Playing with several roadmaps to solve manipulation problems

Fabien Gravot, Rachid Alami, Thierry Simeon

Playing with Several Roadmaps to Solve Manipulation Problems

Fabien Gravot, Rachid Alami, Thierry Siméon

LAAS-CNRS, 7, Avenue du Colonel Roche,
31077 Toulouse CEDEX 04, France.

Abstract

We propose in this paper a resolution scheme that is aimed to be relevant for a large class of manipulation planning problems. This endeavor complements our efforts in developing manipulation planning algorithms [2, 14, 13]. Indeed, we are convinced that a higher level of problems complexity, and particularly those involving multiple robots and multiple objects, will be accessible thanks to the introduction of a symbolic reasoning level.

The resolution scheme relies on Probabilistic Roadmap Methods (PRMs) and on a reasoning level that adaptatively controls the construction and extension of a number of roadmaps.

We consider this symbolic level as a step towards a systematic approach to integrate task planning and geometric planning in better conditions than trough a gross, and somewhat, artificial hierarchical decomposition.

This paper describes the main ingredients of the proposed framework, and its first results.

1 Introduction

The Problem. We address the manipulation planning problem[1]. We have developed a resolution scheme that is able to build plans for a wide range of manipulation problems and that is aimed to be the basis for future work on multi-robot manipulation. The current implementation is already able to deal with manipulation problems involving one robot (holonomic or not) with one movable object and continuous grasps and placements[14, 13]. The presence of objects that can only be moved by the robot leads to complex instances of the motion planning problem. Indeed, the robot may be forced because of kinematic constraints or because of collision avoidance to find intermediate positions and re-grasping steps for the movable object.

Motion planning [10] is a challenging problem that involves dealing with elaborate physical constraints and high-dimensional configuration spaces. The fastest existing deterministic planner has a complexity exponential in the number of degrees of freedom of the robot. On the other hand, a class of randomized planners, in particular, *Probabilistic Roadmap Methods* (PRMs) [8, 12], have been used successfully in high-dimensional configuration spaces. More recently, the use of several roadmaps to solve motion planning with kinematic constraints [6, 11] or manipulation tasks [14, 13], have been proposed. In our approach, we develop a resolution scheme based on a number of methods for reasoning on roadmaps construction and handling.

The Key Ideas. The planner will be based on an "adaptative" search on various roadmaps. Each roadmap type is built with specific random and local path functions.

Planning generally involves several distinct elementary tasks. In the manipulation context, performing an object *transfer* or a *transit* motion between two grasps, are two distinct task types that must be treated separately and combined in a precise way[1, 14, 13]. To deal with such types of tasks, we propose to associate a roadmap to each one.

When dealing with closed kinematic chains, it has been shown that it could be useful to deal separately with different operations. For instance, closing a kinematic chain and moving in the environment can be done in two steps [6, 11]. Therefore, we also allocate a specific roadmap to each operation.

The last type of roadmaps that we propose to define are purely *heuristic* ones. They will be used to perform search in relaxed instances of problems. For instance, one may decide to search for a certain type of path that may be invalid, but that will serve as a good basis for further refinement.

The planner is guided by a set of heuristics rules that will allow it to choose which roadmap should be extended and how to do so. These rules exploit the specificities of the different roadmaps.

The roadmaps are not necessarily built to catch directly the topology of the configuration space of all environment parameters but only of subsets of it. The key idea is to work on several roadmaps of kinematics subset in order to be able to deal with their composition without explicitly computing it.

In order to limit the size of the search space, the search process will tend to give priority to the most constrained roadmaps or to sub-manifolds involving a small number of mobile objects.

To find a valid solution, we use a multi-roadmap solver that can handle several roadmaps to ensure the validity of all of the environment configurations.

2 Configurations Definition

A roadmap are specific to one robot, or more precisely to a sub-set of elementary kinematic chains (noted $\mathcal{EKC}$), associated with configuration definition.

Classically, a configuration is a set of parameters which allows to specify the state of all objects in the environment. With kinematic constraints[11, 6, 4] the number of configuration parameters may be greater than needed. Indeed, through the kinematic constraints, several degrees of freedom must be recomputed. Likewise, we have chosen, in our application, to not to have a minimal representation.

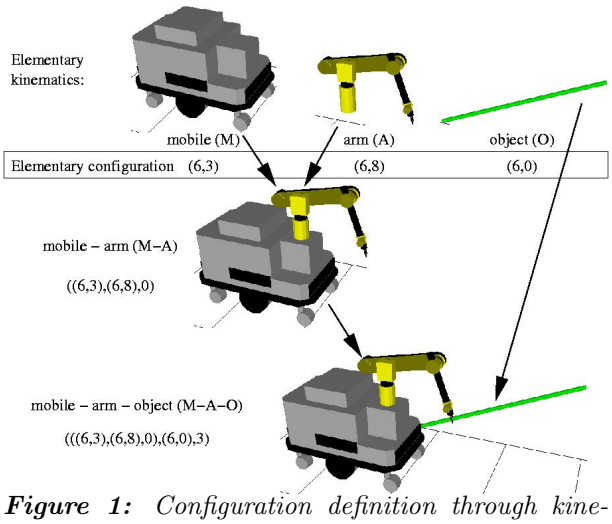

Figure 1: *Configuration definition through kinematic chains*

Let us consider a manipulation environment with a mobile manipulator and a movable object (besides fixed obstacles). As shown in Figure 1, we consider a set of $\mathcal{EKC}$ that define the state of all objects. In our representation we split a configuration into a placement part (6 degrees of freedom) and an actuator part (degrees of freedom controlled by robots). For example the arm configuration has 14 degrees of freedom: 6 for the placement part, 6 for a classical arm configuration and 2 for the gripper.

Given a set of $\mathcal{EKC}$, one can combine them in order to describe more complex robots. For instance a mobile arm (M-A) can be modeled with a mobile base (M), an arm (A) and a fixed attachment between them. The model will then be completed by a configuration which groups the two elementary configuration sets and no other degree of freedom because of the fixed attachment: $((6,3),(6,8),0)$. This allows to take advantage of specific methods associated to each $\mathcal{EKC}$ in order, for instance to combine their local-path methods. We have defined a set of methods that allow this combination, like using Reed and Sheep local-path for the mobile base and straight-lines in the C-space for the arm.

In the same way, we define a chain (M-A-O) where the robot holds the object. The resulting configuration includes all the $\mathcal{EKC}$ configurations and 3 additional degrees of freedom for the continuous set of grasps: $(((6,3),(6,8),0),(6,0)3)$. For a parallel jaw gripper, this attachment depends on 3 parameters: two translations parallel to the grasped face and one rotation around the axis perpendicular to the faces[14, 13]. Using such combinations, the arm forward kinematics can be used in order to generate a transfer configuration (the robot holds the object), and its inverse kinematics to obtain a grasp configuration (the robot grasps the object on a specific placement).

In the next section we will show how these combinations of $\mathcal{EKC}$ will be used to build roadmaps for manipulation problems.

3 Roadmap Definition

As already mentioned, our approach is based on the construction of a number of roadmaps. Each roadmap is associated with a task and is used according to it. It corresponds to one of the previously defined combinations of $\mathcal{EKC}$. In our representation, the link between a roadmap and its semantics is done through 4 attributes:

Roadmap type: see §3.1.
Edge method: defines the local-path used to connect two nodes in the roadmap
Roadmap links: see §3.2.
Random sampling method: see §4.1.

This attributes allow to extend the roadmaps definitions used in the manipulation problems, or in kinematics roadmap [6, 11]. They have been devised not only to solve the class of problems presented here, but also to model and solve multiple robots, multiple objects manipulation problems.

3.1 Roadmap Type

A type is associated to each roadmap in order to describe how to use it. Up to now, we have defined four roadmap types: `Cross`, `Refine`, `Heuristic`, `Relative`.

The first two types are used to extract the solution.
`Cross`: defines a roadmap where all the nodes and edges are valid subset of configurations and movements. The solution plan must be expressed with paths from such roadmaps.

`Refine`: defines a roadmap where all the nodes are valid, but where the edges do not represent a valid motion but satisfy the *reduction property* [2]. This property allows to systematically transform an edge of this graph into a finite number of valid movement operations. Therefore, a solution plan to a manipulation problem can include those elementary paths. The final plan will be obtained after the refinement of such paths into valid action sequences.

The last two roadmap types are used for the sake of increasing the speed of the planner. While a solution cannot include edges borrowed from such graphs, they can provide paths that can guide the search.

`Heuristic`: defines a roadmap where several constraints has been relaxed. They will generally be domain specific.

`Relative`: defines a roadmap where the $\mathcal{EKC}$ configuration is expressed relatively to a joint. Such a roadmap does not take into account the static environment. It allows the computation of complex local configurations and motions. This processing can be useful when the kinematic chain becomes complex and/or when the auto-collision checking or the kinematic computations are time consuming [6, 11].

Table 1: *Roadmap Definition*

Kin.	Roadmap	Edge method	Type
O	Placement Po	none	Cross
O	Movement Mo	linear	Heuristic
M-A	Movement Ti (Transit)	local-path holonomic or not	Cross
M-A-O	Transfer Ta	linear composition fixed attachment	Cross
M-A-O	Grasp $\cap$ Placement GP	linear composition with constraints	Refine
M-A-O	Grasp G	”	Relative
M-A-O	Attachment A	none	Heuristic

In our example, we have defined the set of roadmaps presented in Table 1. The objet alone has two roadmaps. The first $\mathcal{R}(Po)$ represents the valid configurations of the objet when it is not grasped (stable positions globally named Placement). In this case, no movement is allowed, so the roadmap contains only unlinked nodes. In the second roadmap, the configuration space is the same but movements are allowed, resulting in links between nodes. As the object cannot move by itself, this graph is a heuristic one. It is aimed to catch the topology of the valid stable object positions. This will be useful when it will be necessary to find intermediate objects positions (see §4.3). The roadmaps $\mathcal{R}(Ti)$ and $\mathcal{R}(Ta)$ represent respectively the “standard” transit and transfer paths.

As already mentioned, $\mathcal{R}(Ti)$ does not involve all the $\mathcal{EKC}$; hence, any valid path extracted from it should be tested to be free of any collision with the object. The $Graspb \cap Placement$ roadmap $\mathcal{R}(GP)$ satisfies (by construction) the reduction property[2]. So, any path included in this graph can be transformed into a finite set of alternate valid transit and transfer paths. It is the most constrained roadmap with a closed kinematic chain induced by the fact that any configuration corresponds to a stable object placement as well as a grasp/ungrasp robot configuration[14, 13]. Note that $\mathcal{R}(G)$ can be used to store valid closed kinematic chain relative configurations of the robot and the object, and to reuse them in $\mathcal{R}(GP)$.

Another heuristic roadmap, $\mathcal{R}(A)$, is intended to store the gripper/object grasp transforms used. The planner will try to re-use as much as possible the same grasp transforms. This is done in order to simplify the search for new transfer motions.

After this description of the various roadmaps that can be used, we will now see the links between them and how they allow to search for a plan through a set of roadmaps.

3.2 Roadmap links

In order to be able to build a plan as a result of a search through several roadmaps, the planner will incrementally build, explore and switch between them. We define two types of links between those roadmaps:

Inheritance : This link connects a constrained roadmap $\mathcal{R}_1$ to a less constrained roadmap $\mathcal{R}_2$. All valid configurations in $\mathcal{R}_1$ are valid in $\mathcal{R}_2$.

Compose : Such a link connects one or more roadmaps to a more complex roadmap to make new configurations that are a composition of the nodes of these roadmaps. These links are based on the inheritance links.

Inheritance. As said before, the inheritance links connects a constrained roadmap $\mathcal{R}_1$ to a less constrained roadmap $\mathcal{R}_2$. This link also defines a filter used to derive a a valid sub-node in $\mathcal{R}_2$ from a node in $\mathcal{R}_1$. Each time a node is added to $\mathcal{R}_1$, it is linked to its sub-node in $\mathcal{R}_2$.

There are two conditions for the existence of such a link. First, it cannot connect a `Relative` roadmap to a not `Relative` one. Second, the $\mathcal{EKC}$ of $\mathcal{R}_2$ must be included in $\mathcal{R}_1$. When $\mathcal{R}_2$ is `Relative`, a change of reference frame is applied.

For example (Figure 2), $\mathcal{R}(GP)$ is the most constrained roadmap, so all its nodes are linked to the other roadmaps. Whenever a node N is created in $\mathcal{R}(GP)$, a corresponding node is added to $\mathcal{R}(Ta)$. Such a node may serve as beginning or end of a transfer path. Likewise, a node is created in $\mathcal{R}(Ti)$. Besides, a closed kinematic chain and a relative grasp

transform are extracted from N and respectively stored in $\mathcal{R}(G)$ and $\mathcal{R}(A)$.

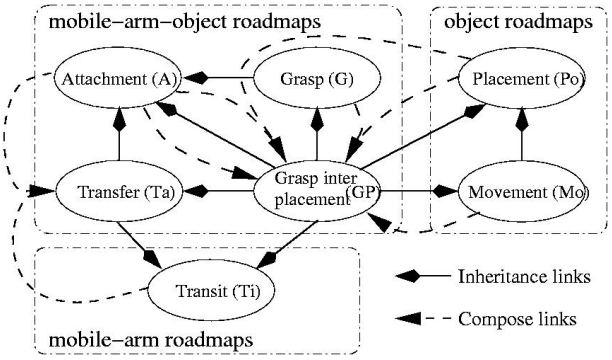

Figure 2: *Roadmap links*

Compose Links. These links define how to make a node in a complex roadmap $\mathcal{R}_c$ from nodes belonging to other roadmaps $\mathcal{R}_1 \ldots \mathcal{R}_n$ that inherit from $\mathcal{R}_c$. The node composition is based on the inverse functions of the original inheritance link.

4 Roadmap Construction

The last roadmap attribute to consider is the random sampling method. Now we study the not so classical random sampling method.

4.1 Random Sampling

For random sampling we mainly use two methods (and their combination): the Random Loop Generator (RLG)[4] and the creation of nodes based on compose links between roadmaps.

The RLG allows to choose a new node with short generation time. In fact the random space is reduced to only allow configurations with the possibility to be updated to satisfy a set of kinematic constraints.

The other possibility is the use of composed links. A set of nodes in sub-roadmaps are chosen at random (or not) to make a new node N_c. In fact this link does not necessary define completely a configuration of N_c. So, there is often a combination of node composition and RLG.

Let us recall that there is a third way to obtain nodes in a roadmap through link inheritance.

Each roadmap can have several ways to make a new node. We use a set of heuristic rules to determine what are the nodes needed, and in which roadmap the search should be performed.

4.2 Heuristic Rules

Recent progresses in task planning [3, 5, 7] have been based on the development of new heuristics. The manipulation problem is both a geometrical and a task problem. So it seems natural to borrow techniques from these domains.

The heuristic rules are, for us, the first level of symbolic control of the purely geometrical PRMs search methods.

Rules Precondition. Rules preconditions enable to compute the probability to select a rule. While in task planning the best rule is always selected, with PRMs, if we want to keep the property of probabilistic completion, we must allow a non-null probability to all applicable rules.

The preconditions are based on roadmaps analysis. Several parameters can be used: the roadmaps nodes number, the number of connected components, the current mean computation time,...

For instance, it is interesting to increase the size of $\mathcal{R}(GP)$ and of $\mathcal{R}(Ti)$ at the beginning to avoid to link not useful nodes of $\mathcal{R}(GP)$ with transit or transfer paths. Indeed, it is better to invest until we have "sufficiently" captured the topology of $\mathcal{R}(GP)$.

The preconditions can also be based on a more detailed analysis of the roadmaps. For example:

$$\begin{array}{c} \exists N_1 \in \mathcal{C}(GP)_1, N_2 \in \mathcal{C}(GP)_2, N_o \in \mathcal{R}(Po) \\ \exists N_s \in \mathcal{R}(Ti), N_e \in \mathcal{R}(Ti)/ \\ N_o \subset N_1 \textit{ and } N_o \subset N_2 \textit{ and} \\ N_s \subset N_1 \textit{ and } N_e \subset N_2 \end{array} \quad (1)$$

This precondition is used in a rule that tries to link two connected components $\mathcal{C}(GP)_1$ and $\mathcal{C}(GP)_2$ of $\mathcal{R}(GP)$ with a transit path. The precondition tests if a common object position exists in the two components. The right part of the rule is discussed in the next subsection.

4.3 Rules Specification

We have two levels of rules. The high level allows to select a rules subset. For instance, the following rule selects the set of rules that require to perform an extension of GP roadmap.

$$Increase\ \mathcal{R}(GP) \quad (2)$$

The low level rules specify (1) the roadmap that must be increased (2) how it will extended (by choosing a new node randomly or by trying to link two nodes with a specific-search) and (3) the associated method.

$$\begin{array}{l} In\ \mathcal{R}(GP),\ random(),\ with\ RLG \\ In\ \mathcal{R}(GP),\ random(),\ with\ Compose(Po, G) \end{array}$$

The first rule draws a new node in $\mathcal{R}(GP)$ with the Random Loop Generator method [4]. The second uses the compose link (Figure 2) from the Grasp roadmap and the Object position roadmap to build a new random node, The interest of such a rule is that all the closed kinematic and the auto-collision computations have been already done; it is sufficient to test only collision with the static environment to validate the node.

Rules can also specify more specific actions. For example, if two nodes N_s and N_e satisfy the precondition 1, then we can try to link them:

$$In\ \mathcal{R}(Ti),\ specific\_search(N_s, N_e),\ with\ RRT$$

5 Problem Solving

The heuristic rules explain how to build the roadmaps, and now we must explain how to search for a solution in the set of roadmaps.

5.1 Kinematic-Component

We define a topological structure across the different roadmaps that will serve as a search space for the planner. We call them the kinematic-components (noted $\mathcal{KC}()$).

A kinematic-component does not link roadmap nodes but connected components of two different roadmaps or of other kinematic-components.

They are defined on `Cross` and `Refine` roadmaps and the edges are derived from roadmaps inheritance links. Besides, each kinematic-component is defined on a subset of elementary kinematic chains ($\mathcal{EKC}$).

In our example, we have three types of kinematic-components: the robot kinematic-component ($\mathcal{KC}(M\text{-}A)$) that stores $\mathcal{C}(Ti)$ components, the object kinematic-component ($\mathcal{KC}(O)$) that stores $\mathcal{C}(Po)$ components, the grasp kinematic-component ($\mathcal{KC}(M\text{-}A\text{-}O)$) that stores $\mathcal{C}(Ta)$, $\mathcal{C}(GP)$ and the pairs of kinematic-components ($\mathcal{KC}(M\text{-}A)$, $\mathcal{KC}(O)$) previously defined.

This hierarchical structure will be mainly used in multiple robots problems.

To solve the problem, the planner must find a solution in a kinematic-component in which all the $\mathcal{EKC}$ are defined. In our case we need to link the kinematic-components $\mathcal{KC}(M\text{-}A\text{-}O)$ of the initial and goal configurations.

5.2 Validity of edges

To find a solution, the planner must incrementally increase the roadmaps, under the control of the heuristic rules until it is able to link the kinematic-components of the initial and the goal configurations. But even if a path is found, it may be not valid. The edges of some roadmaps have not yet be fully tested.

With the mono-robot, mono-object manipulation problem, the solution is found into the robot crossable roadmaps ($\mathcal{R}(Ta)$, $(\mathcal{R}(Ti), \mathcal{R}(Po))$, $\mathcal{R}(GP)$). All these roadmaps are valid but not the robot movement roadmap $\mathcal{R}(Ti)$. The roadmap has been built without the object, so the robot may collide the object during its movement. The selected trajectories must be checked with the current object position (given by $\mathcal{C}(Po)$ coupled with $\mathcal{C}(Ti)$). Like [14, 13], we use RRT[9] to find a valid path based on non-valid trajectory. If no valid path is found, we invalidate the link between those kinematic-components and search for another path in the set of components. This can lead to re-grasping operations.

With multi-robot or with multi-object manipulation, finding a path through valid edges could be much more challenging. To solve this problem we foresee the use of another level of symbolic methods much closer to task planning.

6 First results

The manipulation planner has been implemented within the software platform Move3D[1] currently developed at LAAS-CNRS. Even if the preliminary version of our implementation is not yet fully functional, the first result (Figure 3) shows the roadmaps construction for a simple manipulation task, and its solution with a mean time of 1 minute with a SunBlade 100. In this example we have a non holonomic robot with an arm which can grasp a bar under continuous grasps and placements.

The problem is to change the position of the object. The planner re-uses as much as possible other roadmaps (Figure 3) . The Grasp∩Placement roadmap has nodes composed on the initial and goal position of the object. In the same way the transfer roadmap is very close to the transit one.

7 Conclusion

We have presented a new approach to manipulation planning that deals with a set of roadmaps and introduces new heuristics with more symbolic specifications. This approach has been tested on mono manipulation problems but is intended to deal with a much wider class of problems, even the multi-robot and multi-object manipulation.

The first results are encouraging, but many improvement could be done in order to have a fully functional planner. The most important part will be the realization of the multi-robot solver that could have good performances for searching valid edges through the set of roadmap and possibly take into account other non-purely manipulation tasks . This new solver must be a link between PRMs and tasks planning.

Acknowledgments

We are very grateful to Anis Sahbani and Juan Cortès who have contributed to many ideas in this paper and developed several algorithms that are used in this planner.

[1] See `http://www.laas.fr/ nic` and for a commercial version of this prototype distributed by the spin-off *Kineo*, see `http://www.kineocam.com`

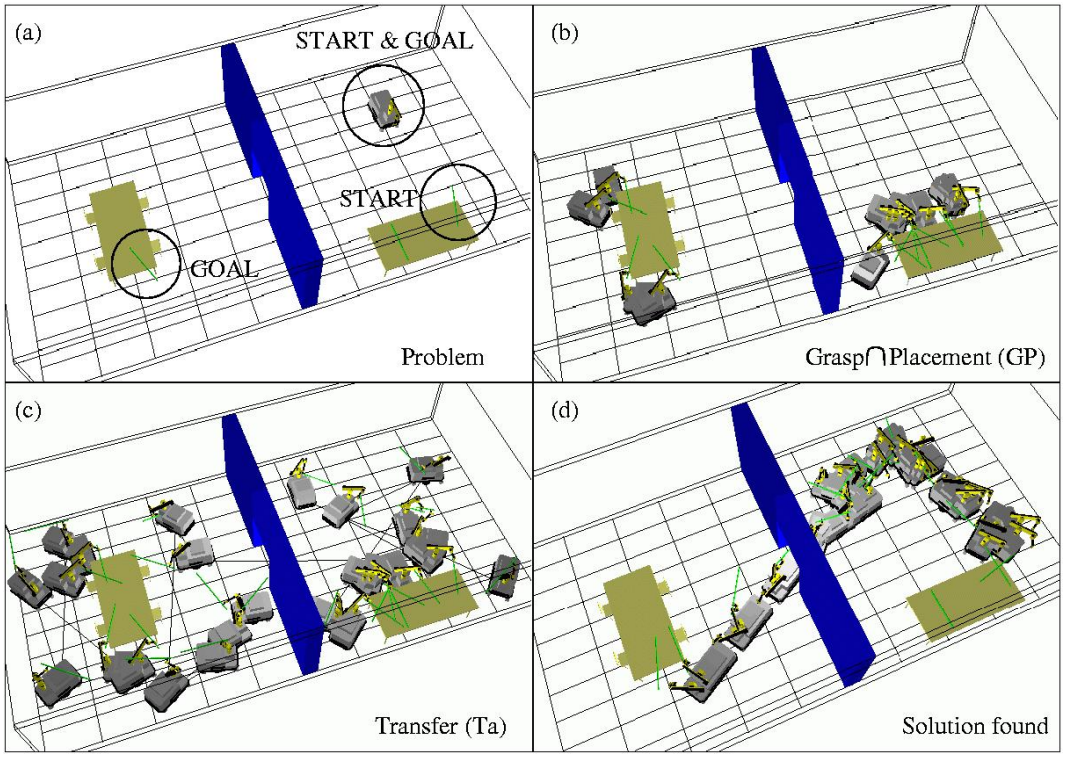

Figure 3: *Example of a manipulation problem with a non-holonomic mobile maniptlator: (a) the problem consists in tranferring the bar from one table to another; (b) and (c) represent two different roadmaps that have been incrementally built during the search (d) illustrates one solution*

References

[1] 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*, Tokyo, August 1989

[2] 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.

[3] F. Bacchus and F. Kabanza: "Using Temporal Logic to Control Search in a Forward Chaining Planner". In *New Directions in Planning*, M. Ghallab and A. Milani, (Eds.) IOS Press, 1996, pp 141-153

[4] J. Cortès, T. Siméon and J.P. Laumond: "A Random Loop Generator for Planning the motions of closed kinematic chains with PRM methods". In *IEEE International Conference on Robotics & Automation*, Washington D.C., May 2002

[5] K. Erol: "HTN Planning: Formalization, Analysis, and Implementation". PhD. Dissertation, Computer Science Dept., University of Maryland, 1995

[6] Li Han and N.M. Amato: "A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems". In *Proceedings of the Workshop on Algorithmic Fundations of Robotic (WAFR '00)*, March 2000, pp. 233-246

[7] J. Hoffmann and B. Nebel: "The FF Planning System: Fast Plan Generation Through Heuristic Search". In *Journal of Artificial Intelligence Research*, 2001, Vol. 14, pp. 253-302

[8] L. Kavraki and J.C. Latombe: "Ramdomized preprocessing of configuration space for fast path planning". In *IEEE International Conference on Robotics & Automation*, San Diego, May 1994

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

[10] J.C. Latombe: "Robot Motion Planning". Kluwer Academic Publishers, Boston, MA, 1991

[11] S.M. La Valle, J.H. Yakey and L.E. Kavraki: "A Probabilistic Roadmap Approach for Systems with Closed Kinematic Chains". In *IEEE International Conference on Robotics & Automation*, Detroit, Michigan, May 1999

[12] C. Nissoux, T. Siméon and J.P. Laumond: "Visibility based probabilistic roadmaps". In *IEEE International Conference on Intelligent Robots & Systems*, Korea, October, 1999

[13] A. Sahbani, J. Cortès and T. Siméon: "A Probabilistic Algorithm for Manipulation Planning under Continuous Grasps and Placements". Submitted to *IEEE International Conference on Robotics & Automation*, 2002

[14] T. Siméon, J. Cortès, A. Sahbani and J.P. Laumond: "A Manipulation Planner for Pick and Place Operations under Continious Grasps and Placements". In *IEEE International Conference on Robotics & Automation*, Washington D.C., May 2002