Regrasp Planning for Discrete Objects

Máximo A. Roa and Raúl Suárez

***Abstract*— This paper proposes an algorithm for regrasp planning of 2D and 3D discrete objects, such that the regrasp trajectory ensures a force-closure (FC) grasp while the regrasp motion is performed. The approach takes advantage of a method that quickly explores the grasp space, and relies on the use of independent contact regions and non-graspable regions, which provide large regions of the FC or non-FC subspaces starting from a single sample. Application examples are included to show the relevance of the results.**

***Index Terms*— Regrasp planning, independent contact regions, non-graspable regions.**

I. Introduction

A manipulation problem appears when an object grasped by a multi-fingered hand needs a grasp change during the execution of a task; it implies the hand ability to change the position and orientation of the manipulated object from an initial to a final position. The final position can be achieved by simply moving the object inside the hand's workspace, changing the position of the hand joints. The range of possible movements that can be imparted on the object without changing the actual grasp is determined by the physical limits in the finger joints and the possible collisions between the fingers and the object. If the final position is not achievable in this way, then the contacts between the fingers and the object must be changed at some point during manipulation, and at least one contact point must be located in a different position; in this way the object can achieve a wider range of positions. Under this assumption, two different ways of manipulation can be established: finger gaiting and regrasping.

Finger gaiting (or finger repositioning) involves the relocation of one or more fingers on the object surface while keeping the FC grasp with the remaining fingers (at least 2) [1]. The change of a grasp from n to $n-1$ fingers involves a change in the problem conditions, as the degrees of freedom of the hand-object system may increase when one contact is lost. The sequence of movements starts with an $n-1$ finger grasp. The object is rotated without changing the contact points until one of the fingers reaches its workspace limits, then, a redundant (free) finger must be located to generate another $n-1$ FC grasp that allows repositioning the limiting finger [2]. Another approach simply changes between different FC grasps by using one or more free fingers, rotating the object until the desired position is reached [3].

This work was partially supported by the projects DPI2007-63665 and DPI2008-02448.

M. Roa is with the Institute of Industrial and Control Engineering (IOC), Technical University of Catalonia (UPC). He works for the R&D Department at Hewlett Packard, Spain. maximo.roa@ieee.org

R. Suarez is with the IOC - UPC, Barcelona, Spain. raul.suarez@upc.edu

On the other hand, the regrasping approach (or multi-fingered manipulation) solves the manipulation problem by simultaneously using all the available fingers; the positions of the fingers can only be changed by rolling or sliding them along the object surface. The theoretical basis for rolling contacts has been established considering the finger-object system [1] and also including the hand kinematics [4]. Manipulation by rolling has been simulated, even using wheeled fingertips [5], but real applications are limited to simple experimental setups such as a 2-finger hand manipulating a ball [6], mainly due to control and stabilization problems, as well as limitations in the workspace of the fingers [7].

Finger sliding is a process that repositions the fingers by sliding them along the object surface; the theoretical basis for this process has been studied [8], [9], but it is hard to be mechanically implemented as the fingers must touch the surface during all the sliding movement [10], which requires tactile sensors with high accuracy and a very controlled hand dynamics. Assuming that the manipulation is performed at low velocities, then the interaction forces between the fingers and the object are dominant compared to the inertial forces, and the manipulation can be considered as quasi-statical, which simplifies the problem formulation.

A dexterous manipulation planner that takes advantage of the quasi-statical formulation for 3D smooth objects has already been proposed [11]. However, computation of regrasp trajectories for general 3D objects has only been recently tackled. When dealing with 3D arbitrary shaped objects, a common approach to describe their surface is by using a cloud of points or a triangular mesh. A method was proposed to avoid dealing with the large amount of data involved in these representations; the approach starts with a triangular mesh, which is simplified and used to build a regrasp roadmap [12].

This work discusses the problem of searching the trajectories for the fingertips on an object surface, in order to change from an initial FC grasp to a final desired one while ensuring the FC condition (i.e. ensuring the resistance to external disturbances) during the finger movements. A solution to the problem is presented based on the concepts of independent contact regions (ICRs) and non-graspable regions (NGRs) [13]. ICRs are defined such that the positioning of a finger in each ICR ensures an FC grasp, independently of the exact position of each finger. NGRs are defined such that a finger contact in each NGR always produce a non-FC grasp, independently of the exact position of each finger.

The approach used in this work focuses only on the object geometry and the FC property to find the trajectories for the fingertips on the object surface, i.e. it is object-centered.

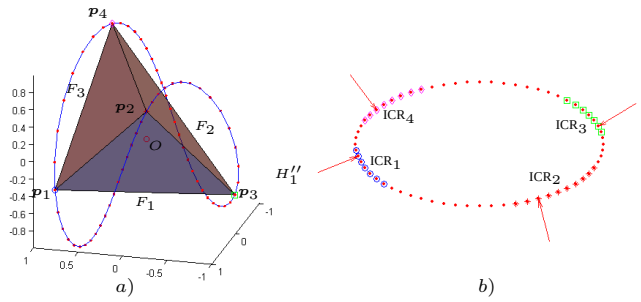

Fig. 1. ICRs for a discretized ellipse: a) FC grasp in the wrench space; b) ICRs on the ellipse.

The kinematics of the grasping device is not considered. It is assumed that the manipulation is performed at low velocities, therefore the manipulation can be considered quasi-statical.

The rest of the paper is organized as follows. Section II provides a background for the regrasp planning problem. Section III describes the approach proposed to plan a regrasp movement on a discrete object, and discusses the problems that appear when the approach is applied to 3D objects. Section IV shows two examples to illustrate the approach, and, finally, Section V presents the conclusions of the work.

II. Background

A. Assumptions

The following assumptions are considered in this work. There is a frictional punctual contact between each finger and the object, with friction being modeled according to Coulomb's law. The object surface is discretized with a large enough set Ω of points $\boldsymbol{p}_i$, whose positions are described by one or two parameters u for 2D or 3D objects, respectively. The normal direction $\hat{\boldsymbol{n}}_i$ pointing toward the interior of the object at $\boldsymbol{p}_i$ is known. Besides, each point is connected with a set of neighboring points forming a mesh of interconnected points on the object surface.

B. Independent Contact Regions and Non-Graspable Regions

Independent contact regions and non-graspable regions are defined on the object surface in such way that a finger located in each ICR or NGR, independently of the exact finger position, always ensures that an FC or non-FC grasp is obtained, respectively. Fig. 1 shows an example of an FC grasp on a discretized ellipse and in the wrench space; it also shows the ICRs for each one of the 4 grasping points on the ellipse. 3,920 different FC grasps can be obtained from the possible combinations of finger positions inside the ICRs.

Fig. 2 shows a 4-finger non-FC grasp for the ellipse in the wrench space and on the ellipse boundary. For a non-FC grasp, different sets of non-graspable regions can be computed [13]; each set is called an NGRH. For the example in Fig. 2, NGRH$_1$ and NGRH$_2$ allow 44,100 and 2,313,441 different non-FC grasps, respectively. Algorithms to compute ICRs and NGRHs have been already presented in a previous work [13].

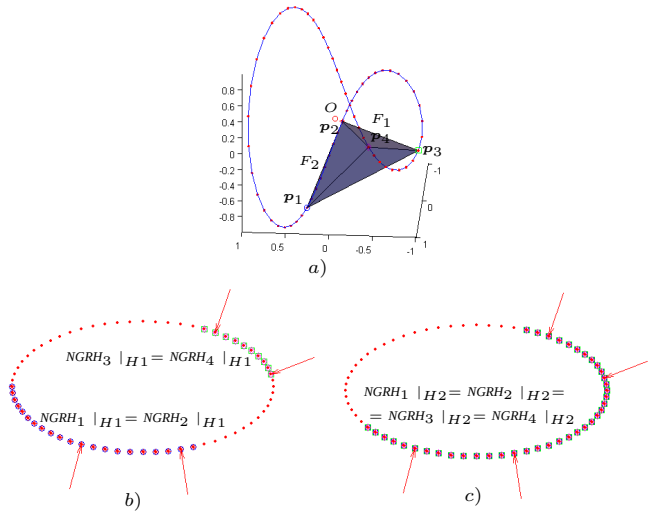

Fig. 2. Sets of NGRs for a discretized ellipse: a) Non-FC grasp in the wrench space, b) First set of NGRs (NGRH$_1$); b) Second set of NGRs (NGRH$_2$).

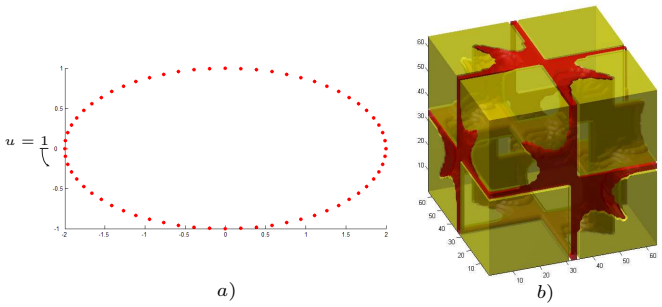

Fig. 3. Grasp space for a 2D object with 3 frictional contacts: a) Discretized ellipse; b) Grasp space.

C. Grasp space

An n-finger grasp G is described by the set of parameters u_i that define the positions of the fingers on the grasped object surface, i.e. $G = \{u_1, \ldots, u_p\}$, with $p = n$ for 2D objects and $p = 2n$ for 3D objects. The p-dimensional space representing the position of the possible contact points defined by $u_1, \ldots, u_p$ is called the grasp space $\mathcal{G}$.

The grasp space $\mathcal{G}$ is divided into two complementary subsets: the FC space, formed by the points that represent FC grasps, and the non-FC space, whose points represent non-FC grasps. Fig. 3 shows the grasp space $\mathcal{G}$ for an ellipse discretized with 64 points using 3 frictional fingers (i.e. each point of $\mathcal{G}$ defines 3 contact points on the ellipse). The grasp space $\mathcal{G}$ contains $64^3 = 262,144$ grasps, with 12.1% being FC grasps and 87.9% being non-FC grasps, as shown in Fig. 3b with dark and light colors, respectively. This ellipse will be used in this work to illustrate the proposed approach for solving the regrasp problem.

The grasp space $\mathcal{G}$ has some symmetries, as any grasp $G = \{u_1, \ldots, u_p\}$ accounts for K different grasps, where $K = n!$ is the total number of possible permutations of the fingers on the object while keeping the same contact points,

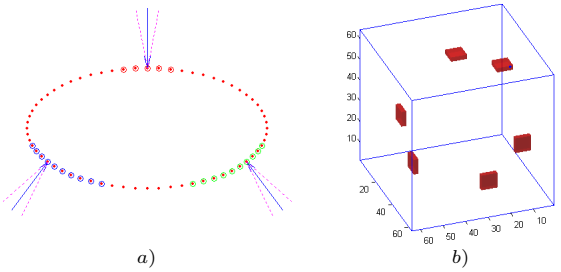

Fig. 4. ICRs: a) An FC grasp with the corresponding ICRs on the discretized ellipse; b) Correspondent BI regions in the grasp space.

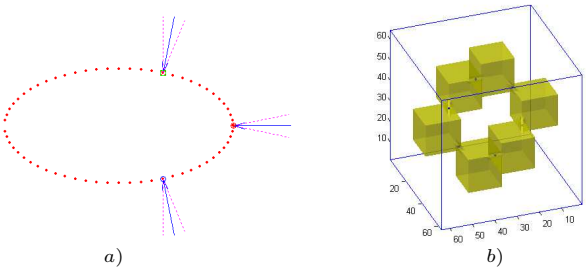

Fig. 5. NGRHs: a) A non-FC grasp on the discretized ellipse; b) Correspondent BN regions in the grasp space.

i.e. the fingers may change their positions with all the other fingers without changing the contact points and the obtained grasps on the object are the same (as long as there are no specific assignments of the fingers to the contact points).

The ICRs computed on the boundary of the object correspond to an axis-aligned region in the grasp space, hereafter called BI region, which encloses a number of FC grasps. Due to the symmetry described above, the ICRs computed for one grasp are actually mapped to K axis-aligned regions BI in the grasp space, as shown in Fig. 4. A 3-finger grasp is used in the example to allow a graphical representation of the grasp space, which is 3-dimensional for this case.

The NGRHs also correspond to axis-aligned regions, hereafter called BN regions, that enclose a number of non-FC grasps in the grasp space. Thanks to the symmetry of $\mathcal{G}$, the NGRHs computed for a non-FC grasp are mapped to K axis-aligned regions BN, as shown in Fig. 5b for the non-FC grasp shown in Fig. 5a. Note that both the BI and BN regions are stored by using $2p$ parameters, representing the lower and upper limit of the correspondent box along each axis of $\mathcal{G}$.

III. Regrasp Planning

A. Parametrization

Since a grasp is a combination of n discrete points on the object, the p-dimensional grasp space is discretized to represent the potential grasps. The simplest way to obtain such discretization is the creation of an uniform grid based on an ordered numeration of the points that represent the boundary of the object. This numeration is straightforward for 2D objects, as their boundary is a closed curve and the discrete points on the boundary can be uniquely identified by a single parameter u. The parameter that identifies each point

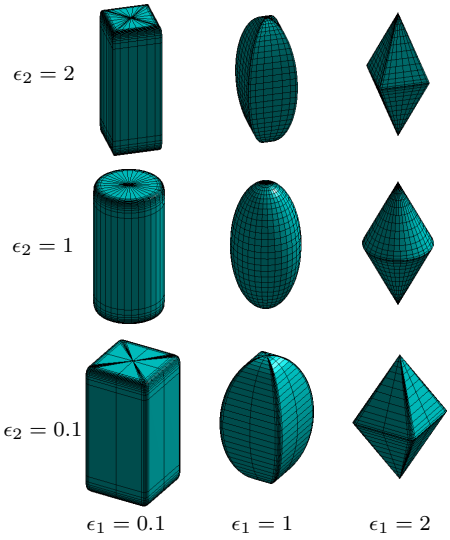

Fig. 6. Superellipsoids with different shapes given by the parameters ϵ_1 and ϵ_2 (the size parameters a_1, a_2 and a_3 are kept constant).

is an ordinal number that starts from an arbitrary origin where $u = 1$. This numbering procedure produces a well-ordered set, where the neighboring relations (required to compute the ICRs) are obtained very easily: a point with code u_i has two neighbors, u_{i+1} and u_{i-1}. The exceptions are the points $u = 1$ and $u = N$, neighbors in the physical space but not in the ordered set; this can be solved readily by identifying them as neighbors in the parameter space.

A similar parametrization for 3D objects can be obtained by using a special subset of 3D objects called superquadrics, which are mainly used for solid object modeling and scene representation, and to generate 3D solids from an unstructured cloud of points [14]. There are four kind of superquadric surfaces: superellipsoids, supertoroids, and superhyperboloids of one or two sheets, but only the first two define closed surfaces. Superellipsoids are defined as

$$s(\phi,\eta) = \begin{pmatrix} a_1 \cos^{\epsilon_1}\phi \cos^{\epsilon_2}\eta \\ a_2 \cos^{\epsilon_1}\phi \sin^{\epsilon_2}\eta \\ a_3 \sin^{\epsilon_1}\phi \end{pmatrix}, \quad \begin{matrix} -\pi/2 \le \phi \le \pi/2 \\ -\pi \le \eta < \pi \end{matrix} \tag{1}$$

with the parameters a_1, a_2 and a_3 being the size factors along the three coordinate axes, and ϵ_1, ϵ_2 the parameters that determine the shape of the superellipsoid. The exponentiation with ϵ_i is a signed power function defined as $\cos^{\epsilon_i}\phi = \text{sign}(\cos\phi)\,|(\cos\phi)|^{\epsilon_i}$. In order to get convex shapes, the parameters should be $\epsilon_1 \le 2$, $\epsilon_2 \le 2$ (Fig. 6).

Supertoroids are defined as

$$s(\phi,\eta) = \begin{pmatrix} a_1(a_4 + \cos^{\epsilon_1}\phi) \cos^{\epsilon_2}\eta \\ a_2(a_4 + \cos^{\epsilon_1}\phi) \sin^{\epsilon_2}\eta \\ a_3 \sin^{\epsilon_1}\phi \end{pmatrix}, \quad \begin{matrix} -\pi \le \phi < \pi \\ -\pi \le \eta < \pi \end{matrix} \tag{2}$$

where a_4 is an additional parameter related to the radius of the supertoroid (Fig. 7).

The normal direction to the object surface for both the

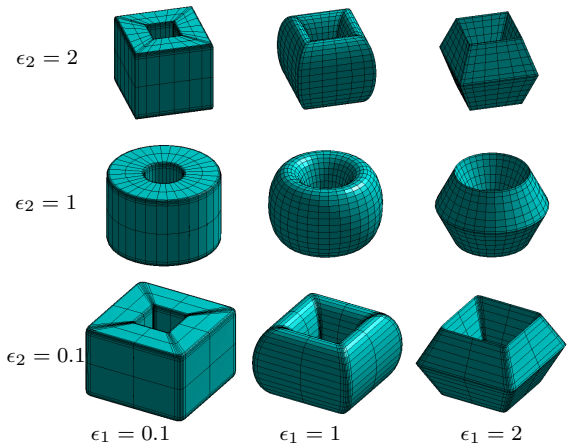

Fig. 7. Supertoroids with different shapes given by the parameters ϵ_1 and ϵ_2 (the size parameters a_1, a_2, a_3 and a_4 are kept constant).

superellipsoids and supertoroids is given by

$$s(\phi,\eta) = \begin{pmatrix} \frac{1}{a_1}\cos^{2-\epsilon_1}\phi\cos^{2-\epsilon_2}\eta \\ \frac{1}{a_2}\cos^{2-\epsilon_1}\phi\sin^{2-\epsilon_2}\eta \\ \frac{1}{a_3}\sin^{2-\epsilon_1}\phi \end{pmatrix} \quad (3)$$

For these objects, a point on the superquadric surface is considered to have 4 neighbors (up, down, left, and right neighboring points).

B. Algorithm

The regrasp planning problem is formulated as follows: given an initial and a final FC grasps, G_i and G_f respectively, find a trajectory for each finger contact on the object surface that allows the grasp change while continuously keeping the FC property (i.e. ensuring the resistance to any external disturbance appeared during the regrasp process). The sequence of movements corresponds to a path between the points G_i and G_f in the grasp space $\mathcal{G}$ such that all the points in the path are FC grasps.

The regrasp algorithm is based on the computation of ICRs and NGRHs (which define BI or BN regions in the grasp space) to find the regrasp path. Each BI region is represented as a node in an auxiliary regrasp graph (hereafter called RG). Two nodes representing a pair of contiguous BIs in the grasp space are connected with an arc between them in RG. First, the regrasp algorithm computes the BI regions for G_i and G_f. Note that although several BIs are identified for a single FC grasp G, only one BI contains the point G_i or G_f; these BIs are identified as BI_i and BI_f, respectively. Then, the algorithm takes a sample grasp from $\mathcal{G}$, identifies whether it is FC or not, and builds the corresponding region around it. If it is an FC grasp, then the computed region BI is added to RG, and the contiguity relations for the new BI are tested, i.e. new arcs are added to RG between the nodes representing BIs that intersect each other. If the sample grasp is a non-FC grasp, then all the possible grasps included in the corresponding region BN are labeled as non-FC grasps. The iterative procedure goes on until a continuous path is obtained in the regrasp graph RG (or, equivalently, in the grasp space $\mathcal{G}$). The algorithm is as follows.

Algorithm: Regrasp planning

1) For the initial and final grasps, G_i and G_f respectively:
   a) Compute the ICRs that define the regions BI_i and BI_f
   b) Label all the possible grasps inside the BIs as FC grasps
   c) Represent BI_i and BI_f as nodes in a regrasp graph RG
2) Get a sample grasp G_s from $\mathcal{G}$
3) If G_s has already been labeled, go to Step 2
4) If G_s is FC then
   a) Compute the ICRs that define the region BI_s
   b) Label all the possible grasps inside BI_s as FC grasps
   c) Represent BI_s as a new node in RG
   d) Determine the contiguity relations between BI_s and the existing BIs in RG

   Else (i.e. if G_s is non-FC)
   a) Compute the NGRHs that define the region BN_s
   b) Label all the possible grasps inside BN_s as non-FC grasps
5) If there is a path in RG between BI_i and BI_f then
   a) Find the intersections in the grasp space between each pair of contiguous BI regions
   b) Find the centroids of each intersection zone, G_c
   c) Compute the regrasp trajectory from G_i to G_f passing through all the G_cs

   Else, go to Step 2

Fig. 8 illustrates the algorithm for a hypothetical 2-dimensional grasp space. It is considered that the order of parameters u in the grasps G_i and G_f respects a predefined assignment of fingers.

The sampling method used in Step 2 is based on a structured grid that identifies each cell of $\mathcal{G}$ with a unique numerical code [15]. The sample selection follows a deterministic sequence that ensures the completeness of the method (a complete deterministic sequence covers the whole grasp space).

Step 5 checks whether there is a path between the initial and final grasp; this is performed using a Dijkstra algorithm applied to the regrasp graph RG. For a quicker convergence of the algorithm (to a solution or to completely cover $\mathcal{G}$ and decide that there is no solution at all), Step 5 could be executed every certain number of generated samples. When there is a path between the initial and final grasps, obtained as a sequence of BIs in RG, the regrasp trajectory must be computed in $\mathcal{G}$; different criteria can be used to compute such trajectory (for instance, minimizing the number of finger movements). The regrasp trajectory that this planner provides is based on one-at-a-time movement of the fingers, i.e. the trajectory of the regrasp sequence in the grasp space follows the direction of the axis (Fig. 8).

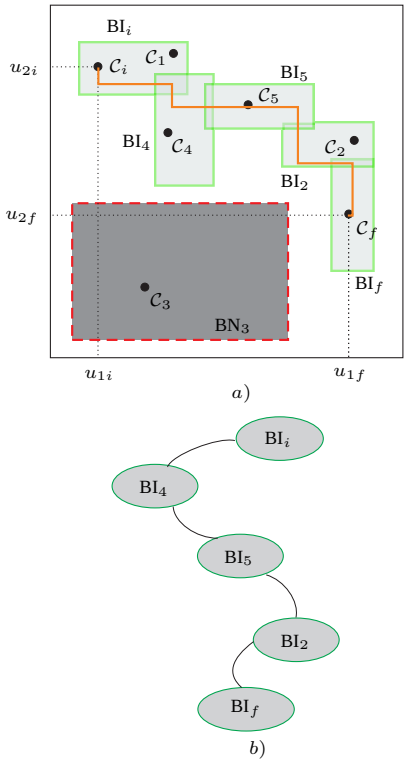

Fig. 8. Regrasp planning: a) Hypothetical 2-dimensional grasp space with the initial and final grasps and BIs, four sampled FC grasps ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_4$, $\mathcal{C}_5$) and one non-FC grasp ($\mathcal{C}_3$); b) Regrasp graph RG with the contiguity relations between the nodes that represent the BIs.

IV. Examples

To illustrate the proposed approach, the algorithm was implemented in Matlab on a Pentium IV 3.2 GHz PC. The first example shows the regrasp planning process for a 3-finger frictional grasp on a discretized ellipse, which provides further insight into the algorithm behavior. The second example shows a regrasp computation for a 3D object.

A. Example 1

The first example uses the discretized ellipse previously presented in Fig. 3. The FC grasp space explored while searching for the regrasp sequence is shown in Fig. 9a. Fig. 9b shows the regrasp path inside the contiguous BIs that connect the initial and final grasp. As a reference, Fig. 9c shows the whole FC grasp space.

In 20 trials of regrasp computations between the same initial and final grasp, the averaged total time ellapsed to get the regrasp sequence was 17.1 s, and 101 evaluations of ICRs and 61 evaluations of NGRHs were required. Table I shows these results and the averaged results for the exploration of the whole grasp space using the deterministic sampling process [15]. Note that the regrasp computation provides a feasible trajectory in a very short time when compared to the time required for the total exploration of the grasp space. Fig. 10 illustrates the regrasp sequence between the initial and final grasp for this example.

TABLE I

Results for the regrasp computation in Example 1

Parameter	Regrasp computation	Total grasp space
time $[s]$	17.1	2,871
Number of samples	3,115	262,144
ICRs computed	101	566
% of the FC space	66.7	100
NGRHs computed	61	313
% of the non-FC space	97.7	100
% of the grasp space	94.0	100

B. Example 2

The second example computes a regrasp trajectory for a superellipsoid, shown in Fig. 11, with 4 frictional fingers. Fig. 12 shows the final grasp, with the trajectories of the fingers on the object surface. The total computational time required to solve the regrasp problem is 8,875 s.

V. Conclusions

This paper has presented an approach to generate a regrasp trajectory in the grasp space for discretized objects with any number of fingers. The proposed method is based on the concepts of independent contact regions (ICRs) and sets of non-graspable regions (NGRHs). The approach is based on a discretization of the grasp space, and a sampling method that provides grasp samples, used to build regions of the FC or non-FC space. With a low number of samples, a large portion of the grasp space is covered. The search of a regrasp path is converted into a graph search in a regrasp graph, that keeps trace of the contiguity relations between different portions of the FC space.

The algorithm presented in the paper has been implemented and some application examples are given. Although the procedures are valid for 3D objects with high-dimensional grasp spaces, its application requires an efficient way to save the data, because the grasp space has a high dimensionality (for instance it is 8-dimensional for a 4-finger frictional grasp on a 3D object). The development of an efficient storage method to speed up the application of the proposed algorithm to 3D discrete objects is an interesting line of future work.

References

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

[2] L. Han and J. Trinkle, "Object reorientation with finger gaiting," in *Proc. 2nd IMACS Int. Conf. Computational Engineering in Systems Applications*, 1998.

[3] J. Hong, G. Lafferriere, B. Mishra, and X. Tan, "Fine manipulation with multifinger hands," in *Proc. IEEE Int. Conf. Robotics and Automation - ICRA*, 1990, pp. 1568–1573.

[4] L. Han and J. Trinkle, "The instantaneous kinematics of manipulation," in *Proc. IEEE Int. Conf. Robotics and Automation - ICRA*, 1998, pp. 1944–1949.

[5] E. Noohi, H. Moradi, and M. Ahmadabadi, "Manipulation using wheeled tips benefits and challenges," in *Proc. 39th International Symposium on Robotics*, 2008, pp. 442–447.

[6] L. Han, Y. Guan, Z. Li, Q. Shi, and J. Trinkle, "Dextrous manipulation with rolling contacts," in *Proc. IEEE Int. Conf. Robotics and Automation - ICRA*, 1997, pp. 992–997.

[7] A. Bicchi, "Hands for dexterous manipulation and robust grasping: A difficult road towards simplicity," *IEEE Trans. Robotics and Automation*, vol. 16, no. 6, pp. 652–662, 2000.

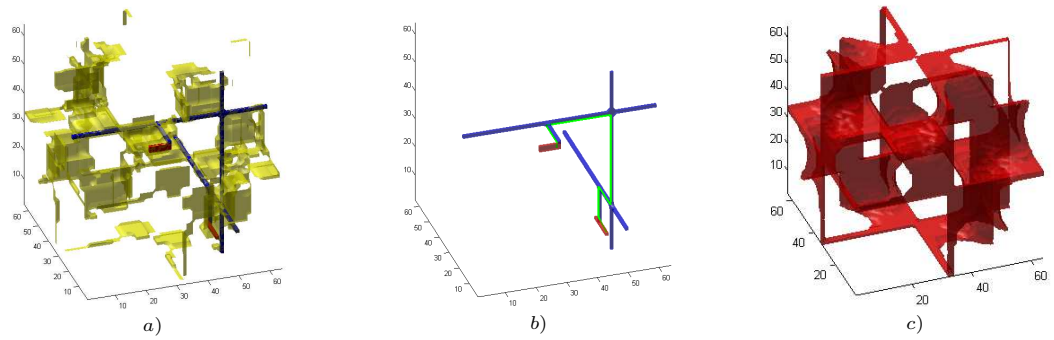

Fig. 9. Regrasp planning for Example 1: a) FC space explored while searching the regrasp sequence; b) Contiguous BIs that provide the regrasp path between the initial and final grasp; c) Total FC space for the example.

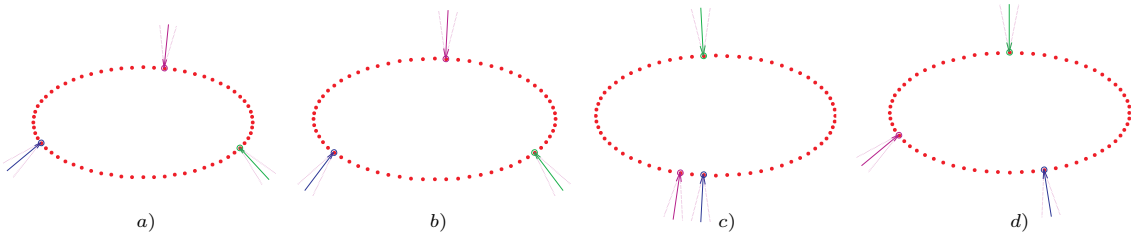

Fig. 10. Sequence of grasps for Example 1: a) Initial grasp G_i; b) and c) Intermediate grasps; d) Final grasp G_f.

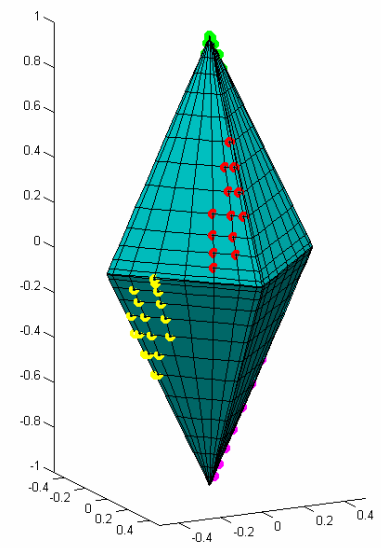

Fig. 11. Superellipsoid for Example 2: Initial grasp G_i, with its corresponding ICRs. Each color represents points within a given ICR.

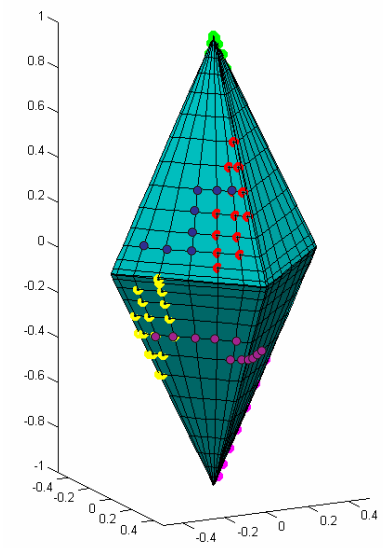

Fig. 12. Regrasp trajectory for Example 2: Final grasp G_f, with the finger trajectories on the object surface. Each color represents the trajectory for one finger.

[8] D. Brock, "Enhancing the dexterity of a robot hand using controlled slip," in *Proc. IEEE Int. Conf. Robotics and Automation - ICRA*, 1988, pp. 249–251.

[9] A. Cole, P. Hsu, and S. Sastry, "Dynamic control of sliding by robot hands for regrasping," *IEEE Trans. Robotics and Automation*, vol. 8, no. 1, pp. 42–52, 1992.

[10] T. Phoka, P. Pipattanasomporn, N. Niparnan, and A. Sudsang, "Regrasp planning of four-fingered hand for a parallel grasp of a polygonal object," in *Proc. IEEE Int. Conf. Robotics and Automation - ICRA*, 2005, pp. 791–796.

[11] M. Cherif and K. Gupta, "3D in-hand manipulation planning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems - IROS*, 1998, pp. 146–151.

[12] T. Phoka, N. Niparnan, and A. Sudsang, "Hierarchical simplification for 5-fingered 3D regrasp planning on triangular mesh objects," in *Proc. IEEE Int. Conf. Robotics and Biomimetics*, 2007, pp. 571–576.

[13] M. Roa, R. Suárez, and J. Rosell, "Grasp space generation using sampling and computation of independent regions," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems - IROS*, 2008, pp. 2258–2263.

[14] L. Chevalier, F. Jaillet, and A. Baskurt, "Segmentation and superquadric modeling of 3D objects," *J. Winter School of Computer Graphics, WSCG*, vol. 11, no. 1, 2003.

[15] M. Roa, R. Suárez, and J. Rosell, "Influence of the sampling strategy on the incremental generation of the grasp space," in *Proc. 11th Int. Conf. on Climbing and Walking Robots - CLAWAR*, 2008, pp. 812–819.