Manipulation Planning with Probabilistic Roadmaps

Thierry Siméon, Jean-Paul Laumond, Juan Cortés, Anis Sahbani
LAAS-CNRS
Toulouse, France

Abstract

This paper deals with motion planning for robots manipulating movable objects among obstacles. We propose a general manipulation planning approach capable to address continuous sets for modeling both the possible grasps and the stable placements of the movable object, rather than discrete sets generally assumed by the previous approaches. The proposed algorithm relies on a topological property that characterizes the existence of solutions in the subspace of configurations where the robot grasps the object placed at a stable position. It allows us to devise a manipulation planner that captures in a probabilistic roadmap the connectivity of sub-dimensional manifolds of the composite configuration space. Experiments conducted with the planner in simulated environments demonstrate its efficacy to solve complex manipulation problems.

1 Introduction

Manipulation planning concerns the automatic generation of the sequence of robot motions allowing to manipulate *movable objects* among obstacles. The presence of movable objects, i.e. objects that can only move when grasped by a robot, leads to a more general and computationally complex version of the classical motion planning problem [15]. Indeed, the robot has the ability to modify the structure of its configuration space depending on how the movable object is grasped and where it is released in the environment. Also, movable objects can not move by themselves; either they are transported by robots or they must rest at some stable placement. Motion planning in this context appears as a constrained instance of the coordinated motion planning problem. The solution of a manipulation planning problem (see e.g. [2, 15]) consists in a sequence of sub-paths satisfying these motion restrictions. 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*.

Consider the manipulation planning example illustrated by Figure 1. The manipulator arm has to get a movable object (the bar) out of the cage, and to place it on the other side of the environment. Solving this problem requires to automatically produce the sequence of transfer/transit paths separated by grasps/ungrasps operations, allowing to get one extremity of the bar out of the cage; the manipulator can then re-grasp the object by the extremity that was made accessible by the previous motions, perform a transfer path to extract the bar from the cage, and finally reach the specified goal position. In particular, the motion shown onto the second image illustrating the solution requires itself four re-grasping operations to obtain a sufficient sliding motion of the bar. This example shows

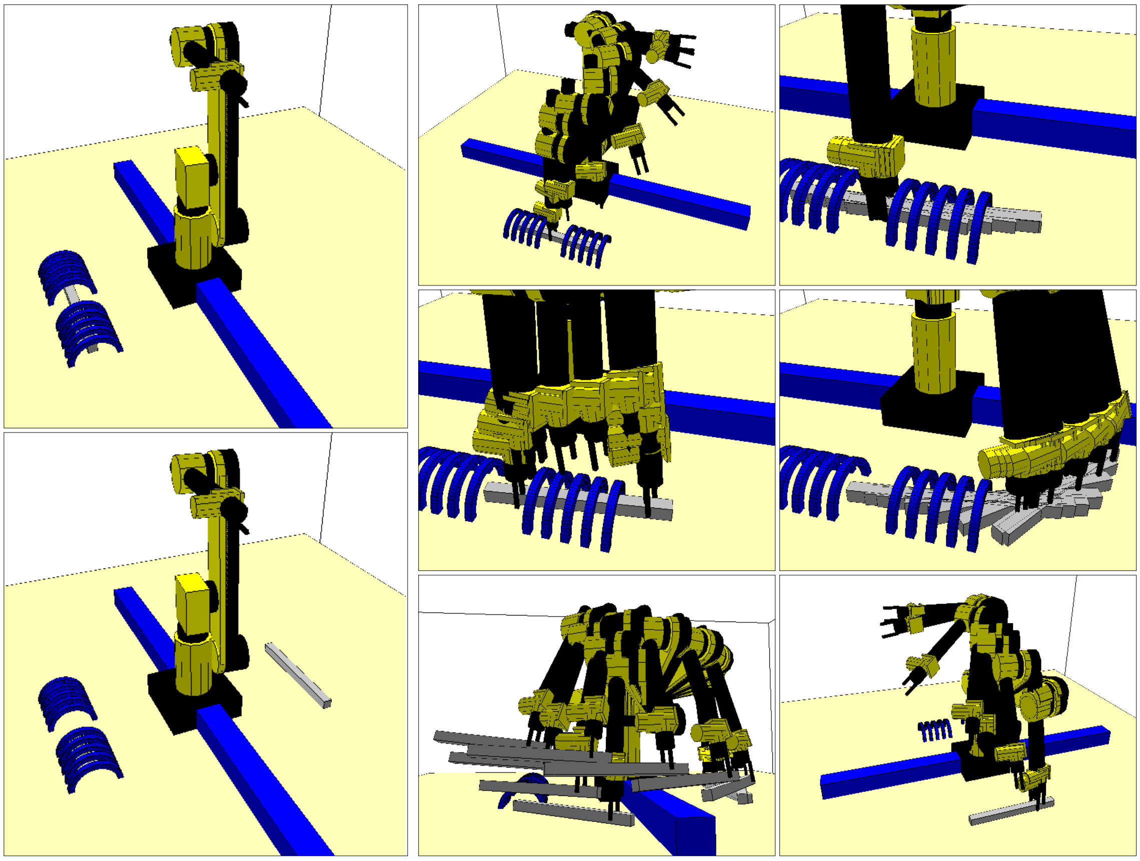


Figure 1: *How to manipulate the bar from its initial position (top,left) to the goal (bottom,left)? The solution (right) requires several pick and place operations*

that a manipulation task possibly leads to a complex sequence of motions including several re-grasping operations. A challenging aspect of manipulation planning is to consider the automatic task decomposition into such elementary collisions-free motions.

Most of existing algorithms (e.g. [1, 2, 5, 13, 21]) assume that finite sets of stable placements and possible grasps of the movable object are given in the definition of the problem. Consequently, a part of the task decomposition is thus resolved by the user since the initial knowledge provided with these finite sets has to contain the grasps and the intermediate placements required to solve the problem. Referring back to the example, getting the bar out of the cage would require a large number of grasps and placements to be given as input data.

In this paper, we describe a general approach based onto recent results presented in [25, 26]. The main contribution is to deal with a continuous setting of the manipulation problem while covering the scope of the previous proposed approaches (Section 2). It allows us to devise a manipulation planner that automatically generates among continuous sets the grasps and the intermediate placements required to solve complicated manipulation problems like the one illustrated onto Figure 1. The approach relies on a topological property first established in [3] and recalled in Section 3. This property allows us to reduce the problem by characterizing the existence of a solution in the lower dimensional subspace of configurations where the robot *grasps* the movable object *placed* at a stable position. Section 4 describes the proposed approach and shows how the connected components of this subspace can be captured in a probabilistic roadmap computed for a virtual closed-chain system. Section 5 details the planning techniques developed to implement the approach. Using the Visibility-PRM algorithm [23] extended to deal with such closed systems [8], we first capture the connectivity of the search space into a small roadmap composed of a low number of connected components (Section 5.1). Connections between these components using transit or transfer motions are then computed by solving a limited number of point-to-point path planning problems (Section 5.2). The details of an implemented planner interleaving both stages in an efficient way are described in Section 5.3. Finally, Section 6 presents some experiments and comments on the performance of the planner.

2 Related Work

One of the challenging issues of manipulation planning is to integrate the additional difficulty of planning the grasping and re-grasping operations to the path planning problem. This interdependency between path planning and grasp planning was first touched upon by work done in the 80's for the development of automatic robot programming systems. In particular, the Handey system [19] integrated both planning levels and was capable to plan simple Pick and Place operations including some re-grasping capabilities. The geometric formulation of manipulation planning [2, 15], seen as an instance of motion planning problem extended by the presence of movable objects, provided a unified framework allowing to better tackle the interdependency issues between both planning levels.

Motion planning in presence of movable objects is first addressed as such in [27]. In this work, an exact cell decomposition algorithm is proposed for the particular case of a polygonal robot and of one movable object translating in a polygonal workspace, assuming a finite grasp set of the movable object.

The manipulation graph concept is introduced in [2] for the case of one robot and several movable objects manipulated with discrete grasps and placements. In this case, the nodes of the manipulation graph correspond to discrete configurations and the edges are constructed by searching for transfer (or transit) paths between nodes sharing the same grasp (or placement) of the movable object(s). Following this general framework, the approach was implemented for a translating polygon [2] and a 3 dof planar manipulator [16]. An exact cell decomposition algorithm is also proposed in [3] for the specific case of a translating polygonal robot capable to manipulate one movable polygon with an infinite set of grasps.

The manipulation planning framework is extended in [12, 13] to multi-arm manipulation where several robots cooperate to carry a single movable object amidst obstacles. In this work, the number of legal grasps of the objects is finite and the movable object has to be held at least by one robot at any time during a re-grasp operation. The planner proposed in [13] first plans the motions of the movable object using an adapted version of a randomized potential field planner [4], and then finds the sequence of re-grasp operations of the arms to move the object along the computed path. This planner relies on several simplifications, but it can deal with complex and realistic problems.

Another heuristic planning approach proposed in [5] is to iteratively deform a coordinated path first generated in the composite configuration space using a variational dynamic programming technique that progressively enforces the manipulation constraints.

Variants of the manipulation planning problem have been investigated. In [20], grasping is replaced by pushing and the space of stable pushing directions imposes a set of nonholonomic constraints that introduce some controllability issues to the problem. The heuristic algorithm described in [7] considers a problem where all the obstacles can be moved by a circular robot in order to find its way to the goal.

Two other contributions extend recent planning techniques to manipulation planning. In [1], the *Ariane's Clew* algorithm [6] is applied to a redundant robot manipulating a single object in a 3D workspace. The method assumes discrete grasps of the movable object; it is however capable to deal in realistic situations with redundant manipulators for which each grasp possibly corresponds to an infinite number of robot configurations. Finally, [21] proposes a practical manipulation planner based onto the extension of the PRM framework [11, 22]. The planner constructs a manipulation graph between discrete configurations; connections are computed using a *Fuzzy PRM* planner that builds a roadmap with edges annotated by a probability of collision-freeness. Computing such roadmaps improves the efficiency of the planner for solving the possibly high number of path planning queries (in changing environments) required to compute the connections.

Contribution: The manipulation planning techniques above only address the discrete instance of the problem. Only the algorithm in [3] considers the case of infinite grasps, but it is however limited to the simple case of a translating robot in a polygonal world. The planning approach proposed in the paper possibly deals with continuous placements and grasps. It is general and practical for realistic problems in 3-dimensional environments. For example, one can describe the set of stable placements by constraining the movable object to be placed on top of some horizontal faces of the static obstacles. Such placement constraints define a 3-dimensional sub-manifold of the object's configuration space (two translations in the horizontal plane and one rotation around the vertical axis). Also, one can consider sets of continuous grasping domains such that the jaws of a parallel gripper have a contact with two given faces of $\mathcal{M}$. Such grasp constraints also define a 3-dimensional domain (two translations parallel to the grasped faces and one rotation around the axis perpendicular to the faces).

3 Manipulation planning

Notations: Let us consider a 3-dimensional workspace with a robot $\mathcal{R}$ and a movable object $\mathcal{M}$ moving among static obstacles. The robot has n degrees of freedom and $\mathcal{M}$ is a rigid object with 6 degrees of freedom that can only move when it is grasped by the robot. Let CS_{rob} and CS_{obj} be the configuration spaces of the robot and the object, respectively. The composite configuration space of the system is $CS = CS_{rob} \times CS_{obj}$ and we call CS_{free} the subset in CS of all admissible configurations, i.e. configurations where the moving bodies do not intersect together or with the static obstacles. The domain in CS corresponding to valid placements of $\mathcal{M}$ (ie. stable placements where the object can rest when ungrasped by the robot) is denoted by CP. The domain in CS corresponding to valid grasps configurations of $\mathcal{M}$ by the robot $\mathcal{R}$ is denoted by CG. Both CP and CG are sub-dimensional manifolds in CS.

Manipulation Constraints: A solution to a manipulation planning problem corresponds to a constrained path in CS_{free}. Such a solution path is an alternate sequence of two types of sub-paths verifying the specific constraints of the manipulation problem, and separated by grasp/ungrasp operations:

- *Transit Paths* where the robot moves alone while the object $\mathcal{M}$ stays stationary in a stable position. The configuration parameters of $\mathcal{M}$ remain constant along a transit path. Such motions allow to place the robot at a configuration where it can grasp the object. They are also involved when changing the grasp of the object. Transit paths lie in CP. However, a path in CP is not generally a transit path since such path has to belong to the sub-manifold corresponding to a fixed placement of $\mathcal{M}$. Transit paths induce a foliation[1] of CP (Figure 2a).

- *Transfer Paths* where the robot moves while holding $\mathcal{M}$ with the same grasp. Along a transfer path, the configuration of $\mathcal{M}$ changes according to the grasp mapping induced by the forward kinematics of the robot: $q_{obj} = \mathcal{G}(q_{rob})$. Transfer paths lie in CG. They induce a foliation of CG (Figure 2b).

Problem: Consider the two sets of constraints defining the stable placements and feasible grasps. A manipulation planning problem is to find a manipulation path (i.e. an alternate sequence of transit and transfer paths) connecting two given configurations q_i and q_f in $CG \cup CP$ (Figure 2c). Manipulation planning then consists in searching for transit and transfer paths in a collection of sub-manifolds corresponding to particular grasps or stable placements of the movable object. Note that the intersection $CG \cap CP$ between the sub-manifolds[2] defines the places where transit paths and transfer paths should be connected. The manipulation planning problem appears as a constrained path planning problem inside and between the various connected components of $CG \cap CP$ (Figure 2d).

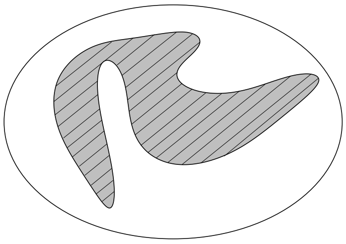
Reduction Property: Two foliation structures are defined in $CG \cap CP$: the first one is induced by the transit paths; the second one is induced by the transfer paths. As a consequence, any path lying in a connected component of $CG \cap CP$ can be transformed into a finite sequence of transit and transfer paths (the proof of this property[3] appears in [3]). Therefore two configurations which are in a same connected component of $CG \cap CP$ can be connected by a manipulation path.

[1] A *foliation* of a n-dimensional manifold $\mathtt{M}$ is an indexed family $\mathtt{L}_\alpha$ of arc-wise connected m-dimensional sub-manifolds $(m < n)$, called *leaves* of $\mathtt{M}$, such that:

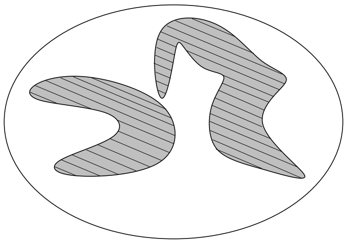
- $\mathtt{L}_\alpha \cap \mathtt{L}_{\alpha'} = \emptyset$ if $\alpha \neq \alpha'$
- $\cup_\alpha \mathtt{L}_\alpha = \mathtt{M}$
- every point in $\mathtt{M}$ has a local coordinate system such that $n - m$ coordinates are constant.

[2] The intersection $CG \cap CP$ is also a sub-manifold. Note however that $G \cup P$ is not a sub-manifold.

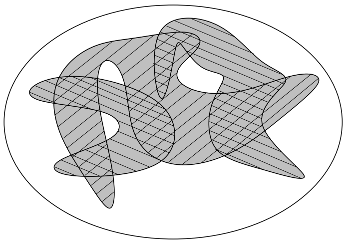
[3] Note that this property holds for a single movable object under the hypothesis that the robot does not touch the static obstacles.



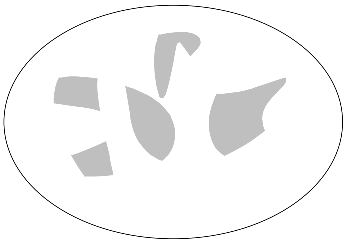
a) *The placement space CP*



b) *The grasp space CG*



c) $CG \cup CP$



d) *$CG \cap CP$ has five connected components*

Figure 2: *Moving along transit (resp. transit) paths induces a foliation of the placement (resp. grasp) space. Both foliations intersect themselves in $CG \cap CP$.*

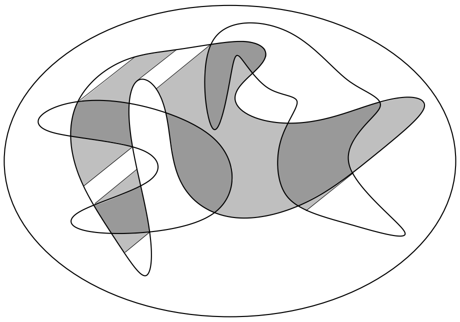
It is then sufficient to study the connectivity of the various components of $CG \cap CP$ by transit and transfer paths. Let us consider a transit (or transfer) path whose endpoints belong to two distinct connected components $(CG \cap CP)_i$ and $(CG \cap CP)_j$ of $CG \cap CP$. From the reduction property above one may deduce that any configuration in $(CG \cap CP)_i$ can be connected to any configuration in $(CG \cap CP)_j$ along a manipulation path.

Manipulation Graph: It is then possible to build a graph MG whose nodes are the various connected components of $CG \cap CP$ while an edge between two nodes $(CG \cap CP)_i$ and $(CG \cap CP)_j$ indicates the existence of a transit (or transfer) path whose endpoints belong respectively to $(CG \cap CP)_i$ and $(CG \cap CP)_j$. Figure 3 illustrates the graph structure for the example introduced in Figure 2. Examples of manipulation paths are shown in the bottom left picture: q_1 is not a valid configuration for the manipulation problem (it does not belong to $CP \cup CG$). Configuration q_6 is in CG; nevertheless it cannot escape from its leaf in CG. A manipulation path exists between q_3 and q_5 and between q_2 and q_4. No manipulation path exists between q_5 and q_4.

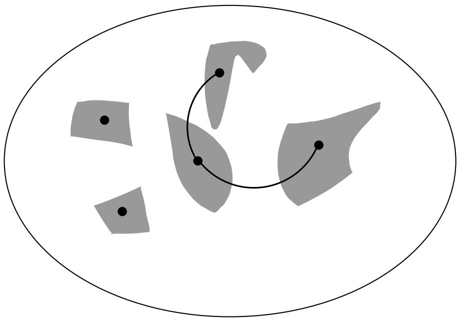
Let q_i and q_f be two configurations in $CG \cup CP$. There exists a manipulation path between q_i and q_f *iff* there exist two nodes $(CG \cap CP)_i$ and $(CG \cap CP)_f$ in MG such as:

- there exists a transit (or transfer) path from q_i to some point in $(CG \cap CP)_i$,
- there exists a transit (or transfer) path from some point in $(CG \cap CP)_f$ to q_f,
- $(CG \cap CP)_i$ and $(CG \cap CP)_f$ belong to a same connected component of the graph MG.

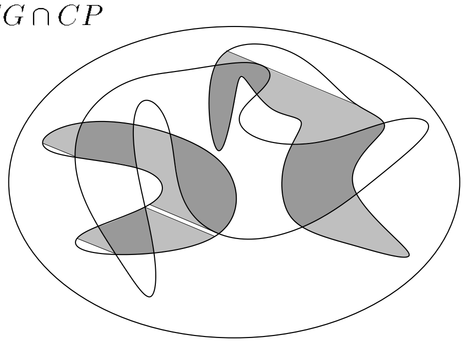
MG is said to be a *manipulation graph*.



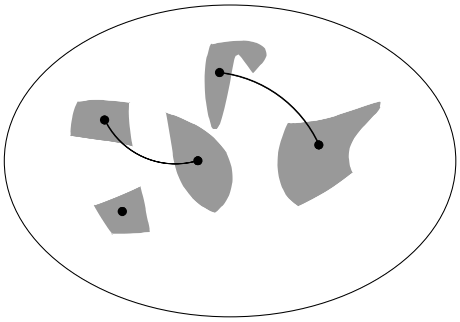
a) *Set of configurations reachable by a transit path starting at a configuration in* $CG \cap CP$



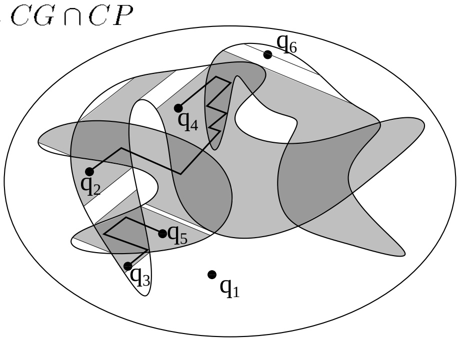
b) *Adjacency of* $CG \cap CP$ *components via transit paths*



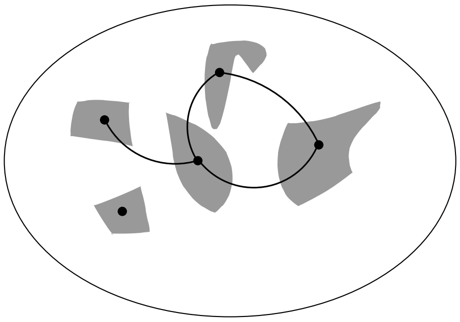
c) *Set of configurations reachable by a transfer path starting at a configuration in* $CG \cap CP$



d) *Adjacency of* $CG \cap CP$ *components via transfer paths*



e) *Examples of manipulation paths*



f) *The Manipulation Graph*

Figure 3: *The topology of* CS *induced by the manipulation problem constraints can be captured by a so-called Manipulation Graph.*

Combinatorial issues: How to capture the various connected components of $CG \cap CP$? How to capture their adjacency by transit and transfer paths? These are the two key issues in manipulation task planning. All the techniques overviewed above fall in this general framework.

4 A General Approach to Manipulation Planning

We now describe our approach for solving manipulation problems in the general setting of continuous grasp and placement constraints. The proposed approach relies onto the structuration of $CG \cap CP$ discussed in the previous section. The main idea is to exploit the reduction property of Section 3 to decompose the construction of the manipulation graph at two levels:

- compute the connected components of $CG \cap CP$.
- determine the connectivity of $CG \cap CP$ components using transit and transfer paths.

A Two-level Probabilistic Manipulation Roadmap: The manipulation graph is computed as in [21] using a probabilistic technique [11, 22], but our construction of the manipulation roadmap integrates a specific step allowing us to directly capture the connectivity of the sub-manifold $CG \cap CP$ inside the roadmap. The structure of a manipulation roadmap computed using this approach is illustrated by Figure 4.

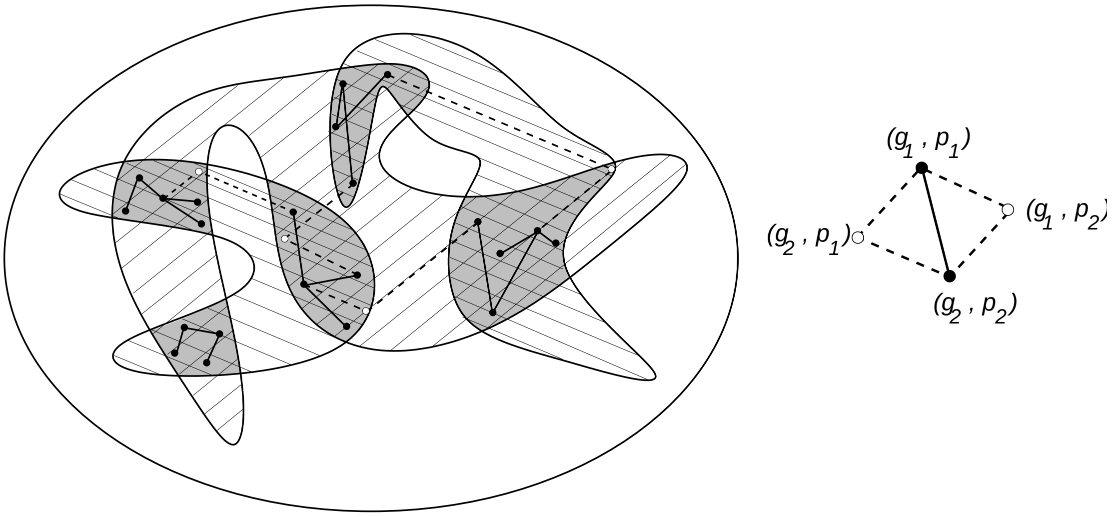


Figure 4: *A probabilistic roadmap as a manipulation graph: nodes belong to $CG \cap CP$ while edges model paths belonging to either $CG \cap CP$, CP or CG. Two types of adjacency are considered: direct $CG \cap CP$ paths (plain segments) or elementary sequences of* Transit-Transfer *(or* Transfer-Transit*) paths (dashed segments).*

The roadmap is composed into a small number of nodes (the connected components of $CG \cap CP$) connected together with transit or transfer paths. Each $CG \cap CP$ component is captured into a sub-roadmap computed using a local planner that generates feasible $CG \cap CP$ motions (the black edges in Figure 4) between nodes (in black) randomly sampled in $CG \cap CP$.

These sub-roadmaps are connected via transit and transfer paths (the dotted edges) using some intermediate nodes (in white). The intermediate nodes are defined as follows. Consider two configurations in $CG \cap CP$ that can not be directly connected by a collision-free path in $CG \cap CP$ (i.e. configurations that do not belong to the same connected component of $CG \cap CP$). These configurations correspond to fixed grasps and placements of the movable object, noted $(g_i, p_i)_{i=1,2}$. Using motions outside $CG \cap CP$, they can only be connected by following the particular leaves of CP and CG issued from both configurations. We then define the intermediate nodes as (g_1, p_2) and (g_2, p_1). An edge between (g_1, p_1) and (g_2, p_2) is added if at least one of the intermediate nodes (g_1, p_2) and (g_2, p_1) belongs to $CG \cap CP$ and is reachable from (g_1, p_1) and (g_2, p_2) by a collision-free transit/transfer path. The connection between two randomly sampled configurations of $CG \cap CP$ is then possible if one of the three types of adjacency (Figure 4) exists:

- `Type1`: a direct path from (g_1, p_1) to (g_2, p_2) lying inside $CG \cap CP$ is collision-free.
- `Type2a`: a transfer path from (g_1, p_1) to (g_1, p_2) followed by a transit path from (g_1, p_2) to (g_2, p_2) are both collision-free.
- `Type2b`: a transit path from (g_1, p_1) to (g_2, p_1) followed by a transfer path from (g_2, p_1) to (g_2, p_2) are both collision-free.

Once the manipulation roadmap is computed, queries are solved by searching for a path inside MG. Note that the obtained solution possibly alternates transfer/transit paths (when traversing edges of MG using `Type2` adjacencies) with other elementary paths computed in $CG \cap CP$ (`Type1` adjacencies inside nodes of MG) which do not correspond to feasible manipulation paths. However, thanks to the reduction property, any such `Type1` paths can be transformed in a post-processing stage into a finite sequence of `Type2` transit and transfer paths.

Capturing $CG \cap CP$ Topology via Closed-Chain Systems The main critical issue of the approach is to capture into a probabilistic roadmap the topology of $CG \cap CP$ which is a sub-manifold of the global configuration space CS with a lower dimension. The idea here is to explore $CG \cap CP$ as such. For this, we consider that $CG \cap CP$ is the configuration space of a single system consisting of the robot together with the movable object placed at a stable position. Maintaining the stable placement while the object is grasped by the robot induces a closed chain for the global system (Figure 5).

We now explain how the closed chain used for the exploration of $CG \cap CP$ is defined. A fixed grasp of the movable object corresponds to a transformation matrix T_g positioning the end-frame of the robot to the coordinate frame of the object. The set of continuous grasps can then be defined by a transform matrix $T_g(q_{grasp})$ where q_{grasp} denotes a set of varying parameters. The CG subspace corresponds to the set of free configurations (q_{rob}, q_{obj}) for which the configuration q_{obj} of $\mathcal{M}$ changes according to the grasp mapping induced by the forward kinematics of the robot and by the grasp of the object: $q_{obj} = \mathcal{G}(q_{rob}, q_{grasp})$. CG is therefore parameterized by the configuration vector (q_{rob}, q_{grasp}) associated to a composite robot obtained by adding a new joint q_{grasp} between the last link of $\mathcal{R}$ and the object $\mathcal{M}$.

On the other hand, the set of stable placements is defined by a transformation matrix $T_p(q_{place})$ relating the object's frame to the world frame, where q_{place} denotes the set of varying placements parameters. The CP sub-manifold corresponds to configurations where q_{obj} changes according to the mapping $q_{obj} = \mathcal{P}(q_{place})$. The $CG \cap CP$ space can then be parameterized as the set of configurations $(q_{rob}, q_{grasp}, q_{place})$ satisfying the closure constraints $\mathcal{G}(q_{rob}, q_{grasp}) = \mathcal{P}(q_{place})$.

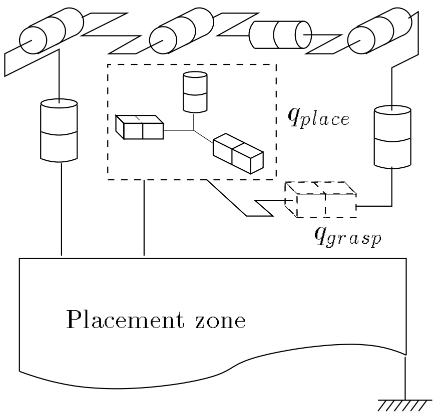


Figure 5: *Closed chain system formed in $CG \cap CP$ by the robot and the movable object*

Facing such sub-dimensional manifolds is a challenging problem for motion planning. In particular, applying a purely randomized PRM framework [11, 22] to closed chain mechanisms is prohibited by the fact that the probability to choose a configuration at random on a given sub-dimensional manifold is null [18]. However, several recent contributions [18, 10, 8] extended the PRM framework to face this issue. Section 5 describes the planning technique used in our implementation.

Connections with transit and transfer paths Computing such connections requires to solve multiple point-to-point path planning problems, as for the case of discrete grasps and placements. Here, the issue is to provide efficient solutions for searching such collision-free transit (or transfer) paths in the various leaves of CP (or CG). For example, the fuzzy roadmap technique [21] could be used to gain efficiency by limiting the number of collision tests performed when solving the queries. Our implemented planner uses however another kind of speed-up. It relies onto a simple technique sharing a similar idea with the kinematic roadmaps [10]. It exploits the fact that each planning problem has to be performed in a *partially* modified environment to re-use a precomputed *static* roadmap that is *dynamically updated* when solving the planning queries. This planning technique is also further explained in the section below.

5 The Planning Techniques

We now detail the planning techniques developed to implement the approach. The two basic primitives required for computing the `Type1` and `Type2` motions are respectively described in subsections 5.1 and 5.2. Then, we explain how both primitives are combined by the algorithm used to build the manipulation roadmap.

5.1 Closed-chain planner for `Type1` motions

As explained above, our approach requires to apply planning techniques for closed chain systems in order to capture the topology of $CG \cap CP$. Several recent contributions extended the PRM framework to deal with closure constraints [18, 10, 8]. In particular, we use the algorithm [8] that demonstrated good performance onto complex 3D closed chains involving more than twenty degrees of freedom. As initially proposed in [10] the loop is broken into two open sub-chains, called the active and passive sub-chains. Random closure configurations (i.e. valid nodes) are computed by the *Random Loop Generator* (*RLG*) algorithm. *RLG* combines random sampling techniques with simple geometrical operations that compute approximated reachable workspaces of various sub-chains to iteratively generate the configuration for the active chain. Then, it performs inverse kinematics for the remaining passive part of the loop in order to force the closure constraint. The advantage of the *RLG* algorithm is that it produces random samples for the active chain that have a high probability to be reachable by the passive part. This significantly decreases the cost of computing and connecting closure configurations. Roadmap's edges are computed using a local planner limited to act onto the active joints, while the passive part of the loop follows the motion of the rest of the chain. The practical efficacy of our approach results from the good performance reached today by these closed-chain extensions of the *PRM* framework.

The $CG \cap CP$ roadmap is then computed using *Visibility-PRM* [23, 17]. This technique keeps the roadmap as small as possible by only adding two types of useful samples: guards that correspond to samples not already "seen" by the current roadmap, and connectors allowing to merge several connected components. Its interest is first to control the quality of the roadmap in term of coverage and second, to capture the connectivity of possibly complex spaces into a small data structure. We believe that the small size of the visibility roadmaps, combined with the proposed structuring of $CG \cap CP$ contributes to the overall efficiency of our approach by limiting the number of costly path-planning queries to be performed during the second stage, when searching the connections with collision-free transfer or transit paths.

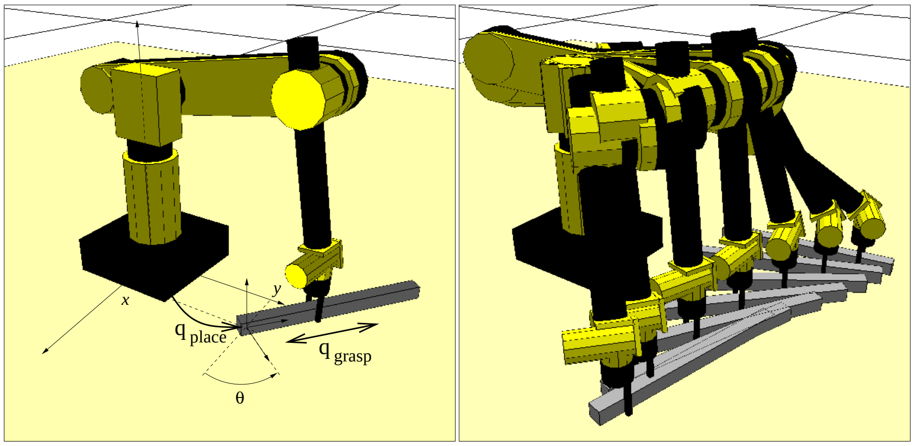


Figure 6: *Virtual closed-chain system and a feasible $CG \cap CP$ motion (the bar moves on the floor while sliding into the gripper's jaws)*

Figure 6 shows for the manipulation example of Figure 1, the closed chain system formed by the 6 *dof* arm manipulating the long bar. The bar moves in contact with the floor while sliding within the gripper. The sliding motion of the gripper results from the additional degree of freedom q_{grasp} introduced in the system to characterize the infinite set of grasps. In this example q_{grasp} is chosen to allow a translation of the parallel jaw gripper along the bar. Similarly, the set of stable placements corresponds to the planar motions parameterized by a 3-dimensional vector q_{place} (two horizontal translations and a vertical rotation), that maintain the contact of the bar with the floor. The motion shown in the right image of Figure 6 is a feasible motion in $CG \cap CP$. It is not admissible from the manipulation problem point of view. However, thanks to the reduction property it can be transformed into a finite sequence of feasible transit and transfer paths.

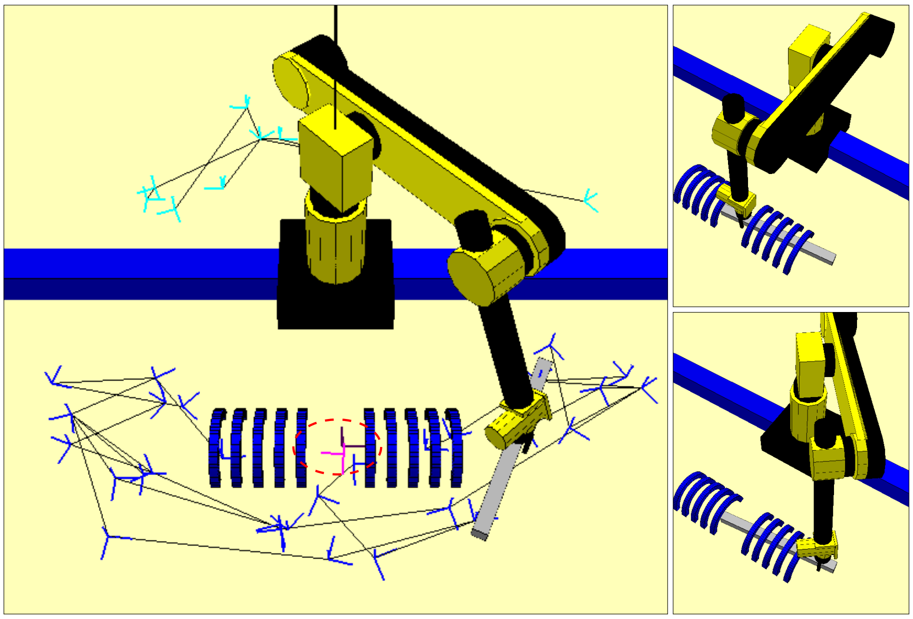


Figure 7: *A Visibility Roadmap computed in $CG \cap CP$ (left) and two placements of the system inside two different connected components of $CG \cap CP$ (right)*

Figure 7 shows the visibility roadmap computed in $CG \cap CP$ by the algorithm for the introduction example shown in Figure 1. While the collision-free configuration space of the arm alone is connected, $CG \cap CP$ is not. The computed roadmap has four connected components: two main components separated by the long static obstacle, and two other small components that correspond to placements of the movable object inside the cage obstacle while it is grasped by the arm through the open passage in the middle of the cage. These two small components (inside the dashed circle of the left image) correspond to similar positions of the bar, rotated or not by 180 degrees. One associated placement of the system is shown onto the top right image. The bottom right image corresponds to a node of the main component with the bar placed at the same position, but using a different grasp. Connecting this node to the small component is not possible because of the cage obstacle that limits the continuous change of grasp. Such re-grasping requires the computation of collision-free paths outside $CG \cap CP$ as explained below.

5.2 Connection planner for `Type2` motions

`Type2` connections require a basic routine to find elementary collision-free transit and transfer paths. Each of the planning problems corresponds to a particular grasp or placement of the movable object. Then, the queries have to be performed in a partially modified environment. The motivation of the two-stage method used by the connection planner is simply to amortize the cost of dealing with such partial changes by re-using at each query some of the paths precomputed during the first stage regardless of the movable object.

First, we compute a roadmap for the robot and the static obstacles, without considering the presence of the movable object. Then, before to solve a given (transit or transfer) path query, the roadmap is updated by checking whether each edge is collision-free with respect to the current position of the movable object. Colliding edges are labeled as blocked in the roadmap.

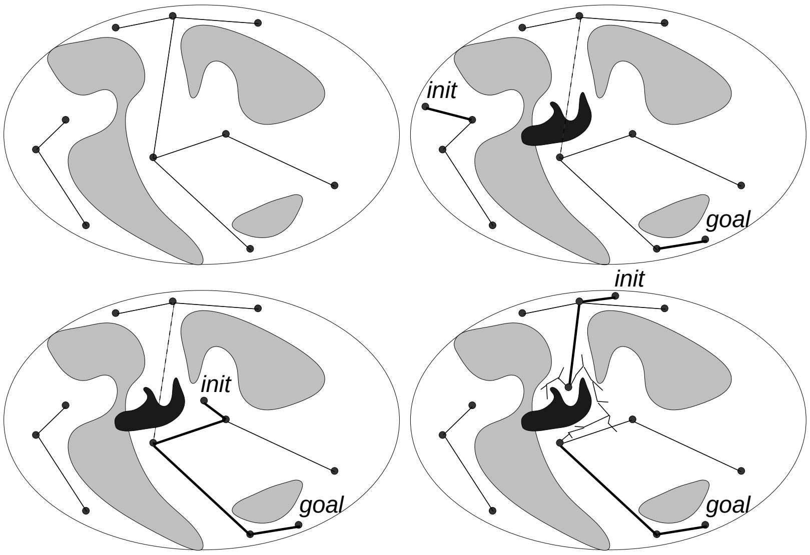


Figure 8: *A static roadmap is computed in the configuration space of the robot (top,left). During queries, it is* labeled *according to collisions with $\mathcal{M}$. If the query fails, there is no solution (top, right). Otherwise, either there exists a solution path in the roadmap avoiding* labeled *edges (bottom left) or not not (bottom,right). In the later case the colliding part of the path is locally updated using a* RTT *like technique.*

The search for a given path is then performed within the labeled roadmap. As illustrated by Figure 8, three cases possibly occur. When the search fails, this means that no path exists even in the absence of the movable object; the problem has no solution. Similarly, when the computed path does not contain any blocked edge (dashed edges in Figure 8) then a solution is found. Now let us consider the intermediate situation where the solution path necessarily contains blocked edges. In such case, the algorithm tries to solve the problem locally by using a Rapid Random Tree (*RRT*) algorithm [14] to connect the endpoints of the blocked edges. The principle of *RRT* consists in incrementally building two random trees rooted at the start and goal configurations, such that both trees explore the space around them and advance toward each other through the use of a simple heuristic. This algorithm was originally designed to efficiently process single-query path planning problems. The main interest of *RRT* is to perform well locally. Its complexity depends on the length of the solution path. This means that the approach quickly finds easy solutions. It may be viewed as a dynamic updating of the roadmaps.

Figure 9 shows the connecting paths computed by the planner for the example of Figure 7. The transfer path (left) is used to connect the two main components of $CG \cap CP$, while the transit path connects the small component (inside the dashed circle of Figure 7) to the main one.

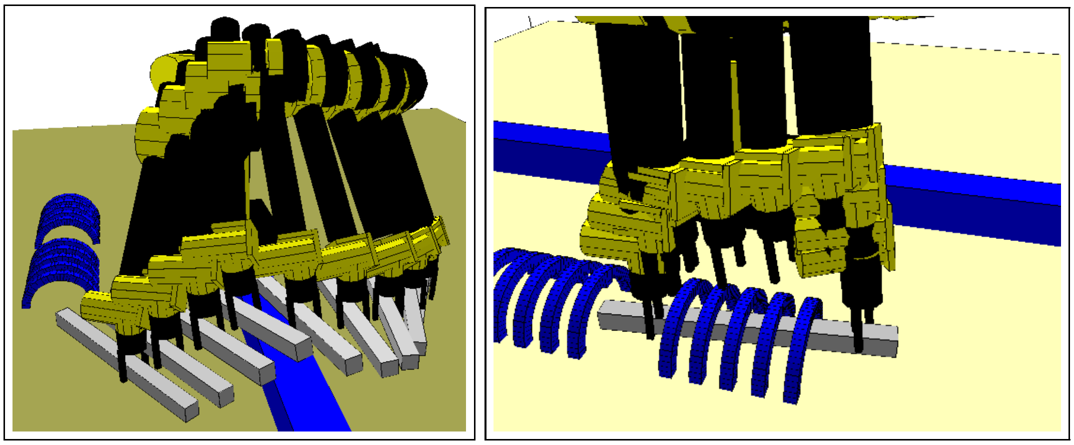


Figure 9: *The transfer path (left) and the transit path (right) computed for connecting $CG \cap CP$ components shown in Figure 7*

5.3 Manipulation planning algorithm

The algorithm incrementally constructs the manipulation roadmap MG by interleaving the two steps of the approach: computing $CG \cap CP$ connected components (`Type1` adjacency) and linking them (`Type2a-b` adjacencies). Following the principle of *Visibility-PRM*, the algorithm stops when it is not able to expand the graph after a given number of tries. This number of failures is related to an estimated coverage of the search space [23] (in our case, the $CG \cap CP$ space). The function EXPAND_GRAPH performs one expansion step of MG. Candidate nodes are first sampled in $CG \cap CP$ and the different types of connections to the graph are then tested.

EXPAND_MANIP_GRAPH(MG)

$q \leftarrow$ NEW_CONFIG(MG)
$Type \leftarrow$ ADJACENCY_CHOICE(MG)
$n_{linked\ comp.} \leftarrow$ TEST_CONNECTIONS(MG, q, $Type$)

if $n_{linked\ comp.} \neq 1$ **then**
  ADD_NODE(q, MG, $Type$)
  UPDATE_GRAPH(MG)
  return TRUE
else
  return FALSE

Node's generation: Our algorithm possibly considers several classes of continuous grasps (resp. placements), each defined by a transformation matrix $T_{g_i}(q_{grasp})$ (resp. $T_{p_j}(q_{place})$) with q_{grasp} (resp. q_{place}) as varying parameters. Therefore, each couple (T_{g_i}, T_{p_j}) induces a particular closed-chain system. A candidate node is generated as follows by the function NEW_CONFIG: it first randomly selects one couple (i, j) of grasps and placement classes. The grasp and the stable placement of the movable object is then chosen by randomly sampling the parameters of vectors q_{grasp} and q_{place} inside their variation interval. The candidate node N is generated when the sampled grasp and placement are collision-free and feasible for the virtual closed system induced by the couple (T_{g_i}, T_{p_j}).

Adjacency's selection: Following the discussion in Section 4, the desired behaviour of the roadmap builder is to start by constructing portions of the roadmap inside $CG \cap CP$ components using `Type1` adjacency, and then to determine connections of the components using `Type2` adjacencies. Rather than considering separately the two stages, the algorithm uses a more sophisticated way to interleave both phases. Function ADJACENCY_CHOICE performs a biased random choice {`Type1`,`Type2`} that depends on the evolution of the size of MG: the first expansion steps start with a low probability to return a `Type2` choice; when the roadmap grows, this probability increases as the percentage of the coverage cov estimated by the fraction $(1 - \frac{1}{ntry})$ (see [23] for details).

A tuning parameter $\alpha \in [0, 1[$ is used to put more or less weight between expanding the $CG \cap CP$ components and connecting them using transit/transfer paths: the probability of choosing the $CG \cap CP$ expansion is determined by $Prob(\texttt{Type1}) = \alpha.(1 - cov)$ and $Prob(\texttt{Type2}) = 1 - Prob(\texttt{Type1})$. With α set to zero, the roadmap builder only considers connections of MG's nodes with transit/transfer paths. When α tends toward 1, the algorithm rarely selects such `Type2` connections before a sufficient coverage of $CG \cap CP$ has been reached. The effect of α on the performance of the algorithm when solving the manipulation problem of Figure 1 is shown by Figure 10.

TEST_CONNECTIONS(MG, q, $Type$)
$n_{linked\ comp.} \leftarrow 0$
for $k = 1$ **to** N_COMP(MG) **do**
  if LINKED_TO_COMP(q, C_k, $Type$) **then**
    $n_{linked\ comp.} = n_{linked\ comp.} + 1$
return $n_{linked\ comp.}$

Edge's generation: The function TEST_CONNECTIONS checks the connection between the candidate node and each connected component C_k of MG using the type of adjacency selected by function ADJACENCY_CHOICE. When the expansion step is performed using `Type1` motions, connections are computed using the closed-chain planner of section 5.1. In this case, note that the connection of the candidate node to the roadmap is only possible with nodes computed for the same classes of grasps and placements (T_{g_i}, T_{p_j}). For each component C_k, nodes with such characteristics are tested until a connection is found feasible for the closed-chain mechanism induced by (T_{g_i}, T_{p_j}). When the expansion is performed using `Type2` motions, function TEST_CONNECTIONS stops checking the component C_k as soon as valid connection is found using the planning technique of section

5.2. Following the visibility principle, the candidate node is added to the graph only if the random sample q was linked to none or to more than one connected component. In the second case, the linked components are merged.

Solving Manipulation Queries: Once the manipulation roadmap is built, queries can be performed using the three following steps. First, the start and goal configurations are connected to MG using the TEST_CONNECTIONS function called with a `Type2` adjacency choice, and the manipulation graph is searched for a path between both configurations. The second step is necessary to transform $CG \cap CP$ portions of the solution path into a finite sequence of transfer/transit paths (see Figure 12). This is done by a simple procedure that iteratively splits the $CG \cap CP$ paths into pieces whose endpoints can be connected by a composition of two transit/transfer motions. Finally, the solution is smoothed by a procedure that eliminates unnecessary motions.

5.4 Performance Analysis

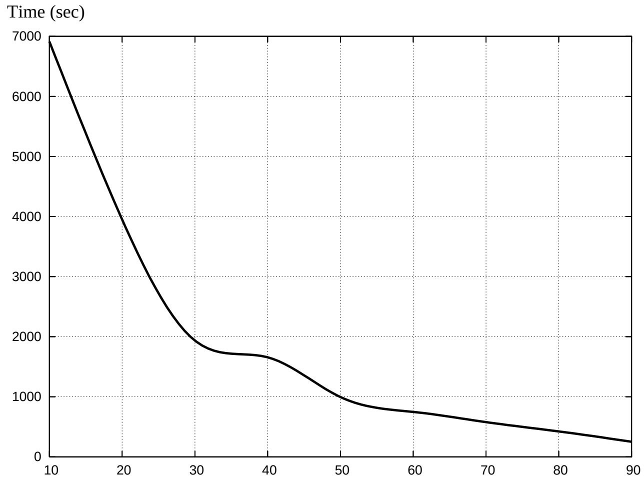


Figure 10: *Performance of the algorithm depending on the percentage of $CG \cap CP$ exploration (`Type1` paths) wrt. to Transit-Transfer (`Type2` paths) used to build the manipulation roadmap for the example of Figure 1. The abcissae corresponds to the values of the tuning parameter $\alpha(\times 100)$.*

Let us discuss the performance of the approach for the manipulation problem of Figure 1 used as illustrative example in this section. The curve displayed in Figure 10 plots the time[4] spent by the algorithm to build the manipulation roadmap according to the tuning parameter α. As explained above, the role of this parameter is to control the rate of connections searched inside $CG \cap CP$ (`Type1` adjacencies) with respect to connections searched outside $CG \cap CP$ along the leaves of the the CG and CP spaces (`Type2` adjacencies). When $\alpha = 0$ the roadmap builder only considers collision-free Transit-Transfer

[4]each time value was averaged over ten runs performed using different seeds to initialize the random generator

paths to connect the random samples generated in $CG \cap CP$. Increasing α allows us to privilege the construction of the $CG \cap CP$ connected components using `Type1` adjacencies before trying possible connections along leaves with `Type2 adjacencies`. As one can note onto the figure, the computation times significantly decrease for runs performed with higher values of α.

The performance of the approach can be explained by the fact that it avoids many searches of collision-free motions along the leaves of CP and CG thanks to the direct exploration of the $CG \cap CP$ sub-manifold. Also note that the $CG \cap CP$ sub-manifold explored in this case is a lower dimensional space compared to the leaves of the placements and grasps spaces. Let us illustrate this by detailing the dimension of the various spaces used in this example. Here, we have $dim(CS_{rob}) = 6$, $dim(CS_{obj}) = 6$ and $dim(CS) = 12$. Placements of the object are allowed only when the bar is placed on the table (3 $dofs$). For a fixed placement of the bar, the robot can freely move its 6 degrees of freedom. Then, the dimension of the placement space is $dim(CP) = 9$. The bar is grasped by the robot by allowing a (1 dof) translating motion along its length; we then have $dim(CG) = 7$. In this example, the leaves in both CP and CG have dimension 6 while $dim(G \cap P) = 4$.

6 Experimental Results

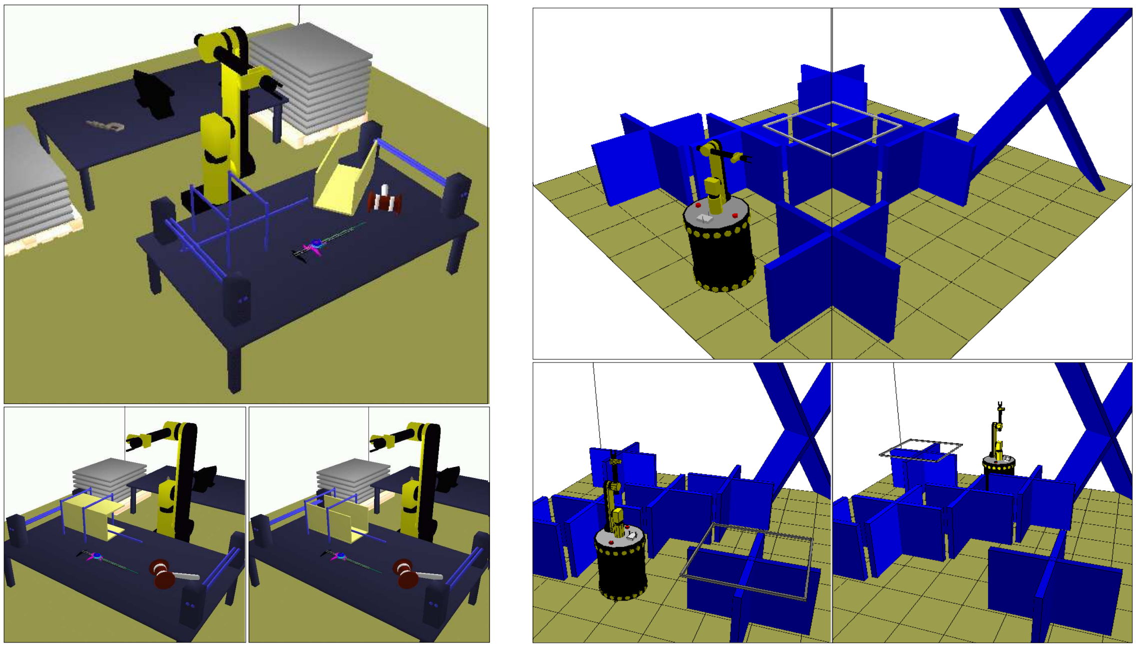


Figure 11: *Scenes and start/goal positions for the* MulGP *and* MobM *examples*

The manipulation planner was implemented within the software platform *Move3D* [24] developed at LAAS. Several environments have been used as test-bed of the planner. In this section, we present the results obtained onto three of them. The computation times correspond to experiments conducted on a 330MHz Sparc Ultra 10 workstation. The first example corresponds to the problem of Figure 1. We refer to it as the *Cage* example.

Two other scenes are shown in Figure 11: the left image illustrates a problem (*MulGP*) involving the same arm manipulating a more complicated u-shaped object. Manipulating this object requires to consider multiple classes of grasps and of placements depending on the contact faces used to grasp/release the object. The right example (*MobM*) corresponds to a problem involving a manipulator arm mounted onto a mobile platform.

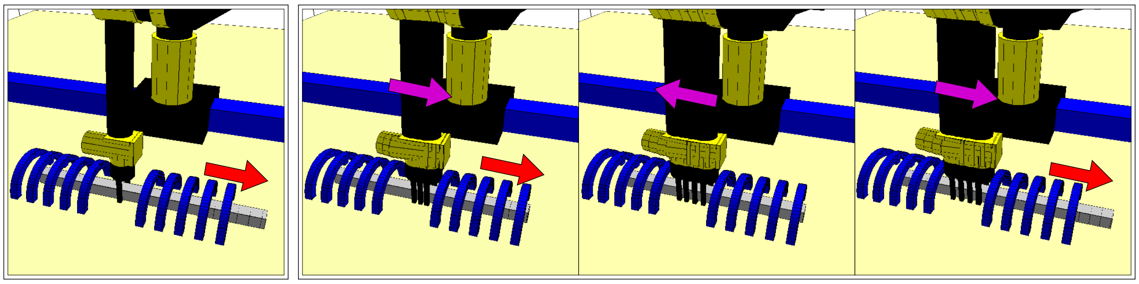


Figure 12: *A $CG \cap CP$ path with a sliding motion of the bar (left) transformed into a sequence of three feasible* transfer/transit/transfer *manipulation paths (right)*

The difficulty with the *Cage* example is the complexity of the manipulation task. Several consecutive re-graspings motions through the middle of *cage* obstacle are necessary to move the bar to a position where it can be regrasped by its extremity. The planner automatically computes the required configurations from only one continuous placement domain (the floor) and one grasping zone all along the bar. The path to get the bar out of the cage is found in the $CG \cap CP$ manifold, and then transformed during the post-processing step in a sequence of transit and transfer paths (see Figure 12). The final path contains 20 elementary paths with 8 re-graspings of the movable object. This difficult manipulation problem was solved in less than two minutes, which demonstrates the efficacy of the proposed approach.

In the example *MulGP*, the manipulation problem is to re-orient the U-shaped movable object, starting from an initial placement where it is trapped by the mechanical device lying at the left of the workplan. This problem was solved by considering 8 grasp classes, each corresponding to a continuous grasp along one of the eight thin faces defined by the U-shaped form of the object. Also, the set of stable placements correspond to positions where one of the three large faces contact with the workplan. We then consider 3 classes of placements according to the orientation of the movable object when placed onto the table. Figure 13 shows the manipulation solution computed by the planner. Here, the presence of several grasp/placements classes, and the larger size of the movable object (which results in more *RRT* calls during the connection stage) increase the overall cost of manipulation planner (see Table 1).

In the example *MobM*, the mobile manipulator (9 *dof*) can only pass from one side of the scene to the other side through the passage under the X-shaped 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. Figure 14 shows the manipulation solution computed by the planner. Here, the manipulation task is simpler compared to the previous examples; less re-graspings are needed to solve the problem. The difficulty illustrated by the example is to deal with a redundant system. An infinite set of solutions exists to achieve the same grasp. Redundancy is a challenge

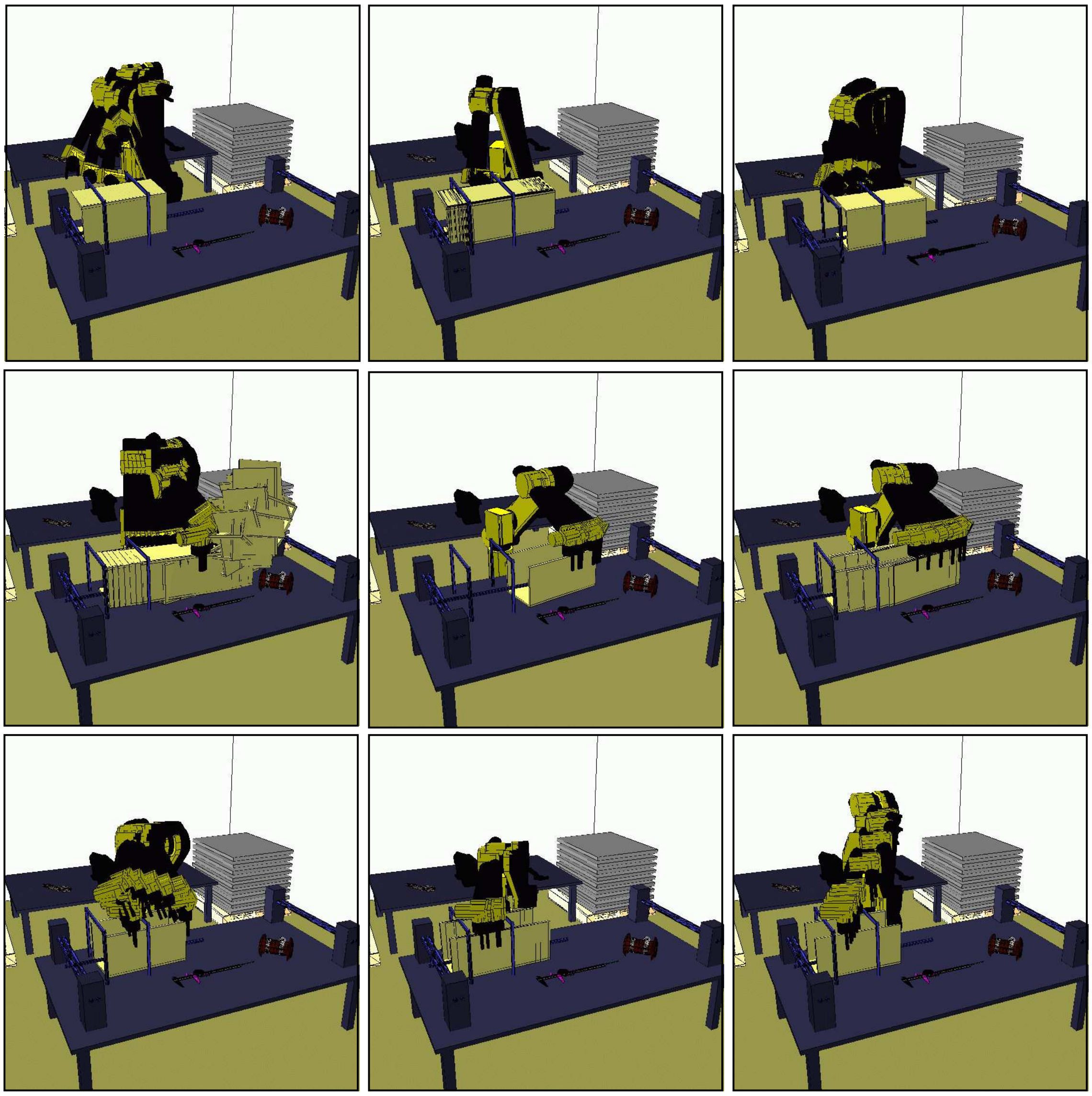


Figure 13: *Manipulation path computed for the* MulGP *problem*

when treating closed chain mechanisms. The exploration of the $CG \cap CP$ manifold for such systems is efficiently performed by using the approach in [8].

Table 1 shows for the three examples numerical results of the performance of the algorithm. All examples were solved after less than five minutes of computation. One can also note that most of the computation time is spent for checking connections with transit and transfer paths; this shows the interest of the proposed approach which limits the number of such connection tests by first computing connected components inside $CG \cap CP$. Also, the use of the visibility technique is the reason for the small size of the manipulation roadmaps; such small roadmaps also reduce the number of connections to be tested.

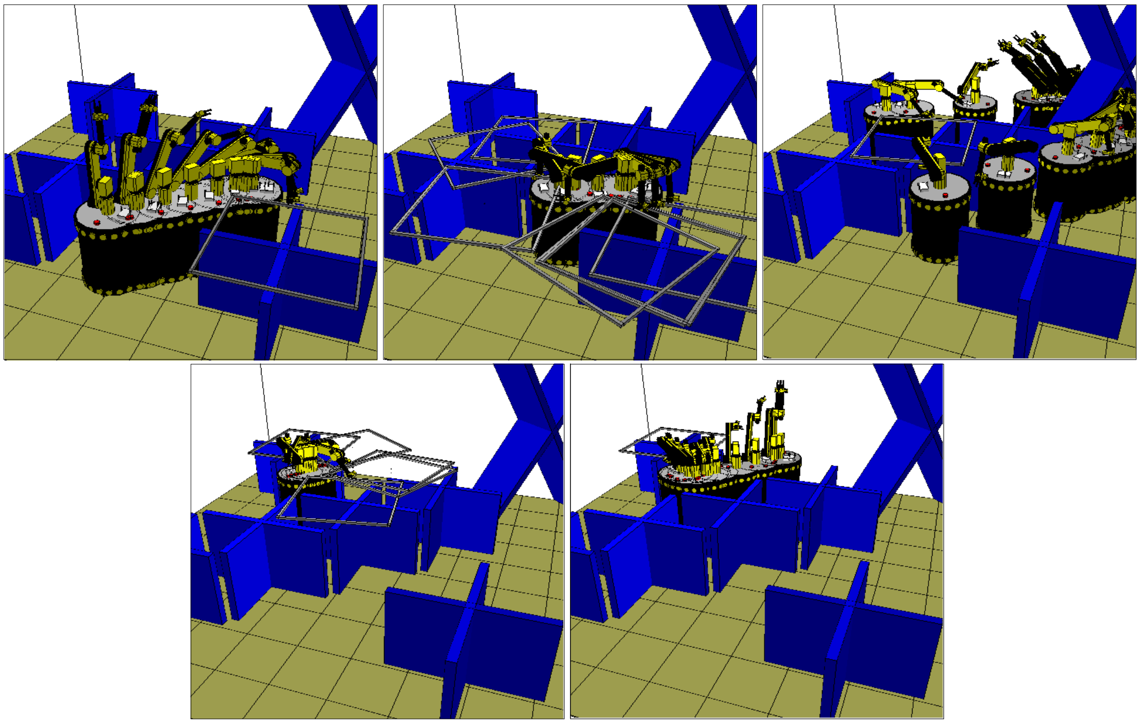


Figure 14: *Manipulation path computed for the* MobM *problem*

Examples:	**Cage**	**MulGR**	**RobM**
Total Time	96 s	330 s	293 s
Computing $CG \cap CP$	23 s	10 s	6 s
Transit/Transfer paths	70 s	306 s	284 s
Total Col. checks	43518	187342	102241
Local $CG \cap CP$ *paths*	3689	1104	190
Transit/Transfer paths	54	168	147
Dynamic updates	15	91	17
N. manip.nodes	32	36	21
N. manip. paths	29	30	21

Table 1: *Numerical results*

7 Conclusion

We have presented a new approach to manipulation planning. The power of the approach lies in the fact that it can deal with continuous settings of the manipulation problem. It relies on a structuring of the search space allowing to directly capture into a probabilistic roadmap the connectivity of the sub-manifolds that correspond to the places where transit paths and transfer path can be connected. This structuring allows us to design a manipulation planner that automatically generates inside continuous domains, the particular grasps and placements that make the problem solvable. Simulations results show the effectiveness of the approach for solving complex manipulation problems.

There remain several possible improvements, in particular to improve the performance of the connection planner which remains the most costly operation. This addresses the issue of improving the efficiency of PRM planners when facing dynamic changes of the environment. Also, our manipulation planner is currently restricted to a single movable object manipulated by a single robot. Considering the case of multiple movable objects and robots first requires studying the conditions under which the reduction property can be extended to such situations. We also begin to investigate a more general approach [9] combining a symbolic task planning level with our geometric manipulation planner for solving a higher level of problems complexity in presence of multiple objects and robots.

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, J.P. Laumond, A geometrical Approach to planning Manipulation Tasks. The case of discrete placements and grasps." In *Fifth 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 J.C. Latombe. Robot motion planning: a distributed representation approach. In *International Journal of Robotics Research*, 10(6), 1991.

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

[6] P. Bessiere, J. Ahuactzin, T, El-Ghazali and E. Mazer. The "Ariane's clew" algorithm: Global planning with local methods. In *IEEE Int. Conf. on Robots and Systems*, 1993.

[7] P.C. Chen and Y.K. Hwang. Practical path planning among movable obstacles. In *IEEE Int. Conf. on Robotics and Automation*, 1991.

[8] 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. Conf. on Robotics and Automation*, 2002.

[9] F. Gravot, R. Alami and T. Siméon. Playing with several roadmaps to solve manipulation problems. In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.

[10] L. Han and N. Amato. A kinematics-based probabilistic roadmap method for closed kinematic chains. In *Algorithmic and Computational Robotics (WAFR00)*, B.R. Donald et al (Eds), AK Peters, 2001.

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

[12] Y. Koga and J.C. Latombe. Experiments in dual-arm manipulation planning. In *IEEE Int. Conf. on Robotics and Automation*, 1992.

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

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

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

[16] J.P. Laumond and R. Alami. A geometrical Approach to planning Manipulation Tasks in robotics. In *LAAS Technical Report n° 89261*, 1989.

[17] JP. Laumond, T. Siméon. Notes on visibility roadmaps for motion planning. In *Algorithmic and Computational Robotics (WAFR00)*, B.R. Donald et al (Eds), AK Peters, 2001.

[18] S. LaValle, J.H. Yakey, and L. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *IEEE Int. Conf. on Robotics and Automation*, 1999.

[19] T. Lozano-Perez, J.L. Jones, E. Mazer, P.A. O'Donnell. Handey: a robot task planner. *Massachusetts Institute of Technology*, 1992.

[20] K. Lynch and M.T. Mason. Stable pushing: mechanics, controllability and planning. In *Algorithmic Foundations of Robotics (WAFR94)*, K. Goldberg et al (Eds), AK Peters, 1995.

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

[22] 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.

[23] T. Siméon, J.P. Laumond and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. In *Advanced Robotics Journal* 14(6), 2000 (a short version appeared in IEEE IROS, 1999).

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

[25] 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 Int. Conf. on Robotics and Automation*, 2002.

[26] A. Sahbani, J. Cortés, T. Siméon. A probabilistic algorithm for manipulation planning under continuous grasps and placements. In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.

[27] G. Wilfong. Motion planning in the presence of movable obstacles. *ACM* Symposium on Computational Geometry, 1988.