A Generic Motion Planner for Robot Multi-fingered Manipulation

Jean-Philippe Saut, Anis Sahbani, Véronique Perdereau

A Generic Motion Planner for Robot Multi-fingered Manipulation

Jean-Philippe Saut[1], Anis Sahbani[2] and Véronique Perdereau[3]

[1]*LAAS-CNRS,* [2]*ISIR-CNRS*

[1] *Université de Toulouse ; UPS, INSA, INP, ISAE: 7, Avenue du Colonel Roche, F-31077 Toulouse Cedex 4*

[2,3] *Université Pierre et Marie Curie - Paris 6: 4, Place Jussieu, BC:173, F-75005 Paris*

[1]*jpsaut@laas.fr,* [2]*anis.sahbani@upmc.fr,* [3]*veronique.perdereau@upmc.fr*

Abstract

This paper addresses the dexterous manipulation-planning problem, which deals with motion planning for multi-fingered hand manipulating objects among static obstacles, under quasi-static movement assumption. We propose a general manipulation approach capable to compute object and finger trajectories as well as the finger relocation sequence, in order to link any two given configurations of the composite system hand+object. It relies on a topological property that characterizes the existence of solutions in the subspace of configurations where the object is grasped by the n fingers. This property leads to reduce the problem by structuring the search space. The developed planner captures in a probabilistic roadmap the connectivity of sub-manifolds of the composite configuration space. The answer of the manipulation planning query is then given by searching a path in the computed graph. Simulation experiments are reported for different multi-fingered manipulation task examples showing the efficiency of the proposed method.

keywords: manipulation planning, multi-fingered manipulation, finger gaiting, grasp computation, motion planning

1 Introduction

The specificity of the robotic multi-fingered manipulation, compared to the whole arm manipulation, is that grasp reconfiguration is possible and often necessary because of the limited accessibility domains of the fingers. This specificity allows object manipulation to be performed in a small workspace and avoids the need for pick and place operations if the grasp configuration must be changed to be adapted to a given task. The goal of dexterous manipulation planning (DMP) is to compute feasible trajectories for the grasped object and for the fingers, in order to drive the object from an initial configuration to a final one. The ability of multi-fingered hands to perform grasp reconfiguration raises two issues from the planning point of view. First, the manipulation task is composed of both continuous and

discrete events. The continuous events are finger and object motions whereas the discrete events are the breaking of some contacts between the fingertips and the object. Secondly, the grasp can become unstable if the planning method does not take into account the stability constraint. A DMP method must consequently integrate schemes that will take into account these two aspects. Our main goal is to develop a method, focused on the motion planning aspect of the dexterous manipulation, that will be able to solve problems involving complex manipulation tasks, not only simple movements like minor reorientation or regrasping. Such a method could constitute the core of the top high level of a complete dexterous manipulation framework. Lower levels focused on control, can then use trajectories provided by our DMP method to compute the finger contact forces and their associated joint torque parameters.

The paper is organized as follows. First, section 2 gives a brief overview of the literature concerning dexterous manipulation planning problems and a brief description of a previous work on manipulation planning with a single robot arm, which has been a source of inspiration for the proposed approach. Section 3 presents problem formulation and hypotheses. Section 4 explains the principle of the proposed method and describes its main algorithms. Finally, section 5 illustrates the capacities of our method with some manipulation tasks that are solved by our planner and shows the benefits of the method for reducing computing times. This section also deals with the influence of the main tuning parameter of our algorithm.

2 Related Work

The early works on Dexterous manipulation planning concerned the problem formulation, without presenting any resolution scheme [1, 2] (for more recent formulations see [3, 4]). They only suggested a choice for the configuration space and presented the different geometric and kinematic constraints related to dexterous manipulation.

Different methods were then proposed, that can be classified into two main categories: *local* methods and *global* ones. The methods of the first category only consider a particular aspect of dexterous manipulation like small object movements without grasp reconfiguration or, on the contrary, contact relocation with no regard on finger or object kinematics. The second category contains the methods that try to consider all the aspects of dexterous manipulation, in a global manner.

2.1 The Local Planning Methods

Among the local approaches is the work of *D. Rus* [5]. She got interested in the finger motion that makes the grasped object rotate. She proposed a full dynamics algorithm called the *finger tracking* algorithm. The main idea of this algorithm is to use two fixed fingers that do not move (with respect to the world frame) and a third one that moves to control the reorientation movement. Rus proved the existence of a diffeomorphism between the system configuration space and $SO(3)$. The object orientation is then fully determined by the positions of the moving fingers. A set of differential equations can be found allowing

to find the forces that must be exerted by the fingers in order to give a specified rotational acceleration to the object.

[6] suggested a planning method for regrasping with a planar four-fingered hand manipulating a polygon. It is based upon the construction of a *switching graph*. Each node of this graph is a set of one among two particular kinds of grasps called *parallel grasps* and *concurrent grasps*. Parallel grasps are the three-fingered grasps such that there exists three parallel vectors, each one included in one of the three contact friction cones, and the middle vector direction opposite to the two others. Concurrent grasps are made of the three contact grasps such that there exists three concurrent lines in the corresponding double-sided friction cones, and these three friction cones positively span $\mathbb{R}^2$ (i.e. every vector in $\mathbb{R}^2$ can be written as a linear combination, with positive coefficients, of any vector triplet, each included in one of the friction cones). This condition is used by the authors to prove that the set of all concurrent grasps, for contacts on three different specified edges, can be represented as a set of points in the plane, all of them included in a polygon called *focus cell*. A graph is built, whose nodes are focus cells. Two nodes can be linked if the associated focus cells have a non void intersection and two edges in common.

2.2 The Global Planning Methods

Different techniques have been proposed that consider the DMP in its various aspects. [7] proposed to build a graph whose nodes are qualitative descriptions of the grasp. These descriptions list the contacts between elements of the grasped object and the hand, such as vertices or edges. The nodes are linked thanks to a planning method that works in joint space and the dexterous manipulation problem solution is found when start and goal configurations are linked to the tree. This work is restricted to a manipulation system with a few degrees of freedom.

[8] made use of a principle similar to the *finger tracking* algorithm as a local method to plan the re-orientation of a convex object. Three fingertips are fixed and the motion of a fourth one is used to rotate the object. The planner works on two levels and its goal is to build a path between two given orientations of the object. The higher level computes a nominal trajectory for the object, chosen to be the shortest possible. The higher level also generates an object orientation along the nominal trajectory, close to the previous orientation on the path. The generated orientation will be used as a sub-goal. A lower level chooses one of three possible manipulation modes: object motion while moving all the fingers or only some of them or motion of the fingers without moving the object. The lower level uses contact kinematics and finger inverse models to reach the sub-goal. If it is unreachable, another sub-goal is generated.

More recently, [9, 10] proposed methods based upon nonlinear system control theory. They employed motion planning methods for smooth systems, extending them to deal with the discontinuities of finger gaiting. The configuration space is divided into strata, each of them corresponding to a particular grasping finger combination. Some of these strata intersect to form other strata. The constraints associated to each stratum are different and so are the corresponding differential equations. The principle

of stratified system control is to define a common state space where all the vector fields, defined for each stratum, can be considered. This space uses fictitious inputs. These inputs can be the object configuration (six parameters) and the fingertip positions [11] or the object configuration and the fingertip positions relative to the object surface [10]. These inputs are fictitious since the real inputs are the finger joint parameters. Once a trajectory is found, these parameters have to be computed from the fictitious inputs, employing the inverse kinematic models. To steer the object from a configuration to another, the stratified system planning methods compute the velocity that will allow the change of configuration in a given time. The fingertip positions are another system input. They must be chosen so that they will stay inside the finger accessibility domains all along the movement, while allowing the grasp to verify the force closure stability criterion.

Among the most recent works on dexterous manipulation planning is the randomized planning architecture [12] based on contact modes switching. It considers all possible contact modes (sliding, sliding with rolling, with spinning, etc.). Based on the RRT method [13], a global planner builds a random tree to explore the object configuration space and a local planner tries to link the tree nodes. The tree nodes are the object configurations. At each iteration of the algorithm, a new object configuration X_{rand} is randomly generated. The algorithm then tries to link X_{rand} to X_{near}, the tree node that is the closest to X_{rand}. A contact mode combination is randomly chosen among the different possible modes. A short time interval is also randomly chosen and a polynomial trajectory between X_{rand} and X_{near} is assigned by the local level planner. The local planner then uses the inverse dynamic model of the system and the previously chosen contact mode to test the feasibility of the local trajectory. If it is not feasible, X_{rand} and X_{near} are not linked and a new configuration is randomly generated.

Even more recently, [4] proposed to use joint space representation of the grasps and described the problem as a hybrid automaton, which can be seen as a state machine that takes into account both discrete (finger relocation) and continuous (object or finger trajectories) events. They did not present a full resolution method but this is part of their ongoing works.

In our opinion, the drawbacks of existing methods are of two kinds. First, many methods compute the object trajectory before computing the finger trajectories [8, 9, 10, 14]. As the object trajectory depends strongly on the accessibility domain of the fingers, such methods may not find a solution in many situations (see for instance example 3 in section 5), since they can lead to object positions that are out of finger access range. The second drawback is that some methods explore the configuration space at a too low level [15], having a control approach rather than a motion planning one. As the configuration space dimension of a dexterous manipulation system is very large, this leads to high computation times. Therefore the associated proposed examples are always very simplistic (small reorientation of a sphere or an egg-shaped object). Here, we emphasize the motion planning aspect of the dexterous manipulation planning problem rather than the control one, in order to treat complex tasks. Of course, control inputs always have to be ultimately computed and so at a moment a control approach must be considered. We believe that the solutions computed by a motion planning method can serve as inputs for a planner working at the control level so that control and motion planning approaches are complementary.

The goal of the proposed method is to solve the above weaknesses taking into account the specific structure of the configuration space. The presented planner is inspired by our previous works on motion planning for manipulation tasks [16], deriving from the works of [17]. To clarify the contribution of the paper with respect to these works, we give in the next section a very brief description of them and introduce the specificity of our method.

2.3 Contribution

An important contribution to manipulation planning was given in the mid 90s with the work published in [16]. The authors proposed an original description of the problem. The considered system was composed of a robot arm, equipped with a gripper, an object to manipulate and a set of static obstacles. Inside the configuration space $\mathcal{CS}$, two subspaces of particular interest were introduced: $GRASP$ also called $\mathcal{CG}$ and $PLACEMENT$ also called $\mathcal{CP}$. $\mathcal{CG}$ is the subspace of all the configurations where the robot grasps the object and $\mathcal{CP}$ is the subspace of all the configurations where the object is placed (e.g. on the floor or on a table) at a stable position. Two kinds of elementary paths were proposed by the authors to solve manipulation task problems. The first one is a transfer path: a movement of the robot grasping the object with a fixed grasp. The second one is a transit path: a movement of the robot alone, the object being placed at a stable pose. Transfer paths are paths in $\mathcal{CG}$ and transit paths are paths in $\mathcal{CP}$. A solution of the manipulation planning problem is therefore a sequence of transfer and transit paths.

Instead of using these two kinds of path only to explore $\mathcal{CS}$, the planning method proposed in [16] relies on a topological property that characterizes the existence of solutions in $\mathcal{CG} \cap \mathcal{CP}$. $\mathcal{CG} \cap \mathcal{CP}$ is the subspace of all the configurations where the robot grasps the object placed at a stable position. The originality of the method is to explore the different connected components of $\mathcal{CG} \cap \mathcal{CP}$ using paths in this subspace. These paths correspond to simultaneous change of both grasp and robot configurations and so are not admissible from the manipulation point of view. However, they can be transformed into finite sequences of feasible transit and transfer paths, thanks to the reduction property introduced and demonstrated in [17]. The exploration is performed by building a graph with a probabilistic method. The different connected components of the graph exploring $\mathcal{CG} \cap \mathcal{CP}$ are then connected via transfer-transit paths. This method gives very interesting results for many manipulation task problems and so is a good start to develop a technique for planning manipulation with a multi-fingered hand.

This method, however, can not be directly employed for multi-fingered manipulation. Important modifications have to be done. The first modification concerns the definition and the structure of the configuration space that is quite different in the case of multi-fingered manipulation. Indeed, specific and more numerous subspaces have to be introduced. Another modification concerns the elementary paths that must be used to link the configurations; transit paths are no more valid as the object must always be grasped by the hand. At last, an important new aspect is to take into account, the stability of the grasps which is not considered in the previous work. All these modifications justify a complete redefinition of the problem and the development of a new planning technique. These are the contributions of the

present paper and are detailed in next sections.

3 Problem Formulation and Hypothesis

3.1 Studied System and Configuration Space ($\mathcal{CS}$)

The studied system is composed of an n-fingered hand and a rigid object to be manipulated. Each finger is an open kinematic chain attached to the palm, which is assumed to be motionless. The object movement will only result from the movements of the fingers. The object configuration, or pose, is characterized by a position and an orientation and thus belongs to $SE(3)$. The configuration space of the object is noted $\mathcal{CS}_{object}$.

The hand configuration is characterized by the joint parameters of its n fingers. Each finger i has m_i joints. If we note $\theta_{i,k}$ the value of the k^{th} joint angle of the i^{th} finger, the configuration vector of this finger is $\mathbf{a_i} = (\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,m_i}) \in \mathbb{S}^{m_i}$ where $\mathbb{S}$ is the circle. So the configuration vector of the whole hand is noted $\mathbf{a} = (\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n})$. The configuration space of the hand is $\mathcal{CS}_{hand} = \{\mathbb{S}^{m_1} \times \mathbb{S}^{m_2} \times \ldots \times \mathbb{S}^{m_n}\}$ and the configuration space of the composite hand+object system is $\mathcal{CS} = \mathcal{CS}_{hand} \times \mathcal{CS}_{object}$.

We assume all contacts to be point contacts and the fingertips to be "sharp" enough to neglect the effects of rolling on contact positions. Thus, a contact on the fingertip surface is supposed to occur at a single point only. A contact is consequently characterized by its sole position on the object surface. A finger that participates in the object grasp is called a "grasping finger" while a finger that does not is called an "independent finger". A k-fingered grasp is defined as a set of k grasping fingers and their associated contact points and by the joint parameters of the independent fingers. For a given object pose and a given contact point, the configuration of the corresponding grasping finger can be fully determined as long as the finger inverse geometric model has only one solution. If not, as we use a random sample-based planning technique (see next sections), one solution among the possible ones is chosen randomly.

A grasp configuration is then defined by the positions of the contacts on the object surface. A k-fingered grasp is represented by a vector $\mathbf{g}_k = (u_1, v_1, \ldots, u_k, v_k) \in \mathbb{R}^{2k}$ where (u_i, v_i) are the coordinates of the i^{th} contact position on the object surface.

$\mathcal{CS}_{free}$ is the set of all configurations that do not lead to collision except collisions due to grasp contact *i.e.* contacts between fingertips and object surface. The admissible $\mathcal{CS}_{free}$ configurations are the configurations corresponding to object grasps; the other ones systematically correspond to unstable situations. We introduce a fundamental subspace called *Grasp Subspace* and noted GS_k, that is the subspace of all the configurations q with k grasping fingers and $(n-k)$ independent fingers:

$$GS_k = \left\{ \; q \in \mathcal{CS}_{free} / \text{ the object is grasped by any combination of k (among n) fingers } \right\} \quad (1)$$

The system admissible configuration space is $CS_{admissible} = \bigcup_{k \in [\![2;n]\!]} GS_k$. An important inclusion relation is $GS_{k+1} \subset GS_k, \forall k \in [\![0, \ldots, n-1]\!]$. In fact, a (k+1)-fingered grasp is a particular case of a

k-fingered grasp (one of the independent fingers is in a configuration that makes it contact the object surface). As there can be different possible combinations for the k grasping fingers, it is useful to introduce the spaces noted $GS_k^i,\ i \in [\![1; C_n^k]\!]$, subspaces of GS_k:

$$GS_k^i = \left\{ \ q \in \mathcal{CS}_{free} / \text{ the object is grasped by a given combination of k (among n) fingers } \right\} \quad (2)$$

Hence, $\forall k \in [\![2; n]\!]$, $GS_k = \bigcup_{i \in [\![1; C_n^k]\!]} GS_k^i$. Defining these subspaces is interesting because their connectivity can be linked to the need for finger gaiting. Indeed, due to motion constraints, it is not always possible to link two configurations belonging to the same GS_k, whatever path type is chosen; for instance, when the k grasping fingers change between two configurations, i.e. when the two configurations belong to different GS_k^i. It is then necessary to use a path in GS_{k+1} or in $GS_{k-1} \backslash GS_k$ (GS_{k-1} without GS_k). This is illustrated by figure 1 showing, for a five-fingered hand, how intermediate configurations in GS_5 or $GS_3 \backslash GS_4$ are needed to connect two configurations in GS_4. The need for subspace change is equivalent to the need for finger gaiting (i.e. to perform finger relocations).

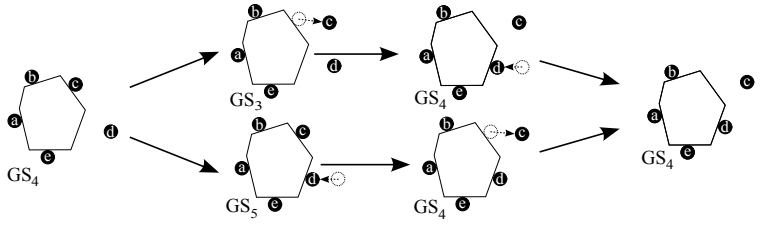

Figure 1: Two ways to link two four-fingered grasps (both belonging to GS_4).

3.2 Constraints and Elementary Movements

A fundamental constraint concerning multi-fingered manipulation is grasp stability, as the object must be held safely during the whole manipulation task. Among all existing stability criteria, we choose the *force closure* one [18, 19]. A grasp satisfies the force closure property if an arbitrary force/torque wrench can be exerted on the grasped object by applying appropriate contact forces. As we assume object and finger movements to be slow enough to neglect inertial effects, we consider that satisfying force closure property at each time is sufficient to guarantee system stability. The force closure property depends on the contact positions and models (point contact with friction, soft contact with elliptic approximation, etc.) and on the values of the possible associated friction coefficients. Another important constraint concerns the kinematics of the system: Object motions are induced by the finger movements. Constraints on object and finger motions lead us to introduce two fundamental kinds of local path, each one corresponding to an elementary manipulation subtask: Grasp reconfiguration (or regrasping) and object displacement. We call the first one *regrasping path* and the second one *transfer path* (by analogy with the transfer path for pick and place problem [20]). During a regrasping path, the object is maintained immobile and some fingers move to change the grasp while during a transfer path, the object is moved but the

grasp remains unchanged (figure 2). The manipulation is performed with a sequence of transfer and regrasping paths.

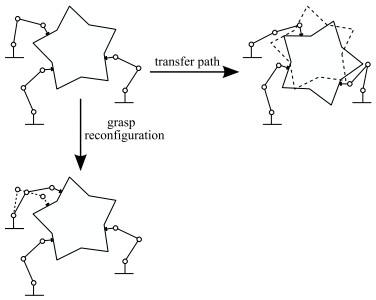

Figure 2: Regrasping and transfer path examples for a planar four-fingered hand.

The goal of the dexterous manipulation planner is then to find such a sequence connecting two given configurations in $\bigcup_{k\in[\![2;n]\!]} GS_k$ while ensuring the grasp stability all along the path sequence.

4 Proposed planning method

The multi-fingered manipulation leads to a sequence of finger movements than can involve 2,3,..., to n-fingered grasps. To each of these grasps is associated a subspace of $\mathcal{CS}$. The solutions of a dexterous manipulation planning problem consequently belong to $\bigcup_{k\in[\![2;n]\!]} GS_k$, the union of all the grasp subspaces. The most exhaustive approach would be to explore this whole set. However, such an approach would lead to prohibitive resolution times because of the very high dimension of the space to be explored. Instead, we choose to favor the search for a solution inside GS_n, for two reasons. The first reason is that privileging GS_n configurations advantages the most stable grasps and the second reason is that GS_n is, among all the grasp subspaces, the one with the smallest dimension, because it is the more constrained. Exploring GS_n will be faster and so will be the resolution of the DMP problem.

So far, no other approach based upon the exploration of GS_n has been proposed because there are no kinematically feasible paths inside GS_n that can link any pair of configurations. Indeed, two configurations in GS_n may correspond to different object poses and to different grasps at the same time and linking these two configurations would mean modifying independently object and grasp configurations. This is not feasible because the object motion is necessarily induced by the finger motions. The originality of the proposed method is to bypass this impossibility, generalizing and exploiting a property that was until now limited to pick-and-place problem. It is the reduction property, introduced in [17].

4.1 Paths in GS_n and Reduction Property

To compute the paths inside GS_n, we use the previously described representation: A configuration in GS_n corresponds to a situation where the object is grasped with n fingers and is so characterized by the object pose and by the contact positions on the object surface i.e. by a vector $(u_1, v_1, u_2, v_2, \ldots, u_n, v_n) \in$

$\mathbb{R}^{2n}$ where (u_i, v_i) is the position of the i^{th} contact on the object surface. By continuously modifying the contact coordinates, the configuration changes in a continuous way, while maintaining all the contacts. These contact parameters can replace the joint parameters and constitute new and less numerous DOFs for the system. For instance, if the hand has four fingers with three DOFs each, the configuration vector for a configuration in GS_n ($n = 4$) has 14 dimensions (6 DOFs for the object and 2 for each of the four grasping fingers) while for an unspecified configuration it has 18 dimensions (6 DOFs for the object and 3 for each of the four grasping fingers). This representation is used to characterize the configurations and to compute paths inside GS_n. An example of a path inside GS_4 is illustrated in figure 3.

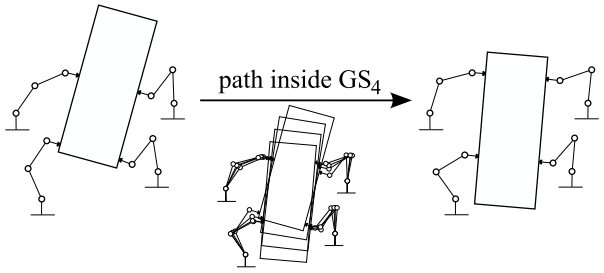

Figure 3: An example of a path computed inside GS_4 for a planar four-fingered hand.

Of course, paths in GS_n are not physically feasible because the object and the contacts can not move independently. However, the reduction property [17] can be extended to the multi-fingered manipulation case. This property states that any collision-free path inside GS_n can be decomposed into a finite sequence of transfer-regrasping paths. We give a demonstration of the extended property in appendix A.

4.2 Principle of the method

We now describe our approach for solving dexterous manipulation planning problems. The proposed approach relies onto the structure of GS_n. The main idea is to exploit the reduction property to decompose the construction of the manipulation graph into the three following steps:

1. explore GS_n and build a graph capturing GS_n connectivity,
2. merge the connected components of the graph with transfer-regrasping paths (through GS_{n-1}),
3. transform GS_n portions of the solution path into a finite sequence of transfer and regrasping paths.

Figure 4 illustrates the principle of the proposed method in the case of a four-fingered hand. Due to obstacles, joint limits or grasp instability, the GS_4 space has several connected components (figure 4 (a)). A graph is built capturing the connectivity of GS_4 (figure 4 (b)) and its connected components are merged with transfer-regrasping paths (dashed curved lines) that pass through GS_3 (figure 4 (c)) .

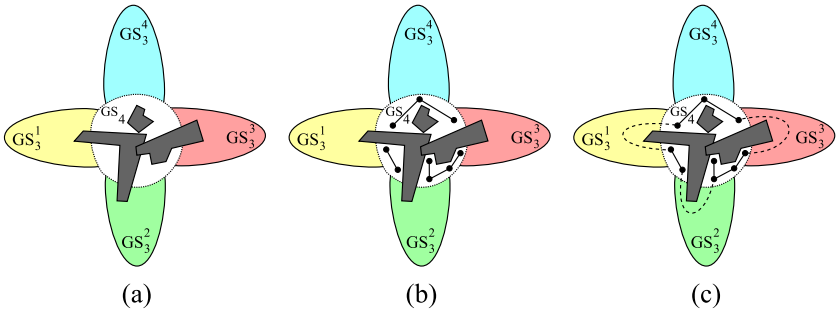

Figure 4: The topology of GS_4, induced by the problem constraints, can be captured in a graph.

4.2.1 The planning technique

We detail in this section our DMP technique. It is based upon the principle of the well-known Probabilistic Roadmap Methods (PRM) [21, 22]. It could be summarized as follows: Initial and goal configurations are the first nodes added to the graph. The graph is then developed inside GS_n, using PRM techniques (figure 4-b), and its connected components are merged using transfer-regrasping paths. These paths are computed with a RRT technique [13] (figure 4-c). Graph construction step stops when the initial and goal configurations belong to a same connected component. A solution path is then searched in the graph using an A*-like algorithm. Once the manipulation path solution of the problem is found, paths inside GS_n are transformed into a finite number of transfer-regrasping paths. This transformation is realized using the algorithm described in section 4.3. Then, regrasping and transfer paths are smoothed using a probabilistic algorithm [21].

Graph Construction

The graph is built alternating two steps:

- Exploring GS_n
- Merging its connected components with transfer-regrasping paths

Algorithm 1 presents the manipulation graph construction. q_{init} and q_{goal} are the start and goal configurations respectively, $\mathcal{G}$ is the graph, $\mathcal{CC}$ the set of $\mathcal{G}$ connected components. l_1 and l_2 are the first connected components of $\mathcal{G}$ containing respectively q_{init} and q_{goal}. The algorithm grows the graph connected components and possibly adds new ones as long as these two initial components l_1 and l_2 are distinct. It ends with the merge of l_1 and l_2 (i.e. when $l_1 \equiv l_2$). The choice of the α parameter is crucial because it has a strong impact on the algorithm convergence. Choosing a small value for α encourages the sampling of GS_n because more configurations are added for each transfer-regrasping connection than for a step of the exploration of GS_n.

The drawback of a dense sampling is an increased computation time. However, if GS_n has numerous connected components, it will be necessary to try to connect them via transfer-regrasping paths. In

```
1  G = {q_init, q_goal} ; CC = {l_1 = {q_init}, l_2 = {q_goal}} ; α ∈]0;1[
2  if l_1 ≡ l_2 then
3      END
4  else
5      while l_1 ≠ l_2 do
6          randomly choose x ∈]0;1[
7          if x < α then
8              explore(q_init,q_goal,G,CC)
9          else
10             connect_components(G,CC)
11         end
12     end
13 end
```

Algorithm 1: Multi-fingered manipulation graph construction algorithm

this case, a value close to 1 should be chosen. Therefore the value of α must be tuned according to the problem characteristics (concerning object size or shape complexity and environment obstacles). The influence of this parameter and the way to choose it are discussed in section 5.2.

4.2.2 The *explore*() function

The main critical issue of the approach is to capture into a probabilistic roadmap the topology of GS_n. The idea is to explore this manifold as such. For this, we consider, as described above, that GS_n is the configuration space of a single system consisting of the hand grasping the object. Additional (virtual) degrees of freedom are added at each contact in order to model the continuity of the grasp (and to allow motion inside GS_n). Exploring GS_n could then be considered as a motion planning problem involving several closed kinematic chains formed by the fingers and the grasped object. One needs to generate configurations verifying chain closures. To solve this problem, RLG algorithm [23] is used. Each chain is divided into an active part and a passive one. The active part configuration is randomly chosen in the accessibility domain of the passive one. The passive part is calculated using the inverse geometric models. In our case, the object represents the active part and the fingers the passive one. The configuration of the object is randomly chosen in an approximation of the hand accessibility domain. For each generated configuration, the grasp stability (i.e. force closure property) is checked. Only stable configurations are added to the graph.

In addition, the *explore*() function tries to link the nodes of the graph (configurations in GS_n) with linear paths in GS_n. It is thus necessary to know how to connect two configurations in this subspace. Indeed, it is not obvious how to compute a movement that combines the continuous changes of both object and grasp configurations. [20] had also to represent a grasp continuous change, for their robotic

arm manipulation planner. They simplified the problem by employing a simple geometry (a parallel-jaw gripper grasping a parallelepiped bar). Thus, only three DOFs (two in translation and one in rotation) are associated with the grasp. In the multi-fingered manipulation case, the problem is more complex because the system has more DOFs and it is desirable to avoid reducing assumptions on the object shape. To keep the generality of the approach, it is crucial to use a grasp parametrization allowing continuous changes.

Since the grasp description needs the contact positions on the object surface, it is necessary to have a parametrization of this surface to compute paths in GS_n. So far, we have implemented star-shaped objects that can be easily parametrized if their surface is approximated by a polyhedron. In a more general case, the parametrization problem can be bypassed by randomly choose points on the object surface and by computing continuous shortest paths on this surface to link these points. A solution is to approximate the surface with a polyhedron. This approximation can be done with an arbitrarily chosen precision. For instance, a geodesic computation algorithm [24] can find the shortest path between two given object surface points. Finally, force-closure property is checked along each computed path in GS_n. In order to satisfy reduction property (allowing the transformation of GS_n paths into a finite sequence of transfer/regrasping paths), additional tests are performed. In fact, for a given configuration belonging to a computed path inside GS_n, the grasp stability is tested for all the grasps obtained by breaking one contact at a time. This is necessary because, at this stage, we do not know yet which finger will have to relocate after the decomposition of the path into a transfer-regrasping path sequence. For every n-fingered grasp along a GS_n path, the stabilities of n (n-1)-fingered grasps are consequently tested. The $(n-1)$ stability tests are also performed for node generation.

4.2.3 The *connect_components*() function

The *connect_components*() function tries to merge two different connected components of the manipulation graph using transfer-regrasping paths. The goal of the the transfer path is to bring the object from its initial configuration to its goal one while the grasp is fixed; and the goal of the regrasping path is to allow grasp change while the object keeps the same configuration. Computing such connections requires that we solve multiple point-to-point path planning problems.

- Transfer path computation

  The movement of the object during a transfer path is a linear trajectory between the two object configurations. The contact positions on the object surface is constant during the movement, so their motion in space is easily computed. The inverse kinematics models of the fingers are used to make their tips follow the motion of their respective contact points. These inverse kinematics models only consider the fingertip positions (not their orientations). The continuity of the joint angles along the path must be checked so that the fingers do not jump from an inverse kinematic solution branch to another.

- Regrasping path computation

To compute the regrasping paths, a collision-free trajectory for each independent finger has to be planned. This is done using the RRT method [13].

4.3 Decomposition of the paths in GS_n

This step is necessary to transform GS_n portions of the solution path into a finite sequence of transfer/regrasping paths. This is done by a dichotomic procedure that iteratively splits the GS_n paths into pieces whose endpoints can be connected by a composition of two collision-free regrasping/transfer paths. The operation of the algorithm is very simple as described in algorithm 2.

Let $q_1 = (o_1, g_1)$ and $q_2 = (o_2, g_2)$ be two configurations linked by a GS_n path. o_1 and o_2 are the configurations of the object and g_1 and g_2 the configurations of the grasp (the contact positions, see section 4.1). The *is_transfer_regrasping_connection_possible* begins by computing a transfer path from (o_1, g_1) and (o_2, g_1) followed by a a regrasping path from (o_2, g_1) and (o_2, g_2). If it fails, a regrasping path between (o_1, g_1) and (o_1, g_2) followed by a transfer path between (o_1, g_2) and (o_2, g_2) are computed. If calculated paths are colliding (or not valid), the configuration halfway along the GS_n portion is generated (noted q_{inter}) and the algorithm is recursively applied to two subpaths connecting this intermediate configuration to the initial and the final ones ($transform\_path$ function).When all the necessary subdivisions are completed, the concatenation of all elementary subpaths is collision-free and respects the manipulation constraints. The process is guaranteed to converge thanks to the reduction property.

```
1  if (is_transfer_regrasping_connection_possible(q1,q2)==true ) then
2      decomposition complete
3      END
4  else
5      if ‖o1 − o2‖ ≥ ε then
6          q_inter ← intermediate_configuration(q1,q2)
7          transform_path(q1,q_inter)
8          transform_path(q_inter,q2)
9      else
10         failure
11         END
12     end
13 end
```

Algorithm 2: Transforming paths inside GS_n into a finite sequence of transfer-regrasping paths

5 Simulation Results and Analysis

5.1 Results

This section presents results obtained from computer simulations, for four different planning problems. We have developed a planner written in C++ that uses the PQP library [25] for collision detection. The simulated hand has four 3-DOF fingers (see figure 5). As the fingers have only 3 DOFs, the inverse kinematics of the fingers has only one solution. If it is not the case, a solution to the inverse kinematics problem is randomly chosen during the sampling phase. The contact model needed for force closure test is the PCWF one (point contact with friction model) and the chosen friction coefficient is 0.8. The graph is built using visibility-PRM method [26]. The computation times correspond to experiments conducted on a PC equipped with an Intel Core2Duo processor (two 2.16 GHz processors) with 2GB RAM. Actually, only one of the processors is employed as the planning program uses only one thread. For each example, we give the resolution times and the number of generated nodes. All the results are averaged on 200 trials. Minimal and maximal times are also given. No planning failures were reported for any of the examples. The probabilistic completeness of the planner could be deduced from the PRM method properties [27].

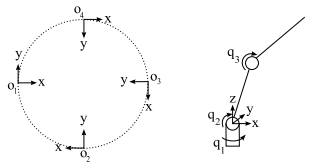

Figure 5: The geometry of the simulated hand.

The first example is very simple and is used for comparison with other existing methods as it is the most common in literature. It concerns the reorientation of a sphere (figure 6). It is solved within a few seconds. The parameter α is constant and set to 0.9.

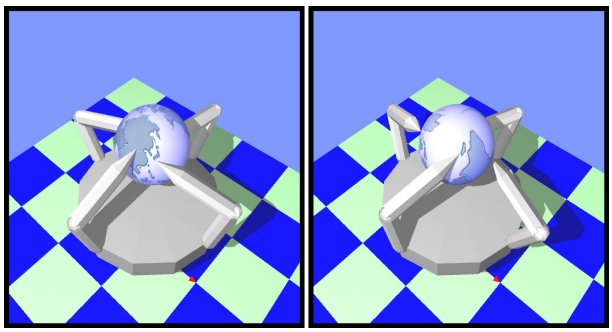

Figure 6: Start and goal configurations for the sphere reorientation problem.

To confirm that exploring GS_4 via paths in this particular space is advantageous, we also solved this example using only "transfer+regrasping" and "regrasping+transfer" paths. We call the associated

method "classic method". The classic method is just a visibility-PRM method [26] whose local method is a single transfer-regrasping or regrasping-transfer sequence (both types are tested for each node connection). This method favors a dense sampling of GS_n (like classic techniques).

Table 1 shows some information on obtained results. The difference between the computing times of the two methods is particularly noteworthy. The exploration of GS_4 allows to considerably reduce the number of samples that is necessary to solve the problem. Far less connections have to be tested and the computation of a connection in GS_4 is much faster than the computation of a connection like "transfer+regrasping" or "regrasping+transfer" because the associated search space dimension is smaller.

Table 1: Results for the sphere example for both proposed and "classic" methods.

Method	Proposed method			Classic method		
	min	mean	max	min	mean	max
resolution time (s)	0.4	5	21	53	580	2141
number of generated nodes	1	41	186	90	858	2733

The purpose of the next example is to study the planner performance for an object whose shape is not as smooth as it is for the sphere. It concerns the reorientation of a box-shaped object. α parameter was initially set to 0.5 and decreased at each trial of connected component merging by a value 0.005 until 0.3. Results are still compared with the ones obtained with the classic method (table 2). Figure 7 shows some steps of the solution obtained for this task. Here, GS_4 has several connected components, unlike for the previous example. The problem takes consequently more time to be solved but the exploration of GS_4 remains very interesting compared to the classic method.

Table 2: Results for the box example for both proposed and "classic" methods.

Method	Proposed method			Classic method		
	min	mean	max	min	mean	max
resolution time (s)	1.2	57	204	54	763	5702
number of generated nodes	14	155	433	138	1165	5703

The third problem's interest comes from the fact that it cannot be solved by a method that first computes the complete trajectory (from start to goal configuration) of the object alone. Such a method

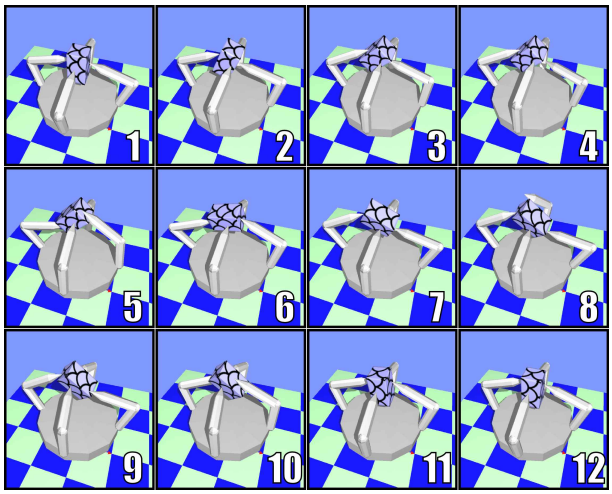

Figure 7: Some steps of a solution obtained for the box problem.

would compute a simple rotation movement whereas the object has also to be translated to always remain reachable by the fingers. The α parameter is constant and set to 0.6. Figure 8 shows some steps of a solution found by our planner. Table 3 gives some information of the planner performance for this example. Tests were not conducted with the classic method as it takes too much time (up to several hours).

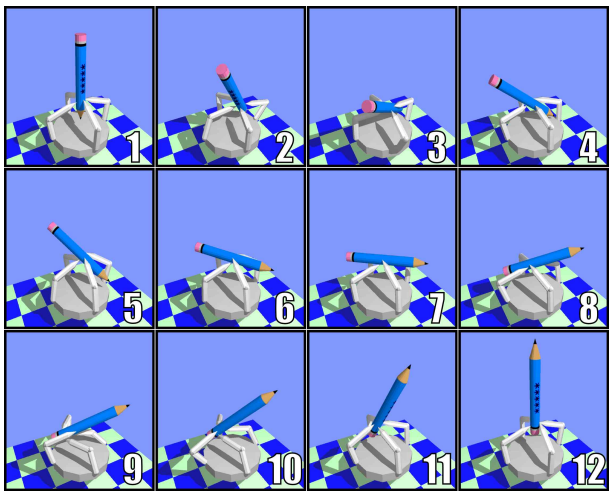

Figure 8: Some steps of a solution obtained for the pencil reversing problem.

At last, we present a more complicated example involving two dexterous manipulation planning subproblems. The goal is first to steer an electric bulb into a plug and then to give it a screwing movement (lift it up and rotate it around a vertical axis). We precise that the socket is just a cylindrical hole; there is no grooves inside. The parameter α is constant and set to 0.8. Figure 9 illustrates some steps of the solution found by our planner. Table 4 shows some information of the planner performance for this example. As expected, the resolution time is much higher than for previous examples since it involves

Table 3: Results for the pencil example for the proposed method.

Method	Proposed method		
	min	mean	max
resolution time (s)	87	699	2423
number of generated nodes	381	1896	6731

the computation of a path into a narrow passage which is a challenging task for most planning methods in general and particularly for probabilistic methods. The obtained results remain very satisfying.

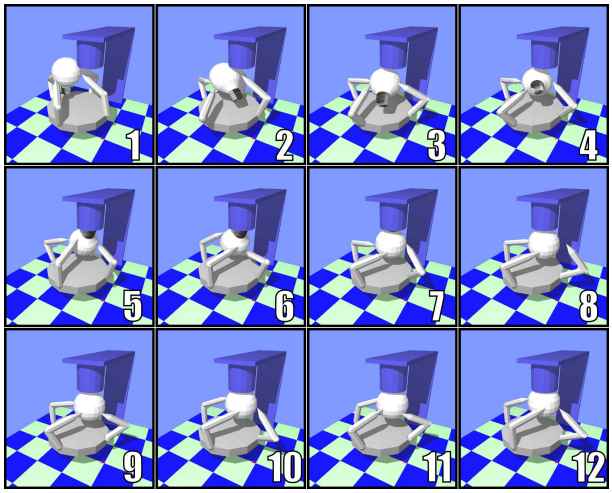

Figure 9: Some steps of a solution obtained for the bulb insertion problem.

Table 4: Results for the bulb example for the proposed method.

Method	Proposed method		
	min	mean	max
resolution time (s)	18	1019	5545
number of generated nodes	28	836	4369

5.2 Influence of the α parameter

So far we chose the values of α that gave what appears to be the best results. The importance of this parameter led us to study its influence on the planner capacity. Therefore experiments measuring the evolution of the resolution time with respect to α, for the first two examples of section 5.1 was conducted. 200 tests were performed for each value of α.

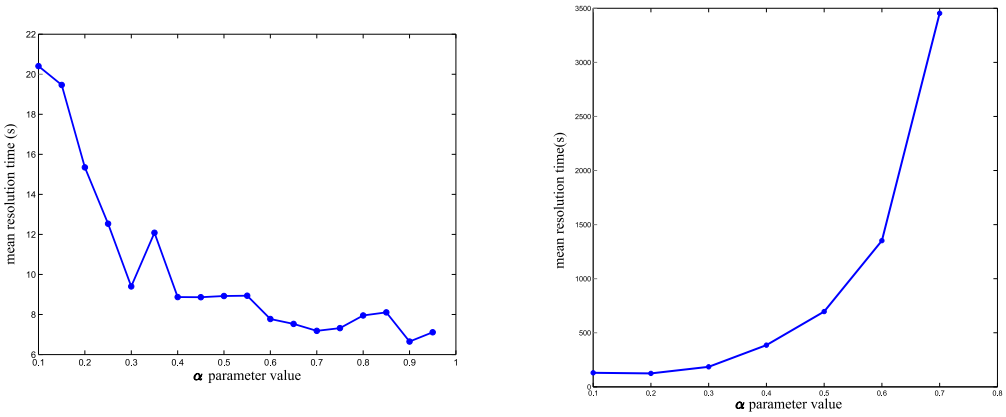

Figure 10: Evolution of the resolution time with respect to the α parameter of the sphere (left) and the box (right) examples.

Logically, figure 10 reveals that the resolution time increases as α decreases (sphere reorientation problem). As GS_4 has only one connected component, it is useless and more time consuming to search for paths in a space of higher dimension as it is done at each connection attempt using regrasping path. In the second example (box problem - figure 10), it is essential to search for regrasping path connections. As the exploration of GS_4 is achieved quickly, the crucial step is merging the connected components of the graph. The bigger is α, the more time the resolution will need and in the extreme case where $\alpha = 1$, the planner can not find a solution. α must so be chosen small if the domain in which the grasp can vary continuously is small, for instance when the object shape is very sharp or if the environment constrains the finger movements. However, the generation of configurations in GS_4 must also be realized as well as the exploration of this subspace. Therefore it is interesting to find a trade-off, using initially a big value of α and decreasing it progressively, e.g. making α vary inversely with the graph size (node number of the graph). As often with the probabilistic methods, the choice of the best value for this parameter remains an issue that would need to be further investigated.

6 Conclusion

We have presented a new generic method for the multi-fingered manipulation motion planning problem with an n-fingered hand. It is based upon a particular description of the configuration space. This description relies on the definition of the grasp subspaces GS_k, that are the subspaces of all the configurations corresponding to k-fingered grasps. The proposed resolution method favors the exploration of GS_n building a probabilistic graph in this subspace. The main originality of our method is to explore mainly GS_n via special paths that are linear paths inside GS_n. This greatly reduces the need to compute regrasping movement trajectories and so the resolution time of the method. The connected components of the graph are merged using elementary paths that pass through GS_{n-1}. Multi-fingered manipulation

tasks have been simulated and have confirmed the validity of the method and very good computing times. In some rare situations, it may be necessary to break more than one contact at a time which is not allowed by the method.

Possible improvements concern the optimization of the computed paths. Another future work could concern the treatment of deformable contacts that offer better grasping possibilities because they permit contacts on sharp parts of the object surface like edges.

REFERENCES

[1] Li, Z., Canny, J., Sastry, S., "On Motion Planning for Dexterous Manipulation, Part I: The Problem Formulation", Proceedings of the IEEE Conference on Robotics and Automation, Scottsdale, USA, pp. 775 - 780 (1989).

[2] Montana,D. "The Kinematics of Multi-Fingered Manipulation", IEEE Transactions on Robotics and Automation, Vol. 11, pp. 491- 503 (1995).

[3] Han, L., Li, Z., Trinkle, J.C., Qin, Z., Jiang, S., "The Planning and Control of Robot Dextrous Manipulation", Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, USA, pp. 263-269 (2000).

[4] Xu,J. and Li,Z., "Kinematic Modelling of Multifingered Hand's Finger Gaits as Hybrid Automaton", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, pp. 3252-3257, (2005).

[5] Rus, D., "In-hand Dexterous Manipulation of 3D piecewise-smooth objects" International Journal of Robotics Research, Vol. 18, No. 4, 355-381 (1999).

[6] Sudsang, A., Phoka, T., "Regrasp Planning for a 4-Fingered Hand Manipulating a Polygon", Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, Vol. 2, pp. 2671- 2676 (2003).

[7] Trinkle, J.C., Hunter, J., "A Framework For Planning Dexterous Manipulation", Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, USA, pp. 775 - 780 (1991).

[8] Cherif, M., Gupta, K., "Planning Quasi-Static Motions for Reconfiguring Objects with a Multi-fingered Robotic Hand", IEEE Transactions on Robotics and Automation, Vol. 4, pp. 491-503 (1999).

[9] Goodwine,B., Burdick,J., "Motion Planning for Kinematic Stratified Systems with Application to Quasi-Static Legged Locomotion and Finger Gaiting", IEEE Transactions on Automatic Control, Vol. 18, pp. 209-222 (2002).

[10] Harmati,I., Lantos,B., Payandeh,S., "On Fitted Stratified and Semi-Stratified Geometric Manipulation Planning with Fingertip Relocations", The International Journal of Robotics Research, Vol. 21, pp. 489-510 (2002).

[11] Goodwine,B., "Stratified Motion Planning with Application to Robotic Finger Gaiting", Proceedings of the IFAC World Congress,1998, Beijing (1998).

[12] Yashima, M., Yamaguchi, H., "Dynamic Motion Planning Whole Arm Grasp Systems Based on Switching Contact Modes", Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington D.C., USA, pp. 2492 - 2496 (2002).

[13] LaValle,S.M., Kuffner,J.J., "Rapidly-exploring Random Trees: Progress and Prospects", Algorithmic and Computational Robotics: New Directions, A K Peters, Wellesley, MA, pp. 293-308 (2001).

[14] Han, L., Trinkle, J.C., "Dextrous Manipulation by Rolling and Finger Gaiting", Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, pp. 730-735 (1998).

[15] Yashima, M., Shiina, Y., Yamaguchi, H., "Randomized Manipulation Planning for A Multi-Fingered Hand by Switching Contact Modes", Journal of the Robotics Society of Japan, pp. 788 - 797 (2004).

[16] Siméon, T., Cortés, J., Sahbani, A., Laumond, J.-P., "A General Manipulation Task Planner", Algorithmic Foundations of Robotics, Springer-Verlag, Vol. 5, pp. 311-328 (2003).

[17] Alami, R., Laumond, J.-P., Siméon, T. "Two manipulation planning algorithms", Algorithmic Foundations of Robotics WAFR94, Springer Verlag (1994).

[18] Bicchi, A., "On the closure properties of robotic grasping", The International Journal of Robotics Research, Vol. 14, #4, (1995).

[19] Liu, Y.-H., "Qualitative test and force optimization of 3D frictional form-closure grasps using linear programming", IEEE Transactions on Robotics and Automation, Vol. 15, pp. 163-173 (1999).

[20] Sahbani, A., Siméon, T., Cortés, J., "A probabilistic algorithm for manipulation planning under continuous grasps and placements", Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Washington D.C., USA, pp. 1560 - 1565 (2002).

[21] Kavraki, L., Latombe, J.C., "Randomized Preprocessing of Configuration Space for Fast Path Planning", Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, USA, pp. 2138-2139 (1994).

[22] Overmars, M., Svestka, P., "A Probabilistic learning approach to motion planning", Algorithmic Foundations of Robotics, Springer-Verlag, San Francisco, USA, pp. 19 - 37 (1994).

[23] Cortés, J., Siméon, T., Laumond, J.-P., "A Random Loop Generator for Planning the Motions of Closed Kinematic Chains using PRM Methods", Proceedings of the IEEE International Conference on Robotics and Automation, Washington D.C., pp. 2141 - 2146 (2002).

[24] Lanthier,M. and Maheshwari,A. and Sack,J.-R., "Approximated Weighted Shortest Paths on Polyhedral Surface", Proceedings on the 13th Annual ACM Symposium on Computational Geometry, Nice, France, pp. 274-283 (1997).

[25] Gottschalk, S., Lin, M.C., Manocha,D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection", Proceedings of ACM Siggraph'96, New Orleans, USA, pp. 171 - 180 (1996).

[26] Siméon, T., Laumond, J.-P., Nissoux, C., "Visibility-based probabilistic roadmaps for motion planning", Journal of Advanced Robotics, Vol. 14, pp. 477- 494 (2000).

[27] Ladd, A. M., Lydia E. Kavraki "Measure Theoretic Analysis of Probabilistic Path Planning", IEEE Transactions on Robotics and Automation, Volume 20, Number 2, pp. 229 - 242 (2004).

APPENDIX

A Reduction property proof

Here we give a proof of the reduction property that derives from [17].

Hypothesis H: The robot hand avoids all forbidden contacts i.e. all contacts with environment obstacles and between any two of its bodies that are not connected by a joint. Accepted contacts are grasp contacts (fingertip/object contacts) and contacts between any two bodies linked by a joint.

The reduction property is then stated as follows:

Reduction property:

Under the hypothesis H, any two configurations of a same connected component of GS_n can be connected by a finite regrasping and transfer path sequence.

Proof:

Let a and b be two configurations of a same connected component of GS_n. There exists a path linking a and b i.e. a continuous function $p : [0, 1] \rightarrow GS_n$ such as $p(0) = a$ and $p(1) = b$.

The object configuration (a vector in $SE(3)$) is noted q_o and the grasp configuration (the positions onto the object surface of n contact points) is noted q_g. As mentioned in section 3, knowing q_o and q_g allows to compute, in a unique way, the robot hand configuration q_r. Let f be the function used to compute the robot hand configuration from the object and grasp configurations:

$$\begin{array}{c} f : CS_{object} \times CS_{grasp} \rightarrow CS_{robot} \\ (q_o, q_g) \mapsto q_r \end{array} \tag{3}$$

$\mathcal{CS}_{object}$ is the configuration space of the object alone ($\mathcal{CS}_{object} = SE(3)$).

$\mathcal{CS}_{grasp}$ is the configuration space of the grasp, characterized by the positions of n points on the object surface. Its topology depends on the topology of the object surface; e.g., if this surface is topologically

equivalent to a sphere, $\mathcal{CS}_{grasp} = (\mathbb{S}^2)^n$. $\mathcal{CS}_{robot}$ is the configuration space of the robot. The robot configurations are characterized by a vector of joint parameters.

Let p_r be the projection of p onto CS_{robot}, p_o the projection of p onto CS_{object} and p_g the projection of p onto CS_{grasp}. Let $c = p(t), t \in [0,1]$ be any configuration along the path p. According to hypothesis H, $p_r(t)$ is included in an open set of $\mathcal{CS}_{robot,free}$, the collision-free configuration space of the robot. So there exists an open ball $B_\epsilon \subset \mathcal{CS}_{robot,free}$, centered on $p_r(t)$, with a radius $\epsilon > 0$.

$f(q_o, q_g)$ is computed from the inverse geometric models of the fingers. These models receive, as an input, the contact positions, depending on object and grasp configurations. The geometric models are assumed to be continuous. Consequently, f is also continuous, meaning that $\forall (q_o^0, q_g^0) \in \mathcal{CS}_{object} \times \mathcal{CS}_{grasp}$:

$$\begin{aligned} \exists \eta > 0,\ \textit{such as} \|q_o - q_o^0\| < \eta \textit{ and } \|q_g - q_g^0\| < \eta, \\ \Rightarrow \|f(q_o, q_g) - f(q_o^0, q_g^0)\| < \epsilon \end{aligned} \tag{4}$$

where the used norms are norms associated to corresponding spaces ($SE(3)$, $\mathcal{CS}_{grasp}$ et $\mathcal{CS}_{robot}$). Besides, p continuity yields directly the continuity of p_o and p_g because the object trajectory is continuous along p and the trajectories of the contact points on the object surface are also continuous along p, by construction of paths in GS_n. Consequently, we have the following continuity relation:

$$\begin{aligned} \exists \eta > 0,\ \textit{such as } \forall (\tau, \sigma) \in]t-\eta, t+\eta[\times]t-\eta, t+\eta[, \\ f(p_o(\tau), p_g(\sigma)) \in B_\epsilon \end{aligned} \tag{5}$$

Let $c_1 = p(\tau_1)$ and $c_2 = p(\tau_2)$ be any two configurations along p such as $(\tau_1, \tau_2) \in]t-\eta; t+\eta[\times]t-\eta; t+\eta[$. Let us prove that c_1 and c_2 can be connected by a transfer path followed by a regrasping path.

Let us consider the path p_1:

$$\begin{array}{rccc} p_1 : & [\tau_1, \tau_2] & \rightarrow & \mathcal{CS}_{object} \times \mathcal{CS}_{grasp} \\ & \tau & \mapsto & (p_o(\tau), p_g(\tau_1)) \end{array} \tag{6}$$

p_1 is a transfer path from $(p_o(\tau_1), p_g(\tau_1))$ to $(p_o(\tau_2), p_g(\tau_1))$.

According to (5) we have:

$$\forall \tau \in [\tau_1, \tau_2], f(p_o(\tau), p_g(\tau_1)) \in B_\epsilon \tag{7}$$

so the robot configuration along path p_1 belong to B_ϵ and p_1 is collision-free.

Let us consider the path p_2:

$$\begin{array}{rccc} p_2 : & [\tau_1, \tau_2] & \rightarrow & \mathcal{CS}_{object} \times \mathcal{CS}_{grasp} \\ & \tau & \mapsto & (p_o(\tau_2), p_g(\tau)) \end{array} \tag{8}$$

p_2 is a regrasping path from $(p_o(\tau_2), p_g(\tau_1))$ to $(p_o(\tau_2), p_g(\tau_2))$.

According to (5) we have:

$$\forall \tau \in [\tau_1, \tau_2], f(p_o(\tau_2), p_g(\tau)) \in B_\epsilon \tag{9}$$

so the robot configuration along path p_2 belong to B_ϵ and p_2 is collision-free. c_1 and c_2 can consequently be connected by p_1 followed by p_2. As the path p_r seen as a set (i.e. the image of $[0,1]$ by function p_r) is a compact set included in an open subset of $\mathcal{CS}_{robot,free}$, there exists a finite number of open balls of $\mathcal{CS}_{robot,free}$ whose union contains p_r (there exists a finite covering of $[0,1]$). The local transformation described above can be realized on each part of p_r covering. The path p between a and b can be decomposed in a finite number of transfer-regrasping sequences.