Combining Motion Planning and Task Assignment for a Dual-arm System

Carlos Rodríguez and Raúl Suárez

Abstract—This paper deals with the problem of combining motion and task assignment for a dual-arm robotic system. Each arm of the system performs independent tasks in a cluttered environment. Robot actions are determined to remove potential obstacles and obtain collision-free paths to grasp the target objects. The approach uses the information provided by the motion planner to build a graph structure in order to represent the obstacles to be removed. The graph is used, first, to decide which is the next motion path to be computed, and, second, to assign the tasks to each arm of the robotic system. The approach has been implemented for a dual-arm robotic system and real experiments are presented in the paper.

I. INTRODUCTION

Object manipulation in a cluttered environment is a task that humans perform daily. Different types of problems are found when humans are replaced with robots that have to perform this type of tasks, like the motion planning in order to move the robot from an initial to the goal configuration, the computation of the object grasping, and the computation of a path to move the object from its initial configuration to a new goal configuration. This kind of problems are considered classics in the literature in robotics.

On the other hand, there are higher level problems to be solved, like planning for task distribution or the precedence relationship between tasks. In the case of two or more robots that execute tasks in a shared environment, planning the task assignment/distribution to each robot is a relevant problem, i.e. defining which robot is in charge of each task, and how the tasks are prioritized in order to avoid collisions of the robots with the obstacles of the environment as well as collisions between the robots.

In this context, this work deals with the motion and task assignment problem for a dual-arm robot in a cluttered environment. Each arm is used in an independent but coordinate way to grasp and manipulate target objects, and there are other objects in the environment acting as potential obstacles. The two arms can be used to remove these obstacles if it is necessary. The target objects can be pre-assigned to each robot or not, depending on the task to be performed.

The proposed approach uses a motion planner to find the robot movements to grasp and manipulate the target objects, and, at the same time, determines which are the obstacles that must be removed. With the information from the motion planning a graph structure is build, which allows to easily represent the tasks that the robots should execute. The graph allows the distribution and the prioritization of the tasks between the arms by using a backtracking search algorithm.

After this introduction the paper is organized as follows. Section II presents a review of related works, Section III presents the proposed approach, and Section IV presents the manipulation task graph. Then, Section V presents some application examples and, finally, Section VI summarizes the work and presents some topics deserving future work.

II. RELATED WORKS

The use of a dual-arm robotic system is becoming quite common to perform manipulation tasks, either involving manipulation of a single object using both arms [1], [2], or manipulation of multiple objects using each arm in an independent way [3], [4].

Motion planning for a dual-arm system or multiple robots with large number of degrees of freedom in a shared environment implies complex problems in high dimensional spaces. The approaches to deal with these problems can be classified into centralized and decoupled [5]. The approach proposed in this work belongs to the decoupled type. For each arm an independent motion plan is computed, and then, the motion plans are coordinated following the approach proposed by [6], in order to avoid collisions between the arms during the execution.

Combining motion and task assignment (hybrid planning) is an open problem in robotics [7], [8], [9]. In the literature there are several relevant algorithms deal with hybrid planning, one of these works employs a high-level planner that acts as constraint and provides a heuristic cost function in the search algorithm in order to speed up the motion planner [10]. Furthermore, a grid-based discrete representation can be also used to combine the information from the task planner and the information from the motion planner in order to obtain a continuous free-collision trajectory for the robot [11]. Another way to combine the task and motion planners is formulated as a representational abstraction between them, where, each action in the task planner can have a multiple instantiations in the motion planner, this increases the possibility of finding a feasible trajectory for the robot [12].

The approach proposed in this work is different to the works mentioned above that are focused on high-level task planners (semantic and symbolic representation) and on how to combine them with the low level planners (motion planner) to solve task-oriented problems. Instead, the proposed approach uses the motion planner to obtain motion
plans that have the information necessary to build a graph representation of the tasks that the robot must execute, as will be explained later in Section IV.

Our work is closely related with the works presented in [13], [14], which propose backtracking search algorithms in the real space in order to know which movable objects in the workspace block the access to a given goal; a related algorithm was also proposed in [15], using a high-level regression-based symbolic planner. Furthermore, Kontiris and Bekris [16] try to avoid the backtracking search by using fast monotone rearrangement solve algorithm.

Stilman et al. [13] solves the problem of rearrange a clutter workspace using a robot in a 2D workspace, and then, they have extended the work to a 3D workspace using a manipulator with high number of degree of freedom [17]. In order to rearrange the workspace and reach the target object, they employ swept volumes to determine, recursively, which objects must be moved. On the other hand, Dogar et. al [14] presents an approach focused on determining free volume space where to put the obstacles that directly block the access to the target object, and includes a non-prehensile manipulation such as pushing in order to move the obstacles that cannot be grasped (pick and place manipulation).

In contrast to [14] and [17], the approach proposed here uses a dual-arm system, and each robot arm may be able to solve the same tasks of removing obstacles and grasp the target objects. Combining the solutions of each arm, it is possible to obtain a better task execution, where the arms work in a independent but cooperative way to achieve a common goal. The proposed approach uses a backtracking search algorithm to explore the graph and obtain the sequence of actions to remove the minimum number of obstacles, and then, distributes the tasks between the robot arms. In contrast, [18] provides algorithms for removing the minimal number of obstacles in order to reach a position, but it does not do recursive removal like in this work.

III. PROPOSED APPROACH

Consider a dual-arm robotic system in a cluttered environment with fixed and removable objects. The goal of this work is to grasp two target objects with the dual-arm system. The objects can be grasped by any arm or can be assigned in advance, depending on the task to be performed. Due to the cluttered environment, the objects of interest may be blocked by other removable objects, which therefore must be removed.

A. Background of the proposed approach

The proposed approach follows the work in [3], which presents a variation of a probabilistic roadmaps planner to compute the robot paths and uses a precedence graph to represent the tasks to be executed by each robot arm. The nodes of the precedence graph represent the target objects and the removable objects $O_j$, $j = 1, ..., n$, and the arcs represent each robot $R_i$, $i = 1, 2$. The motion planner is used to compute a path from the initial configuration of $R_i$ to the grasp configuration of $O_j$. The key point of the motion planner is that the samples generated in the configuration space that imply collisions with removable objects $O_k$ $k = 0, ..., n$ are not rejected like in a classic motion planner; instead, these samples are added to the roadmap as valid samples, and the set of obstacles $SO_{i,j}$ contains the removable objects $O_k$ that generate collisions is stored. At the end, a robot motion path $P_{i,j}$ is generated knowing the involved set of obstacles $SO_{i,j}$. The motion planner is used in a recursive way for each $O_k \in SO_{i,j}$ following a minimum cost strategy until finding a collision-free path $P_{i,j}$ that allows removing the corresponding $O_j$.

B. Contributions and limitations

The main contribution of this work is a framework to use two robot arms to perform independent manipulation tasks in a cluttered workspace considering the removal of obstacles if it is necessary. The framework uses a manipulation task graph that is based on a precedence graph. The graph contains the possible actions that the robot arms can execute, according to the information obtained from the motion planner. A backtracking search algorithm is used to explore the graph and determine the sequence of actions that the robot arms must execute to reach the target objects, having to remove the minimum number of removable obstacles.

A potential limitation of the approach is that the workspace must be known in advance, including the distinction between fixed and removable obstacles. Beside, in the current implementation, a finite set of grasping configurations for each removable object (and the used hand) are given with the object model, then, there is no guarantee that the motion planner can find a solution to manipulate a given object whenever one actually exists using a non-included grasp.

IV. THE MANIPULATION TASK GRAPH

A. Description of the manipulation task graph

The manipulation task graph $G$ is based on an AND/OR graph [19], where the obstacles $O_k$ ($k = 1, ..., n$) of a set of obstacles $SO_{i,j}$ found on the motion plan $P_{i,j}$ for each robot arm $R_i$ ($i = 1, 2$) are represented. Each node represents an object $O_j$ ($j = 1, ..., n$) and each arc from that node represents the robot $R_i$ used to manipulate $O_j$. The root nodes represent the target objects, and the descendant nodes represent the set of obstacles $SO_{i,j}$ of the parent nodes. Figure 1(a) shows the representation of a node of the manipulation task graph $G$. Each node is defined with the following properties: An id of the node name that indicates the represented object $O_j$; A root id that represents which is the corresponding root node of $O_j$; A root cost for each root node representing the number of nodes to be removed from $O_j$ to reach a root node; A local cost for each $R_i$ representing the number of siblings nodes of $O_j$; A state represented by a flag for each $R_i$ associated to $O_j$ that indicates:

- Empty ($\emptyset$), when a path $P_{i,j}$ for $R_i$ to grasp $O_j$ was not computed yet.
- Null (N), when $R_i$ cannot grasp $O_j$ (i.e. $O_j$ is not kinematically reachable by $R_i$).
Step 1: a collision-free paths were found for both robot arms (i.e. task graph with the lower number of obstacles is chosen, in the dual-arm system represented by the robot arms for each).

In order to remove $O_4$ a collision-free paths were found for both robot arms (i.e. no obstacles). Henceforth, the robot states are represented as a pair, i.e. (state of $R_1$, state of $R_2$).

B. Working with the manipulation task graph

The manipulation task graph $G$ is built using the information obtained by the motion planner. First, the target objects are added as root nodes of $G$, then a node is chosen as $O_j$ (node to be explored) and a motion path $P_{t,j}$ is computed for each $R_i$. Then, each $O_k \in SO_{i,j}$ obtained from $P_{t,j}$ is added to $G$ as descendant nodes of $O_j$. Then, the leaf node $O_j$ with the minimum number of obstacles until the target object is chosen, and the motion plan to remove $O_j$ is computed. The process is repeated and $G$ is expanded from the leaf nodes until finding a path to remove each $O_j$ from the target object plan. When there is a path $P_{t,j}$ to remove each obstacle, a backtracking search in $G$ is done to obtain the sequence of action that the robot arms must execute.

Figure 1(b) shows an example in $\mathbb{R}^2$, a cluttered environment is composed of removable square objects $O_j$ in front of the dual-arm system represented by the robot arms $R_1$ and $R_2$. The blue squares represent the target objects $O_1$ and $O_2$ and the red squares the obstacles.

Figure 1(d) shows the resulting $G$ to grasp $O_1$ and $O_2$ with each $R_i$: to grasp $O_1$ with $R_1$ resulted $SO_{1,1} = \{O_6\}$ and for $R_2$ $SO_{2,1} = \{O_5, O_7\}$. The branch of the manipulation task graph with the lower number of obstacles is chosen, in this case the branch that contains $O_4$. In order to remove $O_4$, two parts are executed sequentially until a set of robot actions $SA$ is found to grasp the target objects.

C. Implementation

Algorithm 1 shows the main procedure of the proposed approach. The algorithm receives as input the model of the workspace $W$, which includes the 3D models of the dual-arm system, the fixed and removable objects, as well as their corresponding location in the workspace, the set of target objects $SG$, and the grasping configurations of the removable objects, and returns the task sequence $TS$ that the dual-arm system must execute. The algorithm is divided in two parts, building and exploring the manipulation task graph $G$. These two parts are executed sequentially until a set of robot actions $SA$ is found to grasp the target objects.

Algorithm 1: Main

```
input: W, SG
output: TS
1 SA, TS \leftarrow \emptyset
2 create $G$ with the target objects $SG$ as roots
3 select a target from $SG$
4 while target \neq \emptyset do
5   select an unexplored node $O_{aux}$ with root = target
6   if $O_{aux} \neq \emptyset$ then
7     for each $R_i$ do
8       find the motion plan $P_{t,j}$ to grasp $O_{aux}$
9       if $P_{t,j}$ has $SO_{i,j} \neq \emptyset$ then
10      for each obstacle $O_k$ in $SO_{i,j}$ do
11        if $O_k$ is already in $G$ then
12          if $O_k$ not generates a loop then
13            update the properties of $O_k$
14          else
15            add $O_k$ as child of $O_{aux}$ in $G$
16          update the properties of $O_{aux}$
17        if $O_{aux}$ has at least one state = Free then
18          find solution from $O_{aux}$ to target
19          if there is a solution then
20            add the solution to the set of actions $SA$
21          select a new target from $SG$
22        if $O_{aux}$ has state = Null for each $R_j$ then
23          update the properties of the parent nodes
24      else
25        error: not solution for target
26        break
27    if solution for each target in $SG$ then
28      $TS \leftarrow$ taskDistribution($G, SA$)
29  return $TS$
```

$SO_{1,4} = \{\emptyset\}$, $SO_{2,4} = \{\emptyset\}$. On the other hand, to grasp $O_2$ with $R_1$ resulted $SO_{1,2} = \{O_6\}$ and for $R_2$ $SO_{2,2} = \{O_4\}$, since $O_4$ has a collision-free path this branch is chosen. Figure 1(c) shows the resulting sequence of actions.
V. EXPERIMENTAL RESULTS

The proposed approach has been implemented using a dual-arm robotic system in a simulated and real workspace. The dual-arm system is composed of two 6 DOF Universal Robots (UR5), each one is equipped with an Allegro Hand (AH) with 16 DOF. The simulated workspace includes the corresponding 3D models of the two robots and the two hands, and a data base with different types of objects.

The motion planning and graphical simulations have been performed using The Kautham Project software tool [22], which is a home-developed open source environment that provides several useful tools for the development of planners, like, for instance, random and deterministic sampling methods, metrics to evaluate the performance of the planners (number of generated samples, collision check callings, number of nodes in the graph solution, connected components) and simulation tools (including direct and inverse kinematic models of the robots) that use Coin3D for the graphical rendering, PQP for the collision detection, ROS for the communication layer, and more recently OMPL has been integrated providing the motion planning tool with more options. The experiments were run in a computer with a 2.3 GHz Intel Core i7 processor with 8 Gb RAM, Ubuntu OS 14.04 and ROS Hydro.

In the experiments presented below, several cylinders lie on a table in front of the dual-arm robot. The target objects are the green cylinders \( O_1 \) and \( O_2 \) and there are a set of removable objects (blue cylinders \( O_{1,j} \), \( j \geq 3 \)) that may act as obstacles that block the direct access to the target objects. The goal is that each robot arm grasps a green cylinder.
As mentioned above, a set of grasp configurations of the objects were computed in advance. In this case, the grasp configurations are such that the fingers wrap around the axis of the objects (grasping the objects from the top is not allowed). The set of grasps contains 35 different configurations. The inverse kinematics of the robot arms is computed for each configuration of the set of grasps in order to know whether they are actually reachable. Then, the reachable configurations are used as goal configurations for the motion planner (i.e. the motion planner computes a motion path for each goal configuration). The paths with the minimum number of obstacles are selected. Then, a new set of motion paths are computed to remove the obstacles from the paths toward the target objects (the obstacles are thrown out of the workspace instead of being relocated).

Figure 2(a) and Figure 3(a) show two setups of the workspace used for the experiments. The resulting manipulation task graphs are shown in Figure 2(b) and Figure 3(b).

In the first experiment, the result was as follows. In order to grasp $O_1$ with $R_1$ the computed motion path has collisions with $O_2$, $O_6$ and $O_7$, while to grasp $O_1$ with $R_2$ the objects $O_4$ and $O_5$ have to be removed. Since the paths obtained for both robots are not collision free, the branch of the manipulation task graph with the lower number of obstacles is chosen, in this case the branch that contains $O_4$ and $O_5$. After looking for the motion paths to grasp $O_4$ and $O_5$ it was found a collision-free path for $R_2$ while $R_1$ cannot reach $O_4$ nor $O_5$. On the other hand, to grasp the other target object, $O_2$, a path that collides with $O_3$ and $O_7$, was found for $R_1$ and a path that collide with $O_1$ was found for $R_2$, this generates a loop in $G$, i.e. $O_1$ has to be removed to grasp $O_2$ and vice versa. The path found for $R_1$ to remove $O_3$ has collision with $O_7$ while $R_2$ cannot reach $O_3$ due to a loop in $G$. Finally, in order to grasp $O_7$ a collision-free path was found with $R_2$.

<table>
<thead>
<tr>
<th># Paths</th>
<th>Total time (s)</th>
<th>Average time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>$R_1$</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>$R_2$</td>
<td>18</td>
</tr>
<tr>
<td>Experiment 2</td>
<td>$R_1$</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>$R_2$</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 4(a) shows the resulting task sequence $TS$ describing which object is assigned to each robot in each step of the work, and Figures 4(b-d) show snapshots of the real execution of experiment 1.

In the second experiment, the result was as follows. In order to grasp $O_1$, a motion path colliding with $O_2$ and $O_6$ was found for $R_1$, and a motion path colliding with $O_4$ was found for $R_2$. Then, a collision-free path was found to remove $O_4$ with $R_2$. On the other hand, in order to grasp $O_2$ with $R_1$ a path that collides with $O_6$ was found, and in order to remove $O_6$ a collision-free path was found for $R_1$ while $R_2$ cannot reach $O_6$ because the path has a collision with $O_2$, generating a loop in $G$.

Figure 5(a) shows the resulting task sequence $TS$, and Figures 5(b-d) show snapshots of the real execution of experiment 2.

Table I shows the total number of paths computed for each experiment using a RRT-Connect planner, the total time used to compute all the paths and the average time per path.

VI. CONCLUSIONS AND FUTURE WORKS

This work has presented an approach that combines a motion planner and a task planner for a dual-arm system. One of the advantages of the proposed approach is that it can work in workspaces where there are removable obstacles. The approach generates motion paths for the robot arms (even though the target objects may be blocked by removable obstacles) and the set of motion paths to remove the obstacles. On the other hand, the approach includes a task planner which is in charge of determining the sequence of actions to
be performed by the robot arms to achieve their goals. The proposed approach has been implemented and successfully applied in a simulated and a real environment.

As future work, it is considered the transfer of the objects between the robot arms. Besides, it is considered the inclusion of a grasp planner to determine, when necessary, the grasp configurations on line during the manipulation planning.

REFERENCES


