Combining Navigation and Manipulation Costs for Time-Efficient Robot Placement in Mobile Manipulation Tasks

Fabian Reister, Markus Grotz, and Tamim Asfour

***Abstract*—Mobile manipulation tasks require a seamless integration of navigation and manipulation capabilities. Finding suitable robot placements to pick up and place objects in such tasks is crucial for time-efficient task execution. Sub-optimal robot placements result in infeasible solutions or require larger repositioning of the mobile base to reach target objects, increasing the overall time to complete the task. In this work, we propose an approach that, given a set of objects, autonomously selects the optimal placements of a humanoid robot in conjunction with the best grasp candidate and corresponding arm. In contrast to previous approaches, our method considers both the navigation costs between consecutive robot placements and the manipulation costs to reduce the time needed to complete the task. We evaluate our method on a simulated table clearing task that requires the robot to move between pickup and discard locations and demonstrate the applicability in a real-world experiment on the humanoid robot ARMAR-6. In addition, we perform a run-time analysis and show that our approach can integrate sensory feedback to update the optimal placement in dynamic environments.**

***Index Terms*—Mobile manipulation, motion and path planning, integrated planning and control**

I. Introduction

EFFICIENTLY performing mobile manipulation tasks [1] such as clearing a table or loading a dishwasher is challenging, as it requires a seamless integration of navigation and manipulation capabilities to successfully perform these tasks by robots with usually high number of degrees of freedom. In a mobile manipulation task, a mobile robot like ARMAR-6 (Fig. 1) has to reach several poses by its hands to grasp objects, to follow a certain trajectory of its tool center point (TCP) e. g. while opening a door, to coordinate arm and platform motions and to execute these motions based on sensor feedback. Here, a key aspect is finding suitable robot placements, i. e. positions of the platform, that allow the robot to reach all desired TCP poses without the need of re-positioning during the execution. To increase efficiency in solving a task, unnecessary movements e. g. due to repositioning the platform must be avoided by determining a placement that is optimal for the next manipulation action e. g. grasping or placing action or, in the case of a bimanual mobile robot, by selecting the most appropriate hand to grasp an object.

Manuscript received: February, 24, 2022; Revised May, 28, 2022; Accepted June, 23, 2022. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Competence Center Karlsruhe for AI Systems Engineering (CC-KING) funded by the Ministry of Economic Affairs, Labour and Tourism Baden-Württemberg.

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany {fabian.reister, asfour}@kit.edu

Digital Object Identifier (DOI): see top of this page.

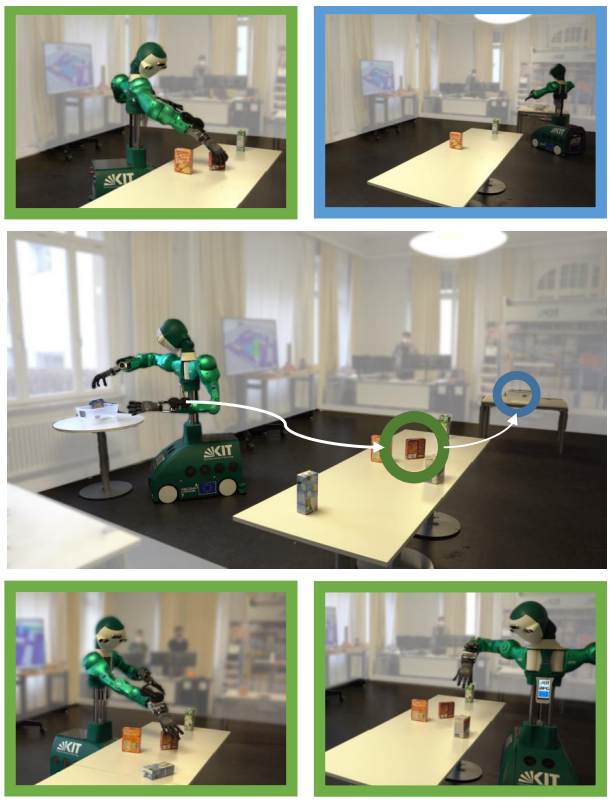


Fig. 1: The humanoid robot ARMAR-6 performing a sequence of mobile manipulation tasks. The figure shows the initial pose as well as exemplary pickup placements. The placements in the bottom row are suboptimal as the arm is close to joint limits (left) or the robot needs to move around the table (right). The top row shows the placements found by our approach.

In this work, we consider the mobile manipulation task of clearing multiple objects from a table. The task involves grasping and subsequently placing multiple objects at a single or multiple locations.

One key aspect of solving this task is to find whole-body goal configurations including the robot's base and the arm to manipulate the object. An optimal robot placement can reduce the total amount of time needed to perform a single pick-and-place action as it minimizes the navigation distance and time while maintaining a high manipulability of the end-effector. Such optimal robot placements not only allow to efficiently manipulate a single object, but also reduce the overall time on a sequence of object manipulations. Fig. 1 illustrates an example for such a pick-and-place task comprising multiple mobile manipulation actions. The goal is to pick up several objects at given positions and place them elsewhere. To

achieve this goal, the robot needs to jointly select the grasp candidate alongside with the corresponding hand as well as the base placement to efficiently perform the task. Planning over the large sequence of object interactions is neither possible in real-time nor meaningful in dynamic environments, in which the robot or a human might change the object arrangement on the table during the task execution.

In our previous work, we introduced reachability inversion [2], by which potential robot base placements for a fixed TCP position can be calculated, instead of defining the reachability of the TCP w.r.t. a fixed robot base pose. We build on our previous work and extend it for the calculation of multiple, optimal robot placements in a mobile manipulation task while taking into account the navigation costs to these placement positions.

Our major contribution in this work is a novel approach for planning of mobile manipulation task that determines consecutive robot placements and grasp candidates jointly in a way that minimize the overall time to complete the task while combining navigation and manipulation tasks. To achieve this, movements of the mobile base and TCP are coordinated to arrive simultaneously at the grasp pre-pose and mobile base placement by introducing navigation and manipulation cost maps (Section IV). The former can be computed online and also be used for optimal navigation planning once the placement is found (Section IV-A). The latter can be derived from prior knowledge about the robot's reachability (Section IV-B). We validate our approach by comparing against several baselines (Section V) within a simulated environment as well as in a real experiment on the humanoid robot ARMAR-6 as shown exemplary in Fig. 1. We also perform a computation time analysis on ARMAR-6 in a dynamic environment.

II. Related work

Finding consecutive robot placements as part of a mobile manipulation task is important for efficient execution and challenging as it requires the generation of kinematically feasible whole-body motions by combining perception, motion planning and control. Pioneering work on generating collision-free whole-body motions has been conducted by Khatib [1] and Brock [3]. In their Elastic Strips framework, they assume that the goal and also an initial collision-free path is given. Approaches that include goal generation can be divided into two categories depending on whether the placement of the mobile platform of the robot for executing manipulation actions is determined implicitly or explicitly.

A. Implicit robot placement

Implicit approaches are often reactive and usually generate platform velocities directly to ensure kinematic feasibility. The authors in [4] use a dynamical systems based approach in conjunction with an approximation of the inverse reachability to generate base and TCP movements. In [5], reinforcement learning is used to control the platform and use the kinematic feasibility as a reward such that the agent learns to move the platform towards the desired goal region. In [6], a whole-body quadratic program is formulated to generate smooth motions for repetitive pick-and-place actions. As the manipulability is considered into the objective function, the controller can locally maximize the manipulability. However, no guarantee can be given regarding the global optimality of the solution.

Although these approaches demonstrate their applicability in dynamic environments, they do not consider the ambiguity and decision that can be made during execution. In both [4] and [5], the right arm is used to perform the action instead of selecting the most suitable one. Furthermore, multiple strategies exist, e. g. to open a door (grasping the handle from top or bottom) or how to grasp an object (multiple grasp candidates), which can improve the overall execution.

B. Explicit methods

The optimal placement can also be planned in advance. Due to the redundancy of the robot's arm, the placement distribution for the pose of the mobile base cannot be described analytically. One approach is known as inversion of reachability maps [2],[7],[8]. Reachability maps represent the robot's ability to reach a certain TCP pose w.r.t. a fixed base position and are calculated by sampling joint configurations q_i, applying forward kinematics to obtain the robot's TCP pose within its base frame ${}^{\text{robot}}\boldsymbol{T}_{\text{TCP}}$ and evaluating a suitable cost function $c(q)$ of the joint configuration q which yields the tuple $({}^{\text{robot}}\boldsymbol{T}_{\text{TCP}}(q), c(q))$. These reachability maps are usually approximated using a fixed-sized 6D grid as in [2] and [8], a task with the complexity of $\mathcal{O}(n^6)$ with n being the number of discretization steps of both position and orientation. To obtain the distribution of the robot placements for a certain TCP trajectory, the inverse reachability map (IRM) is then generated by inverting ${}^{\text{robot}}\boldsymbol{T}_{\text{TCP}}$ and cutting it with the plane describing all robot base positions. In [2] it is assumed that the robot moves within the plane parallel to the ground plane. Recently, this has been extended to surface-orientated reachability maps [9]. To overcome the limitation of the grid-based approximation, [10] learn the reachability and inverse reachability directly by employing density networks.

Selecting the proper cost function $c(q)$ as reachability measure is highly task dependent. It can be a statistical measure of the hit frequency of a cell within the 6D grid, the manipulability and also an extended manipulability as described in [2].

If the inverse mapping is learned directly, it can go beyond joint-space criteria. In [11] a probabilistic placement distribution is directly learned from experience by so-called action-related places. Although being used for scene exploration and object search, the authors in [12] show how multiple criteria such as arm IK, arm reachability, object observability and base reachability can be combined.

Lately, also probabilistic methods have been proposed that reflect the abilities of the perception system to capture platform pose uncertainty [13] as well as uncertainty about the grasp pose [14] or the object pose itself [15].

As the IRMs do not consider collisions of the found solutions, a post-processing step is required to validate the proposed placements. To overcome this limitation, inverse

dynamic reachability maps (iDRM) were proposed in [16] to efficiently incorporate the dynamic scene by means of a voxelized representation such that whole-body goal configurations can be generated in real-time.

C. Consecutive robot placement

Finding a sequence of robot base placements has been studied in the context of coverage path planning [17],[18], inspection [19] and assembly at the example of pick-and place-tasks [20],[21]. In most cases, the objective is to reduce the number of base poses required to fulfill the overall tasks. For example, in [22] a sequence of pick-and-place actions is planned to minimize the number of required robot placements by intersecting base regions defined to reach different boxes containing the objects to be grasped. Yet, their approach is restricted to a fixed orientation of robot.

In [21], an approach is proposed to minimize the number of base placements as well as the distance traveled between these. The formulation is similar to the traveling salesman problem, which can be solved by using simulated annealing. Due to the large search space, the approach does not optimize the orientation of the robot, which is a major restriction of possible placements. Also, the optimal placement is only chosen based on the navigation costs, which can have a major impact on the manipulability. In addition, such a global optimization approach is time consuming. In experiments, the authors report a planning time of over one minute, which limits their approach to offline planning problems.

Yet, most of the presented approaches do not consider the time required to navigate between the placements which, especially for household tasks, can be a dominant factor. The authors in [23] propose an approach which only considers the navigation costs towards the target. Their approach describes a two-part placement strategy in which a first placement on a circle around the object is used to perceive an object. A second placement is then generated based on inverse reachability maps to grasp the object which minimizes the distance to the first one. Yet, their approach does not consider the travel costs to any subsequent target.

III. Problem Formulation

We address the problem of finding multiple consecutive robot placements from which TCP trajectories can be executed. This can be e. g. pick-and-place or other manipulation task such as opening a dishwasher. The goal is to perform such tasks in an efficient way, i. e. as fast as possible. Formally, we define the problem of finding consecutive robot placements as finding feasible placements $\mathcal{R} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_n)$ with $\boldsymbol{r}_i = (x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ from an initial pose $\boldsymbol{r}_0$ to reach a set of n objects at the 6D poses $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ with corresponding TCP trajectories $\tau_1, \dots, \tau_n$. In each step, the concrete TCP trajectory τ_i has to be selected out of a set T_i. In the concrete table clearing scenario, this TCP trajectory must be selected and derived based on a set of grasp candidates associated with the target object and selected arm.

The goal is to find such consecutive robot placements as efficient as possible, i. e. the placements that minimize the objective function

$$f(\mathcal{R}) = \sum_i c(\boldsymbol{r}_{i-1}, \boldsymbol{r}_i, \boldsymbol{r}_{i+1}), \tag{1}$$

with the task specific cost function $c(\cdot)$ that depends of the previous $i-1$ and the next placement $i+1$ if e. g. the distance between the placements is considered. Consequently, this is a coupled optimization problem. In this work, we set $c(\cdot)$ to be the time spent to complete the current task.

IV. Approach

An overview of our approach is given in Fig. 2. We assume that the following is given or can be derived from the robot sensor observations: (i) a scene with n known objects and an object localization algorithm to estimate their 6D poses, (ii) a high-level plan which defines the order in which the objects need to be manipulated, (iii) a grasp planner to generate grasp candidates for the objects and suitable hand trajectories to successfully grasp them, (iv) the navigation cost map (Fig. 3), (v) the reachability maps for the left and right arm, and (vi) the initial robot pose $\boldsymbol{r}_0$. To find time-efficient robot placement, we use a simplified objective function $\tilde{c}$ to approximate c as

$$c \approx \tilde{c} = c_n + c_m. \tag{2}$$

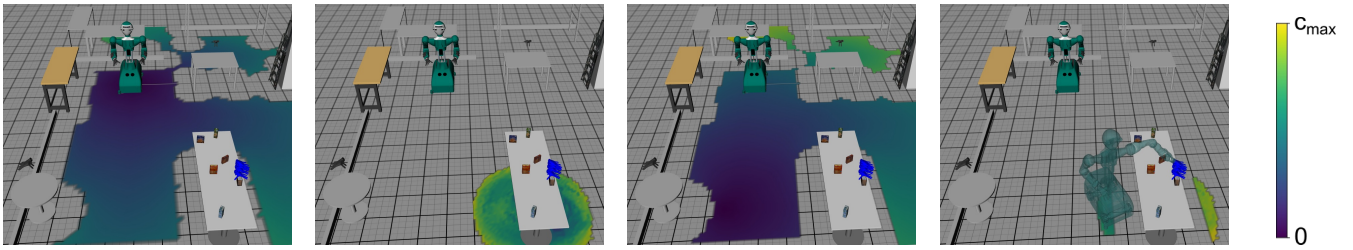
The approximation $\tilde{c}$ consists of two parts: 1) navigation-related costs c_n and 2) manipulation-related costs c_m. The first term in this formulation captures the mobility costs between the robot base placements while the second term evaluates the robot base pose w.r.t. the desired manipulation task. We assume that the time to grasp the object from a pre-pose associated with a grasp is constant and thus we omit it from the optimization problem. Yet, grasp specific costs could easily be integrated into our optimization framework.

The navigation costs c_n are independent of the TCP trajectory but depend on the current and its surrounding robot base poses. Thus, minimizing $\tilde{c}$ becomes a coupled global optimization problem. Due to the curse of dimensionality, optimizing $c_n = \sum_i f(\boldsymbol{r}_{i-1}, \boldsymbol{r}_i, \boldsymbol{r}_{i+1})$ in real-time becomes intractable. Therefore, we suggest to split $\tilde{c}_n$ into two parts and re-formulate the problem as a local optimization problem:

$$c_n \approx c_n(\boldsymbol{r}_{i-1}, \boldsymbol{r}_i) + c_{n,h}(\boldsymbol{r}_i, \boldsymbol{r}_{i+1}). \tag{3}$$

The cost $c_n(\boldsymbol{r}_{i-1}, \boldsymbol{r}_i)$ can be computed by a suitable search algorithm from the initial pose $\boldsymbol{r}_{i-1}$ to a goal region $\mathcal{R}_i$ ($\boldsymbol{r}_i \in \mathcal{R}_i$). As $c_n(\boldsymbol{r}_i, \boldsymbol{r}_{i+1})$ would require to search through $\mathcal{R}_i \times \mathcal{R}_{i+1}$, we approximate it by a suitable heuristic. This detailed computation of c_n will be described in Section IV-A. Even if solving the coupled optimization problem for c_n would be feasible, it would have limited practical applicability in dynamic environments as objects might be moved between pick-and-place actions, e. g. by a human or by the robot itself either unintended or intended by pushing objects aside.

The manipulation costs $c_m(\boldsymbol{r}_i, \tau_i)$ only depend on the TCP trajectory and the current robot placement. The corresponding cost function will be described in Section IV-B.



(a) Navigation costs (b) Manipulation costs (c) Navigation heuristic (d) Optimal placement (e) Legend

Fig. 2: Overview of the proposed method to find robot placements for time-efficient task execution. In this example, the robot's task is to grasp an object from the table on the right and place it on the table in the lower left corner. To find the optimal placement, the method considers multiple cost terms. First, the navigation costs that arise from navigating from the given start pose. Second, the manipulation costs that reflect the manipulability of the end-effector while executing the TCP's grasp trajectory shown in blue. And third, a heuristic to approximate the navigation costs to reach the placement area on the left. By combining all cost maps, the robot can find the optimal base placement.

A. Navigation costs to target

In order to navigate safely and reliably, the robot should avoid close proximity to obstacles. If the robot however needs to move close to objects, it should lower its velocity to be able to react to unforeseen situations and to come to rest before a collision occurs. Therefore, we model the maximal linear velocity at a position $\boldsymbol{p} = (x, y)$ as

$$v(\boldsymbol{p}) = \frac{v_{max}}{1 + \lambda \cdot d_s(\boldsymbol{p})}, \tag{4}$$

with d_s describing the distance to obstacles and d_o the distance to the closest obstacle

$$d_s(\boldsymbol{p}) = \begin{cases} (1 - d_o(\boldsymbol{p})/d_{max})^k & \text{if } d_o(\boldsymbol{p}) < d_{max} \\ 0 & \text{otherwise.} \end{cases}$$

In our experiments, we set the weighting factor λ to 1.3, k to 4 and the maximum distance d_{max} that we consider to 1 m. To find the time-optimal path from a start position to all possible positions in the map, we discretize the search space and convert it to a graph, in which each vertex represents the position within the map and is connected to its 8 surrounding vertices by directed edges. We choose the weight $e_{i,j}$ of an edge connecting vertex n_i and n_j to be consistent with the velocity described by Eq. 4 as

$$e(n_i, n_j) = \|\boldsymbol{p}(n_i) - \boldsymbol{p}(n_j)\| \cdot (1 + \lambda \cdot d_s(\boldsymbol{p}(n_j))). \tag{5}$$

Similar to [24] we use the Shortest Path Faster Algorithm (SPFA) to obtain the minimal costs to reach each node within the graph. However, in [24], only the Euclidean distance is considered as edge weights. In our case, we decouple the computation of d_o and the graph search by pre-computing and maintaining a cost map as depicted in Fig. 3. This allows continuous update based on perceptual information provided by the robot's sensor system.

An exemplary cost map with the distance to the nearest obstacle as well as the navigation costs from an initial start pose is given in Fig. 3. It can be seen that proximity to obstacles is avoided.

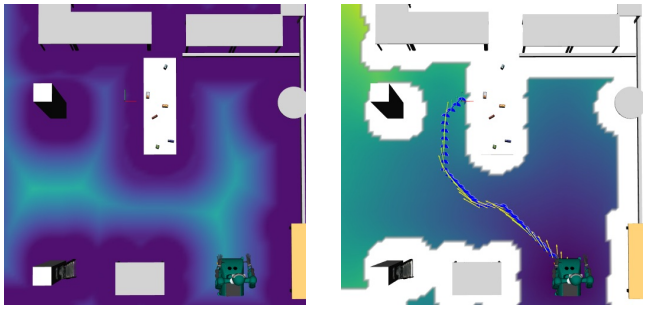


Fig. 3: **Left:** The distance to obstacles cost map. The color coding similar to Fig. 2 indicates the distance to the closest obstacle. **Right:** Cost map obtained by SPFA. In addition, an exemplary distance-aware shortest path is depicted.

B. Manipulation costs

To obtain the manipulation costs, we generate reachability maps (RMs) for ARMAR-6 using the sampling-based strategy described in [2] for the left and right arm. By randomly sampling joint configurations, applying forward kinematics and evaluating the joint configuration, the corresponding cell in the lookup table of the 6D grid is filled. Here, we use the extended manipulability measure that, compared to the Yoshikawa manipulability index [25], takes additional constraints such as joint limits and distance between body parts into account as described in [2]. By optimizing for a high manipulabilty measure, the robot will be able to reach the desired TCP pose even in the presence of object pose estimation and self-localization uncertainty.

To obtain a distribution of the robot's possible placements, we perform reachability inversion as described in [2] for the given TCP trajectory. For each TCP pose of the trajectory, we generate the IRM and combine them into one, which contains the minimal extended manipulability of the individual IRMs.

However, each of the combined IRMs only corresponds to a specific orientation of the robot base. Therefore, we generate n inverse reachability maps to cover the meaningful orientation space. As the robot's head rotation is limited, we restrict the

workspace such that the TCP must be in front of the robot and can only be moved up to $45°$ into the workspace of the other arm. Instead of storing n maps, we combine them into two where the first contains the largest reachability value and the second the corresponding orientation. To ensure a high manipulability, we remove those parts from the inverse reachability map that are below a threshold of 0.0235.

C. Navigation costs for heuristic

To estimate the navigation costs to the next TCP trajectory, we combine both approaches presented in Section IV-A and Section IV-B. Namely, we first compute the reachable sets of robot placements $\mathcal{R}_{i+1,j}$ within $\mathcal{R}_{i+1}$ and identify connected regions as depicted in Fig. 4.

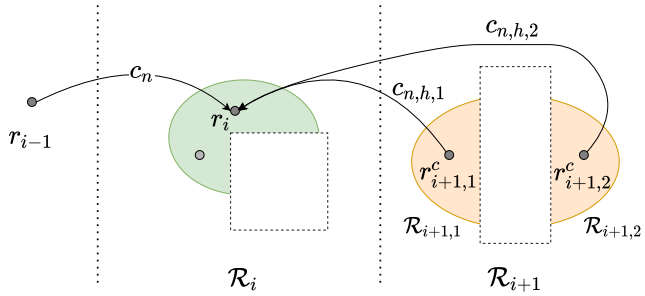


Fig. 4: The computation of the navigation costs for a placement $\boldsymbol{r}_i$. It consists of the navigation costs from the last placement $\boldsymbol{r}_{i-1}$ towards the goal region $\mathcal{R}_i$ and a heuristic that allows us to approximate the navigation costs from an arbitrary placement $\boldsymbol{r}_i$ towards the next goal region $\mathcal{R}_{i+1}$. In the given case, two rectangular objects restrict the placement sets.

These reachable sets $\mathcal{R}_{i+1,j}$ can e. g. represent the left and right side of a table. For each of the centers of the regions $\boldsymbol{r}^c_{i+1,j}$, we then create the navigation cost maps as described in Section IV-A. Finally, we combine the cost maps by a weighted sum based on the area of the region. By doing so, we keep the bias towards any of the possible placements in $\mathcal{R}_{i+1}$ low.

D. Obtaining the optimal placement

Let ζ_n be the navigation cost map initialized from the start position $\boldsymbol{r}_{i-1}$, ζ_m be the manipulation cost map and $\zeta_{n,h}$ be the navigation cost map towards $\mathcal{R}_{i+1}$ as depicted in Fig. 2, then in Eq. 2 becomes

$$c \approx \tilde{c} = c_n + c_m = \lambda_n \zeta_n + \lambda_m \zeta_m + \lambda_{n,h} \zeta_{n,h}. \qquad (6)$$

As each cost map is associated with a mask to represent the feasible regions, we compute the union of region to obtain the feasible region $\mathcal{R}_i$ as depicted in Fig. 2d. The optimal placement for this grasp candidate is the one that minimizes $\tilde{c}$. Consequently, we are then given a set of global optimal placements w.r.t. our cost function for objects as shown in Fig. 5. The best placement does not only have to minimize $\tilde{c}$ but it has to be reachable by the bimanual arm controller. To ensure this, we perform a reachability check and reject the unreachable base placements for the set of grasp candidates.

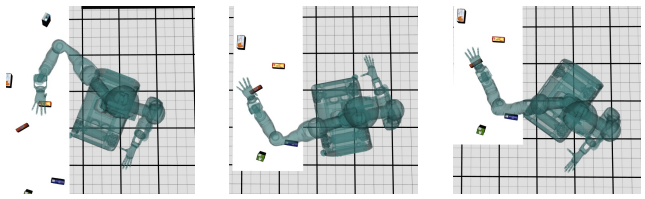


Fig. 5: Exemplary optimal placements and corresponding IK solutions found by our method. Grasping with the right hand (left), grasping with the left hand (middle), grasping with the right hand and flipped elbow (right).

V. Experiments and Evaluation

We evaluated our approach regarding finding time-efficient robot placements in the context of *clearing the table* task with three tables as shown in Fig. 6. We conducted 1) experiments in simulation to provide a quantitative evaluation of the approach, 2) a real-world experiment with a qualitative evaluation on a humanoid robot, and 3) a real-world experiment in a dynamic environment with a quantitative evaluation of the computation time. An accompanying video shows the experiments.

A. System setup

We use the humanoid robot ARMAR-6 [26] for both simulated and real-world experiments. The robot features two 8 DoF arms and five-finger hands and is equipped with a holonomic mobile platform. The approach is implemented within the ArmarX robot framework[1] , extends our previous work [2] as part of the robotics toolbox Simox[2] and uses the ArmarX navigation stack[3] to move the platform between the placements. The global planning step is also based on the cost function described in Section IV-A. To generate the arm movement, we designed a trajectory planner inspired by RelaxedIK [27]. The quadratic program considers multiple objectives and can jointly optimize for the desired TCP poses, avoid joint limits and singularities, and ensures task-space and joint-space smoothness as well as maintaining a certain height above the tables to ensure collision-free movement. Furthermore, for each object to be grasped, a set of grasp candidates using a bounding-box-based grasp planner [28] are generated and used. In this work, we only consider top grasps. To reduce noise induced by visual object localization and by the grasp planner, we assume fixed object poses during every experiment. Throughout the real-world experiments and also to measure the execution time we use a computer with an Intel(R) Core(TM) i7-6700 CPU @ 3.60GHz and 32GB RAM.

B. Baselines

To quantitatively assess our approach we implemented the following baselines:

[1] https://gitlab.com/ArmarX
[2] https://gitlab.com/Simox
[3] https://gitlab.com/ArmarX/skills/navigation/

1) *Pre-defined placements around the table.* For each of the three pick-up tables and on each side of the table, a centered placement is generated with the robot facing towards the table. This yields seven possible placements where we select the one that maximizes the extended manipulability while reaching the grasp pose. This can be seen as an engineered solution to cover the whole table.
2) *Inverse Reachability Maps (IRM).* By setting both navigation cost weights λ_n and $\lambda_{n,h}$ in Eq. 6 to zero, our approach is similar to the reachability inversion approach in [2]. We use the extended manipulability measure similar to our approach.

In addition, we perform an ablation study by comparing the greedy *single-step* and our *sequential* approach by setting the navigation costs for the heuristic $\lambda_{n,h}$ to zero for the first one. If applicable, we chose $\lambda_n = \lambda_{n,h} = \frac{1}{20\,\mathrm{m}}$ and $\lambda_m = 0.002353$.

C. Scenarios

We conducted the experiments in two scenarios in simulation with and without obstacles and in two real world experiments.

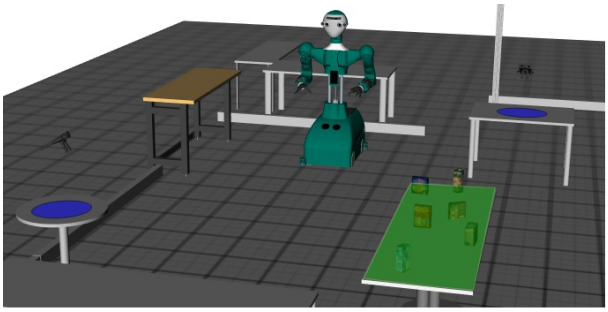


Fig. 6: ARMAR-6 in a simulated scene. The objects are placed on the tables within the green region. Placing areas for grasped object are highlighted in blue.

1) Simulated evaluation scenario: We place the objects on several tables that span an area of $0.8 \times 2.4\mathrm{m}$ as depicted by the green area in Fig. 6. Due to the large workspace of the arms, the robot can reach all positions on the table from both sides. The robot's task is to place the objects alternating on one of the tables highlighted in blue. To reflect the dynamic nature of a table clearing task, we vary the scene by placing the objects randomly on the table and repeat the experiments 45 times. To assess the time to complete the task in simulation, we focus on the contact-free motions. As the concrete grasping execution depends on force-feedback, it can only be modelled insufficiently and therefore we we do not execute the compliant grasp controller in simulation.

2) Simulated evaluation scenario with obstacles: As shown in Fig. 7, we place obstacles around the table which are indicated by red boxes. This restricts the set of feasible robot placements. For each approach, we evaluate how the execution time is affected and repeat the experiment 30 times.

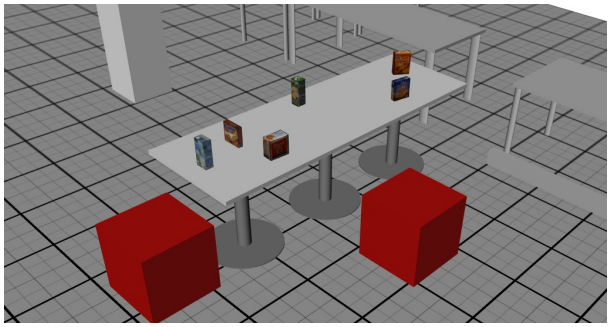


Fig. 7: Simulated scene with two obstacles around the tables. The robot has to find suitable robot placement in the free area around the obstacles, i.e. the two red boxes, to reach and grasp the objects.

3) Real-world robot experiment: We executed the experiment described above also once on the robot with a similar object arrangement as in one of the simulation experiments.

4) Real-world evaluation scenario in dynamic scene: To demonstrate the applicability of our approach in dynamic environments, we let a human enter the scene to update the object arrangement and obstacles as depicted in Fig. 8.

The robot's task is to plan and dynamically update the optimal placement for a top grasp with the right hand. Therefore, we integrate sensory feedback online: 6D object pose estimation [29] is used to update the object-centric manipulation cost maps. In addition, a laser-scanner-based obstacle detection founds the basis of the dynamic navigation cost maps. All cost maps are combined according to Eq. 6 to obtain the optimal robot placement. We record both the initial planning and online update times for 10 different scene configurations while the robot is observing the scene from its initial pose for 2 minutes each.

Fig. 8: Dynamic real-world evaluation scenario. During the experiment, the human moves around and varies the pose of the object that the robot is about to grasp. In addition, the human can move the chairs and boxes on the floor.

D. Results

1) Simulated evaluation scenario: The quantitative evaluation results are shown in Fig. 10, for the simulated scenario

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

Fig. 9: Real-world experiments with ARMAR-6. Our method allows for time-efficient robot placements to increase the robot's utility. Pictures (a)-(d) are selected at different times during the experiment. The accompanying video shows the complete experiment.

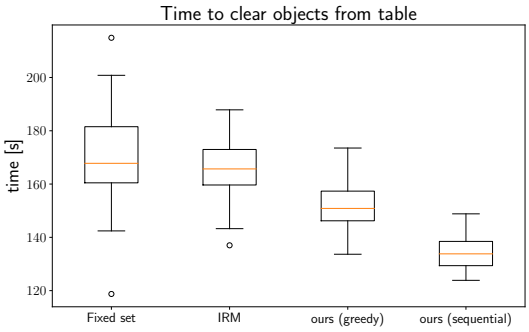


Fig. 10: Execution time for the simulated evaluation scenario.

described in Section V-C1. As can be seen, our approach outperforms all baselines. Compared to the fixed set of possible robot placement and IRM, we achieve a reduction of the time to clear the objects from the table by $21\,\%$ and $18.6\,\%$ for the simulated scenario without obstacles.

2) Simulated evaluation scenario with obstacles: For the simulated scenario with obstacles, we observed a reduction of $18.4\,\%$ in comparison to the fixed set placement and a reduction of $17.7\,\%$ compared to IRM as shown in Fig. 11.

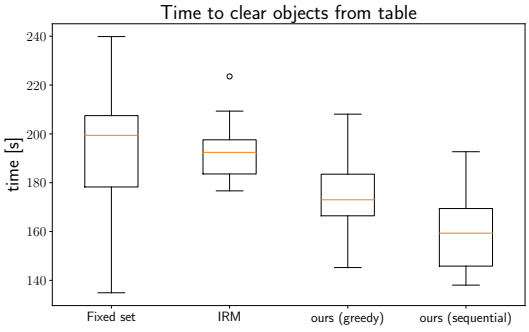


Fig. 11: Execution time for the simulated evaluation scenario with obstacles.

3) Real-world evaluation: When executing the experiment on the real robot, we recorded the following execution times as shown in Tab. I. We include the estimated time from the simulation as a reference. It can be seen that the simulation slightly underestimates the corresponding execution time on the robot. Furthermore, the assumption of a constant grasping time holds

method	simulation	real (navigation)	real (full)
IRM	152.9 s	163 s	202 s
ours	132.3 s	135 s	176 s

TABLE I: Execution time for the real experiment and its simulated counterpart. The simulated experiment does not include the grasping and should ideally be similar to the navigation time of the real experiment.

which is about $40\,\mathrm{s}$ for both approaches. The qualitative results of the real experiment are shown in Fig. 9. The used bimanual arm controller is able to successfully execute the pick-and-place actions from the placements generated by our approach. It can be summarized that by using our approach, the robot is able to clear the table fully autonomously.

4) Real-world evaluation (dynamic): Tab. II reports the planning time to obtain the manipulation and navigation cost maps without any CPU parallelization. The computation time

computation step	time [ms]	
navigation cost map	1.39	± 0.18
manipulation cost map	1053.94	± 48.83
navigation heuristic cost map	58.20	± 5.92
combined cost maps + best placement	4.93	± 1.84
overall	1118.49	± 50.87

TABLE II: Planning time for a single robot placement.

of the inverse reachability maps is the most dominant factor. To obtain the manipulation cost map, 100 IRMs are computed, each for a different robot base orientation.

After the initial planning phase, the manipulation cost map can be updated without the need to recalculate the IRMs. Tab. III lists the update time of the cost maps. As the distance-to-obstacle cost map (Fig. 3) is updated by integrating low-level laser-scanner-based features, we also record its computation time. It can be summarized that our approach

computation step	time [ms]	
distance-to-obstacle cost map	90.25	± 17.0
navigation cost map	1.45	± 0.52
manipulation cost map	0.70	± 0.49
navigation heuristic cost map	2.78	± 1.25
combining cost maps + best placement	5.89	± 2.39

TABLE III: Update time for a single robot placement.

can integrate sensory feedback efficiently to be applicable in dynamic environments.

VI. Conclusion and Future Work

We presented a novel approach for planning of mobile manipulation tasks by combining navigation and manipulation costs for time-efficient robot placements. Our method orchestrates the robot's holonomic mobile base as well as both arms to arrive simultaneously with the end-effector at the object to grasp. We evaluated our method in several simulated and two real-world experiments. The evaluation shows an overall reduction of the time to complete the task. Especially, the navigation time compared to a baseline approach using classical reachability inversion. The conducted run-time analysis in a dynamic environment shows that our method can integrate new sensor data online to update both navigation and manipulation related costs. Future work includes more refined and reactive whole-body control in which the base placement generated by our approach can serve as a prior to overcome local minima. We will also integrate success and failure prediction to make our system more robust.

References

[1] O. Khatib, "Mobile manipulation: The robotic assistant," *Robotics and Autonomous Systems*, vol. 26, pp. 175–183, 1999.

[2] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1970–1975.

[3] O. Brock and O. Khatib, "Mobile manipulation: Collision-free path modification and motion coordination," in *International Conference on Computational Engineering in Systems Applications*, 1998, pp. 1–6.

[4] T. Welschehold, C. Dornhege, F. Paus, T. Asfour, and W. Burgard, "Coupling Mobile Base and End-Effector Motion in Task Space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, 2018, pp. 7158–7163.

[5] D. Honerkamp, T. Welschehold, and A. Valada, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6289–6296, 2021.

[6] J. Haviland, N. Sunderhauf, and P. Corke, "A holistic approach to reactive mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3122–3129, 2022.

[7] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, "Positioning mobile manipulators to perform constrained linear trajectories," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2578–2584.

[8] A. Makhal and A. K. Goins, "Reuleaux: Robot base placement by reachability analysis," in *IEEE International Conference on Robotic Computing (IRC)*, 2018, pp. 137–142.

[9] C. P. Timo Birr and T. Asfour, "Oriented surface reachability maps for robot placement," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1–7.

[10] S. Kim and J. Perez, "Learning Reachable Manifold and Inverse Mapping for a Redundant Robot manipulator," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4731–4737.

[11] F. Stulp, A. Fedrizzi, and M. Beetz, "Action-related place-based mobile manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 3115–3120.

[12] M. Attamimi, K. Ito, T. Nakamura, and T. Nagai, "A planning method for efficient mobile manipulation considering ambiguity," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 965–972.

[13] Y. Meng, Y. Chen, and Y. Lou, "Uncertainty Aware Mobile Manipulator Platform Pose Planning Based on Capability Map," in *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2021, pp. 123–128.

[14] S. Suzuki, D. Endo, and K. Yamazaki, "Posture evaluation for mobile manipulators using manipulation ability, tolerance on grasping, and pose error of end-effector," *Advanced Robotics*, vol. 35, no. 10, pp. 603–618, 2021.

[15] D. Leidner, A. Dietrich, F. Schmidt, C. Borst, and A. Albu-Schaffer, "Object-centered hybrid reasoning for whole-body mobile manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1828–1835.

[16] Y. Yang, V. Ivan, Z. Li, M. Fallon, and S. Vijayakumar, "iDRM: Humanoid motion planning with realtime end-pose selection in complex environments," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 271–278.

[17] F. Paus, P. Kaiser, N. Vahrenkamp, and T. Asfour, "A combined approach for robot placement and coverage path planning for mobile manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6285–6292.

[18] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Optimal TCP and Robot Base Placement for a Set of Complex Continuous Paths," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9659–9665.

[19] M. Forstenhäusler, T. Wetner, and K. Dietmayer, "Optimized Mobile Robot Positioning for better Utilization of the Workspace of an attached Manipulator," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 2074–2079.

[20] I. Chaves-Arbaiza, D. García-Vaglio, and F. Ruiz-Ugalde, "Smart Placement of a Two-Arm Assembly for An Everyday Object Manipulation Humanoid Robot Based on Capability Maps," in *IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, 2018, pp. 1–7.

[21] K. Harada, T. Tsuji, K. Kikuchi, K. Nagata, H. Onda, and Y. Kawai, "Base position planning for dual-arm mobile manipulators performing a sequence of pick-and-place tasks," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 194–201.

[22] J. Xu, K. Harada, W. Wan, T. Ueshiba, and Y. Domae, "Planning an efficient and robust base sequence for a mobile manipulator performing multiple pick-and-place tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11 018–11 024.

[23] S. Natarajan, S. Kasperski, and M. Eich, "An Autonomous Mobile Manipulator for Collecting Sample Containers," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2014, pp. 1–8.

[24] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser, "Spatial Action Maps for Mobile Manipulation," in *Proceedings of Robotics: Science and Systems*, 2020, pp. 1–10.

[25] T. Yoshikawa, "Manipulability of robotic mechanisms," *International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[26] T. Asfour, M. Wächter, L. Kaul, S. Rader, P. Weiner, S. Ottenhaus, R. Grimm, Y. Zhou, M. Grotz, and F. Paus, "ARMAR-6: A High-Performance Humanoid for Human-Robot Collaboration in Real-World Scenarios," *Robotics Automation Magazine (RAM)*, vol. 26, no. 4, pp. 108–121, 2019.

[27] D. Rakita, B. Mutlu, and M. Gleicher, "Relaxedik: Real-time synthesis of accurate and feasible robot arm motion." in *Robotics: Science and Systems*, 2018, pp. 26–30.

[28] R. Grimm, M. Grotz, S. Ottenhaus, and T. Asfour, "Vision-based robotic pushing and grasping for stone sample collection under computing resource constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6498–6504.

[29] P. Azad, D. Münch, T. Asfour, and R. Dillmann, "6-dof model-based tracking of arbitrarily shaped 3d objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5204–5209.