Planning a Minimum Sequence of Positions for Picking Parts from Multiple Trays Using a Mobile Manipulator

JINGREN XU1,2, (Student Member, IEEE), YUKIYASU DOMAE2, (Member, IEEE), TOSHO UESHIBA2, (Member, IEEE), WEIWEI WAN1, (SENIOR MEMBER, IEEE), AND KENSUKE HARADA1,2, (Senior Member, IEEE)

1Robotic Manipulation Laboratory, Graduate School of Engineering Science, Osaka University, Osaka 560-8531, Japan.
2Automation Research Team, Industrial Cyber-Physical Systems Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan.

Corresponding author: Jingren Xu (e-mail: xu@hlab.sys.es.osaka-u.ac.jp).

ABSTRACT 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 paper, we present a planner that plans a minimum sequence of base positions for a mobile manipulator to robustly collect objects stored in multiple trays. We 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. We 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, we 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, we examine our planner on 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.

INDEX TERMS Mobile manipulation, manufacturing automation, part-supply tasks, pick-and-place, task-level planning.

I. INTRODUCTION

There is an increasing demand on robots which are able to flexibly perform tasks in the human environment [1] (such as opening doors [2] and fetch 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, where the robot picks up and transports objects to the desired location. However, these tasks are still heavily occupied by human workers. In our project, we expect to use mobile manipulators to replace human workers to perform the part-supply tasks in an automotive assembly factory, where different categories of assembly parts of automobiles are scattered in a large storage area. To perform the part-supply 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.
FIGURE 1: 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.

A. TASK DESCRIPTION AND MAIN APPROACH

Fig. 1 shows the schematic overview of the task to be performed by the mobile manipulator. The product $P$ to be assembled is comprised of several types of parts: $P_s$, $P_b$, and $P_e$, 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. We 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 the collision with the environment. Notice that we are 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, 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, we 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, we 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, we 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. We discussed and analyzed the feasible policies for different task specifications and parts placements.

B. COMPARED TO EXISTING RESEARCH

Existing research usually optimizes the manipulator configuration or base position for each task, with respect to criteria such as manipulability and reachability \cite{8, 9}. However, very few works consider a sequence of tasks. The most relevant ones we could find are \cite{10}–\cite{13}. In \cite{10}–\cite{12}, 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. In \cite{13}, they 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.

We use the mobile manipulator in a decoupled approach, where the mobile manipulator firstly stops at the planned position and then performs the picking. A few researchers used mobile manipulators in a coupled approach \cite{7}, where the mobile manipulator picks and transports a target object while the base is moving. However, picking from a moving base is sensitive to the base positioning uncertainty and is difficult in real-world implementation. Moreover, we expect the mobile manipulator to pick up multiple objects stored within a small
region. If the base is moving during the picking, the base has to move very slowly to wait for the multiple picking tasks to be finished. Therefore, picking while moving the base may not significantly reduce the operation time. Instead, we employ the decoupled approach and put our effort on reducing the number of base movements.

This paper describes a planner that generalizes our previous conference contributions [14]. The conference version presented a special case of the current version, i.e., planning a globally static base sequence for the mobile manipulator to collect all the objects in a tray. Different from the conference paper, we tackle real-world implementation issues by considering different object placement styles in the trays, and discuss the possible policies for dynamically updating the base sequence as the pick-and-place tasks proceed. Significantly extended numerical analysis of the planner and experimental results are presented to demonstrate the feasibility of our method in different scenarios. The contributions, in addition to the conference version, are summarized as follows:

- We generalize our planner by considering both regularly and randomly placed objects, and propose a method to estimate the base region for randomly placed objects.
- We discuss the possible policies for dynamically updating the base sequence, which further increases the robustness.
- Numerical analysis about the planner is presented. The influence of different parameters on the performance of the planner is analyzed.
- Extended experiments are conducted to demonstrate the feasibility and efficiency of our proposed method in different policies.

II. RELATED WORK

Our work belongs to task-level planning for mobile manipulators considering the motion constraints, i.e., we plan the kinematically reachable base positions for picking parts from multiple trays. We do not plan the low-level motion of the mobile manipulator between these positions, instead, we 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 distinguishes it from the field of task and motion planning (TAMP), where task planning and motion planning are strongly coupled. In our work, the only connection to motion planning during the task planning is that we considered the kinematic feasibility/reachability, i.e., there should be collision-free inverse kinematics solutions in a feasible base position. Therefore, our task-level planning and motion planning are very weakly coupled. The related research is twofold: (1) Positioning the mobile manipulator for performing the tasks. (2) Mobile manipulators performing a sequence of tasks.

A. 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 [15]–[17]. 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 [18]–[20]. Manipulability [8], [9] is a widely used metric for evaluating the flexibility of manipulator configurations in a base position.

Yamamoto et al. [15] proposed a planning and control method to position the mobile manipulator at the preferred region to achieve high manipulability. Du et al. [21] used the manipulability index to determine a suitable base placement. Ren et al. [20] 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. [22] obtained the base placement and grasp for a mobile manipulator to move an object from one configuration to another, by optimizing a scoring function which combines the grasp quality, manipulability and distance to obstacle. OpenRAVE [23] 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. [24] proposed Action-Related Place to associate a base location with a probability of successfully performing a manipulation task, capability map was used to determine if an object was theoretically reachable. Burget et al. [18] 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. [25] took advantage of the reachability map to position a mobile manipulator to perform a linear trajectory in the workspace. Vahrenkamp et al. [26], [27] conducted a series of research on reachability analysis and its application, the base positions with 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 on the completeness of obtained base positions. Some other works that used reachability and capability analysis are referred to [28]–[30].

In addition to the base placement for mobile manipulators, there is a line of research on the optimal placement for fixed-base manipulators [31]–[34]. Feddema et al. [31] 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. [32] considered the obstacles in the workspace, 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 is not a critical issue, thus it is not considered in planning the base positions.

B. MOBILE MANIPULATOR PERFORMING A SEQUENCE OF TASKS

To the best of our knowledge, [11] 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, 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. [12] 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. [13] studied the minimum platform movements to reach a set of poses in the workspace, which is a special case of our task definition. Moreover, we 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, we 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, (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.

III. METHOD OVERVIEW

Fig. 2 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 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, we 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 VII-B.

Algorithm 1: Overview of the algorithm

Input: Grasping poses for objects in target trays
Output: A sequence of base positions

1. base_regions ← {}
2. for tray ∈ target_trays do
3.     base_region ← SolveBaseRegion(tray)
4.     base_regions ← base_regions $\cup$ base_region
5.     robust_intersections ← {}
6.     intersections ← PossibleIntersections(base_regions)
7.     for intersection ∈ intersections do
8.         $r$ ← InscribedRadius(intersection)
9.         if $r \geq$ positioning_uncertainty then
10.             robust_intersections ← robust_intersections $\cup$ intersection
11.         base_sequence ← PlanMinSequence(robust_intersections)
12.         return base_sequence

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 of the base regions (line 11).

Fig. 3a, 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. 3a, all the objects in a tray are used to plan a globally static base sequence. Fig. 3b shows the update of base regions based on the remaining objects in the 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. 2 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.

IV. BASE SEQUENCE PLANNING

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

A. TASK DEFINED AS REACHING THE GRASPING POSES

For grasping the target objects in the target trays, a sequence of tasks are 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 placement, and the grasp planning method considered in this paper is shown in Algorithm 2. For objects regularly placed in the tray, model-based grasp planner [35] can be used to prepare a set of grasps for the target object \( O_j \) in the tray in the offline phase (line 2 and 3). For objects randomly placed in the tray, we treat the objects in the tray as a whole and use a model-free grasp planning method [37] to estimate a set of grasps from depth images (line 5), the details are described in Section VII-B.

Algorithm 2: Planning grasps

| Input: Mesh models of the objects or rendered depth images |
| Output: Planned grasps |
| 1 Function PlanGrasps() |
| 2 if RegularlyPlaced then |
| 3 \[ grasps \leftarrow \text{ModelBasedGraspPlanning}(\text{mesh}) \] |
| 4 else |
| 5 \[ grasps \leftarrow \text{EstimateFromDepthImages}(\text{imgs}) \] |
| 6 return grasps |

B. BASE REGION CALCULATION

Algorithm 3 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, we prepare stable grasping poses with respect to the object for every target object in the tray (line 2), using a grasp planner. Then we uniformly sample base poses \( (x_t, y_t, \phi) \) in front of the target tray (line 3), as illustrated in Fig. 3. Here we assume that the orientation \( \phi \) 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. 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 \( \{G_{t1}, G_{t2}, \ldots, G_{tn}\} \) with respect to the mobile base for object \( O_j \) are obtained for every target object in the tray.

In order to determine the feasibility of a grasping pose, firstly we solve inverse kinematics (IK) and then check if the IK solutions are collision-free. We use the method presented in our previous work [14] to solve inverse kinematics.
Algorithm 3: Calculate the base region for a tray

**Input:** Target objects in the tray  
**Output:** Base region of the tray

```text
Function SolveBaseRegion(tray)

// plan grasps for every target object
grasps_all_objs ← PlanGrasps(objs_in_tray)
sampled_positions ← SampleBasePositions(tray)
base_region ← {}

for position ∈ sampled_positions do

// All target objects should be graspable
all_objs_graspable ← true

for grasps_obj ∈ grasps_all_objs do

// At least one grasp should have collision-free IK

for grasp ∈ grasps_obj do

IKs ← SolveIK(grasp, position)
if ExistCollisionFreeIK(IKs) then
  break
else if IsLastGrasp then
  all_objs_graspable ← false
else
  continue

// if not all the objects are graspable, check the next position
if Not all_objs_graspable then
  break

if all_objs_graspable then
  base_region ← base_region ∪ position

return base_region
```

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

```text
Function SolveIK(grasp, position)

// get grasping pose w.r.t the mobile base
EE_pose ← Transform(grasp, position)
voxel ← QueryDB(EE_pose)
IKs ≈ configs_in_voxel(voxel)

return IKs
```

FIGURE 3: The sampled base positions for one of the trays, 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.

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 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.

### C. ROBUST INTERSECTIONS OF BASE REGIONS

Given a part-supply task to collect the objects in trays \{tray_1, tray_2, ..., tray_n\}, the corresponding base regions \{P_{B1}, P_{B2}, ..., P_{Bn}\} and their intersections can be obtained following the proposed method. As described in line 6 of Algorithm 3, firstly we obtain the possible intersections between all the base regions. Let \( \cap_i^{n} P_{B_i} \) \((1 \leq i \leq n)\), 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 \( \lambda \) be the largest k, then \( \cap_1^{n} P_{B_1}, \cap_2^{n} P_{B_2}, ..., \cap_\lambda^{n} P_{B_\lambda} \) 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 [38] 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 IV.B, in some mobile manipulators, the joint connecting the manipulator and mobile base rotates around a vertical axis. The joint motion can 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, we 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.

D. PATH PLANNING

The function $PlanMinSequence(\text{robust}\_\text{intersections})$ 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 base sequence vector $[x_1, x_2, \ldots, x_N]^T$ and minimizing the sum of its elements, which is formulated as:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} x_i \\
\text{subject to} & \quad x_i \in \{0, 1\} \\
& \quad \sum_{i=1}^{N} a_{1i}x_i \geq 1 \\
& \quad \sum_{i=1}^{N} a_{2i}x_i \geq 1 \\
& \quad \ldots \\
& \quad \sum_{i=1}^{N} a_{ni}x_i \geq 1
\end{align*}
\]

Here, $a_{si}, s = \{1, 2, \ldots, n\}$, 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 [39]. Finally, we 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, we use a simple example where there are 5 base regions of 5 trays, as shown in Fig. 4. One of them is a third-order intersection and four of them are second-order intersections. Notice that, as shown in Fig. 4b, 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., $\text{InscribedRadius}(P_{B1} \cap P_{B2} \cap P_{B3}) < \bar{\sigma}$, 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 of the trays. Then we perform 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 [40].

V. DYNAMICALLY UPDATE THE BASE POSITIONS

The algorithms presented in Section IV explained planning a sequence of base positions for a given task, defined by the grasps for the target objects. In this section, we 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, we 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, we 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.

A. 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. 5. Since the objects are regularly placed in the tray with known poses, the base region $P_{Bij}$ for object $O_j$ in tray $i$ can be calculated in the offline phase for all the objects. In the online phase, firstly we 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. We 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. 6 illustrates a simple heuristic. Firstly we 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 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.

B. 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, we can treat the objects in the tray as a whole and estimate the total grasps using the method described in Section VII-B 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 of remaining objects can be estimated. For instance, in the offline phase, we 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, we have to select the target objects to be grasped and plan the base positions online, which is time-consuming and impractical.

VI. NUMERICAL RESULTS AND ANALYSIS

In this section, we perform numerical analysis on the base sequence planner. We use the Fetch robot [38] with a 7-DOF manipulator mounted on a differential drive mobile base.

Fig. 7 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. We 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. 7, the plotted base regions are the results of parameters: \( m = 12 \times 3 \times 4 \), \( k = 1 \), the tray size is 30cm \( \times \) 40cm. In all the planning, we use uniformly sampled base positions as shown in Fig. 4 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 [14], here we focus on the quantitative analysis of the planner.

A. BASE REGIONS AND INTERSECTIONS

Here we 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. 8 and Table 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. 9 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 we 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).

B. 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 2.

Table 1 shows the result of the calculation time for planning the base region for a single tray. Fig. 10 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 presents the total calculation time of the full planning with respect to the number of trays. As shown in Fig. 11, the calculation grows nearly linearly when the number of trays is smaller than 18, then it grows exponentially due to the exponential complexity of path searching and exploring the minimal number of base positions.

C. ANALYSIS OF DIFFERENT POLICIES

We use some numerical examples to evaluate the performance of different policies. As shown in Table 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

| 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 1: Numerical result for a single tray.

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

TABLE 2: Typical calculation time for different sub-tasks. The result is conducted on 10 neighboring trays, where there are 108 (9 \( \times \) 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.
$m$ objects, $k$ grasps per object

totally $n$ trays

d

FIGURE 7: 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.

FIGURE 8: 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.

| 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: Numerical result for multiple trays (full planning).

less than 10 (excluding the start and goal positions).

For regularly placed objects, we 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, we 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

FIGURE 9: 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.

| 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     | No          |
| Rand/updateR | Unreliable | 0.143 mm           | OK     | No          |
| Rand/updateT | Reliable   | 0.229 mm           | No     | No          |

TABLE 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.
three policies are reliable 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,

$$\text{average}_\text{robustness} = \frac{\sum_{i=1}^{m} \text{robustness}_i}{m}$$

(2)

where \(\text{robustness}_i\) 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 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 nearly 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 4 summarizes all the 6 policies. From the above analysis, 4 of them are feasible for practical application. For randomly placed objects, we 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 \( \text{tray}_1 \) and \( \text{tray}_2 \) and is ready to move to the next base position, another mobile manipulator can immediately join the task and move to pick objects from \( \text{tray}_1 \) and \( \text{tray}_2 \). 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 update.

VII. EXPERIMENTS

We 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 \([38]\), 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 robot. ROS \([41]\) 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 \(0.4 \times 0.3 \times 0.1\)m. The recorded videos of all the experiments are available in the supplementary material and following link: \(\text{https://www.youtube.com/watch?v=JNww18113dI}\) Section VII A and B demonstrate picking regularly and randomly placed objects without updating the base positions, respectively. Section VII 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.

A. REGULARLY PLACED, GLOBALLY STATIC BASE SEQUENCE

The mobile manipulator navigates in an indoor environment as shown in Fig.\([12]\) 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 \(\sigma\) is set as 10cm. Then a sequence of base positions can be planned by the algorithm described in Section \[IV-D\]. As a result, the mobile manipulator should successively move to the center of \(P_{B1} \cap P_{B2}\) and \(P_{B3}\) 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. We 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 VII C is to insert a marker in front of the tray, then the object poses, as well as the grasps, with respect to the robot are easily obtained.

For comparison, we move the robot to the center of \(P_{B1}, P_{B2}\) and \(P_{B3}\) to collect the parts from three trays, respectively. As shown in Fig.\([13]\) in each base position, the mobile manipulator picks up one object from the associated tray. Fig.\([14]\) 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 \(\text{tray}_1\) and \(\text{tray}_2\), then it

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
FIGURE 14: (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 $tray_4$.

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.

B. RANDOMLY PLACED, GLOBALLY STATIC BASE SEQUENCE

In the manufacturing environment, the mechanical components are often randomly placed in the tray. In this experiment, we use a mobile manipulator to pick up multiple mechanical components randomly placed in different trays. The overall experiment setup is shown in Fig. [15].

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 sparse and incomplete point cloud of the object, therefore we use fixed PhoXi 3D scanners to scan the mechanical components to obtain depth images. Fast graspability evaluation [37] 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. [16a] and b. Fig. [16a] represents the contact region where the gripper should contact the object and Fig. [16b] 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, we have to find a set of object poses that approximate the possible poses and plan grasps for these poses. Therefore, we 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 the possible object poses. Fig. [17] shows 4 different object placements, from the corresponding depth images grasp planning is performed to obtain grasps $G_1$, $G_2$, $G_3$ and $G_4$ for each of the object placements, respectively. Actually, for grasping one object, we only need to guarantee that at least one of the grasp candidates be reachable, this is what we did for the case of regularly placed objects. However, for the randomly placed objects, we 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 $\mathcal{G} = G_1 \cup G_2 \cup G_3 \cup G_4$ roughly represents the required grasps for grasping all the objects randomly placed in a tray. We then use $\mathcal{G}$ to calculate the base region of the tray following the method in Section IV. The grasp planning examples for other objects are shown in Fig. [18].

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. [19]. 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 relative to the robot. The online grasp planning functionality is implemented as a
FIGURE 17: 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 18: Planned grasps for different objects.

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. 20 shows the grasping of randomly placed objects during the experiment.

FIGURE 19: 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 tray1 and tray2, then it moves to the center of the third base region to pick up objects in tray4.

FIGURE 20: Pick up randomly placed objects following the planned globally static positions in Fig. 19.

C. 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 of the 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. We assume one object is picked from every target tray in every 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. We also conducted the experiment on updating the base positions based on the remaining objects, which can be seen in the supplementary video.

D. DISCUSSION ON THE EXPERIMENT RESULTS

Through the experiments, we have demonstrated the proposed planner with different policies for different application scenarios. To evaluate the efficiency of the proposed method, we 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, we consider reducing the number of base positions to visit to reduce the operation time for moving the base. Therefore, the time for moving the base is the most appropriate metric for evaluating our planner.

We measure the time for moving the base following the
Xu et al.: Submission for IEEE ACCESS

FIGURE 21: The base positions update based on the target objects to be picked, the update is performed after every round of pick-and-place task.

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

TABLE 5: Average time for moving the base in our experiments. ExperA and ExperB refer to the experiment setup in Section VII-A and Section VII-B respectively.

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. We use the experiment setup in Section VII-A and Section VII-B since their base positions do not change. We run the experiment 10 times and record the time for moving the base. The result is summarized in Table 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. 22 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 increases the average base velocity, thus reducing the operation time.

The current limitations are observed through the physical experiments. We 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, we consider using optimization-based method motion planning method to further reduce the operation time for moving the manipulator.

VIII. CONCLUSIONS
This paper 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, we considered multiple picking tasks in multiple locations, but our planner also applies to other manipulation tasks. In that case, we 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. We also discussed the object placement styles and the update of the base sequence, which are critical for practical applications.

We 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, we 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.

REFERENCES

[1] O. Khatib, K. Yokoi, O. Brock, K. Chang, and A. Casal, “Robots in human environments: Basic autonomous capabilities,” The International Journal of Robotics Research, vol. 18, no. 7, pp. 684–696, 1999.
[2] S. Chitta, B. Cohen, and M. Likhachev, “Planning for autonomous door opening with a mobile manipulator,” in 2010 IEEE International Conference on Robotics and Automation, pp. 1799–1806, IEEE, 2010.
[3] A. Pratkanis, A. E. Leapier, and K. Salisbury, “Replacing the office intern: An autonomous coffee run with a mobile manipulator,” in 2013 IEEE International Conference on Robotics and Automation, pp. 1248–1253, IEEE, 2013.
[4] K. Zhou, G. Ebenerhofer, C. Fitzerger, U. Zimmermann, C. Walter, J. Saenz, L. P. Castroño, M. A. F. Hernández, and J. N. Oriol, “Mobile manipulator is coming to aerospace manufacturing industry,” in 2014 IEEE International Symposium on Robot and Sensors Environments (ROSE) Proceedings, pp. 94–99, IEEE, 2014.
[5] Q. Yu, G. Wang, X. Hua, S. Zhang, L. Song, J. Zhang, and K. Chen, “Base position optimization for mobile painting robot manipulators with multiple constraints,” Robotics and Computer-Integrated Manufacturing, vol. 54, pp. 56–64, 2018.
[6] Y. Zhang, X. Zhao, B. Tao, and H. Ding, “Point stabilization of nonholonomic mobile robot by beizer 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, pp. 1–16, 2020.
[8] T. Yoshikawa, “Manipulability of robotic mechanisms,” The international journal of Robotics Research, vol. 4, no. 2, pp. 3–9, 1985.
[9] B. Bayle, J.-Y. Fourquet, and M. Renaud, “Manipulability of wheeled mobile manipulators: Application to motion generation,” The International Journal of Robotics Research, vol. 22, no. 7–8, pp. 565–581, 2003.
[10] F. 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, pp. 103–107, IEEE, 1990.
[11] F. G. Pin and J.-C. Culioli, “Optimal positioning of combined mobile platform-manipulator systems for material handling tasks,” Journal of intelligent and Robotic Systems, vol. 6, no. 2-3, pp. 165–182, 1992.
[12] W. F. Carriker, P. K. Khosla, and B. H. Krogh, “Path planning for mobile manipulateurs for multiple task execution,” IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 403–408, 1991.
[13] S. Vafadar, A. Olab, and M. S. Panahi, “Optimal motion planning of mobile manipulateurs with minimum number of platform movements,” in 2018 IEEE International Conference on Industrial Technology (ICIT), pp. 262–267, IEEE, 2018.
[14] J. Xu, K. Harada, W. Han, T. Ueshiba, and Y. 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), pp. 11018–11024, IEEE, 2020.
[15] Y. Yamamoto and X. Yun, “Coordinating locomotion and manipulation of a mobile manipulator,” IEEE Transactions on Automatic Control, vol. 39, no. 6, pp. 1326–1332, 1994.
[16] H. Seraji, “An on-line approach to coordinated mobility and manipulation,” in [1993] Proceedings IEEE International Conference on Robotics and Automation, pp. 28–35, IEEE, 1993.
[17] H. Seraji, “A unified approach to motion control of mobile manipulators,” The International Journal of Robotics Research, vol. 17, no. 2, pp. 107–118, 1998.
[18] F. Burget and M. Bennewitz, “Stance selection for humanoid grasping tasks by inverse reachability maps,” in 2015 IEEE International conference on robotics and automation (ICRA), pp. 5669–5674, IEEE, 2015.
[19] P. Abolghasemi, R. Rahmatizadeh, A. Behal, and B. Loloni, “A real-time technique for positioning a wheelchair-mounted robotic arm for household manipulation tasks,” in Proceedings at the Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[20] S. Ren, Y. Xie, X. Yang, J. Xu, G. Wang, and K. Chen, “A method for optimizing the base position of mobile painting manipulators,” IEEE Transactions on Automation Science and Engineering, vol. 14, no. 1, pp. 370–375, 2016.
[21] B. D. Tu, J. Zhao, and C. 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, pp. 1227–1234, American Society of Mechanical Engineers Digital Collection, 2013.
[22] D. Berenson, J. Kuffner, and H. Choset, “An optimization approach to planning for mobile manipulation,” in 2008 IEEE International Conference on Robotics and Automation, pp. 1187–1192, IEEE, 2008.
[23] R. Bankov, Automated Construction of Robot Manipulation Programs, PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.
[24] F. Stulp, A. Fedrizzi, L. Møsenlechner, and M. Beetz, “Learning and reasoning with action-related places for robust mobile manipulation,” Journal of Artificial Intelligence Research, vol. 43, pp. 1–42, 2012.
[25] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, “Positioning mobile manipulators to perform constrained linear trajectories,” in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2578–2584, IEEE, 2008.
[26] N. Vahrenkamp, T. Asfour, and R. Dillmann, “Robot placement based on reachability inversion,” in 2013 IEEE International Conference on Robotics and Automation, pp. 1970–1975, IEEE, 2013.
[27] N. Vahrenkamp, T. Asfour, and R. Dillmann, “Efficient inverse kinematics computation based on reachability analysis,” International Journal of Humanoid Robotics, vol. 9, no. 04, pp. 1250035, 2012.
[28] D. R. Budder and C. Borst, “On avoiding obstacles 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.
[29] J. Dong and J. C. Trinkle, “Orientation-based reachability map for robot base placement,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1488–1493, IEEE, 2015.
[30] F. Zacharias, C. Borst, and G. Hirzinger, “Capturing robot workspace structure: representing robot capabilities in 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3229–3234, Ieee, 2007.
[31] J. T. Feddema, “Kinematically optimal robot placement for minimum time coordinated motion,” in Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, pp. 3395–3400, IEEE, 1996.
[32] D. Hsu, J.-C. Latombe, and S. Sorkin, “Placing a robot manipulator amid obstacles for optimized execution,” in Proceedings of the 1999 IEEE international conference on robotics and automation, pp. 3395–3400, IEEE, 1996.
[33] S. Zeghloul and J.-A. Pamanés-García, “Multi-criteria optimal placement of robots in constrained environments,” Robotica, vol. 11, no. 2, pp. 105–110, 1993.
[34] S. Mitsi, K.-D. Bouzakis, D. Sagrais, 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, vol. 24, no. 1, pp. 50–59, 2008.
[35] K. Harada, K. Kaneko, and F. Kanehiro, “Fast grasp planning for hand-arm systems based on convex model,” in 2008 IEEE International Conference on Robotics and Automation, pp. 1162–1168, IEEE, 2008.
Xu et al.: Submission for IEEE ACCESS

[36] W. Wan, K. Harada, and F. Kanehiro, “Planning grasps with suction cups and parallel grippers using superimposed segmentation of object meshes,” IEEE Transactions on Robotics, 2020.

[37] Y. Domae, H. Okada, Y. Taguchi, K. Sumi, and T. Hirai, “Fast graspsability evaluation on single depth maps for bin picking with general grippers,” in 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1997–2004, IEEE, 2014.

[38] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, “Fetch and freight: Standard platforms for service robot applications,” in Workshop on autonomous mobile service robots, 2016.

[39] P. J. Kolesar, “A branch and bound algorithm for the knapsack problem,” Management science, vol. 13, no. 9, pp. 723–735, 1967.

[40] V. Černý, “Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm,” Journal of optimization theory and applications, vol. 45, no. 1, pp. 41–51, 1985.

[41] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in ICRA workshop on open source software, vol. 3, p. 5, Kobe, Japan, 2009.

WEIWEI WAN received the Ph.D. degree in robotics from the Department of Mechano-Informatics, The University of Tokyo, Tokyo, Japan, in 2013. From 2013 to 2015, he was a Post-Doctoral Research Fellow of the Japan Society for the Promotion of Science, Tokyo, and a Visiting Researcher with Carnegie Mellon University, Pittsburgh, PA, USA. He was a tenure-track Research Scientist with the National Advanced Institute of Science and Technology (AIST), Tokyo.

He is currently an Associate Professor with the School of Engineering Science, Osaka University, Osaka, Japan. His research interests are smart manufacturing and robotic manipulation. He is a member of RSJ and a senior member of IEEE.

JINGREN XU received his B.E. from Shanghai Jiao Tong University and M.E. from The University of Tokyo in 2016 and 2019, respectively. Currently, he is a PhD student in Osaka University working on robotic manipulation. His research interests include mobile manipulation, motion planning, and trajectory optimization for mobile manipulators.

YUKIYASU DOMAE received his BE, ME and PhD degrees from Hokkaido University, Japan, in 2004, 2006 and 2012, respectively. From 2008 to 2013, he worked as a Researcher in the Image Recognition Systems Group of the Advanced Technology R&D Center of Mitsubishi Electric Corporation, Japan. From 2013 to 2018, he worked as a Principal Researcher in the same group. In 2018, he joined the National Institute of Advanced Industrial Science and Technology as the Group Leader of the Manipulation Research Group. Since 2019, he works as the Team Leader of the Automation Research Team in the same institution. His research interests include machine vision and manipulation.

KENSUKE HARADA received his B.Sc., M.Sc., and Doctoral degrees in Mechanical Engineering from Kyoto University in 1992, 1994, and 1997, respectively. He worked as a Research Associate at Hiroshima University from 1997 to 2002. He worked as a research scientist at the National Institute of Advanced Industrial Science and Technology (AIST) from 2002 to 2013. For one year from 2005 to 2006, he was a visiting scholar at the computer science department of Stanford University.

From 2013 to 2015, he was the leader of the manipulation research group, AIST. Currently, he is a professor at the Graduate School of Engineering Science, Osaka University. His research interest includes mechanics and control of robot manipulators and robot hands, biped locomotion, and motion planning of robotic systems.

TOSHIKO UESHIBA obtained a M.Sc. degree from the Department of Electrical Engineering, University of Tokyo in 1986. Since obtaining a position in Electrotechnical Laboratory, Japan in the same year, he has been engaged in research in the field of computer vision. From 1997 to 1998, he joined the Institute for Information Technology of National Research Council Canada as a visiting researcher. He is currently a senior researcher at Artificial Intelligence Research Center of National Institute of Advanced Industrial Science and Technology (AIST). His research interests include geometrical aspects of computer vision, especially camera calibration, multiple view analysis, dynamic scene analysis and 3D reconstruction. He received a Best Paper Award of IPSJ National Convention in 1997 and the Yamashita SIG Research Award in 2003, both from the Information Processing Society of Japan.

***