The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Contact Point Clustering Approach for 5-Fingered Regrasp Planning

Thanathorn Phoka and Attawith Sudsang

Abstract— **We propose a heuristic approach for a regrasp planning problem. The input with a large number of discrete contact points is considered. In this setting, traditional methods of complete solution is not available. Based on wrench space information of the input, our proposed algorithm clusters the input into groups and chooses a representative contact point from each group. A global graph structure for regrasp planning is then constructed using all force closure grasps that can be formed only by representative contact points. Also described are approaches for finding a regrasping sequence from an arbitrary grasp to a grasp in the global structure. Once such regrasping sequences are found for linking the input initial and target grasps to the global graph structure, the regrasp planning problem can be solved as a graph search. The results from preliminary experiments indicate that our method can solve many problem instances efficiently.**

I. Introduction

The central idea of manipulation is about restraining the object being manipulated to have limited or no movement possibility, typically by means of grasping. To verify stability of a grasp, the force closure property is usually considered. Achieving force closure guarantees that end effectors touching the object can exert any possible force and torque to the object with complete restraint about contact points. Hence, any external disturbance to the object can be counterbalanced.

Regrasp planning is inspired by the actual behavior of human manipulation when the object goes through several different grasping configurations while being kept in the hand during the entire process. This ability is referred to as in-hand manipulation. There are several complex tasks that involve not one grasping configuration but a sequence of different grasping configurations, or it might happen that the current grasp is not suitable for the task to perform. This arises the *regrasp planning problem*, given an initial grasp and a target grasp, the goal is to compute a movement sequence of the fingers' contact points that changes the initial grasp to the target configuration while still maintaining the force closure property.

This research is financially supported in part by the Thailand Research Fund through the Royal Golden Jubilee Ph.D. program under grant No. Ph.D. 1.O.CU/49/D.1 and the 90^{th} Anniversary of Chulalongkorn University Fund through the Ratchadapiseksomphot Fund, both are greatly appreciated.

T. Phoka and A. Sudsang are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 10330, Thailand thanathorn.p@gmail.com, attawith@gmail.com

To achieve complete automation, it is vital for the robot to be able to perform sensing by itself. Acquisition of object spatial structure is usually done via range sensing devices such as a laser range finder or stereoscopic cameras. The data obtained through this method usually involve thousands of surface points. To provide a trade-off between accuracy and computational simplicity, Goldfeder *et al.* [1] applied superquadric fitting to parameterize object shapes from the point-cloud input. A grasp planning is performed in the hierarchy of superquadrics from the coarsest to finer approximations. In [2], the minimum volume bounding box approach is used to fit input data by primitive box shapes. The result bounding boxes and data points are iteratively split to yield better box approximations. A grasp planner exploits the approximations as clues to synthesize grasps on arbitrary objects.

While the grasp planning problem for discrete contact points just recently began to receive more attention, the regrasp planning problem in the same setting remains mostly unexplored. The main contribution of this paper is to present a novel method for organizing the input contact points to facilitate the regrasp planning process. Our underlying idea is to partition the input points based on their wrenches. Contact points that can exert similar wrenches are grouped together. For each group, a representative contact point is selected. Every set of representative contact points that can form a force closure grasp is then computed and referred to as a representative grasp. Our idea is motivated by a typical regrasping scenario when a supporting finger need to be placed on the object before some finger in a force closure grasp can be lifted off. Obviously, if the supporting finger can exert wrenches similar to those by the finger to be lifted off, it is likely that the object will still remain in force closure after the finger swap. The most important product of the proposed clustering strategy is the roadmap structure that leads to significant reduction of the search space. This structure is a graph that captures how representative grasps can switch among one another via finger swapping. Since each representative grasp roughly describes a different way wrenches can be aligned to form a force closure grasp, the roadmap somewhat approximates the global relationship that describes how force closure grasps can be switched among one another. Of course, an arbitrary force closure grasp may

not be a member of the roadmap. Therefore, to utilize the roadmap for regrasp planing from an arbitrary initial grasp to a target grasp, we need regrasping sequences that bring the initial and the target grasps to some grasps in the roadmap. We will present methods for computing such sequences based on the clustering information. Preliminary experimental results show that most regrasp planning problems can be solved within a few seconds or a few minutes by our approach whereas exhaustive search takes about a day or longer.

II. Background on Grasping and Regrasp Planning

A grasp is defined by a set of contact points. Each contact point can exert force and torque according to the contact model. A grasp is said to achieve force closure when any force and torque can be countered by positive combination of force and torque exerted by the contact points defining the grasp. In this work we assume hard contact with Coulomb friction model. This model indicates that the ratio between the tangential force and the normal force exerted by an end effector cannot exceed the frictional coefficient to prevent slippage. This condition constrains that the set of net forces on each end effector form a cone. Under these assumptions, it is well-established that four frictional contact points are sufficient to achieve force closure grasp on any object [3].

There exist several works on the problem of force closure analysis, i.e. asking whether a grasp achieves force closure. This problem is very important in regrasp planning since we have to determine the movement of end effectors such that the resulting grasp still maintains force closure in every step of the movement. For 3D frictional grasp, Zhu and Wang [4] proposed Q-Distance algorithm which is based on two linear programming problems. Niparnan and Sudsang [5] proposed an algorithm to filter fault positive grasps based on a necessary condition of positively span of force and torques components. In this work, we apply the algorithm of [5] for fast filtering grasps that do not satisfy the necessary condition. Grasps that pass the filter are then verified for force closure by applying the algorithm of [4].

Finger gaiting [6] or finger switching [7], on the other hand, involves placing one additional end effector on a new position and removing one of the initial end effector. Since four end effectors are sufficient to achieve force closure, we consider 5-finger grasp in this work so that one additional finger can be used to perform regrasping. Assuming that we wish to change from the grasp on point $\{p_a, p_b, p_c, p_d\}$ to $\{p_b, p_c, p_d, p_e\}$, finger switching starts from placing one additional finger on the new position (p_e in this example) and then removes the finger that is not included in the final grasping position (p_a in the example). It comes automatically that two grasps sharing three contact points can be switched to each other.

III. Regrasp Planning on Discrete Point Set

In this work, we assume that the model of the input object is described by triangular meshes. This will result in a polyhedron with a large number of triangular facets. Each facet is usually very small relative to the entire object. We use the centroid of each facet as a possible contact point. This approach is also adopted in [8]. Since we consider discrete contact points, finger rolling and finger sliding which require continuous contact motion are not permitted. Only finger switching is considered in the planning.

A. Representative-Level Roadmap

The first step of regrasp planning is to compute a roadmap of all possible grasping configurations. Since we consider 4-finger grasps, a grasping configuration consists of four contact points. A grasping configuration that satisfies force closure induces a vertex in the roadmap and the configuration is stored in this vertex. An edge joining two vertices exists when two associated grasping configurations can switch to each other. This work assumes five fingers for regrasping. Four fingers are used to securely grasp the object. The remaining finger is used for finger switching. This means that two grasping configurations that have one distinct contact point can perform finger switching which implies that there exists an edge joining the two vertices corresponding to these two grasping configurations.

Suppose there are N contact points to consider. The possible grasping configurations are as many as N^4. For each grasping configuration, there can be as many as $4N$ other grasping configurations that can be reached directly by finger switching. Consequently, size of a roadmap constructed from a large number of contact points could easily become larger than the memory space of an ordinary computer. To overcome these limitations, we propose to cluster the input contact points into groups. Instead of using all contact points, one contact point is picked from each group to be a representative contact point for constructing the representative-level roadmap. The number of groups is a user-defined variable which indicates trade off between completeness and resource used in the computation. Of course, the representative-level roadmap does not cover grasping configurations consisting of some non-representative contact points. A local planner is required to compute a path from such grasping configurations to a grasp in the representative-level roadmap.

B. Spectral Clustering for Contact Point Set

In this work, we apply spectral clustering algorithm to partition the input contact points into meaningful clusters. In spectral clustering, users are free to define how a cluster is meaningful according to their task at hand. Our proposed idea is to define meaningful clusters from similarity of wrenches by means of grasping. Recall the finger switching operation: the remaining finger is placed on the object then one finger

is lifted to change grasping configuration. To maintain force closure, a reasonable heuristic is to ensure that the chosen contact point for the remaining finger and the contact point of the finger to be lifted can produce similar wrenches.

Spectral clustering takes a contact point set as input to compute an affinity matrix which embeds similarities of every pair of contact points. The matrix is solved for eigenvectors corresponding to the k largest eigenvalues. The eigenvectors are then used to determine the clustering of contact points.

1) Affinity Matrix: Affinity matrix contains information that reflects how contact points are grouped according to their applicable forces and torques. Each pair of contact points is measured for pairwise distance. The pairwise distances of all pairs form a matrix that encodes similarity between contact points. An affinity matrix is symmetric and denoted by $A \in \Re^{N \times N}$, where $0 \leq a_{ij} \leq 1$ for all contact points i and j. Element a_{ij} encodes the likelihood that contact points i and j can be clustered into the same group. Let $\mathbf{f}_i, \mathbf{r}_i$ be a unit force perpendicular to the object's surface and the position of ith contact point w.r.t. a reference point $\mathbf{o}$. The associated torque is $t(\mathbf{f}_i, \mathbf{r}_i, \mathbf{o}) = (\mathbf{r}_i - \mathbf{o}) \times \mathbf{f}_i$. The wrench generated by $\mathbf{f}_i$ at this contact point i is therefore $w(\mathbf{f}_i, \mathbf{r}_i, \mathbf{o}) = (\mathbf{f}_i, t(\mathbf{f}_i, \mathbf{r}_i, \mathbf{o}))$. When friction is assumed, the friction cone at contact point i is approximated using an m-sided pyramid bounded by $f_{i1}, \ldots, f_{im}$. The associated wrench cone is defined by $w(\mathbf{f}_{i1}, \mathbf{r}_i, \mathbf{o}), \ldots, w(\mathbf{f}_{im}, \mathbf{r}_i, \mathbf{o})$ and called primitive wrenches.

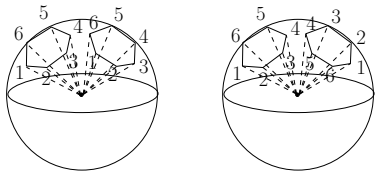
Since a force closure test in the wrench space considers only the directions of wrenches, the distance function is formulated based on measuring the angle between two wrenches from two distinct contact points, which can be calculated from their inner product. However, the torque component of a wrench depends on the choice of the origin assumed in the torque calculation. We therefore use the centroid $\mathbf{c} = (\mathbf{r}_i + \mathbf{r}_j)/2$ of the contact positions $\mathbf{r}_i$ and $\mathbf{r}_j$ as the reference origin. Since $\mathbf{r}_i, \mathbf{r}_j$ and $\mathbf{c}$ are fixed in the object frame, the toque components of these two wrenches are not affected by any rigid transformation applied to the object, i.e., independent from the object's pose.

The proposed distance between two contact points i and j considers the difference between wrenches that the two contact points can exert. Roughly speaking, we compare the geometries of the two wrench cones. Instead of integrating all differences between all pairs of wrenches from the two cones, approximation is taken by comparing only the boundaries of the linearized wrench cones, i.e., angles between the primitive wrenches from the two wrench cones are measured. Each primitive wrench of i is compared with a primitive wrench of j. Obviously, there are many ways to match pairs of primitive wrenches for comparison. We have to select one matching that is appropriate for the distance function. Since the linearization of a friction cone is in either a clockwise or counterclockwise order, the result primitive wrenches are arranged in the same order and a reasonable matching of the primitive wrenches has to preserve this order. The starting index for a wrench cone in the comparison is however not necessary the first index obtained from the linearization. The indices of the primitive wrenches of a contact point can all be shifted by an integer x while preserving the order. Without loss of generality, we apply the shifting x for the primitive wrenches of the contact point j. The angles between $w(\mathbf{f}_{i1}, \mathbf{r}_i, \mathbf{c}), \ldots, w(\mathbf{f}_{im}, \mathbf{r}_i, \mathbf{c})$ and $w(\mathbf{f}_{j(1+x \mod m)}, \mathbf{r}_j, \mathbf{c}), \ldots, w(\mathbf{f}_{j(m+x \mod m)}, \mathbf{r}_j, \mathbf{c})$ are measured pair by pair (Fig. 1). The summation of these angles defines our geometrical difference between these two wrench cones. However, this value is varied by changing x. We imitate the principal of the shortest distance between two bodies in the Euclidean space: by varying x, the minimal summation of angles is used as the pairwise distance.

The difference between two wrench cones is measured as follows

$$M(i,j) = \min \sum_{k=1}^{m} \text{angle}(w(\mathbf{f}_{ik}, \mathbf{r}_i, \mathbf{c}), w(\mathbf{f}_{j(k+x)}, \mathbf{r}_j, \mathbf{c}))$$

where $0 \leq x \leq m-1$ and $k + x \in \mathbf{Z}_m$. Since value of $M(i,j)$ depends on the number of pyramid's sides, the distance function is normalized as $d(i,j) = M(i,j)/m$.



(a) (b)

Fig. 1. Transformation distance : An example of cones in 3D where the numbers show the order of comparison (a) $x = 0$ (b) $x = 2$

The Gaussian similarity function is applied to encode pairwise measures into the affinity matrix. It is formed by an exponential function as

$$a(i,j) = e^{-d(i,j)/2\sigma^2}.$$

Clearly, $0 \leq a_{ij} \leq 1$, and contact points of which their pairwise distance is smaller have larger affinities between them. The issue of choosing σ is neglected. We simply choose σ as the average of all measures, i.e., $\sigma = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} d(i,j)$.

2) Spectral Clustering Algorithm: The spectral clustering algorithm in [9] is applied as follows.

i. Compute the affinity matrix A.
ii. Define D to be the diagonal matrix whose (i,j)-element is the sum of A's i-th row, and construct the matrix $L = D^{-1/2} A D^{-1/2}$
iii. Compute the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of L associated with the k largest eigenvalues.

iv. Construct the matrix $V = [\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_k] \in \Re^{N \times k}$ by stacking the eigenvectors in columns.
v. Form the matrix U from V by normalizing the row sums to have unit length, that is $u_{ij} = v_{ij}/(\sum_k v_{ij}^2)^{1/2}$
vi. Extracting each row of U as a point in $\Re^k$, cluster them into clusters $K_1, \dots, K_k$ by performing k-means algorithm
vii. Assign the original contact point p_i to cluster K_j if and only if row i of the matrix U is assigned to cluster K_j.

After running the clustering algorithm, we need to construct some structures for further uses in regrasp planning. Let the input contact points be stored in a table denoted by T_O. The output of the algorithm are not only the clusters but also the centers of the clusters. The contact point associated with the row vector of U that is closest to the center of K_j is chosen to be the representative contact point of cluster K_j. We construct a matrix E to store Euclidean distances between all row vectors of U and the centers of all clusters. A table T_R is constructed to store representative contact points of all clusters.

C. Constructing Representative-Level Roadmap

We are now ready to compute a roadmap for the representative contact points. All vertices are constructed by checking force closure grasp for every four contact points in T_R. Each grasping configuration satisfying force closure is assigned to a vertex. The number of vertices in a switching graph is equal to the number of grasps found in force closure checking. An edge joining two vertices, whose corresponding grasping configurations can switch to each other, can be easily computed by checking the number of common contact points between the two corresponding grasping configurations. Since 4-finger force closure grasps are considered, two grasping configurations can switch to each other when they share three common contact points. The computed representative-level roadmap is denoted by R.

D. Planning Regrasping Sequence

The regrasping sequence planning process is splitted into two level search: in representative-level roadmap and in local roadmap. In the previous section, the construction of representative-level roadmap has been described. A regrasping sequence acquired from the representative-level roadmap contains only configurations consisting of contact points in T_R which is a subset of of the original contact point set T_O. This means that traversal in the representative-level roadmap does not cover grasping configurations that consist of some contact points in $T_O \backslash T_R$. Therefore, for arbitrary initial and goal grasps, the regrasp planner has to find regrasping sequences that link both grasps to some grasps in the representative-level roadmap.

The planner performs Algorithm 1 which exploits a heuristic of similarity in a cluster to determine which vertex in the representative-level roadmap the initial grasp should be switched to. Let g be the initial grasping configuration which consists of contact points p_a, p_b, p_c and p_d. Let us denote by $p_{a'}, p_{b'}, p_{c'}$ and $p_{d'}$ the representative contact points of the clusters that respectively contain p_a, p_b, p_c and p_d. Also denote by g' the grasping configuration consisting of $p_{a'}, p_{b'}, p_{c'}$ and $p_{d'}$. If g' does not achieve force closure, i.e. the vertex defining g' is not in the representative-level roadmap R, the planner then performs Algorithm 2. Otherwise, a regrasping sequence from g to g' is planned as follows.

For each contact point p_i of g' ($i = a', \dots, d'$), all contact points in T_O that are in the same cluster as p_i are copied to the set S_i. We then compute all possible 4-finger force closure grasps such that the first, second, third and forth contact points are picked from $S_{a'}, S_{b'}, S_{c'}$ and $S_{d'}$, respectively. A local roadmap graph is then constructed such that each of its vertices represents each force closure grasp mentioned above, and each of its edges joins two vertices representing two grasps with three common contact points (finger switching is possible). With a local roadmap, any graph search can be used to retrieve a path from the vertex representing g' to the vertex representing g. If no such path can be found, the planner invokes Algorithm 2.

Algorithm 1

```
Determine p_a', p_b', p_c', p_d' and g'
if g' ∉ R then
    Algorithm 2
else
    Determine S_a', S_b', S_c', S_d'
    L = ConstructRoadmap(S_a', S_b', S_c', S_d')
    if path = FindPath(L, g, g') then
        return path
    else
        Algorithm 2
    end if
end if
```

Algorithm 2 again exploits the information of clusters. It is invoked when g' does not achieve force closure or Algorithm 1 fails to find a regrasping sequence from g to g'. The underlying idea of this algorithm is to relocate one contact point of the given grasping configuration from its cluster to another cluster that its representative contact point can form a force closure grasp with the representative contact points of the other unchanged clusters. A local planning, is then performed among one changed cluster and three unchanged clusters, which guarantees that g and the new representative grasp are force closure. However, this method perturbs the local properties of the changed clusters. To minimize the effect of this transfer and to maximize the use of local property, this algorithm aims to transfer one

contact point in g to its second nearest cluster. The procedure *FindNearestCluster* finds vertices in R that the associated grasping configuration f' has one distinct contact point from g'. Let $p_{x'}$ and $p_{y'}$ be the distinct representative contact point of g' and f'. The associated contact point of $p_{x'}$ is denoted by p_x. We query the matrix E for the distance between p_x and the center of cluster $K_{y'}$ of $p_{y'}$. All grasping configurations in R having one distinct contact point from g' are used to query the distances. There are many possible pairs of a contact point and a cluster that the contact point will be moved to. We again select the pair that induces the shortest distance between them for locality reason. Now we redefine some notations to understand the pseudocode. Let the selected contact point be p_x, its cluster be $K_{x'}$, its second nearest cluster be $K_{y'}$ and the grasping configuration possessing the cluster $K_{y'}$ be f'. This procedure returns the variable X which consists of p_x, $K_{x'}$, $K_{y'}$ and f'. If X is determined, we then temporarily add p_x into $K_{y'}$. Let f' consist of p_q, p_r, p_s, p_t. We perform a local roadmap construction and find a path between g and f' which both are members of the local roadmap.

Algorithm 2

```
Determine p_{a'}, p_{b'}, p_{c'}, p_{d'}
X = FindNearestCluster(p_{a'}, p_{b'}, p_{c'}, p_{d'})
if X is not determined then
    Algorithm 3
else
    Add p_x into K_{y'}
    Determine S_q, S_r, S_s, S_t
    L = ConstructRoadmap(S_q, S_r, S_s, S_t)
    if path = FindPath(L, g, f') then
        Remove p_x from K_{y'}
        return path
    else
        Remove p_x from K_{y'}
        Algorithm 3
    end if
end if
```

If finding a path from g to f' still does not achieve, Algorithm 3 is applied by updating the representative-level roadmap. The contact points p_a, p_b, p_c, p_d are now considered as representative contact points. Let T_G be a set of the contact points $\{p_a, p_b, p_c, p_d\}$. All grasping configurations partially consisting of some contact points in T_G are verified for force closure condition and reported as new vertices in R. New edges are computed among the new vertices and between the new vertices and the recent vertices of R to complete updating R. Clearly, g is associated with a vertex in R. Therefore, a path from g to any grasp in R can be computed by a graph search.

In conclusion, the overall algorithm firstly clusters the input contact points. Then a representative-level roadmap R is constructed from the representative contact points. To query a regrasping sequence from an initial grasping configuration g to a goal grasping configuration h, they have to be linked with R by using Algorithm 1 and 2. These algorithms find a path between g to a vertex in R and so on for h. If the algorithms fail to report a path, Algorithm 3 is performed by adding g or h into R. Finally, a graph search is then applied to find a path connecting two grasps in R.

Algorithm 3

```
T_G = {p_a, p_b, p_c, p_d}
T_A = T_R ∪ T_G
L = ConstructRoadmap(T_A, T_A, T_A, T_G)
Update R by adding L and linking L to R
```

IV. Experiments and Results

The test objects are shown in Fig. 2. They are modeled with about 500 triangular meshes. All objects are clustered into 50 groups for regrasp planning. All run times are measured on a PC with a 2.4 GHz CPU.

(a) (b) (c) (d) (e) (f)

Fig. 2. Test objects

Table I shows the results from the construction of representative-level roadmaps. The results present for each test object the number of force closure grasp vertices of the resulting representative-level roadmap, time spent in the construction and the number of connected components of the roadmap.

The results of local planning are shown in Table II. For each object, we sample 10,000 force closure grasps. For Algorithm 1, each sampled grasp is verified whether the associated representative contact points form a force closure grasp. The numbers of force closure grasps found are shown in percent. For Algorithm 2, the procedure *FindNearestCluster* is executed for each sampled grasp. A grasp passes the verification if the variable X is determined. Then 30 grasps are randomly picked from the sampled grasps that pass the verification for each algorithm. Each grasp is planned for a regrasping sequence that joins it to the representative-level roadmap. The results list the numbers of grasps (out of 30) for which such sequence are found. To measure the efficiency of Algorithm 3, 30 sampled grasps that do not pass the verifications of Algorithm 1 and 2 are randomly picked. We then perform Algorithm 3 for each of them and verify whether it connects with part of the representative-level roadmap that exists before the update by Algorithm

3. The results show the low passing rate of the verification by Algorithm 1 for some objects. Most grasps that pass the verification however can be connected to the representative-level roadmap. Algorithm 2 appears to be more effective than Algorithm 1. The passing rates for the verification of Algorithm 2 are significantly higher than those of Algorithm 1, (with slightly fewer regrasping sequences found). However, Algorithm 1 is still needed because of its considerably higher verification speed. Likewise, although applying Algorithm 3 can connect all grasps to the representative-level roadmap, it takes much longer to update the representative-level roadmap than to run both Algorithms 1 and 2 (the running times are not shown here).

Table III presents the result from querying regrasping sequences. We randomly choose 30 pairs of force closure grasps to serve as the initial and target grasps for each query. Although the number of connected components of the representative-level roadmaps are quite large for some objects, successful regrasping sequences can be found for most pairs of grasps. This implies that the majority of grasps lie in the same connected component. Minimum, maximum and average querying times are also shown in this table. An example of a regrasping sequence is presented in Fig. 3.

TABLE I

RESULT OF REPRESENTATIVE-LEVEL ROADMAP CONSTRUCTION

Fig.	#vertex	#CC	time(s)
(a)	25160	1	38.98
(b)	6602	5	9.95
(c)	2148	13	4.02
(d)	2571	15	4.80
(e)	2372	21	4.38
(f)	19317	1	28.85

TABLE II

RESULT OF LOCAL PLANNING

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	35.18	30	91.21	30	30
(b)	34.29	30	87.92	30	30
(c)	10.02	29	81.15	29	30
(d)	10.77	30	81.55	29	30
(e)	8.94	30	77.87	28	30
(f)	45.67	30	91.20	30	30

V. CONCLUSION

We have proposed a new approach to solve the regrasp planning problem. Our approach provides trade off between completeness and the resource used in the computation. It clusters the input using spectral clustering. The representative contact points from all clusters are used to constrain possible search space. The underlying idea is similar to that of the classical motion planning problem. We construct a partial solution, called a representative-level roadmap. This allows the original problem to be divided into three parts: planning regrasping sequence from the starting grasp to the roadmap, planning regrasping sequence in the roadmap and, finally, planning regrasping sequence from the roadmap to the target grasp. Since the set of representative contact points contains much fewer contact points, solving the problem on the roadmap is much less complex. Nevertheless, this is achieved at the cost of completeness.

TABLE III

RESULT OF PLANNING REGRASPING SEQUENCES

Fig.	#connect	time(s)		
		min	max	average
(a)	30	7.80	109.49	80.31
(b)	30	11.55	43.74	25.77
(c)	29	2.14	11.07	5.42
(d)	30	2.17	14.51	6.99
(e)	29	1.84	9.22	4.91
(f)	30	31.37	145.32	80.07

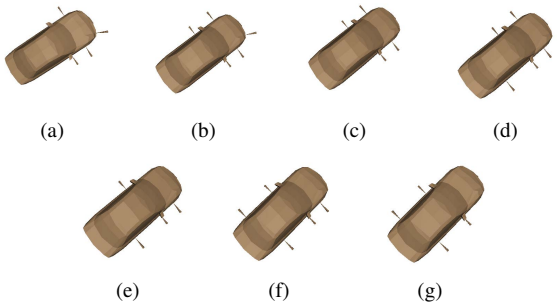


Fig. 3. A regrasping sequence for the object in Fig. 2(b)

REFERENCES

[1] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *IEEE Int. Conf. on Robotics and Automation*, April 2007.

[2] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping," in *IEEE Int. Conf. on Robotics and Automation*, 2008.

[3] J. Ponce, A. Sudsang, S. Sullivan, B. Faverjon, J.-D. Boissonnat, and J.-P. Merlet, "On computing four-finger equilibrium and force-closure grasps of polyhedral objects," *International Journal of Robotics Research*, vol. 16, no. 1, pp. 11–35, February 1997.

[4] X. Zhu and J. Wang, "Synthesis of force-closure grasps on 3-d objects based on the q distance," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, p. 669, August 2003.

[5] N. Niparnan and A. Sudsang, "Positive span of force and torque components of four-fingered three-dimensional force-closure grasps."

[6] L. Han and J. Trinkle, "Dextrous manipulation by rolling and finger gaiting," in *IEEE Int. Conf. on Robotics and Automation*, 1998.

[7] A. Sudsang and T. Phoka, "Regrasp planning for a 4-fingered hand manipulating a polygon," in *IEEE Int. Conf. on Robotics and Automation*, 2003.

[8] M. Roa and R. Suarez, "Independent contact regions for frictional grasps on 3d objects," in *IEEE Int. Conf. on Robotics and Automation*, 2008.

[9] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 849–856.