Dexterous Manipulation Planning Using Probabilistic Roadmaps in Continuous Grasp Subspaces

Jean-Philippe Saut, Anis Sahbani, Sahar El-Khoury and Véronique Perdereau
Université Pierre et Marie Curie-Paris6, LISIF-EA 2385,
3 rue Galilée, Ivry-sur-Seine, 94200 France
jean-philippe.saut@lisif.jussieu.fr

***Abstract*— In this paper, we propose a new method for the motion planning problem of rigid object dexterous manipulation with a robotic multi-fingered hand, under quasi-static movement assumption. This method computes both object and finger trajectories as well as the finger relocation sequence. Its specificity is to use a special structuring of the research space that allows to search for paths directly in the particular subspace GS_n which is the subspace of all the grasps that can be achieved with n grasping fingers. The solving of the dexterous manipulation planning problem is based upon the exploration of this subspace. The proposed approach captures the connectivity of GS_n in a graph structure. The answer of the manipulation planning query is then given by searching a path in the computed graph. Simulation experiments were conducted for different dexterous manipulation task examples to validate the proposed method.**

I. Introduction

A. Dexterous Manipulation

The robotic dexterous manipulation is the kind of manipulation executed with the help of a robotic hand, using small fingertip movements and contact relocations. Unlike for the arm manipulation, only a small workspace is required to manipulate an object and there is no need for pick and place operations in order to change the grasp configuration. Dexterous manipulation implies many research topics such as mechanics, instrumentation, control and planning. This paper addresses the latter. In order to manipulate an object, it is necessary to compute both object and finger feasible trajectories. These trajectories will then be used as the input of the control system of the hand. Computing them is the goal of dexterous manipulation planning (DMP). DMP is a difficult issue because it involves a system which has many degrees of freedom (DOF). Indeed, hands with dexterous manipulation capabilities generally have at least four fingers with at least three DOFs each while the object itself has six DOFs. The exploration of the configuration space of such a system can be consequently very computationally extensive. Another difficulty of DMP is that it implies both continuous (object and finger motion) and discrete (contact relocation) events. A DMP method must therefore integrate schemes taking into account these two aspects.

B. Related Work

The early works on DMP concerned the problem formulation without introducing any resolution scheme ([1],[2]). The first method to be presented was the one by Trinkle and Hunter [3]. The authors proposed to build a graph whose nodes are qualitative descriptions of grasps. These descriptions list the contacts between elements of the grasped object and the hand, such as vertices or edges. The nodes are linked using a planning method working in joint space and the dexterous manipulation problem solution is found when start and goal configurations are linked to the tree. This work was restricted to a manipulation system with few degrees of freedom. Han and Trinkle [4] proposed a framework for the manipulation planning of a sphere with three fingers. A finger needs to be replaced if it is close to its workspace boundary or if it can not ensure a force closure grasp with any of the two others. Rus [5] 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. Cherif and Gupta [6] used the same principle 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. More recently, Goodwine [7] and Harmati *et al.* [8] proposed methods based upon nonlinear system control theory. They use motion planning methods for smooth systems that are extended 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. In each stratum, the vector fields for the control system are smooth, allowing the use of motion planning methods for smooth kinematic non holonomic systems. Yashima *et al.* [9] proposed a randomized planning architecture based on switching of contact modes. The method considers all possible contact modes (sliding, sliding with rolling, with spinning, etc.). Based on the RRT method [10], a global planner builds a random tree to explore the object configuration space and a local planner tries to link the tree nodes. This local planner builds an object trajectory and randomly chooses a contact mode. Then, the inverse problem is used to compute the joint torque trajectories that will lead to the desired object trajectory while satisfying the manipulation constraints. Xu and Li [11] proposed to use joint space representation of the grasps and to describe 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 do not present a full resolution method but this is part of their

ongoing works.

To our opinion, the drawbacks of existing methods are of two kinds. Firstly, many methods compute the object trajectory first and then the trajectories of the fingers ([4], [6], [7], [8]). As the object trajectory depends strongly on the accessibility domains of the fingers, such methods may not find a solution in many situations (see e.g. example 3 in section III). The other drawback is that some methods explore the configuration space at a too low level, having more a control approach than a motion planning approach ([9]). As the configuration space dimension of a dexterous manipulation system is particularly big, this leads to huge computation times. Therefore the associate shown examples are always very simplistic (sphere or egg-shaped object small reorientation). The method we propose in this paper aims to solve these weaknesses taking into account the particular structure of the configuration space.

II. Proposed Method

A. Problem Formulation and hypothesis

1) Studied System and Configuration Space ($\mathcal{CS}$): The studied system is composed of an n-fingered hand and of a rigid object to be manipulated. Each finger is an open kinematic chain attached to the palm, which is considered to be motionless. The object configuration, or pose, is characterized by a position and an orientation. We assume all contacts to be punctual and the fingertips to be "sharp" enough to allow us to neglect the effect of rolling. Thus, the contact on the fingertip surface is supposed to occur only at a single point. A contact is consequently characterized only by its position on the object surface. A finger that participates to the object grasp is called a "grasping finger" while a finger that does not is called an "independent finger". A k finger grasp is defined by a set of k grasping fingers and their relative contact points. For a given object pose and a given contact point, the configuration of the corresponding grasping finger can be determined as long as the finger inverse geometric model has only one solution. If it is not the case, it is possible to reduce the number of solution to one, arbitrarily or using a criterion (e.g. grasp quality criterion). The admissible configurations of $\mathcal{CS}_{free}$ are the ones corresponding to object grasping; the other ones correspond to always unstable situations. Therefore, we introduce a fundamental subspace called *Grasp Subspace k* (GS_k), that is the subspace of all the configurations q with k grasping fingers and $(n-k)$ independent fingers:

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

The system configuration space is composed by the different GS_k subspaces with $k \in [\![0, \ldots, n]\!]$. An important inclusion relation is $GS_{k+1} \subset GS_k, \forall k \in [\![0, \ldots, n-1]\!]$. The reason for this inclusion relation is that a (k+1)-fingered grasp is a particular case of 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:

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

Each GS_k can be seen as the union of C_n^k GS_k^i subspaces. Because of the necessary condition of finger movement continuity, it is not always possible to link two configurations belonging to the same GS_k, whatever path type is chosen. For instance, it is not possible when the k grasping fingers are not the same between these 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 5-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).

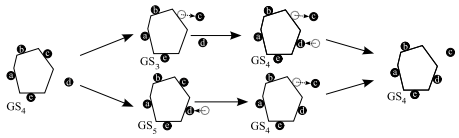

Fig. 1. To link two four-fingered grasps (both belonging to GS_4), it might be necessary to use intermediate three or five finger grasps (belonging to $GS_3 \backslash GS_4$ and GS_5 respectively).

2) Constraints and Elementary Movements: A fundamental constraint that occurs in dexterous manipulation is the grasp stability, as the object must be hold safely during the whole manipulation task. Among all existing stability criteria, we choose the *force closure* one that is certainly the most commonly used. A grasp verifies 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 verifying the force closure property at each instant is sufficient to guarantee the system stability. Force closure property depends only on the contact positions and models (point contact with friction, soft contact with elliptic approximation, etc.). Another important constraint concerns the kinematics of contacts. Indeed, we assume that the movement of the object is induced by the movement of the fingers and that the contacts between the object and the fingertips can not slide on the object surface. This leads us to introduce two fundamental local paths, 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*. 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 (Fig. 2). The goal of DMP is then to find a sequence of transfer and regrasping

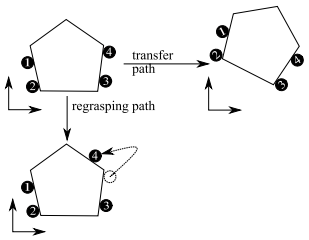

Fig. 2. Regrasping and transfer path examples for a four-fingered hand in the plane.

paths that will link two given configurations both belonging to GS_n while ensuring grasp stability (i.e. force closure) all along.

B. Proposed planning method

1) Main idea of the method: To find a feasible path linking two configurations in GS_n, one needs to explore the system C-space. Classically, we choose to build a graph to explore the configuration space. So far, most existing techniques build a graph by sampling the configuration space and trying to link the obtained samples with elementary paths such as "transfer+regrasping" or "regrasping+transfer" paths. The drawback of such an approach is that it leads to oversample $\mathcal{CS}_{free}$ and consequently to large computing times and to an excessive number of grasp reconfigurations. Indeed, these techniques require subspace change even to link two configurations of the same GS_k^i. We propose instead a new method, advantageously taking into account the GS_k subspace continuity (and so the grasp continuity) and allowing to link directly two configurations belonging to the same GS_k^i. The idea is to add virtual DOFs modeling the continuity of the grasp. These DOFs correspond to continuous contact placements on the object surface. Each contact can then be seen as a sliding joint between the object surface and the fingertip. This representation allows to choose configurations respecting the closure of kinematic chains induced by grasping fingers and object as well as to generate path in a GS_k^i. We can thus define a linear path (Fig. 3) between two GS_k^i configurations. Such a path is obtained by linearly linking the two augmented configuration vectors (augmented with the parameters representing the grasp continuity).

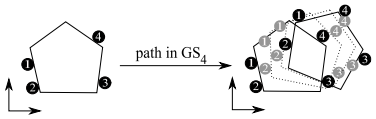

Fig. 3. A linear path in GS_4 for a 4-fingered hand in the plane.

The exploration of the different GS_k^i can then be done using paths included in these subspaces. Consequently, it is possible to directly connect two configurations in GS_k^i, corresponding to both different object poses and different grasps, without having to use in-between configurations (needed to compute a transfer-regrasping path). Introducing intermediate configurations would have led to an oversampling of $\mathcal{CS}$. Of course, paths in GS_k^i are almost always kinematically unfeasible but thanks to the reduction property stated and proved in [12], they can be decomposed in a finite transfer-regrasping path sequence, that is feasible, as long as they are collision free. We explain in the next section, with a 3D four-fingered hand example, the approach we propose and use in our planner.

2) Principle of the method: Let us take the example of a 3D four-fingered hand to illustrate the search space structure described in the previous section. The theory presented in this paper applies to an arbitrary number of fingers but is easier to understand on a practical example. If we choose a point contact with friction model, a grasp must have at least three contacts to be stable. The only interesting subspaces are consequently GS_4 and $GS_3 \backslash GS_4 = \bigcup_{i \in [\![1;4]\!]} GS_3^i$. Figure 4 represents these GS_k subspaces. One can see on Fig. 4 one of the main originality of the proposed DMP technique, which is to link, if it is possible, configurations in GS_4 directly with linear paths in GS_4.

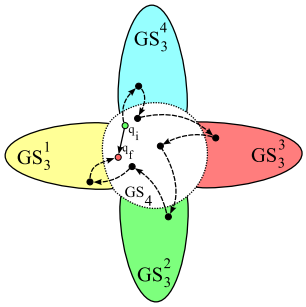

Fig. 4. Linking two configurations q_i and q_f using a path in GS_4 (straight plain line) compared to using transfer-regrasping paths (dashed curved lines). As only one finger can be relocated at a time, at least four regrasping paths are required.

If GS_4 has multiple connected components because of one or more obstacle (obstacles can be joint limits, grasp instabilities or collisions between bodies), paths in GS_4 are not enough. One needs to use paths in $GS_3 \backslash GS_4$ subspace. These paths are regrasping paths (used with transfer paths). The principle of our DMP technique is to explore GS_4 with paths inside this subspace, in order to build a graph. If GS_4 has multiple connected components, the graph exploring GS_4 will have multiple components too and paths in the $GS_3^i, i \in [\![1;4]\!]$ will be used to merge them (transfer-regrasping paths). The obtained graph is the dexterous manipulation graph. Figure 5 shows an example of such a graph. Due to obstacle presence, GS_4 has two connected components. It is necessary to pass through configurations in $GS_3 \backslash GS_4$ i.e. to change the grasp type. This is done using transfer-regrasping paths. Next section detail our planning method to build the dexterous manipulation graph.

3) Dexterous Manipulation Planner: We detail in this section the use of our DMP technique for a 3D four-fingered

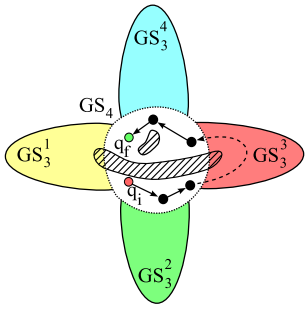

Fig. 5. A dexterous manipulation graph example. Striped zones represent obstacles. q_i and q_f are start and goal configurations. It is necessary to reduce the number of grasping fingers to bypass the obstacles. (Dashed arrow is a regrasping path)

```
1  GM = {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 of the algorithm
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,GM,CC)
9          else
10             connect_ components(GM,CC)
11         end
12     end
13 end
```

Algorithm 1: Manipulation graph construction algorithm

hand with point contact with friction model. The choice of a particular contact model does not modify the way the method works. It only changes the minimum number of fingers that is needed to ensure grasp stability. This method uses the theory described in the previous section. It is based upon the principle of the well-known probabilistic roadmaps [13], [14], in a single query manner. Initial and goal configurations are the first nodes added to the graph. The graph is then developed, until the initial and goal configurations belong to a same connected component of the graph. A path between them is then found in the graph using an A*-like algorithm. Once the global path is found (composed of paths in GS_4 and possible transfer-regrasping paths), the paths in GS_4 are converted into a finite number of transfer-regrasping paths. This transformation is executed using a dichotomic algorithm. Then, the number of regrasping movements is minimized and, at last, regrasping and transfer paths are smoothened using a probabilistic algorithm like the one presented in [13].

Manipulation Graph Construction

The manipulation graph is built alternating two steps:

- GS_4 exploring
- merging its connected components with transfer-regrasping paths

The graph construction algorithm is presented in 1. q_{init} and q_{goal} are start and goal configurations respectively, $\mathcal{GM}$ is the manipulation graph, $\mathcal{CC}$ the set of $\mathcal{GM}$ connected components. l_1 and l_2 are the first connected components that have been found by the algorithm, after adding q_{init} and q_{goal} to $\mathcal{GM}$. $\alpha \in]0;1[$ is an important parameter. The choice of its value is discussed further in the paper.

4) Function explore(): The goal of this function is to build a graph in GS_4 in order to capture this subspace topology. Exploring GS_4 in such a way is a motion planning problem for a system containing several closed kinematic loops. One needs to generate configurations verifying chain closures. To solve this problem, we use RLG algorithm [15]. Each chain is divided into an active part (the object) and a passive one (the fingers). The active part configuration is randomly chosen in the accessibility domain of the passive part. The passive part is calculated using inverse geometric models. The grasp stability (force closure property) is checked for every generated configuration along a path. The function tries to link the nodes of the graph (configurations in GS_4) with linear paths in GS_4. It is thus necessary to be able to connect two configurations in this subspace. Paths inside GS_4 require a specific algorithm. Along such paths, the object moves while the fingers are "sliding" on the surface of the object, in the same time. To keep the generality of the approach, it is crucial to use a grasp parameterization allowing continuous changes. Since grasp description needs the contact positions on the object surface, it is necessary to have a parameterization of this surface to be able to compute paths in GS_4. So far, we have only implemented star-shaped objects that can be easily parameterized if their surface is approximated by a polyhedron. In a more general case, the parameterization problem can be bypassed because actually one just needs to randomly choose points on the object surface and to compute continuous shortest paths on this surface, linking two of these points. A solution is to approximate the surface by a polyhedron. This approximation can be done with an arbitrarily chosen precision. A geodesic computation algorithm is used to find the shortest path between two given object surface points. The path is computed as a set of successive segments. The choice of a random contact point is done by first randomly choosing a facet of the polyhedron using a bias on its area, then by choosing a position on this facet.

5) Function connect_components(): The function connect_components() tries to merge two different connected components of the manipulation graph using transfer-regrasping paths. The transfer path goal is to bring the object configuration from its initial to its goal configuration. The goal of the regrasping path is then to bring the hand to its final configuration.

- transfer path computation
  The movement of the object during a transfer path is a linear trajectory between the two object configurations.
- regrasping path computation

To compute the regrasping paths in a GS_3^i, a collision free trajectory for the free finger has to be planned. This is simply done using the RRT method [10].

Remark on the choice of α:

The choice of α parameter (algorithm 1) is crucial because it influences greatly the algorithm convergence. Having α close to 0 encourages the sampling of GS_4 because configurations are added for each transfer-regrasping connection. However, if GS_4 has numerous connected components, it will improve the convergence speed. A good trade-off is to initialize α with a value close to 1 (to favor GS_4 exploration) and decrease it as the node number of the graph increases (to favor GS_4 connected component merging).

Remark on the number of fingers:

So far, we have chosen a four-fingered hand to illustrate the proposed method. What is changed if the hand has more fingers? For a four-fingered hand with point contacts, the only interesting subspaces were GS_4 and GS_3 for stability reasons. This would not be the case for a different contact model (for instance soft finger contact model) or for a greater number of fingers. In that case, the method remains unchanged: GS_n is explored using paths inside this subspace and the algorithm tries to link its different connected components using transfer paths and regrasping paths in GS_{n-1}. In that case, only GS_n and GS_{n-1} are explored. Solutions requiring that more than one finger relocate at a time can consequently be missed. However such cases seem to be rare. A solution could be then to use regrasping paths in GS_k with $k < n-1$ i.e. to relocate more than one finger at a time. For simplicity reasons, we have so far only implemented our method for a four-fingered hand but plan to do it for more general cases to find the most appropriate approach.

III. Simulation Results

This section presents results obtained from computer simulations, for three different DMP problems. We developed a planner written in C++ that uses the PQP library [16] for collision detection. The simulated hand has four 3-DOF fingers. The contact model used for force closure test is the PCWF one (point contact with friction model) and the chosen friction coefficient is 0.8. We give the computation times that were obtained running the program on a PC equipped with an Intel Core2Duo processor (two 2.16GHz processors) with 2GB RAM. Actually, only one of the processor was used as the planning program uses only one thread.

The first example is very simple and used as a comparison with other existing methods as it is the most common in literature. It concerns the reorientation of a sphere (Fig. 6) and is solved within a few seconds. 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 PRM method whose local method is a single transfer-regrasping or regrasping-transfer sequence (both types are tested for each node connection try). Table I shows some information on the obtained results for 200 tests. The number of generated

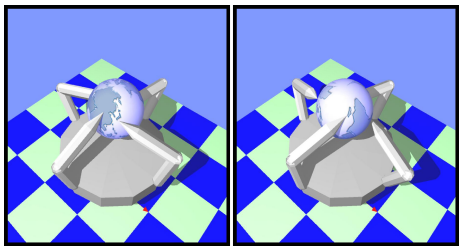

Fig. 6. Start and goal configurations for the sphere reorientation problem. (π rotation around a horizontal axis).

nodes is the number of valid configurations that have been generated in order to be added to the graph. The given computation times only concern the graph construction and research of the solution path inside the graph (equivalent to learning and research phase in PRM method). Decomposition of the paths inside GS_4 and smoothing phase take typically a few seconds, that is marginal compared to the others steps. The difference between the computing time of the

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

TABLE I

Results for the sphere reorientation example for both proposed and "classic" methods, conducted on 200 tests.

two methods is particularly marked. 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 faster than for a connection like "transfer+regrasping" or "regrasping+transfer" because the dimension of the associated search space is smaller.

The next example purpose was to study the planner performance for an object whose shape is not smooth as it is for the sphere. It implies the reorientation of a box-shaped object. Results are still compared with the ones obtained with the classic method (table II). Figure 7 shows some steps of a solution obtained for this task. Here GS_4 has several

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

TABLE II

Results for the box reorientation example for both proposed and "classic" methods, conducted on 200 tests.

connected components, unlike for the previous example. The problem takes consequently more time to be solved but the exploration of GS_4 remains interesting compared to the classic method.

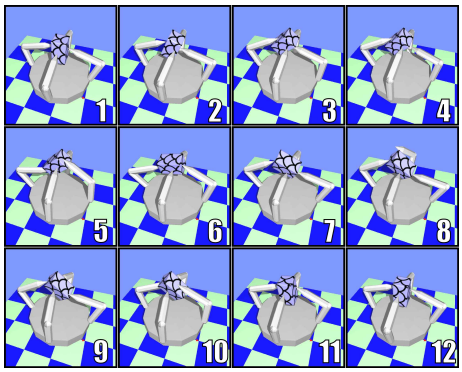

Fig. 7. Some steps of a solution obtained for the box reorientation problem.

The last problem interest comes from that it can not be solved by a method that computes first the trajectory of the object alone. Such a method would certainly compute a simple rotation movement whereas the object has also to be translated to remain always reachable by the fingers. Figure 8 shows some steps of a solution found by our planner (you can also watch the attached video). Table III gives some

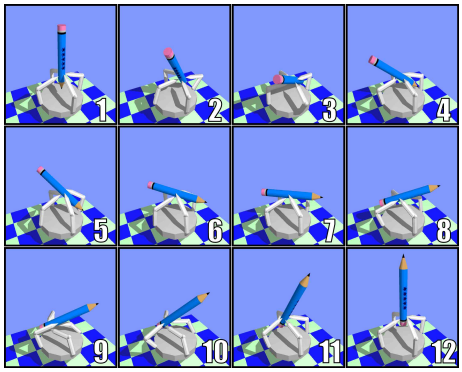

Fig. 8. Some steps of a solution obtained for the pencil inverting problem.

information of the planner performance for this example. Tests were not conducted with the classic method as it takes too many time (up to several hours).

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

TABLE III

RESULTS FOR THE PENCIL INVERTING EXAMPLE FOR THE PROPOSED METHOD, CONDUCTED ON 200 TESTS.

IV. CONCLUSION

We have presented a new method for the dexterous manipulation planning problem with an n-fingered hand. It is based upon a new search space structuring. This structuring relies on the definition of the grasp subspaces GS_k, that are the subspaces of all the configurations corresponding to a k finger grasp. The proposed resolution method builds a graph whose nodes are chosen in the GS_n. The main originality of our method is to explore mainly GS_n via paths included in this subspace, that leads to greatly reduce the dimension of the search space and the number of samples necessary to solve a problem. Dexterous manipulation tasks have been simulated and have confirmed the validity of the method and very low computing times. Improvements are planned concerning the optimization of computed paths. We also plan to study the influence of the only tuning parameter (α) of the method. Another future work concerns the characterization of cases that can not be solved exploring only GS_n i.e. case when more than one finger must relocate at a time.

REFERENCES

[1] Z. Li, J. Canny, and S. Sastry, "On motion planning for dexterous manipulation, part i: The problem formulation," *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, pp. 775–780, 1989.

[2] D. Montana, "The kinematics of multi-fingered manipulation," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 4, pp. 491–503, Aug. 1995.

[3] J. Trinkle and J. Hunter, "A framework for planning dexterous manipulation," *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, pp. 775–780, Apr. 1991.

[4] L. Han and J. Trinkle, "Dextrous manipulation by rolling and finger gaiting," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 730–735, May 1998.

[5] D. Rus, "In-hand dexterous manipulation of 3d piecewise-smooth objects," *International Journal of Robotics Research*, 1997.

[6] M. Cherif and K. Gupta, "Planning quasi-static fingertip manipulations for reconfiguring objects," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 837–848, Oct. 1999.

[7] B. Goodwine and J. Burdick, "Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting," *IEEE Transactions on Automatic Control*, vol. 18, no. 2, pp. 209–222, 2002.

[8] I. Harmati, B. Lantos, and S. Payandeh, "On fitted stratified and semi-stratified geometric manipulation planning with fingertip relocations," *The International Journal of Robotics Research*, vol. 21, no. 5-6, pp. 489–510, Jun. 2002.

[9] M. Yashima, Y. Shiina, and H. Yamaguchi, "Randomized manipulation planning for a multi-fingered hand by switching contact modes," *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, Sep. 2003.

[10] S. LaValle, "Rapidly-exploring random trees: a new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, octobre 1998.

[11] J. Xu and Z. Li, "Kinematic modelling of multifingered hand's finger gaits as hybrid automaton," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3252–3257, Aug. 2005.

[12] R. Alami, J.-P. Laumond, and T. Siméon, "Two manipulation planning algorithms," *Algorithmic Foundations of Robotics WAFR94*, 1994.

[13] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration space for fast path planning," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2138–2139, 1994.

[14] M. Overmars and P. Svestka, "A probabilistic learning approach to motion planning," *In Algorithmic Foundations of Robotics (WAFR94)*, 1994.

[15] J. Cortés, T. Siméon, and J.-P. Laumond, "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 (ICRA)*, pp. 2141–2146, May 2002.

[16] S. Gottschalk, M. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," *Proceedings of ACM Siggraph'96*, 1996. [Online]. Available: http://www.cs.unc.edu/~geom/OBB/OBBT.html