordinated mobility and manipulation. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 28–35. IEEE, 1993.

[26] Homayoun Seraji. A unified approach to motion control of mobile manipulators. *The International Journal of Robotics Research*, 17(2):107–118, 1998.

[27] Felix Burget and Maren Bennewitz. Stance selection for humanoid grasping tasks by inverse reachability maps. In *2015 IEEE International conference on robotics and automation (ICRA)*, pages 5669–5674. IEEE, 2015.

[28] Pooya Abolghasemi, Rouhollah Rahmatizadeh, Aman Behal, and Ladislau Boloni. A real-time technique for positioning a wheelchair-mounted robotic arm for household manipulation tasks. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[29] Shunan Ren, Ying Xie, Xiangdong Yang, Jing Xu, Guolei Wang, and Ken Chen. A method for optimizing the base position of mobile painting ma-

nipulators. *IEEE Transactions on Automation Science and Engineering*, 14(1):370–375, 2016.

[30] Bin Du, Jing Zhao, and Chunyu Song. Optimal base placement and motion planning for mobile manipulators. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1227–1234. American Society of Mechanical Engineers Digital Collection, 2013.

[31] Dmitry Berenson, James Kuffner, and Howie Choset. An optimization approach to planning for mobile manipulation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1187–1192. IEEE, 2008.

[32] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs.* PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[33] Freek Stulp, Andreas Fedrizzi, Lorenz Mösenlechner, and Michael Beetz. Learning and reasoning with action-related places for robust mobile manipulation. *Journal of Artificial Intelligence Research*, 43:1–42, 2012.

[34] Franziska Zacharias, Christoph Borst, Michael Beetz, and Gerd Hirzinger. Positioning mobile manipulators to perform constrained linear trajectories. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2578–2584. IEEE, 2008.

[35] Nikolaus Vahrenkamp, Tamim Asfour, and Rudiger Dillmann. Robot placement based on reachability inversion. In *2013 IEEE International Conference on Robotics and Automation*, pages 1970–1975. IEEE, 2013.

[36] Nikolaus Vahrenkamp, Tamim Asfour, and Rüdiger Dillmann. Efficient inverse kinematics computation based on reachability analysis. *International Journal of Humanoid Robotics*, 9(04):1250035, 2012.

[37] Daniel Leidner and Christoph Borst. Hybrid reasoning for mobile manipulation based on object knowledge. In *Workshop on AI-based robotics at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[38] Jun Dong and Jeffrey C Trinkle. Orientation-based reachability map for robot base placement. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1488–1493. IEEE, 2015.

[39] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3229–3236. Ieee, 2007.

[40] John T Feddema. Kinematically optimal robot placement for minimum time coordinated motion. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3395–3400. IEEE, 1996.

[41] David Hsu, J-C Latcombe, and Stephen Sorkin. Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings of the 1999 IEEE international symposium on assembly and task planning (ISATP'99)(Cat. no. 99TH8470)*, pages 280–285. IEEE, 1999.

[42] Saïd Zeghloul and José-Alfonso Pamanes-Garcia. Multi-criteria optimal placement of robots in constrained environments. *Robotica*, 11(2):105–110, 1993.

[43] S Mitsi, K-D Bouzakis, D Sagris, and G Mansour. Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm. *Robotics and Computer-Integrated Manufacturing*, 24(1):50–59, 2008.

[44] Alonzo Kelly. *Mobile robotics: mathematics, models, and methods*. Cambridge University Press, 2013.

[45] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[46] Yoshio Yamamoto and Xiaoping Yun. Effect of the dynamic interaction on coordinated control of mobile manipulators. *IEEE Transactions on robotics and automation*, 12(5):816–824, 1996.

[47] Homayoun Seraji. Configuration control of redundant manipulators: Theory and implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472–490, 1989.

[48] Homayoun Seraji. An on-line approach to coordinated mobility and manipulation. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 28–35. IEEE, 1993.

[49] Alessandro De Luca, Giuseppe Oriolo, and Paolo Robuffo Giordano. Kine-

matic modeling and redundancy resolution for nonholonomic mobile manipulators. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1867–1873. IEEE, 2006.

[50] Glenn D White, Rajankumar M Bhatt, Chin Pei Tang, and Venkat N Krovi. Experimental evaluation of dynamic redundancy resolution in a nonholonomic wheeled mobile manipulator. *IEEE/ASME Transactions on Mechatronics*, 14(3):349–357, 2009.

[51] Ying Wang, Haoxiang Lang, and Clarence W De Silva. A hybrid visual servo controller for robust grasping by wheeled mobile robots. *IEEE/ASME transactions on Mechatronics*, 15(5):757–769, 2010.

[52] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[53] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[54] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[55] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[56] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[57] Yoshiaki Kuwata, Gaston A Fiore, Justin Teo, Emilio Frazzoli, and Jonathan P How. Motion planning for urban driving using rrt. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1681–1686. IEEE, 2008.

[58] Richard M Murray and Sosale Shankara Sastry. Nonholonomic motion plan-

ning: Steering using sinusoids. *IEEE transactions on Automatic Control*, 38(5):700–716, 1993.

[59] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[60] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[61] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[62] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems*, 1:159–185, 2018.

[63] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[64] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[65] Elon Rimon and Daniel E Koditschek. Exact robot navigation using cost functions: the case of distinct spherical boundaries in e/sup n. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1791–1796. IEEE, 1988.

[66] Daniel E Koditschek and Elon Rimon. Robot navigation functions on manifolds with boundary. *Advances in applied mathematics*, 11:412, 1990.

[67] Howie M Choset, Kevin M Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, Sebastian Thrun, and Ronald C Arkin. *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[68] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[69] Jeffrey Ichnowski, Michael Danielczuk, Jingyi Xu, Vishal Satish, and Ken Goldberg. Gomp: Grasp-optimized motion planning for bin picking. *arXiv preprint arXiv:2003.02401*, 2020.

[70] Jeffrey Ichnowski, Yahav Avigal, Vishal Satish, and Ken Goldberg. Deep learning can accelerate grasp-optimized motion planning. *Science Robotics*, 5(48), 2020.

[71] Cong Wang, Qifeng Zhang, Qiyan Tian, Shuo Li, Xiaohui Wang, David Lane, Yvan Petillot, and Sen Wang. Learning mobile manipulation through deep reinforcement learning. *Sensors*, 20(3):939, 2020.

[72] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. *arXiv preprint arXiv:2112.05251*, 2021.

[73] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(4):6289–6296, 2021.

[74] Max Spahn, Bruno Brito, and Javier Alonso-Mora. Coupled mobile manipulation via trajectory optimization with free space decomposition. In *2021 International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[75] Jianfeng Liao, Fanghao Huang, Zheng Chen, and Bin Yao. Optimization-based motion planning of mobile manipulator with high degree of kinematic redundancy. *International Journal of Intelligent Robotics and Applications*, 3(2):115–130, 2019.

[76] Joshua Fishman, Samuel Ubellacker, Nathan Hughes, and Luca Carlone. Dynamic grasping with a" soft" drone: From theory to practice. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[77] Naresh Marturi, Marek Kopicki, Alireza Rastegarpanah, Vijaykumar Rajasekaran, Maxime Adjigble, Rustam Stolkin, Aleš Leonardis, and Yasemin Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Autonomous Robots*, 43(5):1241–1256, 2019.

[78] Iretiayo Akinola, Jingxi Xu, Shuran Song, and Peter K Allen. Dynamic grasping with reachability and motion awareness. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[79] Arjun Menon, Benjamin Cohen, and Maxim Likhachev. Motion planning for smooth pickup of moving objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 453–460. IEEE, 2014.

[80] Charles Sun, Jędrzej Orbik, Coline Manon Devin, Brian H Yang, Abhishek Gupta, Glen Berseth, and Sergey Levine. Fully autonomous real-world reinforcement learning with applications to mobile manipulation. In *5th Annual Conference on Robot Learning*, 2021.

[81] Bennion R Cannon, Todd D Lillian, Spencer P Magleby, Larry L Howell, and Matthew R Linford. A compliant end-effector for microscribing. *Precision engineering*, 29(1):86–94, 2005.

[82] Kaidi Nie, Weiwei Wan, and Kensuke Harada. An adaptive robotic gripper with l-shape fingers for peg-in-hole tasks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4022–4028. IEEE, 2018.

[83] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1824–1829. IEEE, 2003.

[84] Markus Przybylski, Tamim Asfour, and Rüdiger Dillmann. Planning grasps for robotic hands using a novel object representation based on the medial axis transform. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1781–1788. IEEE, 2011.

[85] Kai Huebner and Danica Kragic. Selection of robot pre-grasps using box-based shape approximation. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1765–1770. IEEE, 2008.

[86] Nikolaus Vahrenkamp, Leonard Westkamp, Natsuki Yamanobe, Eren E Aksoy, and Tamim Asfour. Part-based grasp planning for familiar objects. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 919–925. IEEE, 2016.

[87] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[88] Duc Truong Pham, Nasir Salah Gourashi, and Eldaw Elzaki Eldukhri. Automated configuration of gripper systems for assembly tasks. *Proceedings*

of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 221(11):1643–1649, 2007.

[89] Kensuke Harada, Tokuo Tsuji, Soichiro Uto, Natsuki Yamanobe, Kazuyuki Nagata, and Kosei Kitagaki. Stability of soft-finger grasp under gravity. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 883–888. IEEE, 2014.

[90] Matei Ciocarlie, Claire Lackner, and Peter Allen. Soft finger model with adaptive contact geometry for grasping and manipulation tasks. In *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*, pages 219–224. IEEE, 2007.

[91] Kensuke Harada, Tokuo Tsuji, Kazuyuki Nagata, Natsuki Yamanobe, Kenichi Maruyama, Akira Nakamura, and Yoshihiro Kawai. Grasp planning for parallel grippers with flexibility on its grasping surface. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 1540–1546. IEEE, 2011.

[92] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000.

[93] Richard M Murray. *A mathematical introduction to robotic manipulation.* CRC press, 2017.

[94] Mohammadali Honarpardaz, Martin Meier, and Robert Haschke. Fast grasp tool design: From force to form closure. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 782–788. IEEE, 2017.

[95] Haoran Song, Michael Yu Wang, and Kaiyu Hang. Fingertip surface optimization for robust grasping on contact primitives. *IEEE Robotics and Automation Letters*, 3(2):742–749, 2018.

[96] Alberto Rodriguez and Matthew T Mason. Effector form design for 1dof planar actuation. In *2013 IEEE International Conference on Robotics and Automation*, pages 349–356. IEEE, 2013.

[97] Orion Taylor and Alberto Rodriguez. Optimal shape and motion planning for dynamic planar manipulation. *Autonomous Robots*, 43(2):327–344, 2019.

[98] Nikhil Chavan-Dafle, Matthew T Mason, Harald Staab, Gregory Rossano, and Alberto Rodriguez. A two-phase gripper to reorient and grasp. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1249–1255. IEEE, 2015.

[99] Lionel Birglen and Thomas Schlicht. A statistical review of industrial robotic grippers. *Robotics and Computer-Integrated Manufacturing*, 49:88–97, 2018.

[100] Katharina Hermann, Rafael Hostettler, Markus Zimmermann, and Anand Vazhapilli Sureshbabu. A joint-selective robotic gripper with actuation mode switching. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1532–1539. IEEE, 2019.

[101] Aljaf Kramberger, Adam Wolniakowski, Mads Høj Rasmussen, Marko Munih, Aleš Ude, and Christian Schlette. Automatic fingertip exchange system for robotic grasping in flexible production processes. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1664–1669. IEEE, 2019.

[102] Kensuke Harada, Kento Nakayama, Weiwei Wan, Kazuyuki Nagata, Natsuki Yamanobe, and Ixchel G Ramirez-Alpizar. Tool exchangeable grasp/assembly planner. In *International Conference on Intelligent Autonomous Systems*, pages 799–811. Springer, 2018.

[103] Kento Nakayama, Weiwei Wan, and Kensuke Harada. Designing grasping tools for robotic assembly based on shape analysis of parts. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 1–7. IEEE, 2019.

[104] Corey Goldfeder, Peter K Allen, Claire Lackner, and Raphael Pelossof. Grasp planning via decomposition trees. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4679–4684. IEEE, 2007.

[105] Mark R Cutkosky et al. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on robotics and automation*, 5(3):269–279, 1989.

[106] Masayuki Shimizu, Hiromu Kakuya, Woo-Keun Yoon, Kosei Kitagaki, and Kazuhiro Kosuge. Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Transactions on Robotics*, 24(5):1131–1142, 2008.

[107] Giresh K Singh and Jonathan Claassens. An analytical solution for the inverse kinematics of a redundant 7dof manipulator with link offsets. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2976–2982. IEEE, 2010.

[108] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*, 2016.

[109] Kensuke Harada, Kenji Kaneko, and Fumio Kanehiro. Fast grasp planning for hand/arm systems based on convex model. In *2008 IEEE International Conference on Robotics and Automation*, pages 1162–1168. IEEE, 2008.

[110] Soichiro Uto, Tokuo Tsuji, Kensuke Harada, Ryo Kurazume, and Tsutomu Hasegawa. Grasp planning using quadric surface approximation for parallel grippers. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1611–1616. IEEE, 2013.

[111] Tokuo Tsuji, Soichiro Uto, Kensuke Harada, Ryo Kurazume, Tsutomu Hasegawa, and Ken'ichi Morooka. Grasp planning for constricted parts of objects approximated with quadric surfaces. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2447–2453. IEEE, 2014.

[112] Kensuke Harada, Tokuo Tsuji, Soichiro Uto, Natsuki Yamanobe, Kazuyuki Nagata, and Kosei Kitagaki. Stability of soft-finger grasp under gravity. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 883–888. IEEE, 2014.

[113] Weiwei Wan, Kensuke Harada, and Fumio Kanehiro. Planning grasps with suction cups and parallel grippers using superimposed segmentation of object meshes. *IEEE Transactions on Robotics*, 2020.

[114] Yukiyasu Domae, Haruhisa Okuda, Yuichi Taguchi, Kazuhiko Sumi, and Takashi Hirai. Fast graspability evaluation on single depth maps for bin picking with general grippers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1997–2004. IEEE, 2014.

[115] Peter J Kolesar. A branch and bound algorithm for the knapsack problem. *Management science*, 13(9):723–735, 1967.

[116] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[117] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[118] Steven G Johnson. The nlopt nonlinear-optimization package, 2014.

[119] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming.* SIAM, 2010.

[120] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[121] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249, 2008.

[122] The CGAL Project. *CGAL User and Reference Manual.* CGAL Editorial Board, 5.0.2 edition, 2020.

[123] Ariel Shamir. A survey on mesh segmentation techniques. In *Computer graphics forum*, volume 27, pages 1539–1556. Wiley Online Library, 2008.

[124] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference.* The Eurographics Association, 2008.

[125] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[126] Heiko Mosemann and Friedrich M Wahl. Automatic decomposition of planned assembly sequences into skill primitives. *IEEE transactions on Robotics and Automation*, 17(5):709–718, 2001.

[127] Natsuki Yamanobe, Weiwei Wan, Ixchel G Ramirez-Alpizar, Damien Petit, Tokuo Tsuji, Shuichi Akizuki, Manabu Hashimoto, Kazuyuki Nagata, and Kensuke Harada. A brief review of affordance in robotic manipulation research. *Advanced Robotics*, 31(19-20):1086–1101, 2017.

[128] Arnaud Fréville. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, 2004.

[129] Peter J Kolesar. A branch and bound algorithm for the knapsack problem. *Management science*, 13(9):723–735, 1967.

[130] Truc Le and Ye Duan. A primitive-based 3d segmentation algorithm for mechanical cad models. *Computer Aided Geometric Design*, 52:231–246, 2017.

LIST OF PUBLICATIONS

Journal Articles:

[1] Xu, J., Wan, W., Koyama, K., Domae, Y. and Harada, K., Selecting and designing grippers for an assembly task in a structured approach. *Advanced Robotics*, 35(6), pp. 381-397, 2021.

[2] Xu, J., Domae, Y., Ueshiba, T., Wan, W. and Harada, K., Planning a Minimum Sequence of Positions for Picking Parts From Multiple Trays Using a Mobile Manipulator. *IEEE Access*, 9, pp. 165526-165541, 2021.

[3] Xu, J., Domae, Y., Wan, W. and Harada, K., Whole-body Trajectory Optimization for a Mobile Manipulator to Perform Dynamic Grasping During the Motion. *Submitted to IEEE Robotics and Automation Letters*, 2022.

International Conference Papers (peer-reviewed):

[1] Xu, J., Harada, K., Wan, W., Ueshiba, T. and Domae, Y., Planning an efficient and robust base sequence for a mobile manipulator performing multiple pick-and-place tasks. In 2020 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11018-11024, 2020.

[2] Xu, J., Domae, Y., Wan, W. and Harada, K., 2022, An Optimization-based Motion Planner for a Mobile Manipulator to Perform Tasks During the Motion. In 2022 *IEEE/SICE International Symposium on System Integration (SII)*, pp. 519-524, 2022.

Domestic Conference Papers (non peer-reviewed):

[1] Xu, J., Domae, Y., Wan, W. and Harada, K., Trajectory optimization for cou-

pled mobile manipulator motion in picking tasks. *The Conference of the Society of Instrument and Control Engineers System Integration Division (SICE SI)*, pp. 993–997, 2021.

[2] Xu, J., Domae, Y., Ueshiba, T., Wan, W. and Harada, K., Base position planning for a mobile manipulator to pick-and-transport objects stored in multiple trays. *The Conference of the Society of Instrument and Control Engineers System Integration Division (SICE SI)*, pp. 1167–1172, 2020.

[3] Xu, J., Koyama, K., Wan, W., Domae, Y. and Harada, K., Designing grippers based on model decomposition and primitive fitting. *The Proceedings of JSME annual Conference on Robotics and Mechatronics (Robomech)*, pp. 1P1-B01, 2020.