Regrasp Planning for a 5-Fingered Hand Manipulating a Polyhedron

Thanathorn Phoka and Attawith Sudsang
Department of Computer Engineering,
Chulalongkorn University, Bangkok 10330, Thailand
{phoka,attawith}@cp.eng.chula.ac.th

***Abstract*— This paper addresses the problem of a 5-fingered hand manipulating a polyhedron. In particular, assuming frictional point contacts, we present an approach for computing a sequence of finger repositioning that allows the hand to switch from one grasping configuration to another while maintaining a force-closure grasp of the polyhedron during the entire process. The proposed approach captures ability to switch from one grasp to another in a graph structure, allowing regrasp planning to be reduced to a graph search problem. The proposed approach is implemented and preliminary results are presented.**

I. Introduction

Ability of a multi-fingered hand to change its grasping configuration while maintaining force-closure stability is clearly desirable for in-hand object manipulation tasks. A typical method for a hand to change its grasping configuration is by repositioning its fingers – a process generally known as regrasping or finger gaiting. This paper addresses the regrasp planning problem of a 5-fingered hand manipulating a polyhedron. In particular, assuming hard fingers with frictional point contact, we propose a technique for computing a sequence of finger repositioning that transforms an initial grasp into a desired one while keeping the manipulated object in a force-closure grasp during the entire process.

Extensive review on grasping and dexterous manipulation can be found in [1], [9]. The readers are referred to [11] for necessary background on the classical concept of force closure. Finger gait was suggested in [7] as a method for replacing finger about to reach its workspace limit during object manipulation. A general framework for planning dexterous manipulation using finger rolling and gaiting was proposed in [5] with an example of a 3-finger hand manipulating a sphere. An approach using a branch-and-bound search with nonlinear programming was proposed in [10] for regrasp planning of a polygon.

II. Overview

A hand can change its grasping configuration by executing a sequence of finger repositioning. In this paper, we classify finger repositioning into two different types: *finger switching* and *finger aligning*. Given an object initially in a force-closure grasp, a finger switching is a process that a free finger[1] is appropriately placed on the grasped object to form another force-closure grasp, so that the finger in the original grasp that is not involved in the new grasp can be lifted off allowing the hand to change from the original grasping configuration to the new one while maintaining that the object stays in a force-closure grasp during the finger swapping. Of course, the fingers in the original grasp are sometimes not in the positions where a finger switching can take place. They have to be appropriately repositioned to allow the intended finger switching. This situation calls for a finger aligning. Finger aligning refers to a process that grasping fingers, without breaking contact with the object, adjust their contact positions while maintaining a force-closure grasp during the entire process. With frictional contact assumed, this may be implemented using finger rolling or sliding.

Our approach to in-hand manipulation amounts to computing a sequence of appropriate finger switching and finger aligning. To achieve this, we introduce a structure called *switching graph*. Each vertex of the graph represents a set of force-closure grasps. For every pair of grasps from the same vertex, there always exists a finger aligning between the grasps. Two vertices are adjacent if there exists a finger switching between a grasp from one vertex and another grasp from the other. This property allows regrasp planning to be formulated as a graph search. Note that although finger kinematics and other relevant constraints are not initially taken into account, different search strategies and policies may be later integrated to generate regrasping sequences that meet additional requirement.

Clearly, a finger switch is possible when there is at least one free finger. Since any three-dimensional object has a frictional force-closure grasp with four fingers [8], for a hand equipped with 5 fingers, our approach therefore considers only 4-finger force-closure grasps. As mentioned in [11], 4-finger force-closure grasps can be classified into 3 different types: concurrent, pencil and regulus grasps. The approach for regrasp planning presented in this paper handles only concurrent grasps. Although the other two types are not accounted for, experimental results in Section IV shows that our approach successfully finds a large number of different grasping configurations and possible finger switching for several test objects.

III. Switching Graph

The switching graph concept is based on the idea that a set of concurrent grasps can be represented by a point in three-dimensional space. This representation

[1] a finger currently not in contact with the object

will be explained in detail in Section III-A. We will also show how contiguous points representing concurrent grasps can be grouped together to form a cell. A vertex of a switching graph represents a set of grasps by establishing an association with a cell. The way we form a cell allows us to compute (1) a finger aligning between two grasps within the same cell and (2) a finger switching between a grasp in one cell and another grasp in another cell (associated with a neighboring vertex). This computation will be discussed in Section III-D.

A. Representing Concurrent Grasps

As mentioned earlier, a grasp is geometrically defined by the positions of the fingers on the object's faces. Assuming polygonal object model, the position of a contact point can be defined by specifying an ordered pair representing coordinates of the point on the corresponding grasped face. With four grasping fingers, this amounts to using eight parameters to uniquely define a grasp (with the four grasped faces already chosen). However, using Proposition 1 from [12], we can define a set of concurrent grasps with only three parameters. This proposition requires the following definition.

Definition 1: Let $V_i, i = 1, 2, 3, 4$ be the four cones of half-angle θ centered on vector $\boldsymbol{v}_i$. We say that the four vectors $\boldsymbol{v}_i, i = 1, 2, 3, 4$ θ-positively span $\Re^3$ if any combination of vectors $\boldsymbol{v}'_i \in V_i, i = 1, 2, 3, 4$ positively span $\Re^3$.

To tell whether four given vectors θ-positively span $\Re^3$, we may verify that for any triple $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ of these vectors, the cones V_1, V_2, V_3 of half-angle θ centered on $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_3$ lie in the interior of the same half-space and the cone $-V_4$ of half-angle θ centered on the direction opposite to the fourth vector $\boldsymbol{v}_4$ lies in the interior of the intersection of the trihedra formed by all triples of vectors belonging to V_1, V_2 and V_3. Geometrically, it can be shown that the intersection of the trihedra is essentially the trihedron bounded by three planes, each of which passes through the origin and touches two of the three cones V_1, V_2, V_3 while separating them into different half-space from the remaining cone.

Proposition 1: A sufficient condition for four fingers to form a force-closure grasp is that the four internal normals at the four contact points θ-positively span $\Re^3$ and there exists a point $\boldsymbol{x}_0$ such that the inverted friction cones at this point (Fig. 1) intersect the four contact faces.

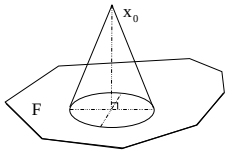


Fig. 1. Inverted friction cone of face F at $\boldsymbol{x}_0$

Note that each point $\boldsymbol{x}_0$ satisfying Proposition 1 yields four *independent contact regions* where fingers can be placed independently while achieving concurrent grasp: these regions are simply the intersection of the inverted friction cones in $\boldsymbol{x}_0$ with the contact faces. As we will discuss in Section III-C, locally adjusting contact points within independent contact regions is a means for finger aligning to move from one grasping configuration to another one belonging to the same vertex in the switching graph.

We are now ready to discuss how a vertex in a switching graph represents a set of grasps. A vertex of a switching graph represents a set of concurrent grasps by having an association with a set of all points $\boldsymbol{x}_0$ satisfying Proposition 1 for a given combination of four faces. Since an inverted friction cone at $\boldsymbol{x}_0$ intersect the corresponding face when $\boldsymbol{x}_0$ lies in the volume defined by the union of all double-sided friction cones at every point on the face (Fig. 2(a)), the set of all $\boldsymbol{x}_0$ satisfying Proposition 1 can be obtained from the intersection of the four volumes each of which is the union of all double-sided friction cones on each face. In the following definition, we name the union and the intersection for future references.

Definition 2: The union of all double-sided friction cones at every point on face F_a will be called the union volume for the face and will be denoted by U_a.

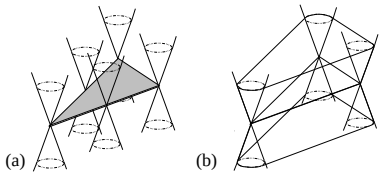


Fig. 2. Union volume: (a) construction, and (b) its shape (see text)

Definition 3: The volume containing all points $\boldsymbol{x}_0$ satisfying Proposition 1 for a given combination of four faces F_i, F_j, F_k and F_l where $i \neq j \neq k \neq l$ will be called the focus cell for the faces and will be denoted by $C_{i,j,k,l}$.

Before proceeding to the next section, it is helpful to discuss briefly about the shape of the union volume and the focus cell. Let us begin by considering an example of a triangular face with its union volume. As shown in Fig. 2(b), the union volume is composed of two symmetric parts (in mirror-like fashion): one above the face, and the other one below. Note that the union volume is an unbounded body. This is because double-sided friction cones are symmetric and unbounded. Clearly from the construction, the boundary of the union volume consists of unbounded rectangular and conic patches (at rounded corners). With conic parts involved, quadric surfaces are needed to exactly describe the union volume's boundary. This requirement implies that to construct a focus cell by intersecting four union volumes, univariate polynomial equations of degree upto 8 are to be solved (e.g., to obtain curved edges from intersecting two conic patches and to obtain a vertex from intersecting three conic patches).

A typical technique to avoid this complexity is to give up some exactness by approximating conic parts of the boundaries of union volumes with multi-facet pyramids. This approximation will allow a union volume to be described as a polyhedron and, in turn, a focus cell can readily be obtained using an algorithm for intersecting polyhedra [6]. This approximation scheme should be used with caution because when the number of facets of the approximating pyramids is too large, the resulting polyhedron will have so many faces that intersecting polyhedra might be slower than using algorithms for computing intersection of quadric surfaces [6]. This issue on construction of focus cells will become important as we discuss how to build a switching graph in Section III-D. Before then, let us explain how focus cells are related to finger switching and finger aligning operations.

B. Finger Switching

Let us consider two focus cells $C_{a,b,c,d}$ and $C_{a,b,c,e}$ such that $C_{a,b,c,d} \cap C_{a,b,c,e} \neq \emptyset$. Let $\boldsymbol{q}$ be a point in $C_{a,b,c,d} \cap C_{a,b,c,e}$. Clearly, $\boldsymbol{q}$ defines two sets of concurrent grasps: one for the combination of faces F_a, F_b, F_c, F_d and the other for the combination of faces F_a, F_b, F_c, F_e. Let us suppose that the fingers 1,2,3 and 4 are respectively on faces F_a, F_b, F_c and F_d and forming one of the concurrent grasps defined by $\boldsymbol{q}$. It is easy to see that the hand can switch to another concurrent grasp (represented by $\boldsymbol{q}$) on faces F_a, F_b, F_c and F_e by placing finger 5 on any point in the intersection between face F_e and its inverted friction cone at $\boldsymbol{q}$ (because $\boldsymbol{q} \in C_{a,b,c,d} \cap C_{a,b,c,e}$). Once finger 5 is on F_e, finger 4 can leave face F_d resulting in a switching from a concurrent grasp on F_a, F_b, F_c, F_d by fingers 1,2,3,4 to another concurrent grasp on F_a, F_b, F_c, F_e by fingers 1,2,3,5. This finger repositioning sequence enables us to plan finger switching by identifying intersection between two focus cells having one different grasped face.

C. Finger Aligning

Clearly, a finger switching cannot occur between two grasps whose corresponding focus cells do not overlap. For example, let us consider focus cells in Fig. 3. Obviously, because $C_{a,b,c,d} \cap C_{a,b,c,f} = \emptyset$, it is not possible to switch directly from a grasp on faces F_a, F_b, F_c, F_d to another grasp on faces F_a, F_b, F_c, F_f using a finger switching discussed in the previous section. However, suppose the current grasp on faces F_a, F_b, F_c, F_d is defined by $\boldsymbol{q}_1$, a finger switching can be performed to switch to another grasp on faces F_a, F_b, F_c, F_e ($\boldsymbol{q}_1$ is in both $C_{a,b,c,d}$ and $C_{a,b,c,e}$) and somehow if the hand can adjust the fingers to change from the grasp defined by $\boldsymbol{q}_1$ to a grasp defined by $\boldsymbol{q}_2$ (which could be any point in $C_{a,b,c,d} \cap C_{a,b,c,e}$), another finger switching at $\boldsymbol{q}_2$ can be applied to switch to a grasp on faces F_a, F_b, F_c, F_f as desired.

In fact, changing grasping configuration within the same focus cell is the process we referred to as finger aligning.

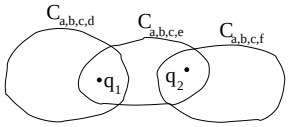


Fig. 3. Moving between non-overlapping cells

This process can be accomplished by taking advantage of the idea that force closure can be maintained during finger sliding, finger rolling (see [5], [2] on how to apply rolling in dexterous manipulation), or finger switching within an independent contact region. To illustrate, let us consider Fig. 4 showing configuration points $\boldsymbol{q}$ and $\boldsymbol{q}'$ in the same focus cell $C_{a,b,c,d}$. The inverted friction cones of the four grasped faces at $\boldsymbol{q}$ intersect the faces in the four independent contact regions R_a, R_b, R_c and R_d and likewise the inverted friction cones at $\boldsymbol{q}'$ intersect the four grasped faces in R'_a, R'_b, R'_c and R'_d. Suppose that the four fingers are at $\boldsymbol{x}_a \in R_a, \boldsymbol{x}_b \in R_b, \boldsymbol{x}_c \in R_c$ and $\boldsymbol{x}_d \in R_d$. This can be represented by $\boldsymbol{q}$. To move from $\boldsymbol{q}$ to $\boldsymbol{q}'$, we move the four fingers from $\boldsymbol{x}_i$ to $\boldsymbol{x}'_i \in R_i \cap R'_i (i = a, b, c, d)$. It is sufficient to ensure force closure during the fingers' motion by maintaining that the fingers are in the independent contact regions of $\boldsymbol{q}$ during the entire process. This can be done by rolling or sliding the fingers on the grasped faces from $\boldsymbol{x}_i$ to $\boldsymbol{x}'_i (i = a, b, c, d)$. Instead of rolling or sliding, it is also possible to apply finger switching within each independent contact region by placing a free finger at $\boldsymbol{x}'_i$ and lifting off the finger at $\boldsymbol{x}_i$. Because there is only one free finger during a concurrent grasp, this kind of finger switching can be performed in one independent region at a time.

By continuity, for any point in a focus cell, there exists a neighborhood for which the four independent contact regions of the point intersect the four independent contact regions of every point in the neighborhood. That is, there always exists a finger repositioning sequence to move between any pair of configuration points in the same focus cell.

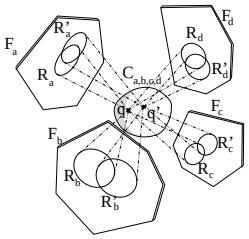


Fig. 4. Moving within a focus cell

D. Computing a Switching Graph

To construct a switching graph, all of its vertices and edges need to be found. To identify all vertices of a switching graph, we compute all focus cells and to identify all edges, we compute all pairs of overlapping focus cells with three common grasped faces.

Computing all focus cells requires identifying all combinations of four faces with concurrent grasps satisfying Proposition 1. Instead of enumeratively checking all combinations, the number of candidates can be significantly reduced by considering only those combinations whose internal normals positively span the plane. Our technique for generating such combinations is based on the fact that when three normals are given, the fourth one must lie strictly inside the trihedron formed by the inverses of the three given normals in order that the four normals positively span $\Re^3$ (otherwise, they would be in the same half space).

It is also important that every combination of four normals is listed without any repetition. This is essentially the problem of generating all k-subsets (i.e., subsets with k members) of a given set with n members. A simple solution for this problem is to assign a totally ordered relation to all members of the set and list every k-subset in the form of a k-tuple for which each element (except the last one) precedes the next one according to the assigned order. Applying this method to our problem, each unit normal is reparameterized using an ordered pair of two angles (α, β) where $\alpha \in [0, 2\pi]$ is the angle between the x-axis and the projection of the normal on the x-y plane, and $\beta \in [0, \pi]$ is the angle between the z-axis and the normal. With this parameterization, a sorted order can be imposed by defining that a normal $\boldsymbol{n}_a = (\alpha_a, \beta_a)$ precedes a normal $\boldsymbol{n}_b = (\alpha_b, \beta_b)$ when $\beta_a < \beta_b$, or when $\alpha_a < \alpha_b$ in the case that $\beta_a = \beta_b$. For clarity, let us present pseudocode of the resulting algorithm. In the pseudocode, the n sorted unit normals are stored in the array $normal[1..n]$ with corresponding indices to faces in the array $faceId[1..n]$ and variable $upwardIndex$ containing the index to the last normal in the array with angle β smaller than $\pi/2$ (i.e., all normal vectors in $normal[1..upwardIndex]$ points in the upward direction).

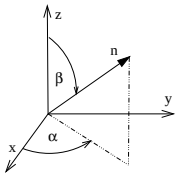


Fig. 5. Parameterization of a unit normal vector

```
1: for i = 1 to upperIndex do
2:     n1 = normal[i]; f1 = faceId[i]
3:     for j = i + 1 to n − 2 do
4:         n2 = normal[j]; f2 = faceId[j]
5:         for k = j + 1 to n − 1 do
6:             n3 = normal[k]; f3 = faceId[k]
7:             m = max(k + 1, upperIndex + 1)
8:             Compute all normal vectors in normal[m..n]
               that is contained in the trihedron formed by
               −n1, −n2 and −n3
```

From line 1 of the above pseudocode, we can see that every first normal is chosen such that it has to point upward (with $\beta < \pi/2$). This is because choosing a first normal with angle $\beta \geq \pi/2$ would result in having all four normals with $\beta \geq \pi/2$ which means that they are all in the same lower half-space and therefore cannot positively span $\Re^3$. The same reason is applied in line 7 to allow a fourth normal to point downward only (with $\beta > \pi/2$), otherwise all four normals would be pointing upward and lie in the same upper half-space. Line 7 also incorporates the fact that, to generate different combinations without repetition, a fourth normal must be after the third normal according to the sorted order (i.e., with greater β than that of the third normal). The following paragraphs describe how line 8 can be implemented.

Because a unit normal can be thought of as a point on the unit sphere, and a trihedron formed by three unit vectors intersects the unit sphere in a triangular region (bounded by three sections of great circles), all normal vectors contained in the trihedron are therefore those vectors corresponding to the points lying inside this triangular region. If we can somehow map the surface of the sphere onto the plane, a range searching algorithm can be applied to find the desired normals.

In fact, we have already mentioned one such mapping. Recall that we parameterize every unit normal using an ordered pair of angles (α, β). This allows each normal vector to be mapped to a point in the α-β plane. The triangular region on the sphere mentioned above will be mapped to a planar region bounded by three vertices and three curved edges (Fig. 6). Each edge of the triangular region on the sphere may contain the highest or the lowest point of the corresponding great circle. By considering the mapping of these points and the three vertices of the region, it is easy to show that the smallest isothetic box[2] covering the planar region can be drawn by considering only the range of the coordinates of all these points. With this bounding box, we can then apply an orthogonal range searching algorithm [4] to find all the points contained in the box (note that before applying the range searching, the bounding box may need to be clipped to ensure that the angle β of a fourth normal is greater than that of the third normal). For each point obtained, its corresponding normal is checked with the three previously chosen normals to tell whether they can positively span $\Re^3$. By using range trees [4] to perform orthogonal range searching, the overall algorithm runs in $O(n^3 \log^2 n)$.

In constructing the bounding box described above, it is important to take into account the nature of the mapping from the spherical to the cartesian coordinates. In particular, when the triangular region on the sphere intersects the arc defined by $\alpha = 0$ (Fig. 7) , two bounding boxes are to be constructed to reflect that the arcs $\alpha = 0$ and

[2]a rectangle with its sides parallel to the axes

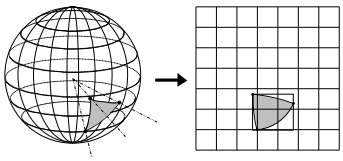


Fig. 6. Mapping from the spherical to cartesian coordinates

$\alpha = 2\pi$ coincide.

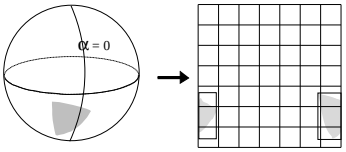


Fig. 7. Two bounding boxes are needed when the triangular region cross over the arc $\alpha = 0$

Another case is when the triangular region covers the "south pole" (bottommost point) of the sphere. When this occurs, the normals corresponding to the three vertices of the triangular region have their normal projection on the x-y plane positively spanning the plane. This should be handled by constructing a bounding box covering the entire range of α (from 0 to 2π).

Every combination of faces found by the algorithm outlined above is also tested whether the corresponding four normal vectors θ-positively span $\Re^3$. This can be done in constant time by following geometric description given after Definition 1. Now that we know all combinations of faces whose normal vectors θ-positively span $\Re^3$, the next step is then to find which ones of these combinations yield focus cells, and which pairs of these focus cells overlap. In this paper, we investigate two different approaches for this task: direct geometric computation, and random sampling.

Direct Geometric Computation: To test whether a combination of four faces F_a, F_b, F_c, F_d (with normals θ-positively spanning $\Re^3$) forms a focus cell, intersection of the union volumes U_a, U_b, U_c, U_d is computed. The intersection, if not empty, is the resulting focus cell $C_{a,b,c,d}$. To find overlapping focus cells corresponding to an edge in the switching graph, all pairs of resulting focus cells with one different face are again checked for intersection.

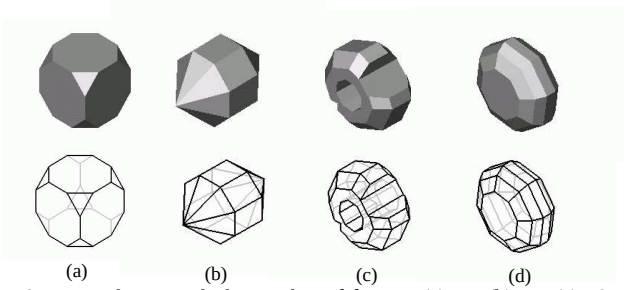
Random Sampling: The underlying idea is that all the points contained in a focus cell are contained in all the union volumes of the faces that form the cell. This implies that if we have found such points, we have an evidence showing that the corresponding focus cell exists. Likewise, we can conclude that two focus cells overlap if we can find some points that are contained in both cells. Following this simple idea, instead of directly computing intersection to explicitly obtain focus cells, a number of points in three dimensions are randomly selected, each of which is then tested to list all faces whose union volumes can contain the point. The resulting list of faces is then scanned for matching with combinations of four faces whose normal vectors θ-positively span $\Re^3$ (obtained from the algorithm previously described). A matching indicates a focus cell found, and any pair of matching with the two corresponding combinations having one different face indicates that the corresponding focus cells overlap and an edge in the switching graph linking the two cells exists.

It is clear that the completeness of the switching graph generated using this approach depends heavily on the number of sampled points and the region in $\Re^3$ where the sampling takes place. To define the sampling region that is guaranteed to cover all focus cells without actually computing them is still an open problem. Our implementation shown in the next section relies on an *ad hoc* alternative by defining the sampling region to be the cube obtained from enlarging the smallest isothetic cube that can contain the object four times about its center. Although a complete switching graph cannot be guaranteed, experimental results show large number of vertices and edges are found within a fraction of the time used by the direct geometric approach.

IV. Implementation and Results

We have implemented the regrasp planning based on the switching graph concept described in the paper. The program is written in C++ using ACIS library [3] for geometric computation. All run times are measured on a PC with a 1 GHz CPU.

Some test polyhedra with 14, 24, 40 and 42 faces are shown in Fig. 8. Test results in Table I show the number of focus cells found, the number of links found and the run time for each object in Fig. 8 when using the direct intersection approach to build the switching graph. Test results from random sampling approach with 1,000, 5,000, 10,000, and 20,000 sampling points are shown in Tables II-V correspondingly (these are average numbers over 20 runs for each test object). Without guaranteeing a complete switching graph, the random sampling approach appears to generate a large portion of the graph when spending only small amount of time compared with the direct intersection approach. In particular, for the object in Fig. 8(b), the random sampling approach is 20 times faster and also producing the complete switching graph. It is of course difficult to give a general statement from only few examples, however we feel that the random sampling approach is very promising especially in its ability to quickly produce a sketch of the switching graph. Fig. 9(a)-(i) show snapshots of a short sequence of finger repositioning generated from the program to transform the initial grasp in Fig. 9(a) into the target grasp in Fig. 9(i). With a switching graph already computed, the program takes less than 0.1 second to generate the sequence.



(a) (b) (c) (d)

Fig. 8. Test objects with the number of faces = (a) 14, (b) 24, (c) 40 and (d) 42

TABLE I

RESULTS FROM DIRECT INTERSECTION APPROACH FOR EACH TEST OBJECT IN FIG. 8

Fig.	# focus cells	# links	time (s)
8(a)	22	24	6.2
8(b)	177	384	99.8
8(c)	684	2043	294.0
8(d)	830	2434	207.2

V. CONCLUSIONS AND FUTURE WORKS

We have presented a method for regrasp planning of a polyhedron by a 5-fingered hand based on the concept of the switching graph and demonstrated an efficient implementation of the proposed approach. Our interest for future works include the comprehensive study of the efficiency of random sampling approach. We are also interested in addition of the other two types of 4-fingered force closure grasps (i.e., pencil and regulus grasps) to our regrasp planning.

VI. REFERENCES

[1] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *IEEE Int. Conf. on Robotics and Automation*, 2000.

[2] A. Bicchi and A. Marigo. Rolling contacts and dexterous manipulation. In *IEEE Int. Conf. on Robotics and Automation*, 2000.

[3] J. Corney and T. Lim. *3D Modeling with ACIS*. Saxe-Coburg Publications, 2002.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 1997.

TABLE II

RESULTS FROM RANDOM SAMPLING APPROACH FOR EACH TEST OBJECT IN FIG. 8 WITH 1,000 SAMPLING POINTS

Fig.	# focus cells	# links	time (s)
8(a)	13	14	0.4
8(b)	111	218	0.8
8(c)	57	92	3.9
8(d)	111	236	5.1

TABLE III

RESULTS FROM RANDOM SAMPLING APPROACH FOR EACH TEST OBJECT IN FIG. 8 WITH 5,000 SAMPLING POINTS

Fig.	# focus cells	# links	time (s)
8(a)	20	24	1.6
8(b)	158	304	2.3
8(c)	184	269	6.3
8(d)	319	701	8.8

TABLE IV

RESULTS FROM RANDOM SAMPLING APPROACH FOR EACH TEST OBJECT IN FIG. 8 WITH 10,000 SAMPLING POINTS

Fig.	# focus cells	# links	time (s)
8(a)	22	24	2.9
8(b)	177	384	4.4
8(c)	482	1071	9.9
8(d)	371	854	13.3

TABLE V

RESULTS FROM RANDOM SAMPLING APPROACH FOR EACH TEST OBJECT IN FIG. 8 WITH 20,000 SAMPLING POINTS

Fig.	# focus cells	# links	time (s)
8(a)	22	24	5.0
8(b)	177	384	7.8
8(c)	560	1346	16.7
8(d)	553	1392	20.3

[5] L. Han and J.C. Trinkle. Dextrous manipulation by rolling and finger gaiting. In *IEEE Int. Conf. on Robotics and Automation*, 1998.

[6] Christoph M. Hoffman. *Geometric and Solid Modeling*. Morgan Kaufmann, San Mateo, California, 1989.

[7] J.W. Hong, G. Lafferriere, B. Mishra, and X.L. Tang. Fine manipulation with multifinger hand. In *IEEE Int. Conf. on Robotics and Automation*, 1990.

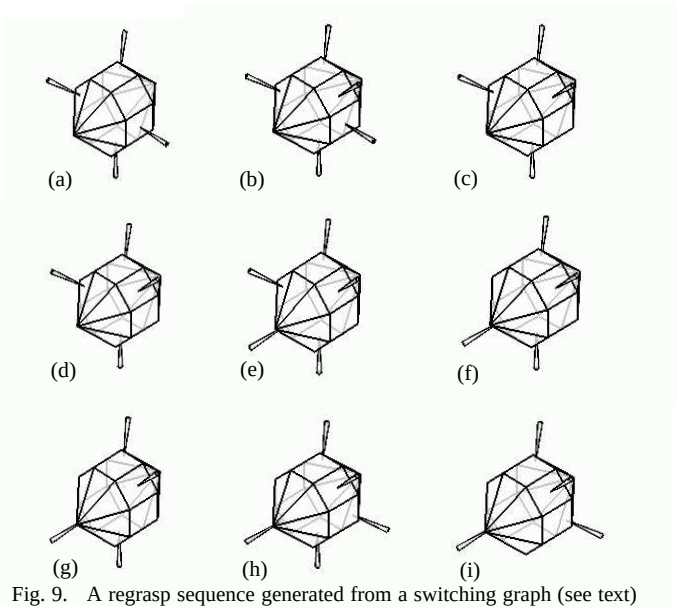
[8] X. Markenscoff, L. Ni, and C.H. Papadimitriou. The geometry of grasping. *International Journal of Robotics Research*, 9(1):61–74, February 1990.

[9] A. Okamura, N. Smaby, and M. Cutkosky. An overview of dexterous manipulation. In *IEEE Int. Conf. on Robotics and Automation*, 2000.

[10] T. Omata and K. Nagata. Planning reorientation of an object with a multifingered hand. In *IEEE Int. Conf. on Robotics and Automation*, 1994.

[11] J. Ponce, S. Sullivan, A. Sudsang, J-D. Boissonnat, and J-P. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research*, 16(1):11–35, February 1997.

[12] A. Sudsang and J. Ponce. New techniques for computing four-finger force-closure grasps of polyhedral objects. In *IEEE Int. Conf. on Robotics and Automation*, 1995.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Fig. 9. A regrasp sequence generated from a switching graph (see text)