Manipulation tasks with a dual arm system including obstacles removing

Carlos Rodríguez Andrés Montaño Raúl Suárez *

Abstract

The paper deals with the problem of planning movements of a two-hand system, considering the possibility of using one hand to remove potential obstacles in order to grasp a desired object with the other hand. The approach is based on a Probabilistic Road Map that does not rule out samples implying collisions with removable objects but instead classify 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 workspace to make the path be actually valid. The approach has been implemented and different tests were performed with considering a real two-hand robotic system with one hand in charge of grasping a desired object and the other in charge of removing the potential obstacles. Some running examples both in simulation and a real workcell are presented in the paper using simulations and real experimentations.

1 Introduction

Moving objects around is a problem of significant relevance in the application of robots, both in industrial and service robotics. It involves several other associated problems, among which there are two main ones: the determination of a proper grasp configuration for the available arm and gripper (considering aspects like the shape of the object and the task to be performed), and the determination of collision free paths to arrive to a grasp configuration and to move the 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 second problem under the following context: two robots are available (see Fig. 1), we want to grasp a particular object with one of the robots, there may be other removable objects in the environment acting as obstacles, and we can use the second robot to remove the obstacles if it is necessary. Note that this is a frequent problem in everyday life for the human beings, and so will be for humanoids robots.

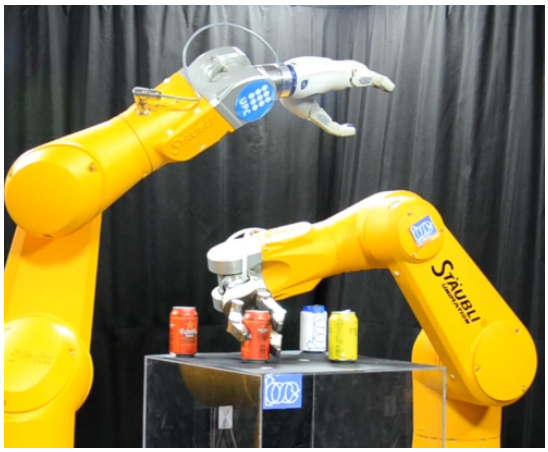


Figure 1. Dual arm system.

The proposed approach simultaneously solve the problem of finding the robot movements to grasp and move the desired object, determining which are the objects acting as obstacles that must be removed, and the problem of finding the robot movements to remove them. The approach allows the consideration of different grasping configurations for each object and selects the ones that allow a real solution to the stated problems.

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

2 Related Work

During the last two decades there has been significant research work concerning robot motion planning problems. Quite effective general strategies have been developed using sampling-based techniques, and among the most relevant approaches are the *Probabistic Road Map planners* (PRM) [6] and *Rapidly-exploring Random Trees planners* (RRT) [9]. In order to speed up query path

*The authors are with the Institute of Industrial and Control Engineering (IOC) - Polytechnic University of Catalonia (UPC), Barcelona, Spain (carlos.rodriguez.pacheco, andres.felipe.montano, raul.suarez@upc.edu).
This work was partially supported by the Spanish Government through the projects DPI2010-15446 and DPI2011-22471.

planning, some variants of PRM planners were proposed. One example is the *Lazy PRM* [2], that builds a roadmap but without checking for collisions in the first phase; if a collision with the obstacles occurs, the corresponding nodes and edges are then removed from the roadmap and the search starts again, the procces is repeated until a collision-free path is found. On the other hand, the *RU-PRM planner* [10] considers scenarios where there exist similar known obstacles and, from this premise, it generates collision-free paths running along the border of the obstacles and then tries to connect these free paths to their nearest neighbors, thus yielding a solution path that skirts the obstacles.

It is important to highlight that the planners mentioned above have been designed to avoid collisions with any obstacle, either fixed or removable. In contrast, one contribution of the approach proposed in this work is the planification of paths considering that the removable object acting as obstacles can be ignored if they are properly removed on time.

Using a dual arm robot to do manipulation tasks moving removable objects [16] implies avoiding collisions between the robot and any other object in the environment, as well as self collisions between the arms or hands, and collisions between the manipulated objects and the robots or any other object in the environment. Relevant works dealing with robot motion planning considering removable obstacles includes [19], that was precursor and has shown that motion planning among removable obstacles is a non deterministic polynomial time hard problem (NP-hard problem), and [3], that created a grid based planner that heuristically tries to minimize the cost of moving obstacles out of the way.

On the other hand, motion planning developments made for humanoid robots address the problem of removable obstacles by building a manipulation space based on a graph that includes all possible geometric paths to move from an initial position to a final one and the respective obstacles encountered in each path [12]. Other approaches solve the planification problem using an RRT-Connect planner and the geometric model of the objects in the environment to plan robot movements to reach the goal by pushing out of the way removable obstacles [17].

There are also proposals that use sensors in order to do an online navigation planning when the robot is in unknown environments; it includes the recognition of removable obstacles and the calculation of how to push them out off the road [5]. Another approach [18] developed a planner considering restrictions on the robot joints, it is applied to the case of a robotic arm on a mobile base that performs manipulation tasks to organize some workshop tools by checking if the robot path can reach a free place to locate a tool or can remove it from a cabinet. This planning algorithm is not able to compute the path of a particular removable object if it is directly blocked by another object.

Our approach is based on a modified PRM, as will be described in next section, to deal with the problem of finding collision free paths for two robots sharing the workspace when one of them must perform a given manipulation task and, if necessary, the other is in charge of removing obstacles. The approach looks for collision free paths and at the same time determines which are the objects to be removed from the workplace in order to allow a valid solution. The robots paths are computed in a decentralized way for each arm-hand system and a precedence tree is constructed indicating which objects and in which order have to be removed in order to allow the goal to be satisfactorily reached without collisions.

3 Proposed Approach

This section describes the proposed approach, first an overview is given and then the approach is formally described by detailing the corresponding algorithms.

3.1 Overview

Consider two robot arms in a workspace where there is an object of interest to be grasped and some other removable objects, i.e. objects that can be removed from the scene by the robots themselves. The removable objects may act as obstacles that do not allow access to the object of interest. In this context the problem to be solved is: find a path for one robot to grasp the desired object and a set of paths for the other robot to remove (if necessary) all the obstacles.

Basically, the proposed approach is as follows. First, a basic vision system is used to identify the removable objects and compute their positions in the real workcell. Using this information the removable objects are added to the 3D model of the workcell. Then, a PRM is used to find a path for the robot in charge of grasping the object of interest (from now on referred to as the main robot), but, as a difference with the typical use of PRM a sample of the robot configuration that implies a collision of the robot with any removable object 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 construction, if there are collisions with the removable objects these are just added to a list 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 a path for the main robot to grasp the object of interest and at the same time an associate list of obstacles that must be removed from the environment in order to make the path being collision free and therefore really valid.

Next step is the search of a path for the other robot (from now on referred to as the assistant robot) to remove each of the obstacles for the main robot. This is done using also a PRMwO for the assistant robot to grasp and remove each obstacle. Since some other objects may act as

new removable obstacles, the procedure is iteratively repeated until a path without obstacles is found for each of the objects to be removed or a loop is found (i.e one object is an obstacle to grasp another one and viceversa). In order to increase the probability of finding feasible valid paths, it is considered that the objects can be grasped with a set of different hand configurations, which are taken as different goal points when the PRMwO is built. The grasping configurations can be obtained using different procedures (see for instance [15, 4]). In this work it is considered 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.

Note that the collision check performed along the configurations of a robot path in the environment 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. In practice, if the arm, the hand and the grasped object are always considered while building the PRMwO, the robots can follow the same path to go toward the object and to remove it from the scene after being grasped.

Once the planning phase is finished, and therefore the necessary geometric paths were determined for the two robots, the trajectories along these paths are determined. This is done with the aim of optimizing the time needed to remove the obstacles and execute the desired task. This coordination, adjusting the temporal evolution of the robots along the obtained paths, has been already developed and implemented [11] but it is a complementary module outside the scope of this work and therefore in not described here.

3.2 Formal description of the approach

This section formally presents the developed algorithms. The following basic nomenclature is used:

R**:** A robot in the workcell (it includes the arm and the hand).

C_o**:** Initial configuration for R.

C_g**:** Goal (grasping) configuration for R.

$path$**:** Path of R from C_o to C_g and return to C_o.

SO**:** Set of obstacles (removable objects) for $path$.

O**:** Removable object in the environment; includes the object model, its configuration C in the workspace and an associate set of grasping configurations G_O.

G_O**:** Robot configuration to grasp the object O referred to the object reference system.

C**:** Robot configuration to grasp the object O referred to the absolute world reference system.

SC**:** Set of configurations C.

G_{PRM}**:** Graph describing the PRM for the search of $path$.

V**:** Vertex of G_{PRM}.

SV**:** Set of vertices of G_{PRM}.

SE**:** Set of edges of G_{PRM}.

PT**:** Obstacle precedence tree.

S**:** Environment model; it includes the model of the environment, the models and initial configurations of the robots, positions and grasping configurations of the removable objects (including the object that must be manipulated).

The subscripts M and A will be used to indicate that the items R, C_o, C_g, and $path$ are associated to the "Main" or to the "Assistant" robot respectively (e.g. R_M and R_A, $path_M$ and $path_A$). The superscript $*$ indicates the selected goal object to be grasped, i.e. O^*.

For simplicity, it is assumed that the task of grasping and manipulating the selected object O^* have been already assigned to one of the robots while the other is in charge of removing the obstacles (in case it is necessary). This means that which robot acts as R_M or R_A is assigned beforehand. Besides, the initial configuration C_o of each robot is also given, and, without lost of generality, C_o is used as final destination to drop the removed objects.

The main procedure is presented in Algorithm 1. First, PT is created with O^* as the root node without descendants, and then a path $path_M$ is sought for R_M using the function findPath(R, O) (describe below). If there are no obstacles along $path_M$ then it is already a solution to the problem, otherwise each obstacle O_i is added to PT as descendant of O^* and an iterative procedure is started looking for paths $path_{A_i}$ for R_A that allow the grasping and removing of each obstacle O_i. If the path $path_{A_i}$ has no obstacles it can be added to the plan and the corresponding obstacle O_i is removed from PT, otherwise the obstacles along $path_{A_i}$ are added to PT as descendants of O_i (i.e. a new level of descendants is generated, see an example in Fig. 2). Note that the maximum number of $path_{A_i}$ to be determined is bounded by the maximum number n_{max} of obstacles in the work environment. The procedure is iteratively repeated for the nodes of lower level in PT, adding new levels of descendants or removing the nodes once their descendants have been removed. The procedure ends with success when it is possible to remove the root node and therefore the selected object O^* can be successfully grasped and manipulated, or the procedure ends with failure when a lower node of PT cannot be removed because it appears at an upper level in the same branch of the tree. This type of failure can eventually be solved considering that the robots can exchange their roles as R_M or R_A.

The function findPath(R, O) is described in Algorithm 2, given a robot R and an object O this function returns: a path for R starting at an initial configuration, going to a grasping configuration C_g associated

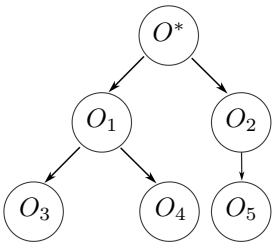


Figure 2. Example of a precedence tree PT. O_1 and O_2 must be removed to allow grasping O^*, O_3 and O_4 must be removed to allow removing O_1, and O_5 must be removed to allow removing O_2.

Algorithm 1 solveProblem(S)

Require: S
Ensure: $\{path_{A_1}...path_{A_n}, path_M\}, \quad n \in \{0....n_{max}\}$
1: Plan = $\emptyset$
2: Add O^* to PT (i.e. create PT)
3: $path_M$, $SO_M \leftarrow$ findPath(R_M,O^*)
4: **if** $SO_M \neq \emptyset$ **then**
5: $\forall O_i \in SO_M$ add O_i to PT as descendent of O^*
6: **while** there are terminal nodes $O_j \neq O^*$ in PT **do**
7: **for** each terminal node O_j **do**
8: $path_{A_j}$, $SO_{A_j} \leftarrow$ findPath(R_A,O_j)
9: **if** $SO_{A_j} \neq \emptyset$ **then**
10: $\forall O_k \in SO_{A_j}$ add O_k to PT as descendent of O_j
11: **else**
12: Add $path_{A_j}$ to Plan
13: Remove O_j from PT
14: **end if**
15: **end for**
16: **end while**
17: **end if**
18: Add $path_M$ to Plan
19: **return** Plan

to O and coming back (with O grasped in the hand) to the initial configuration and the set SO of obstacles that must be removed in order to make the path being collision-free. The first step in this function is the determination of a set of valid (kinematically reachable by R) grasp configurations SC from those already associated to the model of the object O, this is done using the function testGrasp(R, O) described in Algorithm 3. testGrasp(R, O) simply takes each grasping configuration G_{O_i}, transforms it to C_i according to the current configuration of O in the workspace, checks whether C_i can be reached by the robot R (i.e. verifies that the inverse kinematics of R has a valid solution for C_i) and returns the set SC with the reachable configurations C_i. After this, for each reachable grasping configuration C_i a path for the robot is searched using the probabilistic roadmap with obstacles PRMwO(C_o,C_g,R) presented in Algorithm 4, that, besides the $path$, returns the list of objects that are

Algorithm 2 findPath(R,O)

Require: : R, O
Ensure: : $path$ for R and the set of obstacles SO
1: $SC \leftarrow$ testGrasp(O, R)
2: define $N >$ #objects_in_workspace
3: **for** each $C_i \in SC$ **do**
4: $path_i$, $SO_i \leftarrow$ PRMwO(C_o,C_i,R)
5: **if** range(SO_i) $< N$ **then**
6: $path \leftarrow path_i$
7: $SO \leftarrow SO_i$
8: $N \leftarrow$ range(SO_i)
9: **end if**
10: **end for**
11: **return** $path$, SO

Algorithm 3 testGrasp(R, O)

Require: : R, O
Ensure: : A set $SC(O)$ of configurations C reachable by R
1: $SC = \emptyset$
2: **for** each $G_{O_i} \in$ to SG_O **do**
3: Compute C by combining C and G_{O_i}
4: **if** Inverse Kinematics of (R) in C is reachable **then**
5: Add C to SC
6: **end if**
7: **end for**
8: **return** SC

obstacles for R when moving along $path$. The function range(SO_i) returns the number of obstacles in $path_i$ (i.e. the number of elements in SO_i) and is used to select the $path$ with minimum number of obstacles as possible candidate to solve the task.

The function PRMwO(C_o,C_g,R) initializes a PRM with the initial and goal configurations C_o and C_g for the robot R. Then, N_m samples Smp are generated. If a sample Smp is at a distance smaller than a given threshold D_m from a vertex V of G_{PRM}, the segment defined by Smp and V is checked for collisions of R with the environment. If there are collisions the sample is rejected, otherwise the segment is added as an edge to the set of edges SE of G_{PRM}. Then, SE is checked for collisions of R with the removable objects O_i and, if there are collisions, the set of collided objects are associated to SE. The procedure is repeated until C_o and C_g are connected. At that point the graph G_{PRM} is searched for the shortest path O_i between C_o and C_g with minimum number of collision with removable objects, which is returned as solution path together with the set SO of objects that generate collisions along the path.

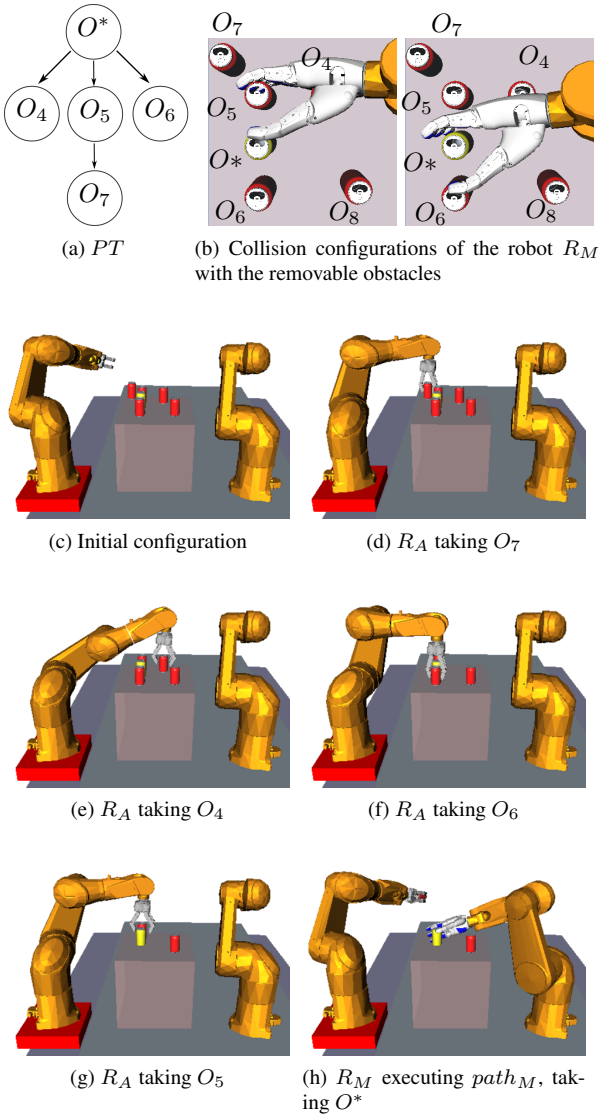
4 Experimental Results

The proposed approach has been implemented inside the home-developed path planning framework called *the Kautham Project*[1] [13], which was developed with the

[1] Website of The Kautham Project where is shows the development, applications, publications and download link of this simulation tool. `https://sir.upc.edu/projects/kautham/`

Algorithm 4 PRMwO(C_o,C_g,R)

Require: : C_o, C_g,R
Ensure: $path$ and SO
1: $SE = \emptyset$, $SO = \emptyset$
2: $SV \leftarrow \{C_o, C_g\}$
3: **for** $N = 0$ to $N < N_m$ (N_m: given maximum size of the sample set) **do**
4: Smp= getSample()
5: Select $V \in SV$
6: Add Smp to SV
7: **if** distance$(V, Smp) < D_m$ (D_m: given maximum neighborhood distance) **then**
8: $E_{s_i} \leftarrow connectSamples(Smp, V)$
9: **if** E_{s_i} does not imply collision of R with the environment **then**
10: **if** E_{s_i} implies collision of R with $O_i \in SE$ **then**
11: $SO_i \leftarrow O_i$
12: **end if**
13: **end if**
14: **end if**
15: **if** C_o and C_g are connected **then**
16: Break
17: **end if**
18: **end for**
19: $path \leftarrow \{$shortest $path$ between C_o and $C_g \in G_{PRM}\}$
20: $SO \leftarrow \{SO_i$ associate with $E_{s_i} \in path\}$
21: **return** $path$ and SO



(a) PT (b) Collision configurations of the robot R_M with the removable obstacles

(c) Initial configuration (d) R_A taking O_7

(e) R_A taking O_4 (f) R_A taking O_6

(g) R_A taking O_5 (h) R_M executing $path_M$, taking O^*

Figure 3. Example 1: (a) Precedence tree PT, (b) Collision configurations, (c) to (h) Snapshots of a simulated execution.

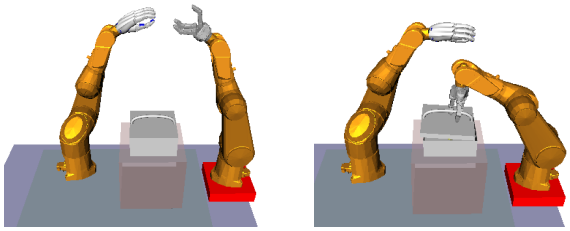
open source and cross-platform directives in mind and using libraries Qt [1] for the user interface, Coin3D [7] for the graphical rendering, PQP [8] for the collision detection and ROS [14] for the communications layer. This framework provides the developer with several tools needed for the development 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 (e.g. 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 with 6 DOF, a Schunk Anthropomorphic Hand (SAH) with 13 DOF, and a Schunk Dexterous Hand (SDH2) with 7 DOF. The object to be grasped is a yellow can and there are some objects in the scene that act as potential obstacles that do not allow a direct access to it. The model of each object has associated eight different grasp configurations for the hand. The initial configuration for each robot is given, and it is also used as final position for the robot paths, i.e. R_M must finish with the yellow can in that position and R_A drops there the removed obstacles. 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.

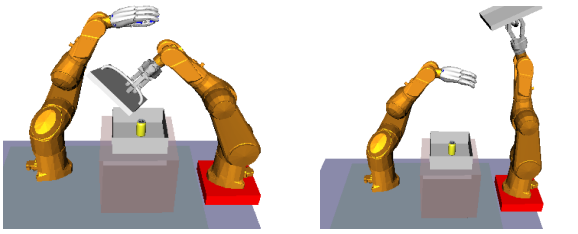
In the first example a yellow can and several red cans are on a table, as shown Fig. 3. The given eight grasping configurations of the cans correspond to axial grasps, i.e. grasping the can from the top is not allowed. A $path_M$ was found with a collision with the red cans O_4, O_5, O_6 that were added to PT as childs of O^*. To remove O_4, a $path_{A_4}$ was found without collisions, then, to remove O_5, a $path_{A_5}$ was found but it has a collision with the red can O_7, this is added to PT as child of O_5 and new path $path_{A_7}$ was found to remove O_7 without collisions. The path $path_{A_6}$ was found without collisions. Fig. 3 illustrates the execution of this example.

In the second example, the can is located inside a box (see Fig. 4). $path_M$ was found with a collision with the box lid (Fig. 4b shows a configuration of R_M colliding with the obstacle). Then, a path $path_A$ was found for the assistant robot R_A to remove the box lid. Fig. 4 illustrates

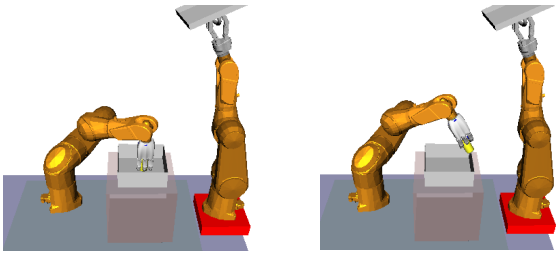
(a) PT (b) Collision configuration of the robot R_M with the lid box



(c) Initial configuration (d) R_A taking O_4

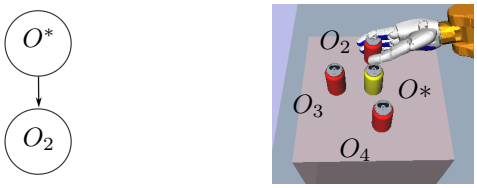


(e) R_A taking O_4 (f) Scene without obstacles

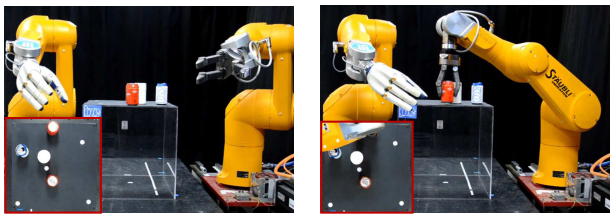


(g) R_M executing $path_M$ (h) R_M taking O^*

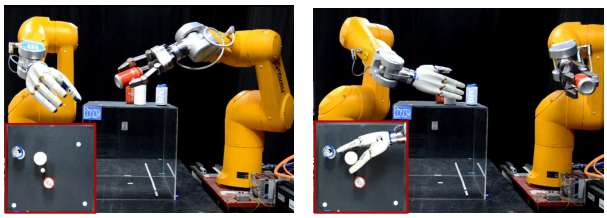
Figure 4. Example 2: (a) Precedence tree PT, (b) Collision configuration, (c) to (h) Snapshots of a simulated execution.



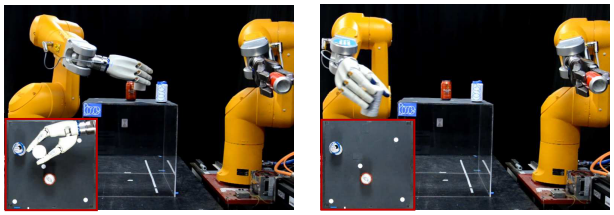
(a) PT (b) Collision configuration of R_M with O_2



(c) Initial configuration (d) R_A taking O_2



(e) R_A taking O_2 (f) R_M taking O^*



(g) R_M taking O^* (h) Final configuration

Figure 5. Example 3: (a) Precedence tree PT, (b) Collision configuration, (c) to (h) Snapshots of a real execution.

the execution of $path_M$ and $path_A$ with some snapshots.

In the third example, as in the first one, several cans lie on a table and the target is the yellow one (see Fig. 5). A $path_M$ was found with a collision with the red can O_2 that was added to PT as child of O^*. To remove O_2, a $path_{A_2}$ was found without collisions for R_A. Fig. 5 illustrates the real execution of $path_M$ and $path_{A_2}$ with snapshots of each path at the time of grasping the corresponding can[2].

The results of the three examples are summarized in Table 1 where it is shown: the times in seconds to computed the paths, the number of PRMwO computed, and the number of vertices generated for each path.

[2]The following link shows a video of the real execution of example 3: `https://iocnet.upc.edu/usuaris/raul.suarez/proyectos/muma/MUMA-videos-es.html`

5 Summary and Future Works

The paper has presented a new approach for the computation of robot paths considering that it may be necessary to remove objects from the environment in order to find a solution. 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 to be removed to make the path feasible. The PRMwO has been implemented and successfully applied to a dual arm system, considering that one arms must grasp an object from the scene and the other is in charge of removing the potential obstacles. A natural extensions of the implemented work is that the removal of each object could be assigned to each of the robots in a dynamic way, minimizing a cost function that considers, for instance, the current positions of the hands and the complexity of the expected movements.

Example 1

Paths	Time (s)	#PRMwO	#Vertices
$path_M$	4.9	3	56
$path_{A_4}$	4.7	1	113
$path_{A_5}$	20.9	4	409
$path_{A_6}$	64.5	1	411
$path_{A_7}$	13.8	1	108
Total	108.8	10	1097

Example 2

Paths	Time (s)	#PRMwO	#Vertices
$path_M$	2.1	3	56
$path_{A_4}$	1.3	1	52
Total	3.4	4	108

Example 3

Paths	Time (s)	#PRMwO	#Vertices
$path_M$	1.7	4	56
$path_{A_2}$	4.5	1	87
Total	7.6	5	143

Table 1. Running information for the three examples.

References

[1] J. Blanchette and M. Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[2] R. Bohlin and L. Kavraki. Path Planning Using Lazy PRM. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 521–528, 2000.

[3] P. Chen and Y. K. Hwang. Practical path planning among movable obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 1, pages 444–449, 1991.

[4] F. Gilart and R. Surez. Determining Force-Closure Grasps Reachable by a Given Hand. In *10th IFAC Symposium on Robot Control, SYROCO*, pages 235–240, September 2012.

[5] Y. Kakiuchi, R. Ueda, K. Kobayashi, K. Okada, and 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*, pages 1696–1701, 2010.

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

[7] Kongsberg Oil & Gas Technologies. Coin3D - 3D Graphics Development Tools. www.coin3d.org, December 2010.

[8] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast Proximity Queries with Swept Sphere Volumes. In *Proc. of Int. Conf. on Robotics and Automation*, pages 3719–3726, 2000.

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

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

[11] A. Montao and R. Surez. An On-Line Coordination Algorithm for Multi-Robot Systems. In *18th Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA*, September 2013. Accepted.

[12] K. Okada, A. Haneda, H. Nakai, M. Inaba, and 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*, pages 1174–1179, 2004.

[13] A. Pérez and J. Rosell. A Roadmap to Robot Motion Planning Software Development. *Computer Applications in Engineering Education*, September 2009.

[14] M. Quigley, B. Gekey, K. Cnley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. *Workshop on Open Source Robotics in IEEE Intl. Conf. on Robotics and Automation*, 2009.

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

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

[17] M. Stilman, K. Nishiwaki, S. Kagami, and J. Kuffner. Planning and executing navigation among movable obstacles. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, October 2006.

[18] M. Stilman, J.-u. Schamburek, J. Kuffner, and T. Asfour. Manipulation planning among movable obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2007.

[19] G. Wilfong. Motion planning in the presence of movable obstacles. In *Proc. of the 4th Annual ACM Symposium on Computational Geometry*, pages 279–288, 1988.