A General Manipulation Task Planner

Thierry Simeon, Juan Cortés, Anis Sahbani, Jean-Paul Laumond

A General Manipulation Task Planner

Thierry Siméon, Juan Cortés, Anis Sahbani, and Jean-Paul Laumond

LAAS-CNRS, Toulouse, France

Abstract. This paper addresses the manipulation planning problem which deals with motion planning for robots manipulating movable objects among static obstacles. We propose a general manipulation planning approach capable to deal with continuous sets for modeling both the possible grasps and the stable placements of the movable object, rather than discrete sets generally assumed by the existing planners. The 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. This property leads to reduce the problem by structuring the search-space. It allows us to devise a manipulation planner that directly captures in a probabilistic roadmap the connectivity of sub-dimensional manifolds of the composite configuration space. Experiments conducted with the planner demonstrate its efficacy to solve complex manipulation problems.

Several applications such as robot programming, manufacturing or animation of digital actors require some real or virtual artifact to move and manipulate objects within an environment composed of obstacles.

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 [14]. 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. Motion planning in this context appears as a constrained instance of the coordinated motion planning problem. Indeed, movable objects can not move by themselves; either they are transported by robots or they must rest at some stable placement. The solution of a manipulation planning problem consists in a sequence of sub-paths satisfying these motion restrictions. In related literature (e.g. [2,14]), 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 ex-

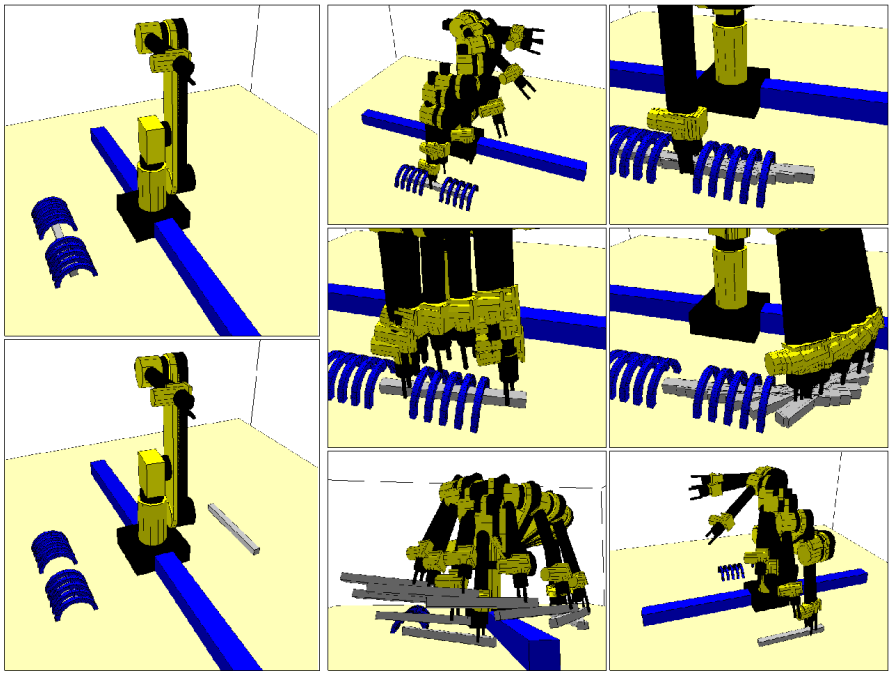

Fig. 1. A manipulation planning problem (left) and its computed solution (right)

tract the bar from the cage, and finally reach the specified goal position. This example shows 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. [1,2,5,12,18]) 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 precise grasps and placements to be given as input data.

In this paper, we describe a more general approach based onto recent results presented in [22,23]. The proposed approach is capable to deal with a continuous setting of the manipulation problem. It allows us to design 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 1. 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 2 describes the proposed two-stage approach. First, Section 2.1 shows how the connected components of this sub-space can be captured in a probabilistic roadmap computed for a virtual closed-chain system. Using the visibility technique [20] extended to deal with such closed systems [7], this first stage of the approach captures the connectivity of the search space into a small roadmap generally composed of a low number of connected components. Connections between these components using transit or transfer motions are then computed in a second stage (Section 2.2) by solving a limited number of point-to-point path planning problems. The details of an implemented planner interleaving both stages in an efficient way are described in Section 3. Finally, we discuss in Section 4 some experiments and the performance of the planner.

1 Problem Statement

Let us consider a 3-dimensional workspace with a robot $\mathcal{R}$ and a movable object $\mathcal{M}$ moving among static obstacles[1] . 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 C_{rob} and C_{obj} be the configuration spaces of the robot and the object, respectively. The composite configuration space of the system is $CS = C_{rob} \times C_{obj}$ 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.

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.

- *Transfer Paths* where the robot moves while holding $\mathcal{M}$ with the same grasp. Along a transfer path, the configuration parameters q_{obj} of $\mathcal{M}$ change according to the grasp mapping induced by the forward kinematics of the robot: $q_{obj} = g(q_{rob})$.

Let P (resp. G) denote the set of stable placements (resp. grasps) of $\mathcal{M}$. Both types of paths lie in lower dimensional sub-spaces of CS_{free}. These sub-spaces are defined as follows:

[1] See [2,14] for a more formal and general description of the manipulation problem

- The *Placement* space CP is the sub-space of CS_{free} defined as the set of free configurations where the mobile object is placed at a stable position, i.e. a position $p \in P$ at which $\mathcal{M}$ can rest when it is not grasped by the robot. Each transit path lies in CP. However, note that a path in CP is not generally a transit path since such path has to belong to the sub-manifold corresponding to a given placement p.

- The *Grasp* space CG is the sub-space of CS_{free} defined as the set of free configurations under all possible grasps $g \in G$ of the object $\mathcal{M}$. A transfer path belongs to the sub-manifold defined by a particular grasp kept constant along the path.

Problem: Given the two sets (discrete or continuous) P and G defining the stable placements and feasible grasps, the 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 of $CG \cup CP$. 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-manifold defines the places where transit paths and transfer paths can be connected. The manipulation planning problem appears as a constrained path planning problem inside (and between) the various connected components of $CG \cap CP$.

1.1 Discrete Case

In the case of a discrete number of grasps and stable placements, the intersection $CG \cap CP$ consists of a finite set [2] of configurations. In this case, it is possible to build a *Manipulation Graph* (MG) where each node corresponds to a configuration of $CG \cap CP$, and edges are constructed by searching for transfer (or transit) paths between nodes sharing the same grasp (or placement) of the mobile object(s).

Most implemented manipulation planners consider that such discrete sets P and G are given as input to the planner. The general framework proposed in [2] for one robot and several moving objects with discrete grasps and placements was first implemented for two simple robots: a translating polygon [2] and a $3dof$ planar manipulator [15]. This framework was extended in [11,12] to multi-arm manipulation planning where several robots cooperate to carry a single movable object amidst obstacles. The planner proposed in [12] first plans the motions of the movable object using a RPP technique [4], and then finds the transfer/transit motions of the arms to move the object along the computed path. This heuristic approach led to impressive

[2] Note however that the case of a redundant robot considered in [1] also leads to an infinite cardinality of the set $CG \cap CP$.

results for solving realistic problems. Another heuristic planner proposed in [5] iteratively deforms a coordinated path first generated in the composite configuration space using a variational dynamic programming technique that progressively enforces the manipulation constraints.

More recently, two other contributions extended existing 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 (i.e. the cardinality of $CG \cap CP$ is infinite). Finally, [18] proposed another practical manipulation planner based onto the extension of the PRM framework [10,19]. The planner builds as in [2] a manipulation graph between discrete configurations of $CG \cap CP$, but connections are computed using a *Fuzzy PRM* planner that builds a roadmap with edges annotated by a probability of collision-freeness. Such roadmaps improve the efficiency of the planner for solving the possibly high number of path planning queries in changing environments required to compute the connections.

1.2 Continuous Case

The structure of the search space is more complex when dealing with continuous sets P and G. For example, one may 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 would define P as a domain in a 3-dimensional manifold of CS_{obj} (two translations in the horizontal place and one rotation around the vertical axis). Also, one may consider G as a set of continuous domains such that the jaws of a parallel gripper have a sufficient contact with two given faces of $\mathcal{M}$. With such grasp constraints, G also corresponds to a 3-dimensional manifold (two translations parallel to the grasped faces and one rotation around the axis perpendicular to the faces).

In such cases, $CG \cap CP$ is defined by continuous configuration domains. Extending the notion of manipulation graph to continuous domains, first requires that the connectivity of $CG \cap CP$ adequally reflects the existence of a manipulation path. This is not *a priori* obvious since paths computed in $CG \cap CP$ correspond to a continuous change of the grasp, while transfer paths need to stay in the sub-manifold defined by a constant grasp.

Reduction Property: It is shown in [3] [3] that two configurations which are in a same connected component of $CG \cap CP$ can be connected by a manipulation path. This property means that any path in $CG \cap CP$ can be transformed into a finite sequence of transit and transfer paths. It is then sufficient to

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

study the connectivity of the various components of $CG \cap CP$ by transit and transfer paths.

Manipulation Graph: The reduction property allows to extend the Manipulation Graph to the continuous case: $MG's$ nodes correspond to the connected components of $CG \cap CP$ and an edge between two nodes indicates the existence of a transit/transfer path connecting two configurations of different connected components. This property was used in [3] to derive, for the specific case of a translating polygonal robot, a cell decomposition algorithm capable to manipulate one movable polygon with an infinite set of grasps.

2 A General Approach to Manipulation Planning

We now describe a more general approach for solving manipulation problems under continuous manipulation constraints. The main idea is to exploit the reduction property of Section 1 to decompose the construction of the manipulation graph into the two following steps:

- compute the connected components of $CG \cap CP$. Section 2.1 describes a method that directly captures the topology of $CG \cap CP$ inside a probabilistic roadmap [10,19].

- determine the connectivity of these connected components using transit-paths and transfer-paths. Section 2.2 explains the method used to compute such connections.

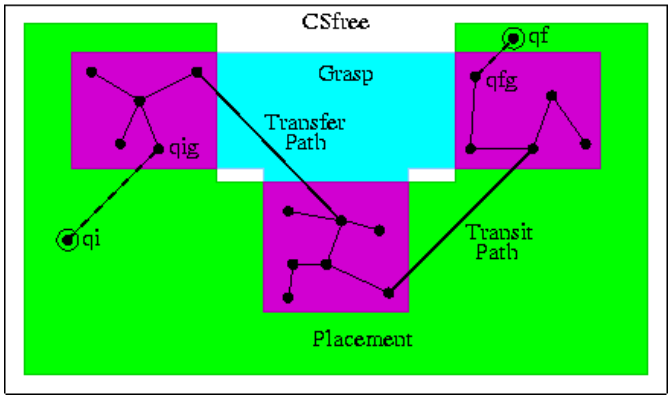

Fig. 2. Structure of the Manipulation Roadmap

Figure 2 illustrates the structure of a manipulation roadmap computed using this approach. The roadmap is structured into a small number of nodes

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

2.1 Capturing $CG \cap CP$ Topology via Closed-Chain Systems

The main critical issue is to capture into a probabilistic roadmap the topology of $CG \cap CP$ which is a sub-manifold of the global configuration space with a lower dimension. The difficulty in using probabilistic roadmap approaches is to face sub-dimensional manifolds. Indeed the probability to choose a configuration at random on a given sub-dimensional manifold is null.

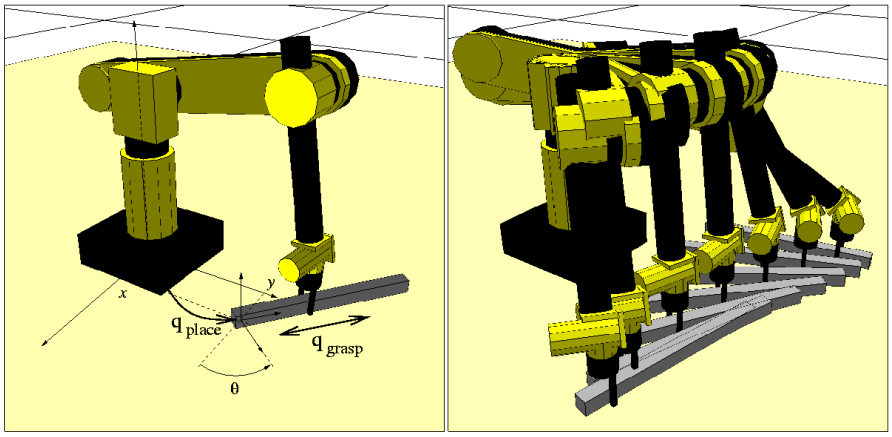

Fig. 3. Virtual closed-chain system and a feasible $CG \cap CP$ motion (the bar moves on the floor while sliding into the gripper's jaws)

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 3 shows the example of a closed chain system formed by a $6dof$ arm manipulating a 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 degrees 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 P of stable placements corresponds to the planar motions parameterized by a vector $q_{place} = (x, y, \theta)$ (two horizontal translations and a vertical rotation), that maintain the contact of the bar with the floor. The motion shown in the Figure is a feasible motion in $CG \cap CP$. It is not admissible from the manipulation problem point of view. However, it can be transformed into a finite sequence of feasible transit and transfer paths.

We then propose to apply planning techniques for closed chain systems to capture the topology of $CG \cap CP$. Indeed, several recent contributions extended the PRM framework [10,19] to deal with such closed chain mechanisms [17,9,7]. In particular, we use the algorithm [7] that demonstrated good performance onto complex 3D closed chains involving tenths of degrees of freedom. As initially proposed in [9] the loop is broken into two open sub-chains, but the random node generation combines a sampling technique called *Random Loop Generator* (RLG) with forward kinematics for the *active* chain and inverse kinematics for the remaining *passive* part of the loop in order to force the closure. The advantage of the RLG algorithm is that it 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. The practical efficacy of our approach results from the good performance reached today by these closed-chain extensions of the PRM framework.

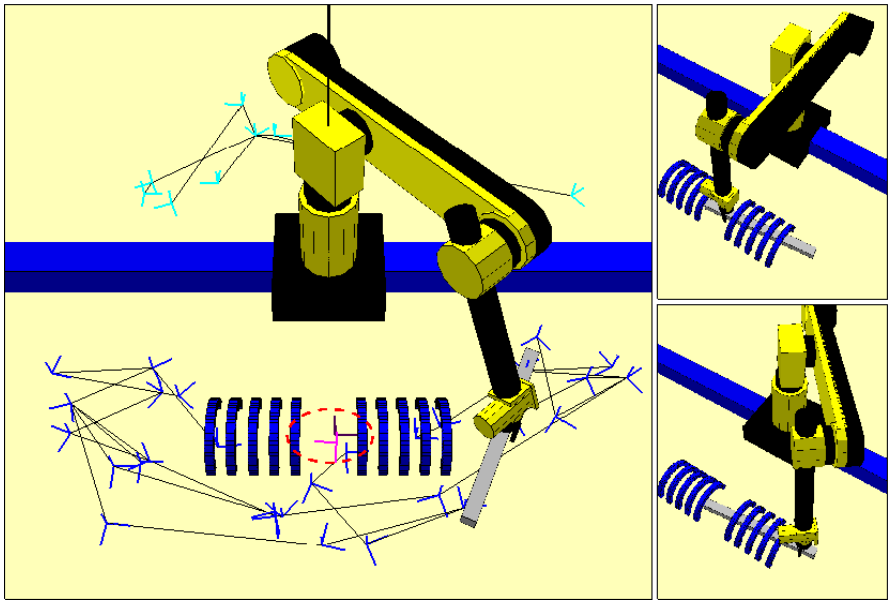

Fig. 4. A Visibility Roadmap computed in $CG \cap CP$ (left) and two placements of the system inside two different connected components of $CG \cap CP$ (right)

The $CG \cap CP$ roadmap is computed using the *Visibility-PRM* technique [20,16]. 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 also 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 4 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 a similar position 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.

2.2 Connections with transit and transfer paths

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 MG). These configurations correspond to fixed grasps and placements of the movable object, noted $(g_i, p_i)_{i=1,2}$. The problem is now to determine whether both configurations can be connected by collision-free transit/transfer paths searched inside the corresponding leaves of CP and CG. Such a connection is possible if there exists:

- 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).
- or 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).

We next explain the two-stage method used by the planner to solve these path-planning queries. Note that each query has to be performed in a partially modified environment since it corresponds to a particular grasp or placement

of the movable object. The motivation 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 by disregarding 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.

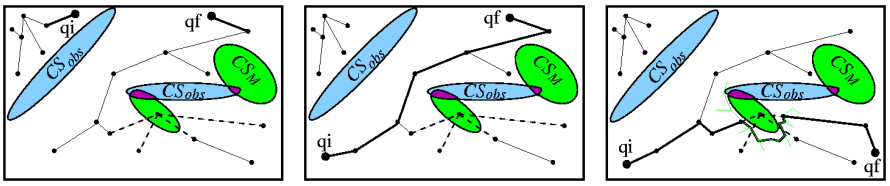

Fig. 5. Search into the *labeled* roadmap: **a/** no solution exists **b/** a solution is found **c/** local update of the *blocked* edges

The search for a given path is then performed within the *labeled* roadmap. As illustrated by Figure 5, 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 onto the Figure) then a solution is found. Now let us consider the intermediate situation where the a path necessarily contains blocked edges. In such case, the algorithm tries to solve the problem locally by using a Rapid Random Tree (RRT) algorithm [13] to connect the endpoints of the blocked edges. 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.

3 Manipulation Planning Algorithm

We now describe in more details the algorithm that was developed to implement the approach presented above. The algorithm takes as input several classes of possible continuous grasps G_i. 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, each 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. For each couple of sets (G_i,P_j), the resulting virtual closed-chain system is considered when analyzing the connectivity of $CG \cap CP$ sub-spaces corresponding to the couple.

3.1 Building the Manipulation Graph

The algorithm incrementally constructs the manipulation graph MG by interleaving the two steps of the approach: computing $CG \cap CP$ connected components and linking them. Following the principle of the *Visibility-PRM* technique, the algorithm stops when it is not able to expand the graph after a given number of tries (MAX_NTRY). This number of failures is related to an estimated coverage of the composite configuration-space by MG.

EXPAND_GRAPH(MG)
$ntry \leftarrow 0$
$expanded \leftarrow$ FALSE
while ($ntry <$ MAX_NTRY(MP)) and ($\neg expanded$)
  $N \leftarrow$ NEW_NODE(MP)
  $expanded \leftarrow$EXPAND_G∩P(MG,N)
  if ($\neg expanded$) and (TRY_OTHER_LINKS($ntry$)) **then**
    $expanded \leftarrow$CONNECT_OUTSIDE_G∩P(MG,N)
  $ntry = ntry + 1$
return *expanded*

The function EXPAND_GRAPH performs one expansion step of MG. Candidate nodes are first sampled in the $CG \cap CP$ manifold and the different types of connections to the graph are then tested.

A candidate node N is generated as follows by the function NEW_NODE: it first randomly selects one of the P_j sets given as input to MP and a stable configuration $p \in P_j$ of the movable object is then chosen by randomly sampling q_{place}. Similarly, one of the grasp classes G_i and the corresponding parameters q_{grasp} are also randomly chosen. The candidate node N is generated when the sampled grasps and placements are collision-free and feasible for the virtual closed system.

Following the discussion in Section 2.1, it is more efficient to explore $CG \cap CP$ than to check connections outside. Therefore, the sample N is first used by EXPAND_IN_G∩P which checks connections to the graph components using $CG \cap CP$ paths. This function keeps N to expand the graph when the node creates a new component or merges existing components of the graph. In the other case, the choice of testing other connections with CONNECT_OUTSIDE_G∩P is made by the boolean function TRY_OTHER_LINKS, based onto a biased random choice {True, False} that depends on the evolution of the size of MG: the first expansion steps starts with a low probability to call CONNECT_OUTSIDE_G∩P; when the roadmap grows, this probability increases as the percentage of the coverage estimated by the fraction $(1-\frac{1}{ntry})$ (see [20] for details).

Expansion in $CG \cap CP$. The function EXPAND_G∩P carries out the exploration of the $CG \cap CP$ manifold using LINKED_IN_G∩P to test the link

EXPAND_G∩P(MG,N)
$n_{linked\ comp.} \leftarrow 0$
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

between N and each connected component of MG by means of paths in $CG \cap CP$. Note that such connections can be achieved only with the nodes computed from the same couple (G_i, P_j) as the one used to generate N. For each component C_i, nodes with such characteristics are tested until a connection is feasible or all those nodes have been checked. The connections are determined from local paths computed in $CG \cap CP$ for the virtual closed system induced by (G_i, P_j). Following the visibility principle, 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.

Connections with transit and transfer paths. The function EXPAND_G∩P fails when N can be linked by $CG \cap CP$ paths to only one connected component of MG. In this case, the role of CONNECT_OUTSIDE_G∩P is to test the link to the other connected components using the composition of a transfer and a transit path, as explained in Section 2.2. Function LINKED_OUTSIDE_G∩P tests the connection between N and each node of C_i using such paths. This function stops checking a component when a valid connection is found. In such case the component of node N is merged with the linked C_i.

CONNECT_OUTSIDE_G∩P(MG,N)
$n_{linked\ comp.} \leftarrow 0$
for i=1 **to** (N_COMP(MG) | $C_i \neq$COMP(N)) **do**
  if LINKED_OUTSIDE_G∩P(N,C_i) **then**
    $n_{linked\ comp.} = n_{linked\ comp.} + 1$
if $n_{linked\ comp.} \neq 0$ **then**
  UPDATE_GRAPH(MG)
  return TRUE
else
  return FALSE

3.2 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 LINKED_OUTSIDE_G∩P function described above 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. 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.

4 Experimental Results

The manipulation planner was implemented within the software platform *Move3D* [21] currently 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 6: the left image illustrates a problem (*MobM*) involving a manipulator arm mounted onto a mobile platform. The right example (*RoBr*) corresponds to an industrial logistics problem treated in the framework of the European Project *MOLOG* [4] . The manipulation device is a rolling bridge.

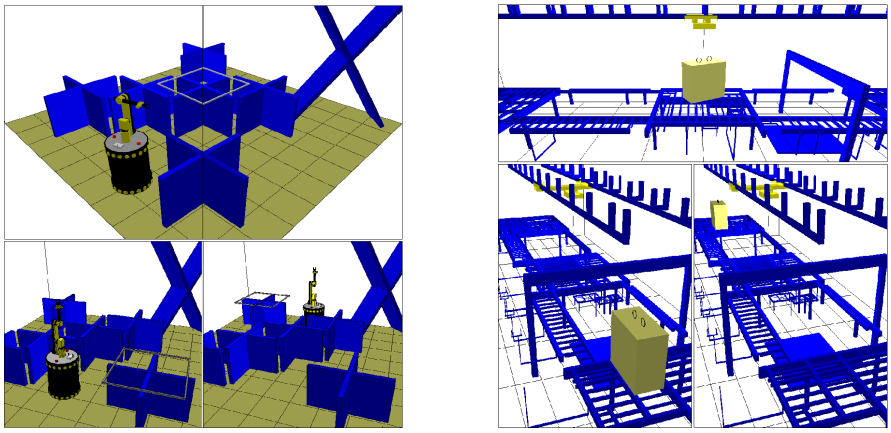

Fig. 6. Scenes and start/goal positions for the *MobM* and *RoBr* examples

[4] see `http://www.laas.fr/molog`.

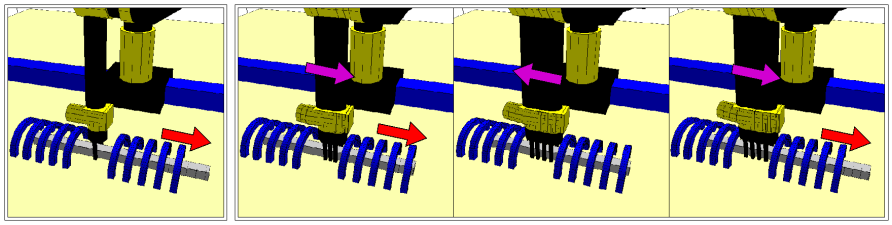

Fig. 7. A $CG \cap CP$ path with a sliding motion of the bar (left) transformed into a sequence of three feasible *transfer/transit/transfer* manipulation paths (right)

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 7). 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 *MobM*, the mobile manipulator ($9dof$) 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

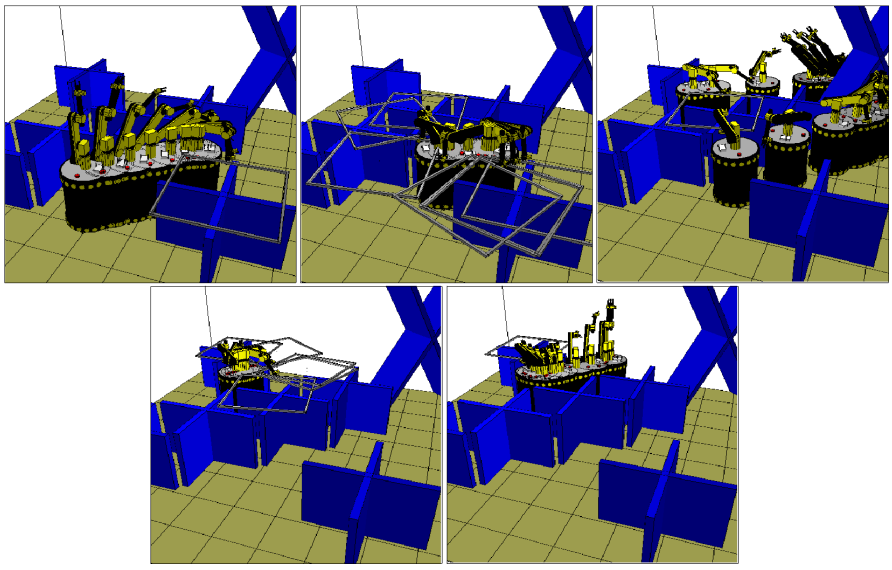

Fig. 8. Manipulation path computed for the *MobM* problem

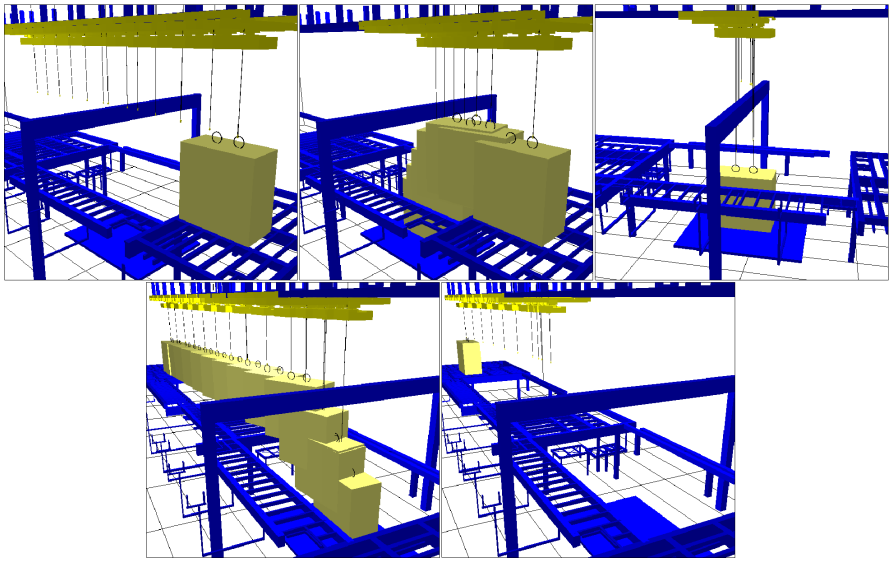

Fig. 9. Manipulation path computed for the *RoBr* problem

(the square frame). A continuous grasping set is defined all around this object. The frame can be placed on the central obstacles. Figure 8 shows the manipulation solution computed by the planner. Here, the manipulation task is simpler compared to the *Cage* example; 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 when treating closed chain mechanisms. The exploration of the $CG \cap CP$ manifold for such systems is efficiently performed by using the approach in [7].

In the example *RoBr*, transporting the load between the extreme sides of the scene requires to generate an intermediate placement under the "arc" obstacle such 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 manipulation solution computed by the planner. Here, the rolling bridge a simple system ($4dof$). However, the geometric complexity of the static scene (≈ 10.000 facets) and the large size of the movable object (which results in many RRT calls during the second stage) increase the overall cost of this connection stage.

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 reduces the number of connections to be tested.

Table 1. Numerical results

Examples:	**Cage**	**MobM**	**RoBr**
Total Time	96 s	293 s	146 s
Computing $CG \cap CP$	23 s	6 s	1 s
Transit/Transfer paths	70 s	284 s	143 s
N. manip.nodes	32	45	10
N. manip. paths	12	14	6

5 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 grasps and placements. 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 in the way to interleave the construction of $CG \cap CP$ with the connection of its connected components via transit or transfer paths. 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 an alternative approach [8] combining a symbolic task planning level with our 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. 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.
8. 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.
9. 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.
10. L. Kavraki and J.C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE Int. Conf. on Robotics and Automation*, 1994.
11. Y. Koga and J.C. Latombe. Experiments in dual-arm manipulation planning. In *IEEE Int. Conf. on Robotics and Automation*, 1992.
12. Y. Koga and J.C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. on Robotics and Automation*, 1994.
13. J. Kuffner and S. Lavalle. RRT-Connect: an efficient approach to single-query path planning. In *IEEE Int. Conf. on Robotics and Automation.*, 2000.
14. J.C. Latombe. *Robot Motion Planning*, Kluwer, 1991.
15. J.P. Laumond and R. Alami. A geometrical Approach to planning Manipulation Tasks in robotics. In *LAAS Technical Report n^o 89261*, 1989.
16. 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.
17. 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.
18. Ch. Nielsen, L. Kavraki. A two-level fuzzy PRM for manipulation planning. In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2000.
19. 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.
20. 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).
21. 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.
22. 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.
23. 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.