Planning manipulation movements of a dual-arm system considering obstacle removing

Carlos Rodríguez, Andrés Montaño and Raúl Suárez

Institut d'Organització i Control de Sistemes Industrials (IOC)
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

Abstract

The paper deals with the problem of planning movements of two hand-arm robotic systems, considering the possibility of using the robot hands to remove potential obstacles in order to obtain a free access to grasp a desired object. The approach is based on a variation of a Probabilistic Road Map that does not rule out the samples implying collisions with removable objects but instead classifies them according to the collided obstacle(s), and allows the search of free paths with the indication of which objects must be removed from the work-space to make the path actually valid, we call it *Probabilistic Road Map with Obstacles* (PRMwO). The proposed system includes a task assignment system that distributes the task among the robots, using for that purpose a precedence graph built from the results of the PRMwO. The approach has been implemented for a real dual-arm robotic system, and some simulated and real running examples are presented in the paper.

Keyword: Motion planning, Grasping, Dual-arm, Manipulation.

1 Introduction

Manipulating objects with more than one hand is a problem of significant relevance in a human environment, and so it is in an environment where several robot arms interact, both in industrial and service robotics. Manipulating objects involves several associated problems, among which there are two main ones: the determination of a proper grasp configuration for the available hand or gripper (considering, for instance, aspects like the shape of the object and the task to be performed with the grasped object), and the determination of collision free paths to reach the grasp configuration and to move an object from its initial configuration to a desired one; grasping and path planning are already classic problems in robotics.

In this work we deal with the problem of finding collision free paths under the following assumptions: two robots (hand-arm systems) are available in a common workspace to grasp and move objects, the final goal is to grasp a particular object, there may be other removable objects in the environment acting as obstacles, and the two robots can be used to remove these obstacles if it is necessary. Note that this is a frequent problem in everyday life for the humans, and so will be for humanoids robots.

The proposed approach solves the problem of finding the robot movements to grasp and manipulate the desired object determining at the same time which are the objects acting as obstacles that must be removed. The approach allows the consideration of different grasping configurations for each object and selects the one that allows a real solution to the stated problem.

After this introduction the paper is organized as follows. Section 2 presents a review of related works and Section 3 presents the proposed approach, giving first an overview, Section 4 presents the description of the approach including the proper algorithms. Then, Section 5 deals with the implementation and presents some application examples and, finally, Section 6 summarizes the work and presents some topics deserving future work.

2 Related Work

The existence of more than one robot arm performing manipulation tasks in a common workspace is becoming quite frequent, either because two or more fixed or mobile robots have access to the same workspace or due to the increasing number of multi-arm robots, particularly with anthropomorphic features. Planning manipulation movements for these systems is a research topic of increasing interest [1], and is closely related with the grasping problem [2, 3]. Robots sharing the workspace can cooperatively work acting as a close kinematic chain, like for instance assembling a nut and a bolt [4] or a peg-in-hole task [5], in both cases with each part being manipulated by a different arm, or cooperating with open chain coordinated movements, like in the problem considered in this paper where the robots must work coordinately in order to properly remove obstacles to make a desired object reachable.

Motion planning for multiple robots in a shared physical space usually implies complex problems in high dimensional spaces, and the approaches to deal with them can be classified into centralized and decoupled [6]. In the

centralized approaches multiple arms are considered as a single multi-body robot with the number of degrees of freedom (DOF) of the whole system, and the planning algorithms simultaneously find a coordinated and collision free path for each robot. In the decoupled approaches each arm is treated as a single independent system and the motion planning process has two phases, first a path for each arm is independently determined and, then, it is required a coordination method to avoid potential collisions between the robots when they simultaneously execute their movements. Even when centralized approaches are complete, they have to deal with a higher number of DOF which implies a planning space of higher dimension, and therefore they are, in general, computationally more expensive. On the other side, the decoupled approaches decompose the problem into smaller subproblems (i.e. one for each robot) but, as a drawback, they need the additional coordination phase.

Several techniques have been developed for robot motion planning. Among the most effective are the sampling-based techniques, which are classified into deterministic and probabilistic depending on the way the samples are generated [7]. Typical examples of the deterministic approaches are visibility graph [8], retraction algorithm [9], A* algorithm [10], Dijkstra algorithm [11], and potential fields [12]. Among the most relevant probabilistic approaches are the Rapidly-exploring Random Trees planners (RRT) [6] and Probabistic Road Map planners (PRM) [13]. These original probabilistic approaches have some problems when there are narrow passages and in order to overcome them several variations were developed, like a multi-resolution PRM planner [14], a dynamics domain RRTs [15], a retraction base RRTs [16], a adaptive workspace biasing [17], and a sampling method based on Principal Component Analysis [18]. In order to speed up the query path planning, some variants of PRM planners build a roadmap without checking for collisions, then, once a potential solution path was found the existence of collisions is verified and if they occur the corresponding nodes and edges are removed from the roadmap and a new search is started; the process is repeated until a collision free path is found (e.g. the Lazy PRM planner [19]). Some extensions of PRMs include the use of object symmetries in order to improve the performance of the planner [20], and the consideration of scenarios where there are obstacles with known collision free paths around them which have to be connected to the general roadmap, to yield a solution path that skirts the obstacles [21]. The paths obtained with the described planners can be optimized using post processing methods, which search for an optimal subset of

samples in the configuration space that replace some samples from the initial path [22]. The motion planning problem is in a close relation with the task planning problem, which is usually in a higher level of abstraction that may include the use of semantic knowledge to improve the planning capabilities by integrating spatial knowledge with other sources of knowledge [23].

The planners mentioned above have been designed to avoid collisions with any obstacle, either fixed or removable, while the problem of motion planning considering removable obstacles is still an open problem in robotics. A relevant pioneering works considering removable obstacles have shown that motion planning among obstacles is an NP-hard problem [24] and proposed a grid based 2-D planner to heuristically try to minimize the cost of pushing obstacles out of the way [25].

Approaches to this motion planning problem were particularly developed in the context of humanoid robots, with different proposals depending on whether there is an a priori full knowledge of all the obstacles in the environment, including whether they are fixed or not. An example of a work considering a fully known environment can be found in [26], where the problem is solved as follows. The initial configuration, the goal configuration and the configurations describing the position of the obstacles are considered as nodes of a graph, a collision free path is searched between every pair of these configurations and when it is found the corresponding nodes are connected in the graph, and finally, the task solution path is found searching the graph for a branch from the initial to the final configuration; every intermediate node in this branch represents an obstacle that must be removed to execute the task. When the objects in the environment are not completely known, the planner works while the environment is being explored to distinguish fixed and removable obstacles. The classification of the objects as fixed or removable can be done through the object recognition and checking in the database for proper label [27, 28], or, by trying to push the objects and classifying them according to some sensor inputs (e.g. measurements of force sensors on board of the robot) [29]. In all these works the obstacles that do not a allow a robot path towards the goal are pushed out of the path by the robot itself.

In another type of problem the desired object is known but it position is not, thus the robot must execute pick and place tasks to remove the visible obstacles and look for the desired object. An example can be found in [30] that presents a motion planner based on a vision system, the planner considers the visibility and the accessibility to the objects to compute the occluded

space where the target could be, and decides then which obstacle must be removed to look for the desired object.

In the context described above, the approach proposed in this paper to plan the robot paths, and if necessary determine a sequence of robot actions to remove obstacles, belongs to the decoupled category, uses a motion planner based on a PRM, and requires the knowledge of the environment.

3 Overview of the Proposed Approach

3.1 Problem Statement

Consider two robots (hand-arm systems) R_i, $i = 1, 2$, in a shared workspace where there is an object of interest O_0 to be grasped and some other removable objects O_j, $j = 1, ..., n$, which can be grasped and removed from the scene by the robots themselves and that may act as obstacles that do not allow access to O_0. In this context the problem to be solved is: find a geometric path $P_{i,0}$ for a robot R_i in order to grasp the desired object O_0 and, if necessary, a set of geometric paths $P_{l,j}$, $l = 1, 2$ to remove all the obstacles O_j in $P_{i,0}$ using both robots in a coordinated way.

3.2 Contributions and limitations of the approach

The main contributions of this work are, on one side, a path planner able to find paths in environments where there are removable obstacles such that it can returns either directly a collision free path or a path that would be free of collisions if some particular removable obstacles (returned together with the path) are previously removed. A second aspect is the reasoning about the actions to be done by each robot, using for this a precedence graph where the target and the obstacles are represented as nodes and the links indicate the obstacles to be removed in order to be able to do each particular action. The path planner and the reasoning with the graph are simultaneously used until a solution is found, with the solution including an assignment of robot actions to remove (if necessary) obstacles in order to arrive to a desired target and the corresponding collision free paths for the robots to do such actions.

On the other hand, as potential limitations of the approach it can be mentioned that the environment must be known in advance, including the distinction between fixed and removable obstacles, and that, in the current

implementation, the grasping configurations for each removable object with the used hand are also known (they are provided with the object model, and any number of grasping configurations can be considered).

3.3 Approach overview

This subsection presents an overview of the proposed approach with the aim of introducing the main ideas and features, a detailed description is presented in Section 4.

Figure 1 shows a schema with the main blocks of the work, which also indicates the associated algorithms described later in Section 4. The main steps are the following. A graph is built with the root node representing the goal object (block 1), then a motion planner is used to look for a robot path to reach and grasp the goal object, considering each available robot and different grasping configurations (block 2 and 3). If a collision free path (a solution) is found (block 4) it is directly returned, otherwise the graph is expanded adding child nodes representing the objects that are obstacles for the goal abject accessibility (block 5). The object represented by each of the new child nodes is now selected as a goal object to be removed (block 6) in order to clean the path to the object represented by the parent node, and the procedure enters in a loop through blocks 2 to 6. The loop lasts until any of the two circumstances occurs: a) the last added nodes represent objects that can be removed from the environment without any obstacles, allowing free access to the objects represented by the parent nodes, and so on until the original goal object (the root node), meaning that a solution was found, or, b) the last added nodes represent objects that can not be removed, either because they are not reachable due to kinematic constrains of the hand-arm systems or because objects already represented in the graph have to be removed, meaning that there is no solution and the goal object can not be reached.

The core of the approach is the generation of the robot paths, which is done as follows. A PRM is used to find a path $P_{i,0}$ for each robot R_i to grasp the object of interest O_0, but, as a difference with the typical use of PRMs a sample of a robot configuration that implies a collision of the robot with any removable object O_j in the environment is not neglected, instead it is considered for the PRM in an usual way but associating to it a list with the collided obstacles. The same is done when a local planner checks the validity of a local segment connecting two samples for the PRM

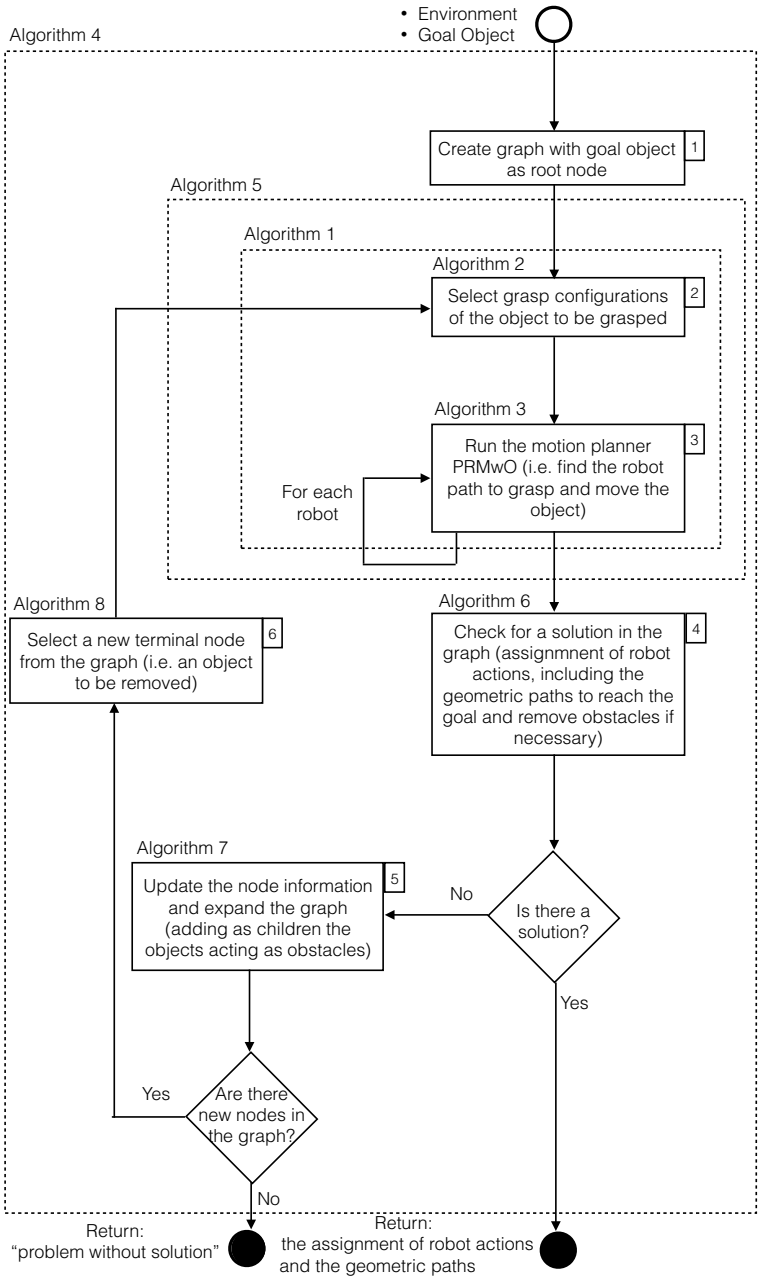


Figure 1: Schema of the proposed approach for the planning of manipulation movements, i.e. the assignment of robot actions and the geometric path of each robot.

construction, if there are collisions with any removable object O_j it is just added to a list of obstacles associated with the segment. We refer to this as *Probabilistic Road Map with Obstacles* (PRMwO). Using the PRMwO as a regular PRM, it is possible to obtain paths $P_{i,0}$ for each robot R_i to grasp the object of interest O_0 and, at the same time, an associate set $SO_{i,0}$ with the obstacles that must be removed from the environment in order to make $P_{i,0}$ be free of collisions. The same procedure is generically applied to look for a path $P_{i,j}$ for the robot R_i to grasp and remove any removable object O_j returning the corresponding set of obstacles $SO_{i,j}$.

Note that the collision check performed for the configurations of a robot path must be done considering the arm and the hand when the robot is going toward the object to be grasped, and considering the arm, the hand and also the grasped object when this is removed from the scene. The hand can have any initial configuration.

When the PRMwO is used to search a solution to grasp and remove an object O_j the search can finish once a first path $P_{i,j}$ with an associate set of obstacles $SO_{i,j} \neq \emptyset$ was found, starting then the search of valid robot paths to remove the objects in $SO_{i,j}$, or the search can continue looking for a path $P_{i,j}$ with $SO_{i,j} = \emptyset$ (i.e. look for a path that does not require removing any obstacle). The balance between these two cases depends on the particular problem, practical criteria to stop the search procedure are: select the first path found without caring about the number of obstacles, predefine a maximum number of samples in the PRMwO, or predefine a maximum searching time.

After finding a path $P_{i,0}$ with associate obstacles $SO_{i,0}$, $i = 1, 2$, for the object of interest O_0, the same procedure is applied to search for paths $P_{i,j}$ with $SO_{i,j}$, $i = 1, 2$ $\forall O_j \in SO_{i,0}$, and iteratively continues $\forall O_l \in SO_{i,j}$ and so on until an appropriate set of paths without obstacles is found (i.e the robots can start removing objects without any obstacles), or a loop is found (i.e one object is an obstacle to grasp another one and viceversa).

The objects can be grasped in different ways with a set of different hand configurations, which are taken as different goals when the PRMwO is built. This increases the probability of finding valid paths to grasp each object. When a predetermined grasping configuration is not actually reachable for the current pose of the object it will be properly disregarded during the planning process, this may happen due to hand or arm kinematic constraints or due to collisions with fixed obstacles. The grasping configurations for each object can be obtained using different procedures (see for instance [31, 32]).

This problem is outside the scope of this work and therefore we assume here that the set of possible grasping configurations of an object has been computed in advance and it is provided with the model of the object.

The other key point of the approach is the representation of the information obtained in the iterative search of robot paths described above in an AND/OR precedence graph G. Each node of G represents an object O_j and the edges starting at O_j link O_j with the nodes representing all the objects $O_l \in SO_{i,j}$ (i.e. the obstacles to be removed to make $P_{i,j}$ be collision free). When $SO_{i,j}$ has more than one element it generates AND edges, meaning that all the objects in $SO_{i,j}$ have to be removed in order to be able to grasp O_j with R_i (i.e. be able to execute $P_{i,j}$ without collisions). On the other hand the edges generated by $SO_{1,j}$ and $SO_{2,j}$ are type OR, meaning that only the objects in $SO_{1,j}$ have to be removed if O_j is to be grasped with R_1 or only the objects in $SO_{2,j}$ have to be removed is O_j is to be grasped with R_2. G has a tree structure with the root node representing always the target object O_0. Fig. 2 shows an example of precedence graph G (a more detailed description of the elements of the graph G will be given in Subsection 4.2).

Going from the final nodes of G towards the root it is possible to determine: the set of objects O_j that must be removed in order to get a path free of collisions up to the object of interest O_0, the robot R_i to be used to remove each object, and the corresponding geometric path $P_{i,j}$. Fig. 3 shows an example of the set of actions SA assigned to the robots, i.e. a description of which objects are going to be manipulated by each robot.

After having assigned the robot actions to each robot and determined the corresponding geometric paths free of collisions, a final step to optimize the performance of the robots consists in a temporal coordination of the robot movements, i.e. in order to optimize the time needed to remove the obstacles and reach the desired goal a robot can execute the assigned movements in parallel with another robot as long as the priorities in the sequence of actions are properly satisfied. This coordination of the robots adjusting the temporal evolution along the geometric paths has been already developed and implemented, but it is a complementary module outside the scope of this work and therefore it is not described in this paper (see [33] for details).

The following section presents a detailed description of the approach describing the used algorithms.

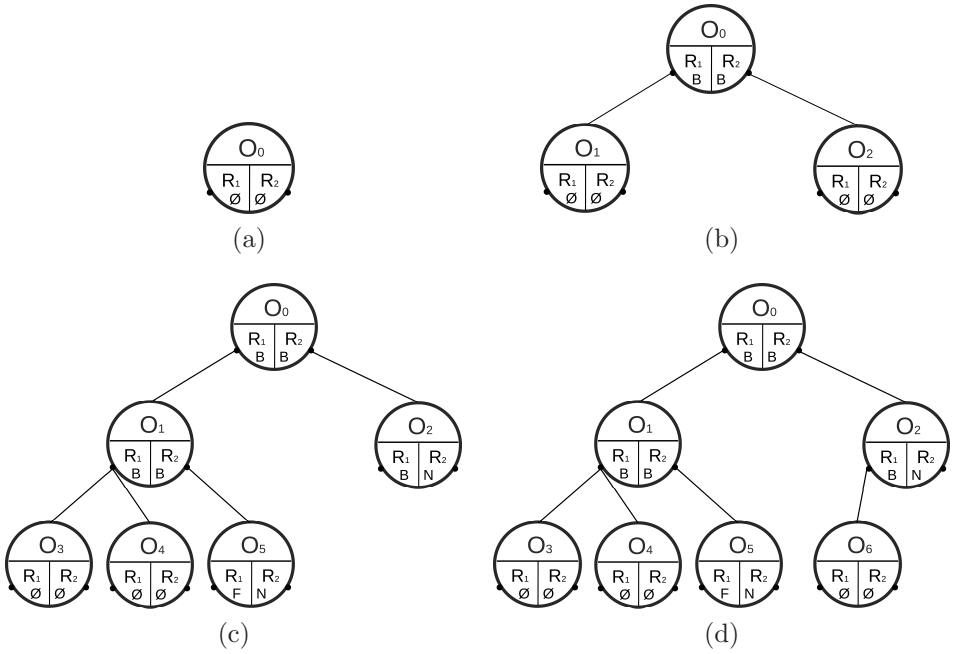


Figure 2: Example of precedence graph G: (a) a root node is added representing the desired object O_0; (b) using R_1 to grasp O_0 produces $SO_{1,0} = \{O_1\}$ and using R_2 produces $SO_{2,0} = \{O_2\}$, then O_1 and O_2 are added as child nodes of O_0 with edges for R_1 and R_2 acting as OR edges; (c) removing O_1 with R_1 returns $SO_{1,1} = \{O_3, O_4\}$ then O_3 and O_4 are added as child nodes of O_1 with two edges for R_1 acting as AND edges, while removing O_1 with R_2 returns $SO_{1,1} = \{O_5\}$ and then O_5 is added as a child node of O_1, in this case with an edge for R_2 acting as OR edge; (d) on the other branch, removing O_2 with R_1 returns $SO_{1,2} = \{O_6\}$ then O_6 is added as child node of O_2 with an OR edge for R_1 and trying to remove O_2 with R_2 does not return any valid path (for instance, because there is no kinematic solution for R_2 grasping O_2) and therefore no child nodes with edge for R_2 are added to G. This procedure is iteratively executed until there are terminal nodes in G representing objects that can be removed without additional obstacles and therefore the assignment of robot actions can be determined to remove current obstacles and arrive to O_0.

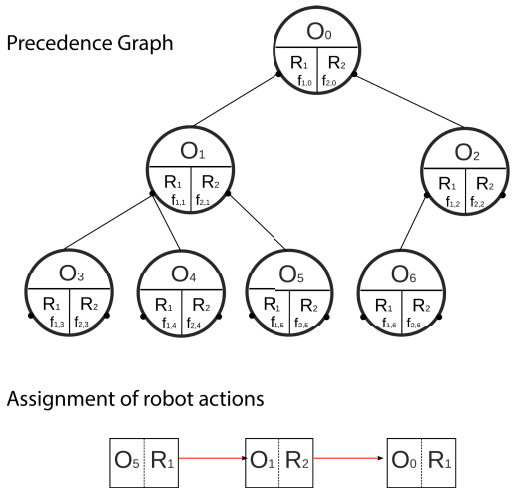


Figure 3: Example of the assignment of robot actions to each robot: one valid solution is removing O_5 with R_1, O_1 with R_2 and finally O_0 with R_1.

4 Detailed description of the approach

The concepts introduced in the previous section are formalized here and the developed procedures are described in detail.

4.1 Determination of the geometric paths of the robots

In order to determine the geometric paths of a robot it is necessary to know which is the grasping configuration of the hand. As it was mentioned in Subsection 3.3, it is assumed that a set grasping configurations in the object reference frame is provided with the model of each object for the used hand. Figure 4 shows an example illustrating four different cylindrical grasp configurations of the Schunk Anthropomorphic Hand (SAH) and the Schunk Dexterous Hand (SDH2) grasping a soda can. A given grasping configurations of the hand is related to the object reference frame, but since the position of the object in the work environment is known, the position of the arm wrist for the grasping configurations can be easily obtained. A grasp configuration with a description of the hand joints and the wrist position is represented as $\mathbf{c}_k$, and a set of these configurations for an object O_j is represented as SC_j.

Now, given an object O_j to be grasped by a robot R_i the procedure to

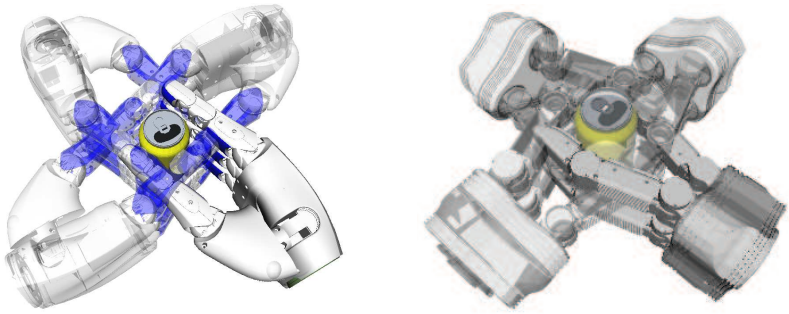


Figure 4: Examples of grasping configurations of two hands (SAH and SDH2) grasping a soda can (the finger positions are the same for all the grasping configurations, but the hand positions with respect to the can reference system is different).

Algorithm 1: findPath

input : R_i, $\mathbf{c}_o$, SC_j
output: $P_{i,j}$, $SO_{i,j}$

$P_{i,j} = \emptyset$;
$SO_{i,j} = \emptyset$;
$SC_{i,j} \leftarrow$ selectGrasps(R_i, SC_j);
$max =$ number of removable objects in the workspace;
for *each* $\mathbf{c}_k \in SC_{i,j}$ **do**
    $P_{aux_{i,j}}$, $SO_{aux_{i,j}} \leftarrow$ PRMwO(R_j, $\mathbf{c}_o$, $\mathbf{c}_k$);
    if $range(SO_{aux_{i,j}}) < max$ **then**
        $P_{i,j} \leftarrow P_{aux_{i,j}}$;
        $SO_{i,j} \leftarrow SO_{aux_{i,j}}$;
        $max \leftarrow$ range($SO_{aux_{i,j}}$);
return $P_{i,j}$, $SO_{i,j}$;

Algorithm 2: selectGrasps

input : R_i, SC_j
output: $SC_{i,j}$

```
1 SC_{i,j} = ∅ ;
2 for each c_k ∈ SC_j do
3     if c_k is reachable by R_i (i.e. inverse kinematics of R_i
      has a solution for c_k) then
4         Add to c_k the arm configuration;
5         SC_{i,j} ← SC_{i,j} ∪ c_k
6 return SC_{i,j} ;
```

look for a geometric path $P_{i,j}$ is shown in Algorithm 1. It requires as input the involved robot R_i, its current configuration $\mathbf{c}_o$, and the set of grasping configurations SC_j for O_j, and returns a geometric path $P_{i,j}$ and the set of obstacles $SO_{i,j}$ to be removed to make $P_{i,j}$ collision free. Algorithm 1 performs the following actions. First, the subset of grasping configurations $SC_{i,j}$ that are kinematically reachable by R_i are selected from SC_j (Step 3). This is done by the function *selectGrasps* detailed in Algorithm 2, that simply solves the inverse kinematics of R_i for each configuration $\mathbf{c}_k \in SC_j$, when there is a solutions adds the arm configuration to $\mathbf{c}_k$, and returns the set $SC_{i,j}$ of configurations with a kinematic solution for R_i. Then, a path $P_{i,j}$ (with the associate set of obstacles $SO_{i,j}$) is searched for each reachable configuration $\mathbf{c}_k \in SC_{i,j}$, and the path with smaller number of obstacles in $SO_{i,j}$ is selected (Steps 5 to 9). $P_{i,j}$ and $SO_{i,j}$ are generated with the function PRMwO detailed in Algorithm 3.

The function PRMwO requires as input the robot R_i to be used, its initial configuration $\mathbf{c}_o$ and a grasping configuration $\mathbf{c}_g$ of the object O_j to be grasped, and returns a geometric path $P_{i,j}$ for R_i to grasp and remove O_j and the set of associate obstacles $SO_{i,j}$ that must be removed in order to make $P_{i,j}$ be collision free. Algorithm 3 performs the following actions. First, the set of vertices SV of a PRM is initialized with the configurations $\mathbf{c}_o$ and $\mathbf{c}_g$ (step 1), and then a searching loop is executed until the initial and final configurations are connected in the PRM or a predefined maximum number of configuration samples has been generated (step 3). Within this loop, each iteration starts with the generation of a random configuration

Algorithm 3: PRMwO

input : R_i, $\mathbf{c}_o$, $\mathbf{c}_g$
output: $P_{i,j}$, $SO_{i,j}$

$SV = \{\mathbf{c}_o\ ,\mathbf{c}_g\}$;
$k = 0$;
while $(k < k_m) \vee (\mathbf{c}_o$ and $\mathbf{c}_g$ are not connected) **do**
    generate a new configuration $\mathbf{c}$;
    $SO_{(\mathbf{c})} \leftarrow$ collisionCheck($\mathbf{c}$);
    if there are not fixed obstacles in $SO_{(\mathbf{c})}$ **then**
        associate $SO_{(\mathbf{c})}$ to $\mathbf{c}$;
        add $\mathbf{c}$ to SV;
        find the nearest neighbor $\mathbf{c}_{nn} \in SV$ to $\mathbf{c}$;
        generate the segment $s = \overline{\mathbf{c}\mathbf{c}_{nn}}$;
        $SO_{(\mathbf{s})} \leftarrow$ localPlanner($\mathbf{s}$);
        if there are not fixed obstacles in $SO_{(\mathbf{s})}$ **then**
            associate $SO_{(\mathbf{s})}$ to $\mathbf{s}$;
            add $\mathbf{s}$ to PRM;
        else
            reject $\mathbf{s}$;
    else
        reject $\mathbf{c}$;
    $k = k + 1$;
$P_{i,j} \leftarrow$ find the path with minimum number of vertices in the PRM;
$SO_{i,j} \leftarrow$ collect the obstacles $SO_{(\mathbf{c}_i)}$ and $SO_{(\mathbf{s}_i)}$ associate with each vertice $\mathbf{c}_i$ and segment $\mathbf{s}_i$ contained in $P_{i,j}$;
return $P_{i,j}$, $SO_{i,j}$;

sample $\mathbf{c}$ in the C-space of R_i (step 4), and the function *collisionCheck* is run to test whether R_i located at $\mathbf{c}$ implies a collision with any fixed or removable object, with another robot, or with itself (step 5); if there are collisions the collided obstacles are added to the set of obstacles $SO_{(\mathbf{c})}$. If $SO_{(\mathbf{c})}$ contains only removable obstacles, then $SO_{(\mathbf{c})}$ is associate with $\mathbf{c}$ and it is added to SV (steps 7 and 8). The nearest neighbor $\mathbf{c}_{nn}$ to $\mathbf{c}$ in SV is determined and used to generate the segment $\mathbf{s}$ between $\mathbf{c}$ and $\mathbf{c}_{nn}$, and the function *localPlanner* is used to check for the existence of collisions when R_i is moved along $\mathbf{s}$, and, as in the case of the sample $\mathbf{c}$, if there are collisions the collided obstacles are added to a set of obstacles $SO_{(\mathbf{s})}$ (Steps 9 to 11). If the obstacles in $SO_{(\mathbf{s})}$ are all removable objects then $SO_{(\mathbf{s})}$ is associate to $\mathbf{s}$ and $\mathbf{s}$ is added to the PRM, otherwise $\mathbf{s}$ is rejected (steps 12 to 16). If $SO_{(\mathbf{c})}$ contains fixed obstacles, then $\mathbf{c}$ is directly rejected (step 18). Finally, when $\mathbf{c}_o$ and $\mathbf{c}_g$ are connected by the PRM, the path $P_{i,j}$ with minimum number of vertices between them is searched in the roadmap and the set $SO_{i,j}$ that contains the obstacles associated with every node $\mathbf{c}_i$ and segment $\mathbf{s_i}$ included in $P_{i,j}$ is computed (steps 20 and 21). The algorithm returns $P_{i,j}$ and $SO_{i,j}$.

4.2 Assignment of robot actions

As it was conceptually described in Subsection 3.3 the assignment of robot actions is determined using a precedence graph G. Each node of G represents a removable object, and, for simplicity, we will refer to each node using the name of the represented object (i.e. the "node O_j" represents the object "O_j"). Each node has an associate flag for each robot available in the workspace that eventually could grasp it. The flag of the node O_j associate with the robot R_i is indicated as $f_{i,j}$. Each flag $f_{i,j}$ has the following possible states:

- Empty ($\emptyset$), when a path for R_i to grasp O_j was not computed yet.
- Null (N), when R_i cannot grasp O_j (i.e. when it does not exist a kinematic solution for R_i to grasp O_j, this is, $selectGrasp(R_i, SC_j) = \emptyset$).
- Free (F), when a path for R_i to grasp O_j was found and it has no obstacles (i.e. $SO_{i,j} = \emptyset$).
- Blocked (B), when a path for R_i to grasp O_j was found but there are some obstacles along it (i.e. $SO_{i,j} \neq \emptyset$).

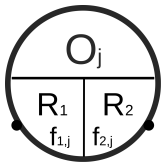


Figure 5: Graphical representation of a node of G representing the object O_j and two flags $f_{1,j}$ and $f_{2,j}$ for the robots R_1 and R_2 respectively.

- Discarded (D), when a path for R_i to grasp O_j was found but there are some obstacles along it (i.e. $SO_{i,j} \neq \emptyset$) and at least one object $O_h \in SO_{i,j}$ was already included in the same branch of G and O_j should be removed from the scene to reach O_h.

For the case of two robots, the nodes of G will be graphically represented as shown in Figure 5 with the state of the two flags $f_{i,j}$ shown in the two lower quarters of a circle.

The main procedure for the assignment of robot actions is presented in Algorithm 4, it requires as input the desired object O_0 and a description of the workspace W (that includes the models of the non removable objects, the models, the positions and a set of grasping configurations of the removable objects, and the models and the initial configurations of the robots), and returns an assignment of robot actions to each robot (that includes the corresponding geometric paths) to grasp the desired object and to remove, if necessary, some obstacles.

Algorithm 4 creates G with the root node O_0 (step 2) and selects it as an auxiliar one O_{aux} for an iterative procedure (step 3) where the following actions are performed. The auxiliar node is analyzed (step 5) using the function *AnalyzeNode* described in Algorithm 5 that changes the state of the two flags $f_{1,aux}$ and $f_{2,aux}$ associate with O_{aux}, indicating whether O_{aux} can be manipulated by the robots and whether there are obstacles that must be removed to allow it. Then, the graph G is searched for a solution starting from O_{aux} (step 6) using the function *checkGraph* described in Algorithm 6. If a solution is not found then G is expanded (step 8) by adding to the node O_{aux} the child nodes given by $SO_{1,aux}$ and $SO_{2,aux}$ (i.e the obstacles for R_1 and R_2 to grasp O_{aux}), this is done using the function *expandNode* described in Algorithm 7. After this, a new terminal node is selected to be analyzed (step 9) using the function *selectNode* described in Algorithm 8 and a new iteration is started. When the function *selectNode(G)* returns $O_{aux} = \emptyset$, it

Algorithm 4: Main

input : W, O_0
output: SA

```
1  SA ← ∅ ;
2  Create G with O_0, f_{1,0} = ∅ and f_{2,0} = ∅ ;
3  O_aux ← O_0 ;
4  while SA = ∅ do
5      O_aux ← AnalyzeNode(O_aux, SC_aux) (update the flags f_{i,aux}) ;
6      SA ← checkGraph(G, O_aux) ;
7      if SA = ∅ then
8          G ← expandNode(G, O_aux) ;
9          O_aux ← selectNode(G) ;
10         if O_aux = ∅ then
11             return Error: problem without solution;
12 return SA
```

means that there are not more expandable nodes, all the terminal nodes of G have been analyzed and none can be removed directly, either because $f_{i,j} = N$ or $f_{i,j} = D$, in this case the problem does not have a solution (step 11).

The functions used in the main procedure mentioned above are the following.

The function *AnalyzeNode*, described in Algorithm 5, requires as input a node O_j of G with the flags $f_{i,j} = \emptyset$ and SC_j, and returns as output the same node with an updated status of the flags (and the corresponding relevant information, i.e. the paths $P_{i,j}$ and the sets $SO_{i,j}$). *AnalyzeNode* uses the functions *selectGrasps* (Algorithm 2) and *findPath* (Algorithm 1) described above to look for $P_{i,j}$ and $SC_{i,j}$ and, according to the result, establishes the values of the flags $f_{i,j}$.

The function *checkGraph*, described in Algorithm 6, requires as input the graph G and a node O_j, and returns the assignment of actions SA to be executed by the robots. The function *checkGraph* verifies if there exists a valid path in G from the terminal node O_j to the root node O_0 and assigns the robot actions SA to remove the corresponding obstacles. The algorithm uses the functions *getSiblingNodes* and *selectParentNode* to select the sibling nodes and the parent node respectively in order to explore the graph while

Algorithm 5: AnalyzeNode

input : O_j
output: O_j with updated information about its flags

```
1 for each R_i do
2     P_{i,j}, SO_{i,j} ← findPath(R_i, SC_{i,j}) ;
3     set f_{i,j} according to the resulting P_{i,j}, SO_{i,j} and SC_{i,j} ;
4 return O_j (with the updated information, i.e. f_{i,j}, P_{i,j} and SO_{i,j}) ;
```

searching for a solution SA.

The function *expandNode*, described in Algorithm 7, requires as input the graph G and the node O_j to be expanded (which includes the information about the set of obstacles $SO_{i,j}$), and returns G properly expanded with the children of O_j. This function simply adds to G as children nodes of O_j the elements in the set of obstacles $SO_{i,j}$ when O_j cannot be removed by any robot with a collision free path (i.e. $\nexists i / f_{i,j} = F$) but it could be removed by a robot R_i if the obstacles in $SO_{i,j}$ are previously removed.

The function *selectNode*, described in Algorithm 8, requires as input the graph G and returns one of its terminal nodes O_j, which will be analyze to check for the existence of a solution. *selectNode* chooses among all the terminal nodes of G the one that has the lowest cost, considering as the cost of a node the number of nodes that, starting from it, represent objects that must be removed to let O_0 (the root node) be free of obstacles. The function selects terminal nodes that have not been analyzed previously (i.e. with $f_{i,j} = \emptyset$), computes their costs, and returns the terminal node with the lowest cost (if there are several nodes with the same lowest cost the function returns the one that was first analyzed).

5 Experimental Results

The proposed approach has been implemented inside the home-developed path planning framework called *The Kautham Project*[1], which was developed with open source and cross-platform directives in mind and using libraries Qt for the user interface, Coin3D for the graphical rendering, PQP

[1] For detailed information about The Kautham Project, see `https://sir.upc.edu/projects/kautham/`

Algorithm 6: checkGraph

input : G, O_j
output: SA

```
1  O_aux = O_j;
2  while O_aux ≠ O_0 do
3      SO_aux ← getSiblingNodes(O_aux);
4      [R_l, O_h] ← selectParentNode(O_aux);
5      c ← getCardinality(SO_aux);
6      while SO_aux ≠ ∅ do
7          select O_aux' ∈ SO_aux;
8          if ∃i/f_i,aux' = F then
9              if i/f_i,aux' = F for more than one i then
10                 select a random [R_i, O_aux'] for i ≠ l;
11             SA ← SA ∪ {[R_i, O_aux']};
12             delete O_aux' from SO_aux;
13             c = c − 1;
14         else
15             if ∀i f_i,aux' = N then
16                 update O_h with f_l,h = N;
17                 SO_aux = ∅;
18                 SA ← ∅;
19     if c = 0 then
20         update O_h with f_l,h = F;
21         O_aux = O_h;
22     else
23         SA ← ∅;
24         break;
25 return SA;
```

Algorithm 7: expandNode

input : G, O_j
output: G

1 **if** $\nexists i / f_{i,j} = F$ **then**
2 $\forall i / f_{i,j} =$ B add as children of O_j the elements of $SO_{i,j}$;
3 **return** G;

Algorithm 8: selectNode

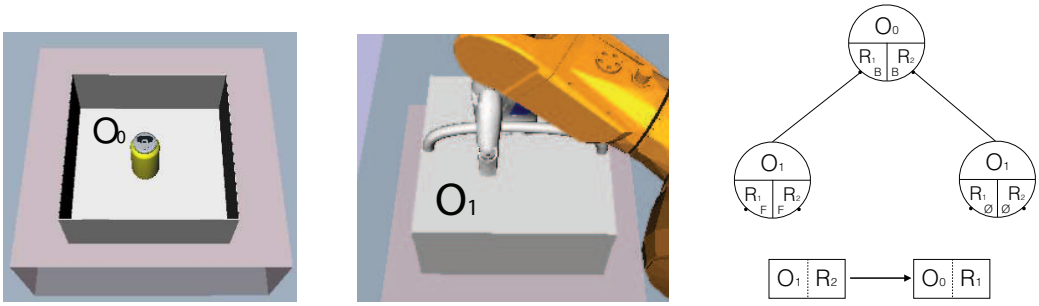
input : G
output: O_j

1 $max =$ number of removable objects in the workspace;
2 $O_{aux} = \emptyset$;
3 **for** each non-analyzed terminal node $O_j \in G$ *(i.e. nodes with* $f_{i,j} = \emptyset$*)* **do**
4 compute the minimum *cost* of O_j;
5 **if** $cost < max$ **then**
6 $O_{aux} = O_j$;
7 $max = cost$;
8 **return** O_{aux};

for the collision detection and ROS for the communication layer. This framework provides the developer with several tools needed for the development of planners, like, for instance, direct and inverse kinematic models of the robots (arms and hands), random and deterministic sampling methods, metrics to evaluate the performance of planners (number of generated samples, collision check callings, number of nodes in the graph solution, connected components) and simulation tools.

The following three examples illustrate the ability of the proposed approach to find the paths for a two hand-arm robotic system composed by two Stäubli TX-90 robots arms with 6 DOF, the robot R_1 is equipped with a Schunk Anthropomorphic Hand (SAH) with 13 DOF, and the robot R_2 is equipped with a Schunk Dexterous Hand (SDH2) with 7 DOF. The object to be grasped is always a yellow soda can, and there are other objects in the scene that act as obstacles that do not allow a direct access to the yellow can. The initial configuration for each robot is given, and it is also used as final position for the robot paths. The examples have been run in a computer with a processor Intel Core2 2.13GHz and 4Gb RAM, Debian OS 7.0 and ROS Groovy.

In the first example the yellow can O_0 is located in the center of a box that has a lid O_1 acting as an obstacle that has to be removed in order to make the yellow can be reachable (see Fig. 6a). O_0 has associated four grasp configurations for each of the used hands (all of them grasping the object from the top) and O_1 has six grasp configurations for each hand (at three different points of the lid handle with two opposite orientations of the wrist), so both robots R_1 and R_2 can grasp the box lid and the yellow can. When the procedure is executed, a path to grasp O_0 was found for each of the robots R_1 and R_2, i.e. $P_{1,0}$ and $P_{2,0}$, and, as expected, both of them imply a collision with the box lid, i.e. $SO_{1,0} = SO_{2,0} = \{O_1\}$, then the corresponding flags are set as $f_{1,0} = f_{2,0} = B$. Fig. 6b shows the configuration of R_1 colliding with the obstacle O_1 while grasping O_0. Then, O_1 is added as child node of O_0 in the graph G for both R_1 and R_2. Fig. 6c shows the resulting graph G. When the first node representing O_1 is analyzed, the results is that it can be removed using any of the robots without any additional obstacles, i.e. $P_{1,1}$ and $P_{2,1}$ are collision free meaning that the two robots R_1 and R_2 can remove the box lid without obstacles, thus the corresponding flags are set as $f_{1,1} = f_{2,1} = F$. At this point, the analysis of G already gives a valid assignment of robot actions: R_2 is selected to remove O_1 (using a path $P_{2,1}$ without obstacles) and R_1 is used to grasp O_0 (using a path $P_{1,0}$, also without obstacles once



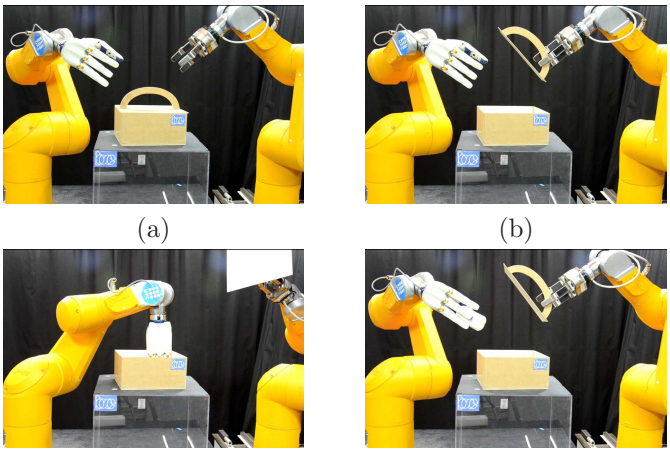
(a) (b) (c)

Figure 6: Example 1; (a) O_0 inside the box (without the lid), (b) collision configuration of the robot R_1 with the lid box, (c) precedence graph G and the assignment of robot actions.

O_1 has been removed), as shown in Fig. 6a. Snapshots of the real execution of this example are shown in Fig. 7.

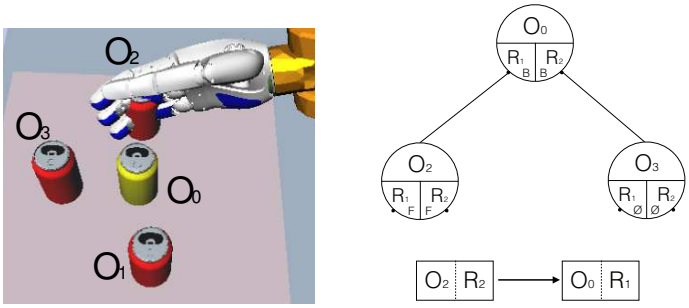
In the second example the yellow can O_0 lies on the table among other red cans O_1, O_2 and O_3 (see Fig. 8a). In this example it is not allowed to grasp O_0 from the top (as it was done in the previous example), i.e. the set of grasping configurations associated with O_0 for each of the two hands includes eight cylindrical grasps (like those in Fig. 4, where for clarity only four grasps were shown), while the red cans can be grasped either with four grasps from the top and with eight cylindrical grasp for each hand. The paths $P_{1,0}$ and $P_{2,0}$ found to grasp O_0 with R_1 and R_2 have collision with the red cans O_2 and O_3 respectively, which were then added to G as children of O_0 for R_1 and R_2 (see the graph G in Fig. 8b). Analyzing O_2, it is found that it can be removed with both robots R_1 and R_2 without collisions, i.e. $P_{1,2}$ and $P_{2,2}$ are collision free. At this point the procedure can already select an assignment of robot actions to solve the task: R_2 is selected to remove O_2 using $P_{2,2}$ and then R_1 is in charge of grasping O_0 using $P_{1,0}$ once O_2 has been removed. Fig. 9 shows some snapshots of the two robots executing their assigned actions to arrive to the yellow can O_0.

In the third example, the yellow can O_0 lies again on the table among several other red cans O_1 to O_5 (see Fig. 10a). In this example, as in the previous one, it is not allowed to grasp O_0 from the top while the red cans can be grasped either from the top or with a cylindrical grasp. In this case, the path $P_{1,0}$ found to grasp O_0 with R_1 has collisions with the red cans O_4 and O_5 that were added to G as children of O_0 for R_1, and the path $P_{2,0}$ found



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

Figure 7: Example 1; (a) initial configuration, (b) R_2 removing O_1, (c) R_1 grasping O_0, (d) final configuration.



(a) (b)

Figure 8: Example 2; (a) collision configuration of the robot R_1 with O_2, (b) precedence graph G and the assignment of robot actions.

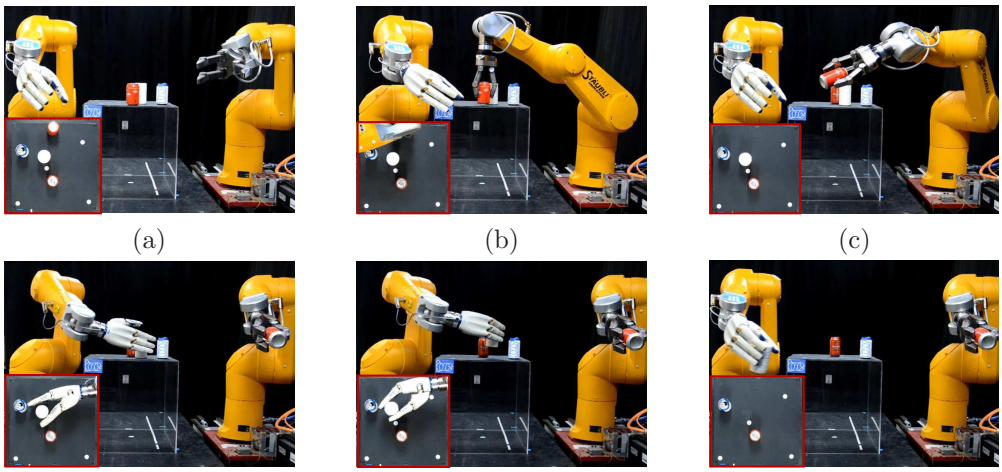
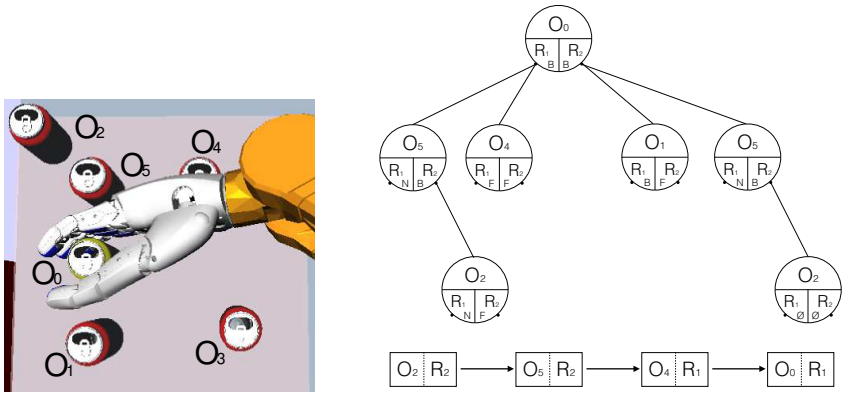


Figure 9: Example 2; (a) initial configuration, (b) R_2 grasping O_2, (c) R_2 removing O_2, (d) R_1 grasping O_0, (e) R_1 moving O_0, (f) final configuration.

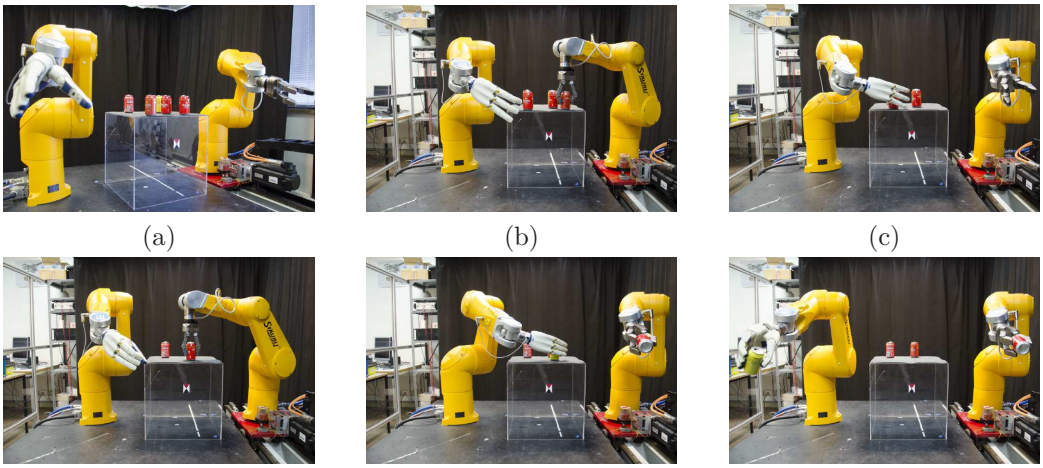
to grasp O_0 with R_2 has collisions with the red cans O_5 and O_1 that were added to G as children of O_0 for R_2. Fig. 10b shows the resulting graph G. Iteratively exploring and expanding the graph give the following results. O_5 is not reachable with R_1 (note the flag $f_{1,5} = N$) while using R_2 a path $P_{2,5}$ was found with a collision with O_2, which is added to G as child of O_5 for R_2. O_4 can be grasped and removed with both robots using $P_{1,4}$ and $P_{2,4}$ that were found without any collision. On the other branch of G, to remove O_1 a path $P_{1,1}$ was found including collisions, but since a collision free path $P_{2,1}$ was also found the node O_1 is not expanded, and O_5 produces the same result as above, so it is expanded adding O_2 as child node for R_2. Now, dealing with O_2, results that it is not reachable with R_1 (note the flag $f_{1,2} = N$) but it is removable with R_2 using a path $P_{2,2}$ found without collisions. At this point the procedure can already assign the robot actions to solve the task: R_2 is selected to remove O_2, R_2 is in charge of removing O_5, R_1 is selected to remove O_4, and finally R_1 is in charge of manipulating O_0 once the other obstacles were removed. Fig. 11 shows some snapshots of the task execution following the assignment of robot actions until being able to manipulate to the yellow can O_0.

In order to give an idea of the running performance of the approach, Ta-



(a) (b)

Figure 10: Example 3; (a) collision configurations of the robot R_1 with the removable obstacles O_4, (b) precedence graph G and the assignment of robot actions.



(a) (b) (c)

(d) (e) (f)

Figure 11: Example 3; (a) initial configuration, (b) R_2 grasping O_2, (c) R_1 grasping O_4 while R_2 has just dropped O_2, (d) R_2 grasping O_5 while R_1 is going to drop O_4, (e) R_1 grasping O_0, (f) final configuration.

Example 1

Path $P_{i,j}$	Total time(s)	#Samples	#Samples connected	#Paths
$P_{1,0}$	6.12	218	186	3
$P_{2,0}$	0.21	92	2	1
$P_{1,1}$	3.93	113	21	1
$P_{2,1}$	0.32	68	7	1

Example 2

Path $P_{i,j}$	Total time(s)	#Samples	#Samples connected	#Paths
$P_{1,0}$	1.63	49	4	4
$P_{2,0}$	1.17	93	7	3
$P_{1,2}$	3.85	101	17	1
$P_{2,2}$	0.38	87	2	1

Example 3

Path $P_{i,j}$	Total time(s)	#Samples	#Samples connected	#Paths
$P_{1,0}$	4.65	62	30	3
$P_{2,0}$	0.34	83	7	1
$P_{1,1}$	1.13	64	27	2
$P_{2,1}$	0.23	87	5	1
$P_{1,2}$	-	-	-	-
$P_{2,2}$	0.18	76	2	1
$P_{1,4}$	1.32	62	30	3
$P_{2,4}$	5.87	202	30	1
$P_{1,5}$	-	-	-	-
$P_{2,5}$	0.44	92	4	3

Table 1: Running information for the three examples.

ble 1 shows for each example: the time in seconds required to computed the paths $P_{i,j}$, the total number of the samples generated in the configuration space, the number of connected samples in the roadmap, and the number of paths $P_{i,j}$ computed for each O_j (note that there is a $P_{i,j}$ for each grasp configuration in $SC_{i,j}$, the set of grasping configurations selected in Algorithm 2 from the set of given grasps SC_j associated with O_j).

6 Summary and Future Works

The paper has presented a new approach for the computation of robot paths to manipulate objects considering that it may be necessary to remove obstacles from the environment in order to make the paths actually feasible. The approach is based on what we call *Probabilistic Road Map with Obstacles* (PRMwO), which returns the path for a robot to reach a particular goal and the list of obstacles that must be removed. The information obtained from the PRMwO allows the generation of a precedence graph that is used to assign robot actions to each robot in order to properly remove the obstacles and reach the desired object.

The whole approach has been implemented and successfully applied to a dual-arm robotic system. It is noteworthy that even when the approach was implemented for the described dual-arm robotic system (available both in simulation and as a real physical system) it could be really applied to the case of more than two robots working in a shared environment. Note than none of the presented algorithms imposes a constraint in the number n of involved robots, number that only appears in iterative loops of the type "for each robot R_i $(i \in \{1, ..., n\})$" (e.g. in line 1 of Algorithm 5), or in questions like "if $\forall i$ $(i \in \{1, ..., n\})$ ..." (e.g. in line 13 of Algorithm 6) or "If $\nexists i$ $(i \in \{1, ..., n\})$ such that ..." (e.g. in line 1 of Algorithm 7), therefore the application of the approach to more than two robots is straightforward.

As future work it is considered the inclusion of a grasp planner such that the grasping configurations are determined during the manipulation planning when necessary. Even when this would increase the computational cost (compared with the use of predefined grasping configurations), it would avoid the need of having predefined grasping configurations for each object and several grasping configurations would be computed only when there are difficulties to reach an abject or when the cost of a first solution is high (e.g. a large number of obstacles have to be removed). Another topic for future work is the optimization of the distribution of actions between the robots according to different criteria, including for instance the possibility that one robot transfers a grasped object to the other in order to be ready to do the next action if in this way it is more efficient.

Acknowledgements

This work was partially supported by the Spanish Government through the projects DPI2010-15446, DPI2011-22471 and DPI2013-40882-P.

References

[1] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, D. Kragic, Dual arm manipulation - A survey, Robotics and Autonomous Systems 60 (10) (2012) 1340–1353.

[2] N. Vahrenkamp, T. Asfour, R. Dillmann, Simultaneous Grasp and Motion Planning, IEEE Robotics and Automation Magazine 19 (2012) 43–57.

[3] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, R. Dillmann, Humanoid motion planning for dual-arm manipulation and re-grasping tasks, in: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2009, pp. 2464–2470.

[4] R. Shauri, K. Nonami, Assembly manipulation of small objects by dual-arm manipulator, Assembly Automation 31 (2011) 263–274.

[5] A. Edsinger, C. Kemp, Two Arms Are Better Than One: A Behavior Based Control System for Assistive Bimanual Manipulation, Lecture Notes in Control and Information Sciences 370 (2008) 345–355.

[6] S. M. LaValle, Planning Algorithms, Cambridge University Press, 2006.

[7] J.-C. Latombe, Robot motion planning, Kluwer Academic Publishers, 1991.

[8] N. J. Nils, A mobile automaton: an application of artificial intelligence techiques, in: Proc. of the 1st. International Joint Conference on Artificial Intelligence, 1969, pp. 509–520.

[9] C. O'Dunlaing, M. Sharir, C. Yap, Retraction: A New Approach to Motion-Planning (Extended Abstract), in: STOC, 1983, pp. 207–220.

[10] P. Hart, N. Nilsson, B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, in: IEEE Transactions on Systems Science and Cybernetics, 1968, pp. 100–107.

[11] W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik, 1959.

[12] O. Khatib, Real-time Obstacle Avoidance for Manipulators and Mobile Robots, in: Proc. IEEE Int. Conf. Robotics and Automation, 1985, pp. 500–505.

[13] L. E. Kavraki, P. Svestka, J. Latombe, M. Overmars, Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, in: Proc. IEEE Int. Conf. Robotics and Automation, 1996, pp. 566–580.

[14] Y. Yang, O. Brock, Adapting the sampling distribution in PRM-Planners based on an Approximated Medial Axis., in: Proc. IEEE Int. Conf. Robotics and Automation, 2004, pp. 4405–4411.

[15] A. Yershova, L. Jaille, T. Simeon, S. LaValle, Dynamic-Domain-RRTs: Efficient Exploration by Controlling the Sampling Domain., in: Proc. IEEE Int. Conf. Robotics and Automation, 2005, pp. 3856–3861.

[16] L. Zhang, D. Manocha, An Efficient Retraction-base RRT Planner., in: Proc. IEEE Int. Conf. Robotics and Automation, 2008, pp. 3743–3750.

[17] J. Zucker, M. Kuffner, J. Bagnell, Adaptive workspace biasing for sampling-based planners., in: Proc. IEEE Int. Conf. Robotics and Automation, 2008, pp. 3757–3762.

[18] J. Rosell, L. Cruz, R. Suárez, A. Pérez, Importance Sampling based on Adaptive Principal Component Analysis, in: Proc. of the IEEE Int. Symposium on Assembly and Manufacturing, 2011, pp. 1–6.

[19] R. Bohlin, L. Kavraki, Path Planning Using Lazy PRM, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 2000, pp. 521–528.

[20] E. Cheng, P. Frazzoli, S. LaValle, Improving the performance of Sampling-Based Planners by using a Symmetry-Exploition Gap Reduction Algorithm., in: Proc. IEEE Int. Conf. Robotics and Automation, 2004, pp. 4362–4368.

[21] J.-M. Lien, Y. Lu, Planning Motion in Similar Environments, in: Proc. of Robotics: Science and Systems., Seattle, USA, 2009.

[22] R. Guernane, N. Achour, Generating optimized paths for motion planning, Robotics and Autonomous Systems (2011) 789–800.

[23] C. Galindo, J. Fernández-Madrigal, J. González, A. Saffiotti, Robot task planning using semantic maps, Robotics and Autonomous Systems (2008) 955–966.

[24] G. Wilfong, Motion planning in the presence of movable obstacles, in: Proc. of the 4th Annual ACM Symposium on Computational Geometry, 1988, pp. 279–288.

[25] P. Chen, Y. K. Hwang, Practical path planning among movable obstacles, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 1, 1991, pp. 444–449.

[26] K. Okada, A. Haneda, H. Nakai, M. Inaba, H. Inoue, Environment manipulation planner for humanoid robots using task graph that generates action sequence, in: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2004, pp. 1174–1179.

[27] M. Stilman, J. Kuffner, Navigation among movable obstacles: Real-time reasoning in complex environments, in: Journal of Humanoid Robotics, 2004, pp. 322–341.

[28] M. Stilman, K. Nishiwaki, S. Kagami, J. Kuffner, Planning and Executing Navigation Among Movable Obstacles, in: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2006, pp. 820–826.

[29] Y. Kakiuchi, R. Ueda, K. Kobayashi, K. Okada, M. Inaba, Working with movable obstacles using on-line environment perception reconstruction using active sensing and color range sensor, in: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2010, pp. 1696–1701.

[30] M. Dogar, M. Koval, A. Tallavajhula, S. S., Object Search by Manipulation., in: Proc. IEEE Int. Conf. Robotics and Automation, 2013, pp. 4973–4980.

[31] C. Rosales, L. Ros, J. M. Porta, R. Suárez, Synthesizing grasp configurations with specified contact regions, International Journal of Robotics Research 30 (4) (2011) 431–443.

[32] F. Gilart, R. Suarez, Determining Force-Closure Grasps Reachable by a Given Hand, in: 10th IFAC Symposium on Robot Control, SYROCO, 2012, pp. 235–240.

[33] A. Montaño, R. Suárez, An On-Line Coordination Algorithm for Multi-Robot Systems, in: 18th Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA, 2013, pp. 1–7.