A Probabilistic Algorithm for Manipulation Planning under Continuous Grasps and Placements

A. Sahbani, J. Cortés, T. Siméon

LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse Cedex France

Abstract

An important skill of autonomous robots is the ability to carry out manipulation tasks. The solution to a manipulation problem generally consists in a sequence of elementary paths where an object is moved by a robot or it stays at a stable placement while the robot performs a re-grasping motion. Most existing planners require a finite set of configurations to achieve this task decomposition. We recently proposed in [13] an approach to automatically compute such intermediate configurations from continuous sets of stable placements and possible grasps of the movable object. This paper describes an improved algorithm based on this approach. It also presents several complex manipulation problems that illustrate the efficiency of the planner.

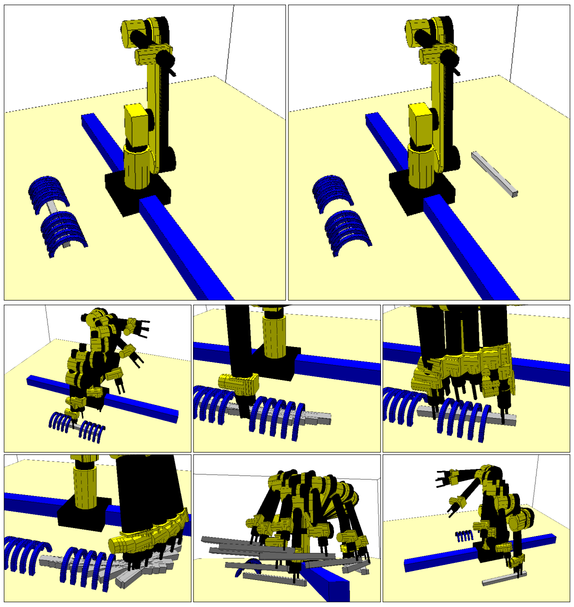

Figure 1: *A manipulation planning problem*

1 Introduction

Motion planning in the manipulation context appears as a constrained instance of the coordinated motion planning problem [10]. Two kinds of systems move in the same environment: robots and movable objects. The constraint is that movable objects can not move by themselves. Either they are transported by robots, or they stay in a stable placement. Considering such constraints leads to a more complex version of the planning problem.

The solution of a manipulation planning problem consists in a sequence of sub-paths satisfying these motion restrictions. In related literature [2, 10], motions of the robot holding the object at a fixed grasp are called *transfer paths*, and motions of the robot while the object stays at a stable placement are called *transit paths*. Figure 1 illustrates a manipulation planning example. The manipulator arm (initially on standby configuration) has to get the movable object (the bar) out of the cage, and place it on the other side of the environment. Achieving this task requires the following motions: the manipulator must first execute a transit path to grasp the movable object at the initial placement; then a sequence of transfer/transit paths separated by grasp/ungrasp operations allow to get one extremity of the bar out of the cage; a motion of the manipulator is performed to re-grasp the object by the extremity that was made accessible by the previous motion; the bar is then moved outside of the cage and a transfer path allows the specified goal position to be reached; finally, the robot moves back to its home position. Manipulation planning concerns the automatic decomposition of the manipulation task into such elementary motion planning sub-tasks, and the solution of each one of them.

Most of existing manipulation planning algorithms assume that finite sets of stable placement and possible grasps of a movable object are given in the definition of the problem (e.g. [1, 2, 4, 8, 11]). Consequently, a part of the manipulation task decomposition problem is thus resolved by the user. Returning to the example, getting the bar out of the cage requires a large number of precise placements and grasps that must be input data for these algorithms. Dealing with continuous grasp and placement sets may allow more sophisticated planners to be designed. The intermediate configurations linking the solution subpaths can be automatically computed. A technique treating this extension of the manipulation problem was recently presented in [13].

The contribution in this paper is to propose a more elaborated algorithm issued from the same ideas to solve manipulation tasks for a robot and a movable object. Section 2 recalls notation and briefly explains the method. The algorithm described in Section 3 computes a *manipulation graph* using *visibility-PRM* notions [15]. The implemented planner has demonstrated very good performance treating difficult manipulation planning problems. Some examples are commented in Section 4.

2 Theoretical Overview

Let $\mathcal{A}$ and $\mathcal{M}$ denote a robot and a movable object in a 3-dimensional workspace. The composite configuration-space of the two systems is $CS = C_{rob} \times C_{obj}$. CS_{free} is the sub-set in CS of all admissible configurations (i.e. configurations where the moving bodies do not intersect together or with the static obstacles). CP is the sub-space of CS_{free} defined as the set of free configurations where $\mathcal{M}$ is placed at a stable position. CG is the sub-space of CS_{free} defined as the set of free configurations corresponding to all possible grasps of the object $\mathcal{M}$. Transit paths are contained in CP while transfer paths are in CG. The intersection $CG \cap CP$ represents the manifold that corresponds to *the robot grasping the movable object on a stable placement.* A manipulation graph MG consists of a number of configurations of $CG \cap CP$ connected by transit and transfer paths.

The *reduction property* [3] shows that two configurations which are in a same connected component of $CG \cap CP$ can be linked by a manipulation path (i.e. a finite sequence of transit and transfer paths). The approach in [13] exploits this property to decompose the construction of MG. First the connected components of $CG \cap CP$ are computed, and then the connectivity of these sub-sets is determined using transit and transfer paths.

The referred approach admits the definition of continuous sets in a manipulation problem MP. Several classes of possible continuous grasps of $\mathcal{M}$, G_i, can be entered. Each G_i is defined by a transformation matrix T_{g_i} and a set of parameters, noted q_{grasp}, varying in a given interval. Similarly, several continuous regions of placement P_j can be defined, being P_j characterized by a transformation matrix T_{p_j} that defines a stable situation of the object, and a vector $q_{place} = (x, y, \theta)$ representing two horizontal translations inside a rectangular domain and a rotation around an axis perpendicular to this plane.

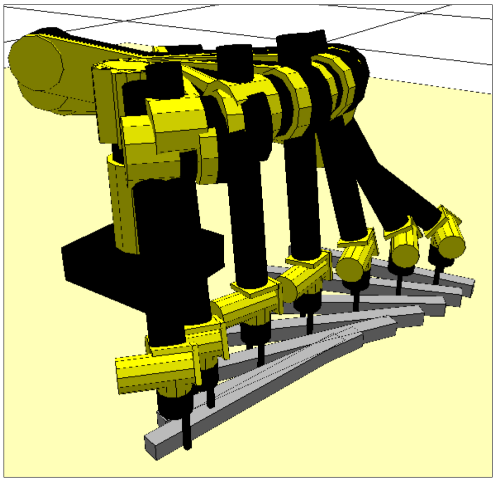

Figure 2: *Illustration of a $CG \cap CP$ path*

For each couple of sets (G_i, P_j), a virtual closed-chain system is formed when the robot grasps the movable object placed at a stable position. The connectivity of sub-spaces of $CG \cap CP$ corresponding to these couples can be analyzed using motion planning techniques for closed mechanisms. Figure 3 shows the kind of possible motions in these sub-spaces: the bar moves on the floor while sliding into the gripper's jaws. We refer to them as *$CG \cap CP$ paths.* The cited approach applies the technique presented in [5] to capture the topology of the $CG \cap CP$ manifold into a visibility-PRM [15]. Figure 3 shows the computed roadmap containing the four connected components of $CG \cap CP$ in the above presented example. Transit and transfer paths linking the different connected components of this roadmap are computed using a method that combines PRM [7, 12] with RRT [9] techniques.

Each connected component of $CG \cap CP$ can be seen as a mega-node of MG. The resulting graph will consist of a small number of such mega-nodes compared to the manipulation graph that would be obtained by considering discrete points of $CG \cap CP$. The proposed structuring significantly limits the number of path planning queries to be performed for connecting the nodes with collision-free transfer or transit paths.

The main difficulty in such a decomposition of the roadmap construction is to find the best way to interleave the two steps: computing $CG \cap CP$ subsets and linking them. Next section describes the algorithm that we propose to achieve the computation of MG.

3 Manipulation Planning Algorithm

The algorithm incrementally constructs a manipulation graph MG until it exceeds a given number

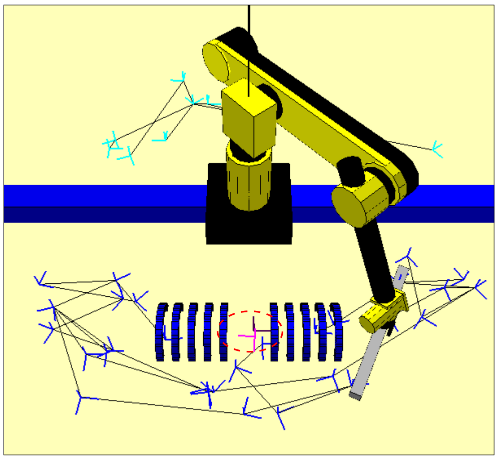

Figure 3: Visibility-PRM computed in $CG \cap CP$

of nodes or it constitutes a solution to the manipulation problem. The algorithm also stops if it is not able to expand the graph after a certain number of tries (MAX_NTRY). This number probabilistically determines the coverage of the composite configuration-space CS_{free} computed in MG.

EXPAND_GRAPH(MG)
$ntry \leftarrow 0$
$expanded \leftarrow$ FALSE
while ($ntry$<MAX_NTRY(MP)) and ($\neg expanded$)
  $S \leftarrow$ RANDOM_STRATEGY(MP)
  case $S = 0$
    $expanded \leftarrow$EXPAND_IN_G∩P(MG)
  case $S = 1$
    $expanded \leftarrow$EXPAND_USING_REGRASPING(MG)
  case $S = 2$
    $expanded \leftarrow$EXPAND_USING_TRANSFER(MG)
  $ntry = ntry + 1$
return $expanded$

Figure 4: Function performing an expansion step of the manipulation graph

The function EXPAND_GRAPH in Figure 4 achieves one expansion step of MG. Several strategies are possible. The choice of one of these strategies is made randomly.

The function RANDOM_STRATEGY performs a biased random selection that depends on the evolution of the size of MG. Following the discussion in Section 2, it seems reasonable to begin the construction by exploring the $CG \cap CP$ set. Therefore, during the first expansion steps, the probability to call EXPAND_IN_G∩P is higher. When the roadmap grows, the link between the different connected components will be tested using transit and/or transfer paths. The probability of calling the other strategies, EXPAND_USING_REGRASPING or EXPAND_USING_TRANSFER, increases as the percentage of the coverage of CS_{free}. This value is estimated by the fraction: $ntry$/MAX_NTRY.

3.1 MG expansion strategies

Expansion in the $CG \cap CP$ set. The function EXPAND_IN_G∩P, illustrated in Figure 5, carries out the exploration of the $CG \cap CP$ manifold. First, the function NEW_NODE generates a node N. This function starts randomly selecting one of the P_j sets given as input to MP. A stable configuration of the movable object, $p \in P_j$, is then chosen by randomly sampling q_{place}. Then, one of the grasp classes G_i and the corresponding parameters, q_{grasp}, are also randomly chosen. If the robot is able to make the grasping (i.e. if a inverse kinematics solution exists), then N is generated.

EXPAND_IN_G∩P(MG)
$N \leftarrow$ NEW_NODE(MP)
$n_{linked\ comp.} \leftarrow 0$
for $i = 1$ **to** N_COMP(MG) **do**
  if LINKED_IN_G∩P(N,C_i) **then**
    $n_{linked\ comp.} = n_{linked\ comp.} + 1$
if $n_{linked\ comp.} \neq 1$ **then**
  ADD_NODE(N, MG)
  UPDATE_GRAPH(MG)
  return TRUE
else
  return FALSE

Figure 5: Function trying to expand the manipulation graph using paths in $CG \cap CP$

Then, LINKED_IN_G∩P tests the link between N and each connected component of MG by means of paths in $CG \cap CP$. Note that this connection can be achieved only with the nodes computed from the same P_j and the same G_i as N. For each component, nodes with such characteristics are tested until a connection is feasible or all those nodes have been checked. Following the visibility principle [15], the node N is added to the graph only if it was linked to none or to more than one connected component. In the second case, the linked components are merged.

Expansion using re-grasping. The manipulation grasp can be expanded by means of two nodes issued from the same placement of the object and the transit path between them. This kind of expansion is interesting when the two nodes are in different connected components of $CG \cap CP$, because these sets can then be connected by a re-grasping motion. The function EXPAND_USING_REGRASPING aims to add such kind of nodes to the graph. Figure 6 shows the algorithm implemented in this function. N_1 and N_2 are nodes corresponding to a same placement of the movable object. Two different strategies to obtain N_1 are possible: to select it between the existing nodes in MG or to generate it. The interest of the first possibility is that the connection of N_1 to the rest of the graph has been already tested. Besides, this option allows more than two nodes to be produced from the same placement of the object, which can be necessary to solve certain manipulation problems. However, the option of generating a new node allows other placements of the object to be tried for the re-grasping motions than the contained in the graph. Our experience with manipulation planning problems has demonstrated the importance of testing such new placements, even if computing time must be spent in testing connections of N_1. Once N_1 has been chosen (or generated), N_2 is computed after randomly selecting a grasp.

EXPAND_USING_REGRASPING(MG)

```
N1, N2 ← SAME_PLACEMENT_NODES(MP,MG)
n_linked comp. ← 0
if not (REGRASP_IN_G∩P(N1, N2)) then
  if (TRANSIT_PATH(N1, N2)) then
    for i=1 to (N_COMP(MG) | Ci ≠ (C⇐N1)) do
      if LINKED_IN_G∩P(N2,Ci) then
        n_linked comp. = n_linked comp. + 1
if n_linked comp. ≠ 0 then
  if not (NODE_IN_GRAPH(N1,MG)) then
    ADD_NODE(N1, MG)
  ADD_NODE(N2, MG)
  UPDATE_GRAPH(MG)
  return TRUE
else
  return FALSE
```

Figure 6: *Function trying to expand the manipulation graph by re-grasping motions*

The function REGRASP_IN_G∩P returns true when the re-grasping motion between N_1 and N_2 is feasible in the $CG{\cap}CP$ set. In this case N_2 is in the same connected component as N_1, so it does not offer new information. Otherwise, the transit path between the two nodes is tested. If this path exists, N_2 is tried to be linked to the components of MG (excepting the component of N_1) using $CG{\cap}CP$ paths. In case of success, N_2 (and N_1, if it is not yet a node of the graph) is added to MG and the linked components are merged.

Expansion using transfer/transit paths. The last expansion strategy of the algorithm consists in linking the different connected components of MG using paths out of the $CG \cap CP$ set (i.e. transfer/transit paths). It is implemented by the function EXPAND_USING_TRANSFER described in Figure 7. An existing node N is randomly selected. Then, the function LINKED_OUT_OF_G∩P tests the connection between N and the connected component of MG (excepting its own component). For each component i, each one of its nodes N_{ij} is tried until a valid path between N and N_{ij} is

EXPAND_USING_TRANSFER(MG)

```
N ← EXISTING_NODE(MG)
n_linked comp. ← 0
for i=1 to (N_COMP(MG) | Ci ≠ (C ⇐ N)) do
  if LINKED_OUT_OF_G∩P(N,Ci) then
    n_linked comp. = n_linked comp. + 1
if n_linked comp. ≠ 0 then
  UPDATE_GRAPH(MG)
  return TRUE
else
  return FALSE
```

Figure 7: *Function trying to expand the manipulation graph using transfer/transit paths*

found. In the general case, such a path is the composition of a transfer and a transit path. Let us represent by (p_N,g_N) and ($p_{N_{ij}}$,$g_{N_{ij}}$) the parameters corresponding to the placement and the grasp of N and N_{ij} respectively. Virtual nodes N_{v_1} and N_{v_2} are computed from (p_N,$g_{N_{ij}}$) and ($p_{N_{ij}}$,g_N) in order to test both possibilities: transit from N to N_{v_1} and transfer from N_{v_1} to N_{ij}; or transfer from N to N_{v_2} and transit from N_{v_2} to N_{ij}. In case of success, the linked components are merged.

3.2 Algorithm parameters

Several parameters control the roadmap construction. We have already mentioned MAX_NTRY. Its value controls the end of the process. We consider that the configuration-space has been sufficently explored when the number of tries to expand the manipulation graph reaches this quantity.

Other important parameters are related to the choice of the expansion strategy. As mentioned above, we consider reasonable to make this choice at random, but dependent of the evolution of the graph construction process. Additionally, different types of probabilistic distributions and their parameters are available.

Although it is not reflected in the figures of this section, some parts of the algorithm can be iterated. For instance, it may be interesting to re-try to generate a valid configuration of the robot to achieve a grasp of the movable object on a valid placement when the first tried grasp is not reachable. The number of iterations of such processes are also parameters that the user can choose.

3.3 Solution Path

A manipulation planning query MP consists in finding the path between two configurations $q_i, q_f \in CS_{free}$ representing initial and final situations of $\mathcal{A}$ and $\mathcal{M}$. The problem is solved when these configurations can be linked to the same connected component of MG.

The final solution path is obtained after a refining process. $CG \cap CP$ paths are transformed in a finite sequence of transfer/transit paths by a simple procedure that iteratively splits the path into such kind of motions. In a further step, the solution is optimized by eliminating unnecessary intermediate operations (re-graspings).

4 Results

In this section we show results obtained from the implementation of the algorithm within the software platform *Move3D* [14]. Computing times correspond to experiments on a 330MHz Sparc Ultra 10 workstation.

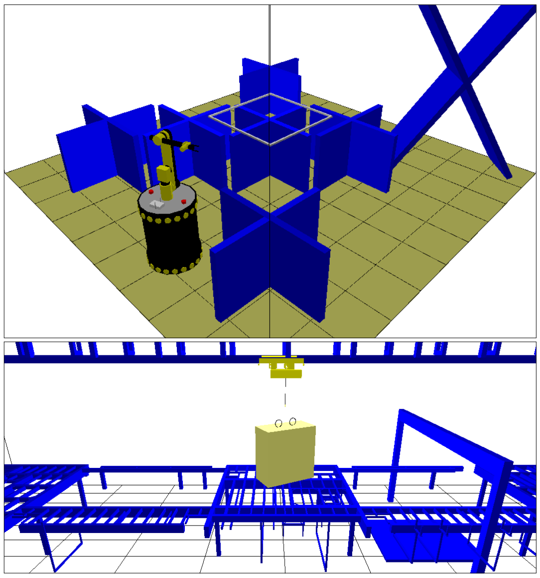

Figure 8: *Scenes of two examples*

Several environments have been used as test-bed of the planner. In this paper we present three of them. The first manipulation example has already been illustrated in Figure 1. We refer to it as example *Cage*. The two other environments are shown in Figure 8. The top image illustrates another robotics manipulation example (that we call *MobM*). In this case the manipulator arm is on a mobile platform (an holonomic robot). The last example (*RoBr*) corresponds to an industrial logistics problem treated in the framework of the project MOLOG [16]. The manipulation device is a rolling bridge.

The difficulty involving the Cage example is the complexity of the manipulation task. A high number of re-graspings are necessary in the solution path. The planner automatically computes the required configurations from a continuous region of placement (the floor) and a grasping zone all along the bar. The path to get the bar out of the cage is found in the $CG \cap CP$ manifold. Within the refining process this paths is decomposed in transit and transfer paths.

Manipulation tasks in the two other examples are simpler. Only a re-grasping is needed to solve them. In the example MobM, the mobile manipulator is able to pass from the one half to the other of the environment through the passage under the big vertical obstacle. However this passage is too narrow for the movable object (the square frame). A continuous grasping set is defined all around this object. The frame can be placed on the central obstacles. In the example RoBr, transporting the load between the extremes of the manufacturing line involves finding a placement under the arc in such a way that the rolling bridge can reach one of the rings at the top of the load from each side of the obstacle. Figure 9 shows the sequence of motions of the solution paths. The difficulty represented by the example MobM is dealing with a redundant system. An infinite set of solutions exist to achieve the same grasp. Redundancy is a challenge when treating closed chain mechanisms. The exploration of the $CG \cap CP$ manifold for such systems is efficiently performed by using the approach in [5]. In contrast, the rolling bridge in example RoBr is a simple system (4 d.o.f.). How-

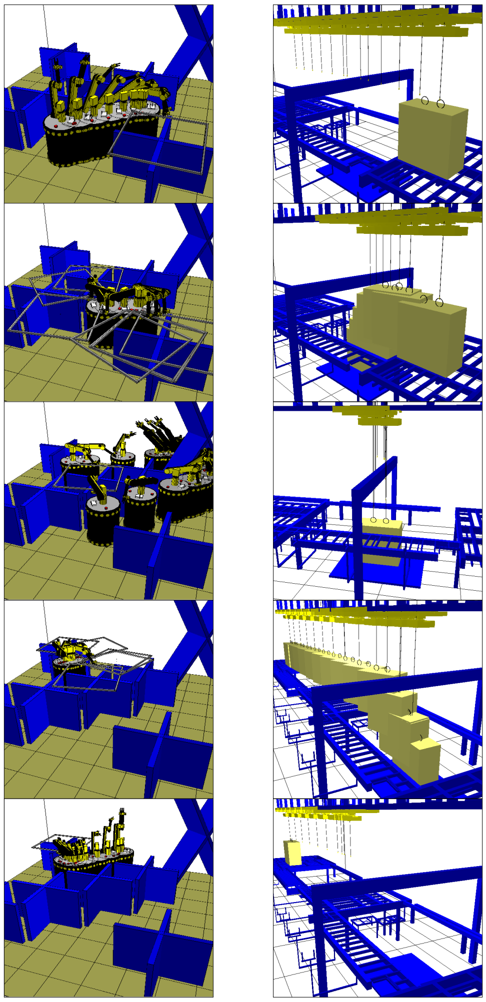

Figure 9: *Sequences of solution sub-paths*

ever, the big size of the load in relation to the handling device and the environment makes the computation of transfer paths more difficult.

	Cage	MobM	RoBr
Total Time	96 s	293 s	146 s
computing $CG \cap CP$	23 s	6 s	1 s
transit and transfer	70 s	284 s	143 s
N. nodes	32	45	10
N. manip. paths	12	14	6

Figure 10: *Numerical results*

The table in Figure 10 shows numerical results of the performance of the algorithm solving the three examples. Computing time is highly conditioned by the cost of testing transit and transfer paths. The lower presence of obstacles in the environment of the Cage example makes this amount smaller than in the other examples. The use of visibility notions is the reason for the small size of the manipulation roadmaps. Computing such small roadmaps reduces the number of connections to be tested. The number of nodes in the table corresponds to the number of configurations in $CG \cap CP$ that are kept in the graph. These configurations are connected by $CG \cap CP$ paths or manipulations paths (i.e. transit (re-grasping) paths and transfer/transit paths). The number of manipulation paths in each one of the roadmaps appears at the bottom of the table.

5 Conclusions

The algorithm described in this paper requires less user intervention when defining the manipulation planning problems than existing planners. The approach allows continuous grasps and placements and automatically decomposes the manipulation task. Results that we have shown demonstrate its efficacy to solve complex manipulation problems. Our objective for the close future is to continue improving this new technique. Rapidly identifying placements corresponding to required regrasping operations will considerably reduce computing cost. We are investigating a procedure that analyzes information provided by collision detection while exploring the $CG \cap CP$ set in order to find such specific placements. We are also studying the possibility to combine a symbolic task planning level with our planner, in order to handle several robots and objects in the manipulation planning problem [6].

Acknowledgments

This work is supported by the European Esprit Project 28226 MOLOG.

References

[1] J.M. Ahuactzin, K.K. Gupta and E. Mazer. Manipulation planning for redundant robots: a practical approach. In *Practical Motion Planning in Robotics*, K.K. Gupta and A.P. Del Pobil (Eds), J. Wiley, 1998.

[2] R. Alami, T. Siméon and J.P. Laumond. A geometrical Approach to planning Manipulation Tasks. The case of discrete placements and grasps." In *International Symposium on Robotics Research*, 1989.

[3] R. Alami, J.P. Laumond and T. Siméon. Two manipulation planning algorithms. In *Algorithmic Foundations of Robotics (WAFR94)*, K. Goldberg et al (Eds), AK Peters, 1995.

[4] J. Barraquand and P. Ferbach. A penalty function method for constrained motion planning. In *IEEE Int. Conference on Robotics and Automation*, 1994.

[5] J. Cortés, T. Siméon and J.P. Laumond. A Random Loop Generator for planning the motions of closed kinematic chains with PRM methods. In *IEEE Int. Conference on Robotics and Automation*, 2002.

[6] F. Gravot, R. Alami and T. Siméon. Playing with several roadmaps to solve manipulation problems. Submitted to *IEEE/RSJ Int. Conference on Intelligent Robotics and Systems*, 2002.

[7] L. Kavraki and J.C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE Int. Conference on Robotics and Automation*, 1994.

[8] Y. Koga and J.C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conference on Robotics and Automation*, 1994.

[9] J. Kuffner and S. Lavalle. RRT-Connect: an efficient approach to single-query path planning. In *IEEE Int. Conference on Robotics and Automation.*, 2000.

[10] J.C. Latombe. *Robot Motion Planning*, Kluwer, 1991.

[11] Ch. Nielsen and L. Kavraki. A two-level fuzzy PRM for manipulation planning. In *IEEE Int. Conference on Intelligent Robots and systems*, 2000.

[12] M. Overmars and P. Švestka. A Probabilistic learning approach to motion planning. In *Algorithmic Foundations of Robotics (WAFR94)*, K. Goldberg et al (Eds), AK Peters, 1995.

[13] T. Siméon, J. Cortés, A. Sahbani and J.P. Laumond. A Manipulation Planner for Pick and Place Operations under Continuous Grasps and Placements. In *IEEE International Conference on Robotics and Automation*, 2002.

[14] T. Siméon, J.P. Laumond and F. Lamiraux. Move3D: a generic platform for path planning. In *4th International Symposium on Assembly and Task Planning*, 2001.

[15] T. Siméon, J.P. Laumond and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. In *Advanced Robotics Journal* 14(6), 2000.

[16] T. Siméon, JP. Laumond, C. van Geem and J. Cortés. Computer Aided Motion: Move3D within MOLOG. In *IEEE Int. Conference on Robotics and Automation*, 2001.