Fast Computation of 4-Fingered Force-Closure Grasps from Surface Points

Nattee Niparnan and Attawith Sudsang
Department of Computer Engineering,
Chulalongkorn University, Bangkok 10330, Thailand
{nattee,attawith}@cp.eng.chula.ac.th

***Abstract*— This paper addresses the problem of computing frictional 4-fingered force-closure grasps of three dimensional objects. The proposed approach searches for force-closure grasps from a collection of sampled points on the object's surface. Unlike most other works, the approach is not limited to the objects with a certain class of shapes. It can be applied to an object in any shape since only the object's surface points and corresponding surface normals at the points are needed. The efficiency of the approach arises from a heuristic for search space pruning which is based on ability to efficiently locate regions in three dimensional space where friction cones intersect and a randomized test for checking force-closure condition. The proposed approach is implemented and preliminary results are presented.**

I. Introduction

Grasp planning plays an important role in grasping and manipulation [1], [10]. Its objective is to find a desirable grasping configuration by computing appropriate contact positions for placing the fingers on the object's surface. Typically, the classical force-closure condition is considered in the computation to ensure that the object can be held securely by the fingers [9], [8]. Most works in force-closure grasp computation require fitting the shape of the object to be grasped with a certain class of geometric models so that the computation can be performed entirely on a model. Since the information regarding the object's surface is crucial to grasp computation, it is natural to use boundary based models in describing shapes. A boundary based model of an object represents a surface enclosing the object as a set of faces [7]. Majority of research in grasp planning focuses on polyhedral models (whose all faces are planar) with an aim in deriving efficient or analytical formulation for characterizing force-closure grasps on a given set of faces [11], [13], [6]. Several techniques have been proposed and shown to be efficient in generating good grasps (according to various criteria) provided that all grasped faces are already selected. The problem of choosing appropriate grasped faces is, however, rarely studied [5]. A straightforward search of all combinations of faces is usually applied, yielding an approach with prohibitive time complexity that can handle, in practice, only objects with small number of faces.

In general, many real objects cannot be represented by a polyhedral model with small number of faces. A standard technique widely used in geometric modeling to represent a general shape (including curved objects) is to describe the surface enclosing its volume using a large number of small triangles. Unfortunately, as mentioned earlier, most works in grasp planning were not designed to handle large number of faces. Several papers demonstrated efficiency of their approaches using only simple test objects with small number of faces (usually fewer than 20). This concern was taken into account in only few papers. In [14], the problem of fixture design from a set of pre-selected frictionless contact points was addressed. Using a local greedy search with D-optimality criterion, the method seek a force-closure set of 7 fixturing locations from the given set of contact points. Operating on set of points enabled the method to handle complex objects. Recently, it was suggested in [2] that acceptable force-closure grasps could be efficiently generated using a randomized selection from a set of contact candidates. The paper also attempted to convince that the resulting grasps achieve the quality comparable with human generated grasps.

The agenda here is that grasp planning need an efficient approach for dealing with real-world complex objects. This paper presents a solution to the problem. More precisely, assuming four hard fingers with frictional point contacts, we address the problem of computing force-closure grasps of arbitrary three dimensional objects. Although the approach shares the flavors of the techniques used in [14] and [2], it has some different standpoints. Unlike both papers, our objective is to efficiently compute as many force-closure grasps as possible as opposed to a single good grasp. Grasp quality is intimately task dependent, so the users should be supplied with enough choices to make a good decision. Moreover, obtaining a number of grasps at once is directly helpful to certain tasks such as manipulation or regrasp planning where several grasps are usually considered. Conceptually, the proposed approach searches for force-closure grasps from a collection of sampled points on the object's surface. It can be applied to an object in any shape since only the surface points and corresponding surface normals at the points are needed. The efficiency of the approach arises from a heuristic for search space pruning which is based on ability to efficiently locate regions in three dimensional space where friction cones intersect and a randomized test for checking force-closure condition. Unlike the local method in [14], our heuristic search captures more global information which allows a variety of force-closure grasps to be generated. We demonstrate in preliminary experimental results that the approach is efficient. It is capable of computing hundreds of force-closure grasps of complex objects with running time below three seconds.

The remainder of the paper begins with some necessary background on grasping in Section II. The detail on how

the search for force-closure grasps is performed is given in Section III. We describe our implementation and present some preliminary results in Section IV, and then conclude the paper with conclusion and future works in Section V.

II. Background

In this section, we give some necessary background on grasping. In particular, the condition given in Proposition 1 provides the most important foundation to the derivation of our search method for finding force-closure grasps.

A hard finger in contact with some object at a point $\boldsymbol{x}$ exerts a force $\boldsymbol{f}$ with moment $\boldsymbol{x} \times \boldsymbol{f}$ with respect to the origin (but it cannot exert a pure torque). Force and moment are combined into a six dimensional zero-pitch wrench $\boldsymbol{w} = (\boldsymbol{f}, \boldsymbol{x} \times \boldsymbol{f})$. Under Coulomb friction, the set of wrenches that can be applied by the finger is:

$$W = \{(\boldsymbol{f}, \boldsymbol{x} \times \boldsymbol{f}) : \boldsymbol{f} \in F\},$$

where F denotes the friction cone at $\boldsymbol{x}$.

A d-finger grasp is defined geometrically by the position $\boldsymbol{x}_i (i = 1, ..., d)$ of the fingers on the boundary of the grasped object. We can associate with each grasp the set of wrenches $W \subset \Re^6$ that can be exerted by the fingers. If we denote by W_i the wrench set associated with the i^{th} finger, we have

$$W = \{\sum_{i=1}^{d} \boldsymbol{w}_i : \boldsymbol{w}_i \in W_i \text{ for } i = 1, ..., d\}.$$

We say that a grasp achieves *force closure* when any external wrench can be balanced by wrenches at the fingertips, i.e. when the corresponding wrench set W is equal to $\Re^6$. Because zero wrench is contained in W for a force-closure grasp, it is then clear that force closure implies equilibrium. Interestingly, it is shown in [13] that the converse of this statement is also true for non-marginal equilibrium, i.e. grasps such that the forces achieving equilibrium lie strictly inside the friction cones at the fingertips. In other word, grasps achieving equilibrium with non-zero forces for some friction coefficient achieve force closure for any strictly greater friction coefficient.

A zero-pitch wrench $\boldsymbol{w} = (\boldsymbol{f}, \boldsymbol{x} \times \boldsymbol{f})$ for the force $\boldsymbol{f}$ can be thought of as the line of action of this force and can be written in Plücker coordinates. Equilibrium therefore implies that the lines (represented as Plücker vectors) associated with the contact forces are linearly dependent. As mentioned in [13], Grassman geometry [3], which characterizes the varieties of various dimensions formed by sets of dependent lines, can be applied to yield a necessary and sufficient condition for non-planar equilibrium, namely, the contact forces must *positively span*[1] $\Re^3$ and their lines of action all intersect in a point (concurrent grasps), lie in two flat pencils having a line in common (pencil grasps), or form a regulus (regulus grasps).

[1]A set of vectors positively spans some space when any vector in the space can be written as a linear combination of the vectors in the set with positive coefficients

Although there are three types of non-coplanar 4-fingered force-closure grasps, the work in this paper is interested in computing concurrent force-closure grasps only. Proposition 1, derived directly from the above discussion, will therefore state only the condition involving concurrent grasps. Despite the omission of the other two types, results given in Section IV show that the proposed approach successfully finds a large number of force-closure grasps.

Proposition 1: A necessary and sufficient condition for four non-coplanar points to form a concurrent force-closure grasp is that: (P1) there exist four lines in the corresponding double-sided friction cones that intersect in a single point, and (P2) the vectors parallel to these four lines and pointing inward the grasped object positively span $\Re^3$.

III. Force-Closure Grasp Search

Our objective is to find as many 4-fingered frictional force-closure grasps as possible while constraining that all contacts must be selected from a set of given surface points. Since the intended number of surface points is well over 200, a straightforward brute-force test of all combinations for force-closure condition will definitely yield unacceptable performance. Very large search space calls for an aggressive pruning technique and an informed search strategy that effectively incorporates the knowledge of the force-closure condition. These features are the foundation of our approach.

The input of our proposed approach consists of a set of points on the surface of the grasped object and corresponding inward normals at these surface points. At present, the surface points are randomly sampled from a model of the object (we are currently investigating several sampling policies that may help improve the efficiency). The proposed approach is developed around a technique for pruning the search space. The underlying idea is based on condition P1 of Proposition 1 which implies that for four contact points to form a 4-fingered concurrent grasp it is necessary that the four friction cones at the points intersect. By considering only combinations that satisfy this condition, large portion of the search space can be ignored since for most general shapes, 4 friction cones at 4 arbitrary points rarely intersect.

Now that we have decided to use friction cone intersection as the pruning condition, an efficient method is needed for computing which combinations of four surface points will yield intersecting friction cones. Clearly, computing exact intersection is too time consuming to be practical. We therefore propose to use intersection in discretized space: The 3D object's space is discretized into a regular 3D grid of cubic pixels[2] (see an example in Figure 1(a)), axes of all friction cones are then drawn on the grid space in a bitmap fashion, and the desired intersection is guided by the pixels drawn over by at least four axis lines. It should be noted here that friction cones' axes are used in the computation instead of the whole friction cones

[2]The term *voxel* is sometimes preferred

themselves. This is because we want to take advantage of an efficient raster based line drawing algorithm simultaneously with the effect of discretization in approximating cones' volume. Intuitive justification behind this heuristic technique is that when axes of a set of friction cones intersect a cube, it is more likely for this cube than the others to contain the intersection of the set of friction cones. This effect is illustrated in Fig. 1(b) where we can see more intersections of friction cones in the lower left cell than the others. As we will see in the next section, this heuristic enables the algorithm to focus the search in the region where condition P1 of Proposition 1 is likely to be satisfied. Note that although our problem is in a 3D setting, intrinsic 2D illustrations will appear to help clarify discussion throughout the rest of this section. This can be done mainly because of great similarity between 2D and 3D concurrent grasps. The similarity is confirmed by a 2D version of Proposition 1 for 2D force-closure grasps given in [12].

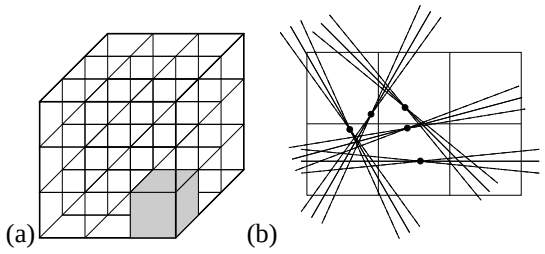

Fig. 1. (a) Cartesian partitioning of a cube into $3 \times 3 \times 3$ identical cubes, (b) Arrangement of friction cones at the given sampled points

Conceptually, the proposed approach is a search that is performed along the hierarchy of discretizing resolutions. It starts from the coarsest resolution where the sets of surface points whose friction cones likely to intersect are identified using the pruning method based on intersection in discretized space mentioned above. For each of these sets, a randomized test is applied to generate force-closure grasps. Then, only on each of these sets, the same search process repeats in a recursive fashion to discover more force-closure grasps at a finer discretizing resolution. This recursion is performed to a specified depth. A pseudo code in the following section describes the approach in detail.

A. Algorithm

In this section, we give a pseudo code of the approach outlined above and discuss how it works in detail. For clarity, let S be the set of all given surface points in the object's space. For any point $\boldsymbol{s} \in S$, let us denote by $\boldsymbol{n}(\boldsymbol{s})$ the corresponding inward normal at $\boldsymbol{s}$, by $l(\boldsymbol{s})$ the line passing through $\boldsymbol{s}$ with direction $\boldsymbol{n}(\boldsymbol{s})$, and by $f(\boldsymbol{s})$ the double sided friction cone at $\boldsymbol{s}$ with half angle θ and axis $l(\boldsymbol{s})$. In the following pseudo code, the input of the procedure `fc_grasp_search` consists of the set $T \subseteq S$ containing the surface points where the search is performed, an isothetic cube[3] C in the object's space, and a recursive depth variable d. In the following, let us list a pseudo code of the procedure `fc_grasp_search` and explain its algorithm. Note that computing force closure grasps amounts to executing the command `fc_grasp_search` $(S, C^*, 1)$ where C^* is an initial isothetic cube containing all surface points in S.

`fc_grasp_search` (T, C, d)
1: if $d > D$ then return
2: Let the cubes $C_j, j = 1, .., m^3$ be the $m \times m \times m$ partitioning of the cube C using cartesian grids of the same spacing
3: Let $T_j = \{\boldsymbol{s} | \ \boldsymbol{s} \in T$ and $l(\boldsymbol{s})$ intersects $C_j\}$
4: for all T_j with $|T_j| \geq 4$
5: if vectors in $L = \{\boldsymbol{n}(\boldsymbol{s}) | \ \boldsymbol{s} \in T_j\}$ positively span $\Re^3$ then {
6: for i = 1 to K {
7: Randomly pick a point $\boldsymbol{p}$ in the cell C_j
8: $M = \{\boldsymbol{s} | \ \boldsymbol{s} \in T_j$ and $\boldsymbol{p}$ is contained in the cone $f(\boldsymbol{s})\}$
9: find_equilibrium_grasps$(M, \boldsymbol{p})$
10: }
11: fc_grasp_search$(T_j, C_j, d+1)$
12: }

The above code begins in line 1 by checking whether the current search exceeds the depth limit. Line 2 defines $C_j, j = 1, ..., m^3$ to be m^3 identical cubes obtained from partitioning the cube C using $m \times m \times m$ grid of the same spacing. These cubes C_j's determine the discretization mentioned earlier. Line 3 computes the sets $T_j, j = 1, ..., m^3$ where each T_j contains all surface points in T whose corresponding axes of friction cones intersect C_j. This computation can be done using an algorithm for drawing lines in 3D bitmap. The detail is given in Section III-D. The main part of the search is the loop in lines 4-12. This loop considers only T_j with at least four members (line 4), i.e., corresponding C_j is drawn over by at least four axes of friction cones. This decision is a pruning heuristic that allows the search to operate on typically smaller sets T_j's instead of the set T. For each T_j considered, in line 5, the inward normals of all members of T_j are tested whether they positively span $\Re^3$. If they do not, the corresponding T_j is skipped from further processing. The detail on how a set of vectors is checked for positively spanning $\Re^3$ is given in Section III-B. It should be noted that a set of surface points whose normals do not positively span $\Re^3$ may sometimes form a force closure grasp, but the resulting grasps are usually not desirable due to their poor ability to resist forces in certain directions. A 2D version of this scenario is illustrated in Fig. 2. Fig. 2(a) shows three surface points with their friction cones and a point $\boldsymbol{p}$ in the intersection of the three cones (shaded region). As we can see in Fig. 2(b), the three inward normals (the cones' axes) do not positively span $\Re^2$, but there exists, as shown in Fig. 2(c), an equilibrium with line of forces intersecting at $\boldsymbol{p}$ and positively spanning $\Re^2$. Despite achieving force-closure, given the maximum magnitude of forces each finger can exert, this grasp is very weak in resisting forces in horizontal directions.

For each set T_j that passes the positively spanning test, the algorithm continues to the code in lines 6-11. This portion of the code is responsible for two tasks: (1) to randomly test and generate force-closure grasps at the

[3]A cube with its sides parallel to the axes

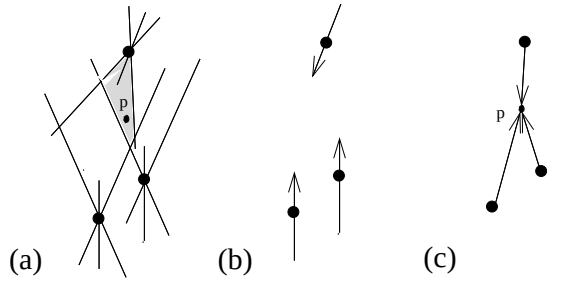

Fig. 2. Three sampled points with inward normals not positively spanning $\Re^2$ but yield a force-closure grasp: (a) three friction cones at the points, (b) the three inward normals, and (c) an equilibrium grasp indicating force-closure.

current level of discretization, and (2) to invoke a recursive search on T_j using finer discretization of C_j. The first task is handled by the K iterations in lines 6-10. The following paragraph describes its detail.

Each of the K iterations in lines 7-9 aims at finding combinations of surface points in T_j that yield non-marginal equilibria satisfying Proposition 1. The first part (lines 7-8) identifies set M, a subset of T_j with intersecting friction cones (so that intersecting lines according to condition P1 of Proposition 1 can be found). To avoid the complexity from applying a direct approach based on computing intersection of cones, a randomized method is used in approximating the set M. The method is based on a simple fact that the existence of a point that is contained in every cone in a given set implies that all the cones in the set intersect. This translates exactly to the code in lines 7-8. Note that checking whether a point is contained in a cone can be easily done by comparing the half angle of the cone with the angle between the cone's axis and the line between the point and the cone's vertex. Once the set M is obtained, the second part of the iteration assumes that the randomly selected point $\boldsymbol{p}$ is the intersection of the four lines of contact forces. This assumption is then used for listing all combinations of four surface points in M that fulfill condition P2 of Proposition 1 (line 7). These combinations thus form force-closure grasps by satisfying both P1 and P2 conditions of Proposition 1. The detail on how the combinations are enumerated is given in Section III-C.

The only remaining detail is about the code in line 11. This code invokes a recursive search `fc_grasp_search` on T_j using the cube C_j, allowing the search to be performed at a finer discretizing resolution of C_j. Varying resolution of discretization is needed in discovering as many force-closure grasps as possible. This is because some grasps can be detected only with a sufficiently large pixel cube. An analogous 2D illustration in Figure 3 demonstrates this situation. We can see that choosing a square smaller than the square C will not be able to cover the three normals and will result in a failure to detect the grasp formed by the three corresponding points. This issue affects how the size of the initial cube is set. Setting the size too small may result in missing opportunity to detect some grasps. In the implementation, the initial cube is usually defined to be much larger than the object. This does not affect the performance due to efficiency of the pruning in the first level. We are in the process of investigating the relationship between the cube's size and resulting grasps in order to derive a strategy for appropriately selecting the initial size and discretizing factor m.

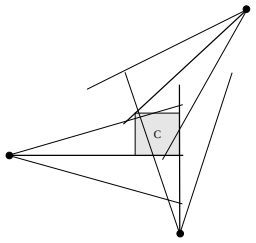

Fig. 3. A square C (shaded region) just large enough to cover the three normals (thick lines)

B. Positively Spanning $\Re^3$ Test

A set of vectors fails to positively span $\Re^3$ when they lie in the same half space bounded by a plane through the origin. This is because any vector in the other half space cannot be written as a positive linear combination of the set of vectors. To determine whether a set of vectors positively span $\Re^3$, let us consider a set of points whose coordinates are given by the vectors in the set and compute the convex hull of these points and the origin. If the origin is in the boundary of the resulting convex hull, it is clear from convexity that all the input points are in one half space bounded by a plane through the origin, which means that the given set of vectors do not positively span $\Re^3$. Note that computing a convex hull of three dimensional point set takes $O(n\log(n))$ where n is the number of input points [4].

C. Enumerating Equilibrium Combinations

Given a point $\boldsymbol{p}$ and a set M containing sampled points whose double-sided friction cones contain the point $\boldsymbol{p}$, we want to find all combinations of four points from M that satisfy condition P2 of Proposition 1 assuming that the four lines mentioned in the proposition intersect at the point $\boldsymbol{p}$.

For a sampled point $\boldsymbol{s}$ in M, let us denote by $\boldsymbol{v}(\boldsymbol{s})$ the unit vector parallel to the line joining $\boldsymbol{p}$ and $\boldsymbol{s}$, and pointing inward the grasped object (in the direction of the inward normal at $\boldsymbol{s}$, i.e. $\boldsymbol{v}(\boldsymbol{s}) \cdot \boldsymbol{n}(\boldsymbol{s}) \geq 0$). Also, let $M' = \{\boldsymbol{v}(\boldsymbol{s}) |\, s \in M\}$. With the definition, it is clear that our problem amounts to finding all combinations of four unit vectors in the set M' that can positively span $\Re^3$. Our technique for generating such combinations is based on the fact that when three vectors are given, the fourth one must lie strictly inside the trihedron formed by the inverses of the three given vectors in order that the four vectors positively span $\Re^3$ (otherwise, they would be in the same half space).

It is also important that every combination of four unit vectors 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 vector 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 vector on the x-y plane, and $\beta \in [0, \pi]$ is the angle between the z-axis and the vector. With this parameterization, a sorted order can be imposed by defining that a vector $\boldsymbol{v}_a = (\alpha_a, \beta_a)$ precedes a normal $\boldsymbol{v}_b = (\alpha_b, \beta_b)$ when $\beta_a < \beta_b$, or when $\alpha_a < \alpha_b$ in the case that $\beta_a = \beta_b$.

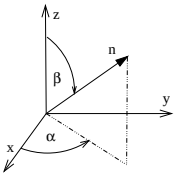

Fig. 4. Parameterization of a unit normal vector

Because a unit vector 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 unit 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 vectors.

In fact, we have already mentioned one such mapping. Recall that we parameterize every unit vector using an ordered pair of angles (α, β). This allows each unit 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. 5). 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 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 vector is greater than that of the third one). For each point obtained, its corresponding vector is checked with the three previously chosen vectors to tell whether they can positively span $\Re^3$. By using range trees [4] to perform orthogonal range searching, the overall enumeration runs in $O(n^3 \log^2 n)$ where n denotes the number of vectors in M'.

In constructing the bounding box described above, it is important to take into account the nature of the mapping

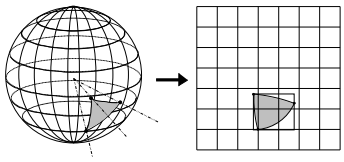

Fig. 5. Mapping from the spherical to cartesian coordinates

from the spherical to the cartesian coordinates. In particular, when the triangular region on the sphere intersects the arc defined by $\alpha = 0$ (Fig. 6) , two bounding boxes are to be constructed to reflect that the arcs $\alpha = 0$ and $\alpha = 2\pi$ coincide.

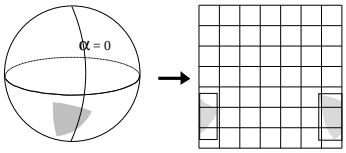

Fig. 6. Two bounding boxes are needed when the triangular region cross over the arc $\alpha = 0$

Another case is when the triangular region covers the "south pole" (bottommost point) of the sphere. When this occurs, the vectors 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).

D. Intersecting Cubes and Lines

To compute which cubic cells in a regular cartesian partitioning is penetrated by a given straight line, we refer to a well known solution of the same problem in two dimensions. In particular, in computer graphics, when a line need to be drawn on a raster display, a line drawing primitive is called to determine which pixels in a rectangular tessellation have to be turned on (see Fig. 7).

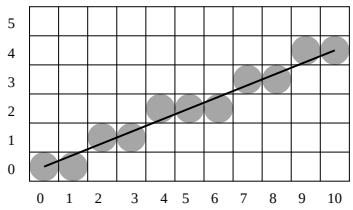

Fig. 7. Pixels drawn using Bresenham's line algorithm

Bresenham's line algorithm [7] is one of the most efficient algorithms for drawing lines in computer graphics. It scan converts a line by sampling the line at unit intervals in one coordinate and determine corresponding integer values nearest the line path for the other coordinate. Our approach is therefore to generalize Bresenham's algorithm for drawing line in three dimensions. This can be done straightforwardly by adding an outer loop together with corresponding counter variables to accommodate additional scanning coordinate. It is important to note that the original

Bresenham's line algorithm does not turn on all pixels penetrated by the line path. It decides to turn on only the pixel that contains more portion of the line at the considered scanning coordinate (e.g. pixel (1,1) in Fig. 7 is not turned on although it is in the line path). To suit our need, all penetrated pixels can be marked by modifying the algorithm to ignore this decision. Our modified line drawing algorithm has the same time complexity as the original Bresenham's, that is $O(n)$ where n denotes the number of pixels drawn. Its detail, however, cannot be described here due to space limitation.

IV. Implementation and Results

Like most heuristic based techniques, the true efficiency is best described from real implementation. We have developed a program for computing 4-finger force-closure grasps from surface points using the algorithm described in the paper. The program is written in C++ and all run times are measured on a PC with a 2.4 GHz CPU.

In the experiment, the half angle of friction cones is set to 10 degrees, the recursion is limited to 4 levels (i.e., $D = 4$), discretizing factor m is set to 10, and the number of points randomly selected in each cubic cell is set to 8 (i.e., $K = 8$). Limited space allows only two test objects to be demonstrated here: a torus (Fig. 8) and a duck doll (Fig. 9). Surface points are generated from 3D models. In each of the following figures, four illustrations are given. Illustration (a) shows 1200 sampled surface points in the initial cube, illustration (b) shows the intersection points of contact forces that yield force-closure grasps, and illustrations (c)-(d) show two examples of force-closure grasping configurations found by the program. Experiment results are shown in Table I and Table II correspondingly. The results shown are average values over 20 runs. In each table, time to generate the first solution, total run time, and the number of distinct force-closure grasps found are listed for varying number of surface points: 400, 800, 1200, 1600 and 2000.

For the case of the torus, we can see that the intersection points of contact forces (Fig. 8(b)) concentrate around the axis of symmetry. This reflects the fact that inward normals of four non-coplanar points on the torus's surface can intersect only at a point on the main axis. The scattering is the effect of friction cones. For the duck doll (Fig. 9(b)), there are dense region in the stomach and the head of the duck. Some intersection points are outside the object due to the effect of concavity. More disperse pattern in this case is due to greater variation of the direction of inward normals.

We can see that the total run times in both cases for 400 sampled points are below 3 seconds. With this amount of time, a number of force-closure grasps are already generated. It should also be noted that the first force-closure grasps in each case arrives much sooner.

V. Conclusion and Future Works

It is not unusual for an accurate representation of an object to require a model with thousands of faces. When dealing with objects with huge number of tiny faces, typical approaches for grasp planning based on intensive computation for finding grasps on selected faces become less desirable. Grasp planning need a new approach to handle real-world complex objects. We have presented a solution to this problem. In particular, we have presented an approach for computing frictional 4-fingered concurrent force-closure grasps of arbitrary 3D objects using surface points. The efficiency of the approach is confirmed by the results from a preliminary implementation. Application of the approach includes the problem of autonomous real-time grasping of unmodeled objects using range data. Of course, the process of acquiring accurate and sufficient amount of data has to be carefully considered. Issues remained to be investigated include appropriate partitioning scheme and random point selection policy.

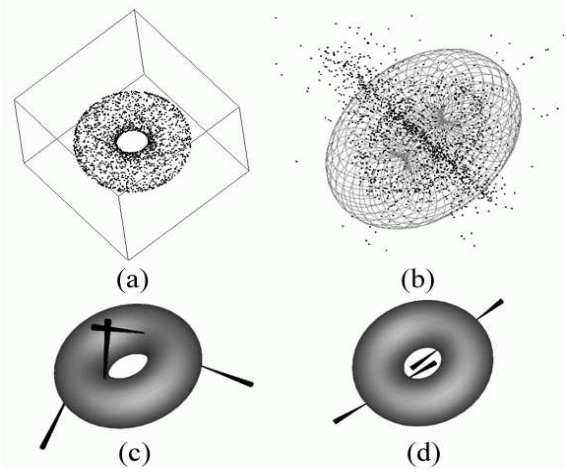

Fig. 8. A torus

TABLE I
AVERAGE RESULTS FROM 20 RUNS USING THE TORUS IN FIG. 8

#points	1st sol (s)	total time(s)	#sol
400	0.225	2.00	290
800	0.341	7.09	975
1200	0.472	14.40	1870
1600	0.615	23.06	3000
2000	0.743	32.82	4322

References

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

[2] Ch. Borst, M. Fischer, and G. Hirzinger. Grasping the dice by dicing the grasp. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003.

TABLE II
AVERAGE RESULTS FROM 20 RUNS USING THE DUCK DOLL IN FIG. 9

#points	1st sol (s)	total time(s)	#sol
400	0.261	2.65	145
800	0.385	9.46	522
1200	0.583	17.79	951
1600	0.833	39.65	1450
2000	1.146	46.49	1903

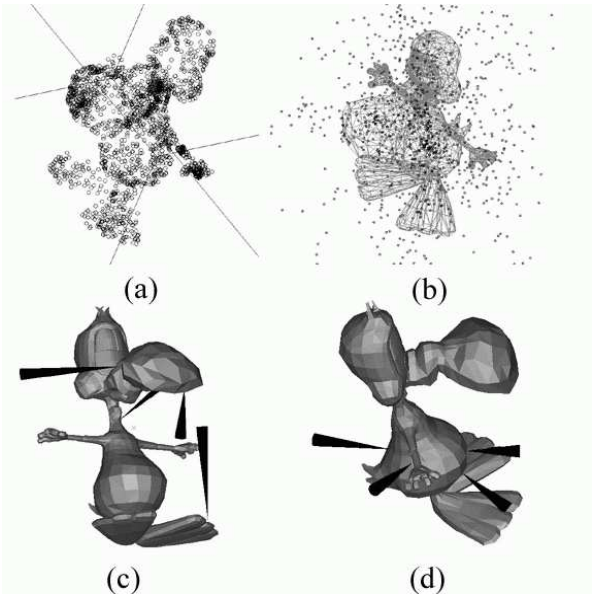

Fig. 9. A duck doll

[3] A. Dandurand. The rigidity of compound spatial grid. *Structural Topology*, 10, 1984.

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

[5] D. Ding, Y. Liu, M. Y. Wang, and S. Wang. Automatic selection of fixturing surfaces and fixturig points for polyhedral workpieces. *IEEE Transactions on Robotics and Automation*, 17(6), 2001.

[6] D. Ding, Y. Liu, and S. Wang. Computation of 3-d form-closure grasps. *IEEE Transactions on Robotics and Automation*, 17(4), 2001.

[7] F.S. Hill. *Computer Graphics Using Open GL*. Prentice Hall, 2001.

[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] V-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3):3–16, June 1988.

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

[11] T. Omata. Finger position computation for 3-dimensional equilibrium grasps. In *IEEE Int. Conf. on Robotics and Automation*, 1993.

[12] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics and Automation*, 11(6):868–881, December 1995.

[13] 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.

[14] M. Y. Wang. An optimum design for 3-d fixtures sysnthesis in a point set domain. *IEEE Transactions on Robotics and Automation*, 16(6), 2000.