Global Grasp Planning Using Triangular Meshes

Shuo Liu and Stefano Carpin

Abstract— **In this paper we present an algorithm to determine the location of contact points to obtain force closure grasps on tree dimensional objects. The shape of the object is assumed to be given by a triangle mesh – a format widely used in CAD software. Our algorithm can handle an arbitrary number of contact points and does nor require any prior information about their initial locations. Through an iterative process, contact point locations are updated aiming at improving a commonly used grasp quality metric. The process is *global* in the sense that during the process the whole surface of the object can be explored, and contact point locations can cross sharp edges that usually represent a problem for optimization algorithms relying on smooth surface representations. Extensive simulation results illustrate the performance of the proposed method, outlining strengths and directions for further research.**

I. Introduction

The ability to perform reliable grasps to restrain objects used in everyday activities continues to be one of the capabilities critically missing in today's robots. If one embraces the long term vision of developing robots capable of assisting and interacting with humans to perform everyday tasks, grasping is one of the missing building blocks. It has been argued that grasping, manipulation, and speech are among the most fundamental human abilities unparalleled by animals [2]. It should then not surprise that many problems in this area are still open.

The concepts of *force* and *form* closure have been extensively used to characterize the different types of grasps one can perform. Due to its practical importance, in this paper we focus on force closure. Two fundamental, interrelated questions arise when considering force closure. The first is how to determine a set of contact points such that by applying appropriate forces the object can be restrained. The second question is how to compare two force closure grasps. These two questions are linked because in general there exist multiple sets of contact points that can ensure force closure, and a criterion to decide which one is better is necessary. Grasp planning can be seen as a search process in the space of possible grasps aiming at identifying a solution maximizing (locally or globally) a given performance metric. With respect to grasp quality metrics, some criteria have been proposed and are commonly used and accepted by researchers and practitioners (see Section II for more details.)

S. Liu and S. Carpin are with the School of Engineering, University of California, Merced, CA, USA.

This work is supported by the National Institute of Standards and Technology under cooperative agreement 70NANB12H143. Any opinions, findings, and conclusions or recommendations expressed in these materials are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the funding agencies of the U.S. Government.

The situation is different on the algorithmic side, in the sense that while various techniques have been presented, no single approach became mainstream or widely used. Multiple reasons account for this situation. Besides the intrinsic difficulty of the problem, there is no canonical representation for the objects to be grasped, nor agreement on how much preliminary knowledge should be made available to the grasp planning system. Another problem slowing progress in the area is the lack of benchmarks and the persistent inability to contrast different solutions because most authors do not make their implementations available to the rest of the community, or when they do they often use self-developed data formats or representations. A fair experimental comparison between different grasping algorithms is to date not possible.

In this paper we present a grasp planner assuming that objects are represented using triangle meshes. Since our eventual application is robot aided manufacturing, this representation is appropriate because numerous CAD software packages use this representation in production processes. Triangle meshes have often been rejected in the past in lieu of other representations (e.g., superquadrics, or other parametric representations) allowing to use optimization methods requiring smooth surfaces. Triangle meshes, however, can approximate arbitrarily complex surfaces, including many that are poorly approximated by superquadrics or patches of other canonical surfaces. Differently from other methods requiring the specification of an initial region for the contact points, our method starts from random placements and iteratively modifies their positions while exploring various areas of the surface. The search can be guided by various performance metrics, including those commonly used for grasp quality evaluation. In this sense our method is *global* because during its search it explores the whole object, while other methods limit the search in the vicinity of the initial location (see Section II and V for more details about this aspect). The reader is however cautioned that *global* does not mean that the algorithm achieves the global optimum. In fact, its solution depends on a randomly chosen initial configuration. To lower the dependency from this random seed, multiple initial configurations are preliminarily generated and the algorithm is eventually started from the most promising one. Our search method allows to move the contact points across *sharp edges* that would normally be an impediment for other methods. Due to the randomized nature of the method, the exploration process terminates when a local optimum is reached. Our experimental validation substantiate our claims, i.e., we can produce force closure grasps by providing to the algorithm just a triangular mesh, and during the search process the position of the contact points crosses boundaries

that normally preclude exploration to other methods. We also show that our method outperforms a baseline random planner that has been used when evaluating other planners.

The paper is organized as follows. Related work is discussed in Section II. In Section III we introduce relevant notation and concepts to present our algorithm in Section IV. Results are given in Section V and conclusions are provided in Section VI.

II. Related Work

Because of its practical importance, grasping has received a significant amount of attention since the very dawn of robotics research. For a general introduction on the topic, the reader is referred to [12], [15]. Literature in grasp planning is vast and different taxonomies could be considered to classify the various approaches proposed. Some methods do not rely on the previous availability of a geometric model [1], [16] and rather rely on sensorial data collected at run time. These algorithms typically produce form closure grasps rather than force closure. To the best of our knowledge, analytic assessments of the quality of the solutions produced by these approaches have not been proposed. Other methods rely on collections of formerly computed grasps for *typical* objects. The Columbia Grasp Data Base is probably the most known dataset supporting these methods [5].

When planning force closure grasps, a common paradigm is to perform a search in the space of possible configurations while trying to optimize one of the criteria described in the following. To this end, it is often assumed that the boundary of the object is represented using surfaces that can be analytically described and are smooth, like superquadrics, planes, spheres, etc. In [8], the authors assume that the object to be grasped is modeled by superquadrics, and, notably, include in the grasp planning stage a set of constraints imposing that the planned grasp can actually be realized by a given robotic hand. In [18], the authors instead assume that the surface of the object is represented by a collection of planes, spheres, and similar entities. Both methods rely on optimization stages utilizing ideas related to gradient-based or interior-point methods. A common limitation of these two approaches that we try to overcome, is that a valid initial placement for the contact points needs to be given. Moreover, during the grasp planning process the contact points are constrained to remain onto the surface where they started. Therefore in the case of a complex object whose shape is given by the union of many surfaces, the planning process does not explore the whole search space, but only a subset because jumping from one surface to the other is not allowed. A method based on triangle meshes was recently proposed in [7]. Their method is based on a hierarchy of triangle meshes at different resolutions. Another related method is presented in [6], where the set of possible grasping points is initially determined sampling in a regular pattern. Our algorithm could be coupled with these techniques and this is the subject of future work.

The problem of establishing the quality of a grasp has been widely considered. The most used metric for force closure grasps was proposed by Ferrari and Canny in [4] and aims at rewarding grasps that can resist arbitrary wrenches while applying minimal forces with the fingers. The reader is referred to [17] for a review about grasp quality metrics. The analytic properties of the metric proposed in [4] were recently studied in great detail in [14], where the authors determine an efficient criterion to reject grasps not giving force closure and propose a planning method based on random grasps. The random planner, however, only returns force closure grasps, but does not rank them according to any criterion.

III. Preliminaries and notation

In this section we shortly recall concepts and notation relevant for the problem we tackle. Our summary is necessarily short and the reader is referred to [12], [15] for an in depth discussion. We suppose that the object to be grasped is a three dimensional rigid body $\mathcal{B}$. To implement the method presented in section IV, it will be necessary to assume that the surface of the object is represented using a triangle mesh, but the concepts presented herein are independent from the representation. The hand is equipped with n fingers and we hypothesize that each finger touches $\mathcal{B}$ in a single point. Each contact is modeled using the *hard finger* model. Accordingly, a grasp can be represented by the coordinates of n contact points, $\mathbf{p}_1, \ldots, \mathbf{p}_n$ and n contact forces $\mathbf{f}_1, \ldots, \mathbf{f}_n$. The coordinates of $\mathbf{p}_1, \ldots, \mathbf{p}_n$ are referred to a given reference with origin in the center of mass of $\mathcal{B}$. Coherently with the contact model, to prevent slippage each force $\mathbf{f}_i$ must lie within the friction cone $\mathbf{C}_i$ defined at $\mathbf{p}_i$. The cone (in particular its opening) encodes information about the friction coefficient at the contact point, so in the following we will not explicitly mention friction because this information is implicitly represented by $\mathbf{C}_i$. Each force $\mathbf{f}_i$ produces a contact wrench $\mathbf{w}_i$ given by

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{p}_i \times \mathbf{f}_i \end{bmatrix}. \tag{1}$$

For computational efficiency it is customary to approximate $\mathbf{C}_i$ using a regular polyhedral cone with m sides (see Figure 1). We will assume that $\mathbf{C}_i$ is normalized, i.e., the length of the normal $\mathbf{n}_i$ is one, and approximated using a regular polyhedron with m sides.

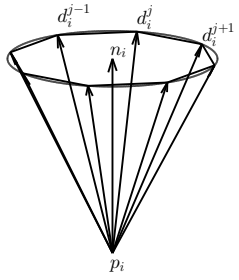

Fig. 1. A regular polyhedral is used to approximate the friction cone $\mathbf{C}_i$. $\mathbf{n}_i$ points inside the object and the opening of the code is defined by the friction coefficient.

When using this approximation we indicate with $\mathbf{d}_i^j$, $1 \leq i \leq n$, $1 \leq j \leq m$ the jth component used to approximate the ith friction cone. Force $\mathbf{f}_i$ can then be written as

$$\mathbf{f}_i = \sum_{j=1}^{m} \alpha_{ij} \mathbf{d}_i^j \tag{2}$$

with $\alpha_{ij} \geq 0$. Substituting Eq. 2 into Eq. 1 we obtain an expression leading to the commonly used test for force closure, i.e.,

$$\mathbf{w}_i = \begin{bmatrix} \sum_{j=1}^{m} \alpha_{ij} \mathbf{d}_i^j \\ \mathbf{p}_i \times \sum_{j=1}^{m} \alpha_{ij} \mathbf{d}_i^j \end{bmatrix} = \sum_{j=1}^{m} \begin{bmatrix} \alpha_{ij} \mathbf{d}_i^j \\ \mathbf{p}_i \times \alpha_{ij} \mathbf{d}_i^j \end{bmatrix} = \\ = \sum_{j=1}^{m} \alpha_{ij} \begin{bmatrix} \mathbf{d}_i^j \\ \mathbf{p}_i \times \mathbf{d}_i^j \end{bmatrix} = \sum_{j=1}^{m} \alpha_{ij} \mathbf{w}_i^j.$$

The grasp achieves force closure if and only if the origin of the six dimensional wrench space lies inside the convex hull (CH) of the set of the nm elementary wrenches $\mathbf{w}_i^j$, i.e.,

$$0 \in \text{int CH}(\mathbf{w}_i^j, 1 \leq i \leq n, 1 \leq j \leq m). \tag{3}$$

It is well known that the relation given in Eq. 3 is valid in theory but often not viable in practice, because it may require the robotic hand to apply arbitrarily large contact forces to resist a possibly modest external wrench. Quantitative measures have then been proposed to identify *good grasps*, i.e., sets of contact points that can resist external wrenches without requiring to apply excessively large contact forces. In the following we use one of the metrics proposed in [4], i.e., the quality Q of a grasp is defined as the radius of the largest ball centered at the origin and totally included in the convex hull of the nm elementary wrenches (see Eq. 3). This metric corresponds to minimizing the maximum finger force needed to resist arbitrary external wrenches. Alternatively, one can also minimize the sum of the finger forces (see [4]). While the method we present in the following embraces the first metric, it can be used with the second as well because it is parametric with respect to the quality measure.

IV. Proposed Approach

We assume that $\mathcal{B}$ is represented using a triangle mesh. This assumption is consistent with current practices in CAD software [3]. Let $\mathbf{T} = \{\mathcal{T}_1, \dots \mathcal{T}_N\}$ be the set of N triangles in the mesh. A grasp on $\mathcal{B}$ will then associate each of the n fingers with one of the N triangles. In general, multiple fingers could be associated with the same triangle, although this placement will usually not result in a satisfactory grasp.

Our grasp planning method works as follows. An initial tentative grasp is generated assigning each of the n fingers to one of the triangles. This assignment can be done randomly, or incorporating prior knowledge as a bias in the random distribution. Then, an iterative improvement process is executed as follows. The positions of $n-1$ fingers are kept fixed, while the position of the remaining finger changes to improve Q. During this phase the position of the finger is not bound to stay inside the triangle it started from, but it can cross the boundary and move to one of the neighboring ones. The ability to move from one triangle to another overcomes some of the major limitations of other methods. In particular we are able to cross *sharp* edges on the boundary of the object. When it is no longer possible to improve the score of the grasp by moving that finger, its position becomes fixed and the same method is applied to another finger. This process repeats until no further improvements are possible. Each of the steps necessary to implement this strategy is described in the following.

Contact point representation. At the core of the method we propose lies a parametric representation for a generic point inside an arbitrary triangle. Figure 2 illustrates the idea.

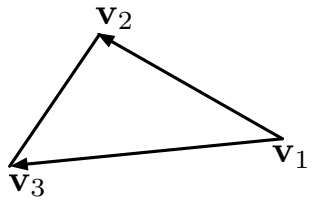

Fig. 2. Any point inside the given triangle can be represented as a constrained linear combination of $\mathbf{v}_2 - \mathbf{v}_1$ and $\mathbf{v}_3 - \mathbf{v}_1$.

Indicating with $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ the coordinates of the three vertices of the triangle $\mathcal{T}$ referred to a given reference frame, the coordinates of a generic point $\mathbf{p}$ inside $\mathcal{T}$ can be written as

$$\mathbf{p} = \alpha_1(\mathbf{v}_2 - \mathbf{v}_1) + \alpha_2(\mathbf{v}_3 - \mathbf{v}_1) \tag{4}$$

subject to the constraints $\alpha_{1,2} \geq 0$ and $\alpha_1 + \alpha_2 \leq 1$. In the following, we indicate with $\mathbf{p}_i$ the contact point of the ith finger. We furthermore indicate as $\mathcal{T}^{(i)} \in \mathbf{T}$ the triangle in which $\mathbf{p}_i$ is located, and let $\alpha_1^{(i)}, \alpha_2^{(i)}$ the coefficients to obtain $\mathbf{p}_i$ using Eq. 4. We will also implicitly assume that the numbering of the vertices in $\mathcal{T}^{(i)}$ is unambiguously determined[1] and stored together with $\mathcal{T}^{(i)}$.

Score function. The score function Q maps a set of contact points to a real value. We therefore write $Q(\mathbf{p}_1, \dots, \mathbf{p}_n)$ to indicate this value.

Local improvement. Let us assume the ith finger is the one whose contact point position is being modified to increase the objective function. Its coordinates can then be written using Eq. 4. Six different vectors $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_6$ are then computed as follows to generate six new candidate contact points:

$$\boldsymbol{\mu}_1 = (\alpha_1^{(i)} + S)(\mathbf{v}_2 - \mathbf{v}_1) + \alpha_2^{(i)}(\mathbf{v}_3 - \mathbf{v}_1) \tag{5}$$

$$\boldsymbol{\mu}_2 = (\alpha_1^{(i)} - S)(\mathbf{v}_2 - \mathbf{v}_1) + \alpha_2^{(i)}(\mathbf{v}_3 - \mathbf{v}_1) \tag{6}$$

$$\boldsymbol{\mu}_3 = \alpha_1^{(i)}(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} + S)(\mathbf{v}_3 - \mathbf{v}_1) \tag{7}$$

$$\boldsymbol{\mu}_4 = \alpha_1^{(i)}(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} - S)(\mathbf{v}_3 - \mathbf{v}_1) \tag{8}$$

$$\boldsymbol{\mu}_5 = (\alpha_1^{(i)} + S)(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} - S)(\mathbf{v}_3 - \mathbf{v}_1) \tag{9}$$

$$\boldsymbol{\mu}_6 = (\alpha_1^{(i)} - S)(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} + S)(\mathbf{v}_3 - \mathbf{v}_1). \tag{10}$$

[1]Eq. 4 relies on a precise ordering of the vertices in the triangle. If the order is changed, the expression is still valid, but the coefficients change.

The parameter S (step size) determines the magnitude of the local exploration and is iteratively altered during the computation, according to a schedule described later. The score function Q is then computed for all grasps $\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}, \boldsymbol{\mu}_j, \mathbf{p}_{i+1}, \ldots \mathbf{p}_n$, $(1 \leq j \leq 6)$ obtained substituting $\mathbf{p}_i$ with $\boldsymbol{\mu}_j$. If no improvement is obtained, the local improvement for the ith finger is stopped. Otherwise $\mathbf{p}_i$ is replaced by the new point $\boldsymbol{\mu}_j$ achieving the highest value for the score function and the process continues.

It is evident that there exists combinations of values for $\mathbf{p}_i$ and S such that one or more of the new candidate contact points $\boldsymbol{\mu}_j$ may fall outside the triangle $\mathcal{T}^{(i)}$. Assume $\boldsymbol{\mu}_j$ lies outside $\mathcal{T}^{(i)}$. Then, the segment between $\boldsymbol{\mu}_j$ and $\mathbf{p}_i$ must cross one of the edges of $\mathcal{T}^{(i)}$ and the neighboring triangle $\mathcal{T}_k$ can be univocally[2] and efficiently determined, for example using a doubly-connected edge list data structure [3] (see Figure 3).

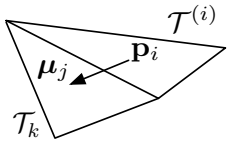

Fig. 3. When $\boldsymbol{\mu}_j$ is outside $\mathcal{T}^{(i)}$, the neighboring triangle $\mathcal{T}_k$ can be efficiently determined.

In this case $\boldsymbol{\mu}_j$ is replaced with a random point inside $\mathcal{T}_k$. Then, if the score function obtained substituting $\mathbf{p}_i$ with $\boldsymbol{\mu}_j \in \mathcal{T}_k$ is the best among the new candidates, the ith finger moves to the new triangle, i.e., $\mathcal{T}^{(i)}$ is replaced by $\mathcal{T}_k$. $\mathcal{T}^{(i)}$ and $\mathcal{T}_k$ are not bound to lie on the same plane, but could be arbitrarily positioned (e.g., they could lie on two orthogonal planes.) This feature allows the local improvement phase to cross boundaries that normally limit the exploration range of optimization methods relying on smooth surfaces.

Step size update: Eqs. 5-10 depend on the step size parameter S. The parameter is initially set at 0.5. This choice is motivated by the constraints on $\alpha_{1,2}^{(i)}$ ensuring $\mathbf{p}_i$ is inside $\mathcal{T}^{(i)}$. A larger value would provide a too strong bias towards points $\boldsymbol{\mu}_j$ outside $\mathcal{T}^{(i)}$, while we strive to balance exploration inside and outside the triangle. As the process continues, S decreases according to the formula[3] (S' is the new step)

$$S' = S\frac{v-b}{|v|}$$

where v is the value of the score function for the newly determined best grasp, and b is the previous best value. Therefore, as the magnitude of the improvement decreases, the new step size decreases too and eventually approaches 0 when v approaches b. In the experimental section we show that this approach enables the exploration of various triangles in the given mesh.

[2]Degenerate cases of course exist, for example if the segment between $\boldsymbol{\mu}_j$ and $\mathbf{p}_i$ intersects a vertex of the triangle. Cases like this can be arbitrarily solved, e.g., randomly selecting one of the edges defining the vertex.

[3]We use the absolute value of v because in the beginning of the process it could be the case that the grasp does not yet achieve force closure, and then v is negative, i.e., it is the distance of the convex hull form the origin.

Algorithmic sketch: Putting all the previous steps together we obtain algorithms 1 and 2. Algorithm 1 is the high level control part. It first generates K random grasps and the best score is set to be the threshold Thr. Then we again generate random grasps until the score of the grasp is larger than Thr and use algorithm 2 to improve the score and push it into a local optimum. We do R random restarts for this phase, and the grasp with highest score is chosen to be the final grasp. The process for generating random grasps is extremely rapid. So the first step which calculates Thr will favor us in a way that we won't start the optimization process from a particularly disadvantageous initial grasp. Higher values of K are evidently desirable but come at the cost of a longer preprocessing step. In our experiments we set K to 90 and R to 10, but the overall performance is not too sensitive to these parameters, neither in terms of quality of the solution, nor in terms of time.

Algorithm 1 Global optimization algorithm

```
1:  Data : T = {T_1, ... T_N}, n
2:  Result : p_1, ..., p_n
3:  Thr ← −∞
4:  for i ← 1 to K do
5:     (p_r1, ..., p_rn) ← RandomGrasp
6:     if Q(p_r1, ..., p_rn) > Thr then
7:        Thr ← Q(p_r1, ..., p_rn)
8:  Best ← −∞
9:  S ← 0.5
10: for i ← 1 to R do
11:    Converged ← false
12:    while Q(p_1, ..., p_n) < Thr do
13:       (p_1, ..., p_n ← RandomGrasp
14:    repeat
15:       RP ← RandomPermutation(1, ..., n)
16:       for j ← 1ton do
17:          p_R(i), v_i ← Improve(R(i), p_1, ..., p_n, S)
18:          if max_i v_i > Best then
19:             Best ← max_i v_i
20:             S ← NewStep
21:          else
22:             Converged ← true
23:    until Converged = true
```

It is immediate to show that the algorithm eventually converges, i.e., it does not get stuck going back and forth between two sets of contact points. This is true because the algorithm can never go back to a previously visited configuration since it moves away from it only when it finds one with a better score and never goes to a configuration with an inferior ranking. Hence the algorithm eventually converges to a local optimum.

V. Experimental Results

In this section we present the results we obtained testing our algorithm on various objects. The code is written in

Algorithm 2 Local improvement step

$i, \mathbf{p}_1, \ldots, \mathbf{p}_n, S$
$\mathbf{p}_i, v$
repeat
  Compute $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6$ from $\mathbf{p}_i$ and S as in Eq. 5-10
  for $j \leftarrow 1$ **to** 6 **do**
    if $\boldsymbol{\mu}_j \notin \mathcal{T}^{(i)}$ **then**
      Determine triangle $\mathcal{T}_k$ neighbor of $\mathcal{T}^{(i)}$
      $\boldsymbol{\mu}_j \leftarrow$RadomPoint in $\mathcal{T}_k$
  $\mathbf{p}_i \leftarrow \arg\max_{\boldsymbol{\mu}_j, \mathbf{p}_i} g(\mathbf{p}_1, \ldots, \boldsymbol{\mu}_j, \ldots \mathbf{p}_n)$
  if needed **then**
    Update $\mathcal{T}^{(i)}$
until $\mathbf{p}_i$ does not change anymore
$v \leftarrow g(\mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots \mathbf{p}_n)$

C++ and relies the *qhull*[4] library to compute the convex hull of the discretized friction cones needed in the definition of Q. Our implementation is in no-way optimized, therefore results involving times should only be considered to compare the impact of design choices (e.g., dependency between the time and the complexity of the mesh) and not to gauge the absolute efficiency of the algorithm. All results discussed in this section refer to the case where grasps involving 4 contact points are computed.

Figure 4 shows three of five objects we used. In particular the first one (referred to as *joystick* in the following) was chosen because it replicates one of the test cases used in [18]. The two objects not shown are a cube and a sphere. Table I shows the number of triangles in each mesh.

Object	# Triangles
Joystick	1134
C shape	96
Teapot	992
Cube	12
Sphere	960

TABLE I
NUMBER OF TRIANGLES IN THE MESHES

Figure 5 displays the result of 100 runs for the teapot and the joystick objects. Each episode consists of 10 independent executions of the algorithm and the planner eventually returns the best grasp obtained in the 10 restarts. The score on the y axis is the Q grasp quality metric formerly described. Evidently, increasing the number of restarts would reduce variability and increase the average score, but this will come at the cost of additional time.

Similar trends were obtained for the other objects and are displayed in tabular form in Table II and III.

It is interesting comparing Table III with Table I because we see that large variations in the number of triangles in the mesh do no translate to dramatic changes in computational time. One can therefore in practice afford to utilize triangular meshes with a large number of triangles to provide high

[4]http://www.qhull.org.

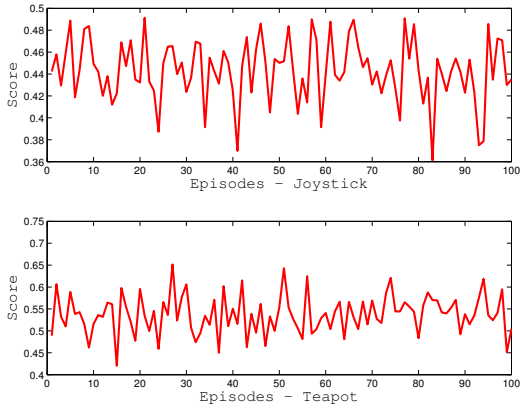

Fig. 5. Performance on the joystick and teapot objects.

Object	Avg. Q	Std. Dev. Q
Joystick	0.4432	0.0281
C shape	0.4279	0.0263
Teapot	0.5377	0.0441
Cube	0.3619	0.0376
Sphere	0.6667	0.0260

TABLE II
GRASP QUALITY FOR DIFFERENT OBJECTS. AVERAGES AND STANDARD DEVIATIONS ARE COMPUTED OVER 100 EPISODES.

fidelity approximations of the object shape. With regard to computational time, it is important to observe that like all methods based on independent restarts, this algorithm can be easily parallelized and take advantage of multi-core architectures, with each core executing an equal share of the restarts. Therefore, one can expect a speedup linear in the number of cores. This is an advantage over strictly sequential algorithms. All results presented in this section do not take advantage of the multi-core approach and consequently displayed times could be easily reduced by a factor of 4 or 8 when run on nowadays entry level multi-core processors. Additional performance gains could be obtained integrating our recently developed PQH algorithm to accelerate the evaluation of the grasp metric Q [9]. These improvements are left for future work.

Figure 6 illustrates one of the major features of the algorithm we proposed, namely the ability to move the contact points between the different parts of an object. The *joystick* object was used in [18] and can be defined as the union of 8 different surfaces: 6 planes to define the cubical part, one to define the cylinder, and one for the sphere.

Object	Avg. T	Std. Dev. T
Joystick	8.48	7.6422
C shape	10.17	15.5095
Teapot	8.33	11.4297
Cube	8.09	9.8105
Sphere	3.26	0.8287

TABLE III
TIME SPENT TO COMPUTE GRASPS FOR DIFFERENT OBJECTS. AVERAGES AND STANDARD DEVIATIONS ARE COMPUTED OVER 100 EPISODES.

Fig. 4. Three of the objects used to test the algorithm. In the following they are referred to as *joystick*, *c shape* and *teapot*. The other two objects are a cube and a sphere.

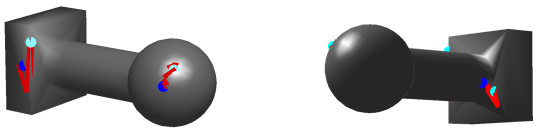

Fig. 6. The figure shows the path followed by three contact points for the testcase *joystick* (red line). Each contact point starts from the cyan location and end in the blue point. The path is plotted in red.

Object	Contact #	Avg. N	Std. Dev. N	Max N
Joystick	1	1.42	1.0641	12
	2	1.43	1.1661	15
	3	1.42	1.1075	9
	4	4.68	5.1763	33
C shape	1	1.09	0.6279	5
	2	1.10	0.6923	7
	3	1.08	0.5962	5
	4	2.00	1.9519	21
Teapot	1	1.39	0.9699	8
	2	1.40	0.9993	9
	3	1.38	0.9674	7
	4	4.26	4.2510	28
Cube	1	1.02	0.5224	4
	2	1.01	0.4961	4
	3	1.02	0.5290	5
	4	1.56	1.3320	19
Sphere	1	1.25	0.7925	5
	2	1.24	0.8283	8
	3	1.33	0.9255	10
	4	3.75	3.1650	23

TABLE IV

AVERAGE, STANDARD DEVIATION, AND MAXIMUM NUMBER OF TRIANGLES (N) EXPLORED BY EACH FINGER DURING THE SEARCH. DATA IS AVERAGED OVER 100 RUNS.

To run[5] the algorithm described in [18] one has to specify in which of the 8 surfaces each contact point is initially located. During the optimization process each contact point cannot leave the surface it started from. For example, if one point is initially located on the cylinder, it has to remain on that surface. Figure 6 instead shows that with our method the contact points are moved along the parts of the object spanning different faces. Hence, our planning method is *global* because it does not restrict the range of motions for the contact points. Similar behaviors are observed on the *c shape* and the *teapot* objects, but figures are omitted for lack of space. Table IV quantitatively substantiate our claim that the algorithm explores multiple triangles during the search process. For each object the table shows the average number of triangles explored by each contact point. Note that the highest number is always recorded for the fourth contact point, because the exploration starts from that finger. To put these numbers into perspective, it is also necessary to recall that the exploration starts from the best of the initial $K = 100$ random grasps and often times this provides a good enough starting point so that limited exploration is needed. Moreover, since we are using four contact points, force closure is reached sooner, whereas with three fingers more exploration would be needed.

A. Comparison with random sampling

In this last subsection we compare our algorithm with a baseline planner relying on randomly selected grasps. This is the same comparison proposed in [7], and we opt for this approach given the difficulty in obtaining fully functional implementations of third party grasp planners[6]. As term of comparison we use the number of grasps evaluated during the planning process. This is a meaningful metric because grasp quality evaluation implies the computation of the convex hull, and this is the most time consuming step. The randomized grasp planner repeatedly generates one random grasp and computes its score. It retains the grasp if it is the best generated so far, otherwise it discards it, and it repeats the process. This algorithm is anytime, in the sense that the more time it spends, the higher the quality of the solution it finds. Figure 7 shows the results, with the red curve showing our solution and the blue one displaying the random sampler. Presented data are averages over 100 repeated independent trials. Note that we stopped our planner after 10 restarts and on average that required around 3000 grasp evaluations, whereas we let the randomized planner to continue until 5000 grasps were generated and evaluated. The gap is larger for the joystick case, where it is evident that the randomized planner can increase its performance only very slowly. The gap is narrower for the teapot object, but it is worth observing two

[5]We thank the authors of [18] for having provided us the code implementing their method.

[6]A freely available implementation of our algorithm is available on `http://robotics.ucmerced.edu`.

facts. First, the randomized planner increases its performance only very slowly and will therefore take a long time to close the gap even though it is small. Second, although the gap may be small in absolute terms, it is significant from a practical point of view, in the sense that even a small increase in the score metric is important. Finally, it worth observing that the two curves initially overlap, because our planner also starts generating 100 random grasps to initialize the optimization process. The chart shows that the search method we proposed succeeds in driving the exploration process towards valuable grasps that cannot be easily reached by random sampling only.

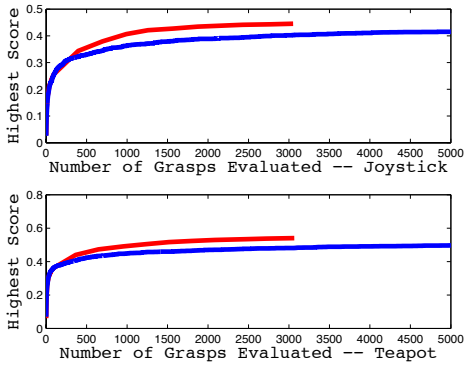

Fig. 7. Comparison between the proposed algorithm and a randomized grasp planner.

VI. Conclusions and Future Work

We proposed a method to plan grasps to obtain force closure on objects whose surface is represented using a triangle mesh. The algorithm can plan grasps for an arbitrary number of contact points and the result is obtained through an iterative process aiming at optimizing a commonly used grasp quality function. Besides the triangle mesh, the algorithm does not require any other preliminary information, like the initial placement for the contact points. Our simulations outline two major findings. First, the computational time grows very slowly with the number of triangles in the meshes and this enables the use of meshes with a large number of triangles to provide accurate approximations of objects featuring curved surfaces. We have also demonstrated that the algorithm is indeed *global* in the sense that during the exploration stage the contact points can move through different patches of the surface, thus overcoming one of the limitations of methods requiring smooth representations and imposing constraints on the area to be explored. The algorithm also clearly outperforms a baseline randomized planner.

We anticipate a number of improvements for our method. First, we will integrate a *hand compatibility* test, similarly to what has been implemented in [8], so that we can reject grasps that cannot be physically realized by the robotic hand. This test can be included during the local improvement step. Next, in order to expedite the process we will implement a parallel version of the algorithm, aiming at a speedup linear in the number of cores utilized. Finally, noticing the difficulty in providing quantitative comparisons with other grasping algorithms, we believe it will be important for the research community to develop benchmark problems, standardized representations, and canonical implementations of quality metrics to promote repeatable and comparable research in this area. Software packages like *Graspit!* [10] provide a step in the right direction, but the community still lacks standardized assessment procedures for repeatable performance evaluation.

References

[1] B. Balaguer and S. Carpin. Efficient grasping of novel objects through dimensionality reduction. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1279–1285, 2010.

[2] A. Bicchi and V. Kumar. Robotic grasping and manipulation. In S. Nicosia, B. Sicilano, and A. Bicchi, editors, *Ramsete: Articulated and mobile robots for services and Technology*, volume 270 of *LNCS*, pages 55–74. Springer, 2001.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, 3rd edition, 2008.

[4] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2290–2295. IEEE, 1992.

[5] C. Goldfeder, M. Ciocarlie, H. Dang, and P.K. Allen. The Columbia grasp database. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1710–1716, 2009.

[6] K. Hang, J.A. Stork, and D. Kragic. Fingertip space for multi-fingered precision grasping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1641–1648, 2014.

[7] K. Hang, J.A. Stork, F.T. Pokorny, and D. Kragic. Combinatorial optimization for hierarchical contact-level grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 381–388, 2014.

[8] S. El Khouri, L. Miao, and A. Billard. Bridging the gap: One shot grasp synthesis approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2027– 2034, 2012.

[9] S. Liu and S. Carpin. Fast grasp quality evaluation with partial convex hull computation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015.

[10] A.T. Miller and P.K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics Automation Magazine*, 11(4):110–122, 2004.

[11] H. Moravec. *The future of robot and human intelligence*. Harvard University Press, 1998.

[12] R.M. Murray, Z. Li, and S.S. Sastry. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[13] A.M. Okamura, N. Smaby, and M.R. Cutkosky. An overview of dexterous manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 255–262, 2000.

[14] F.T. Pokorny and D. Kragic. Classical grasp quality evaluation: New theory and algorithms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3493– 3500, 2013.

[15] D. Pratticchizzo and J.C. Trinkle. Grasping. In B. Siciliano and O. Khatib, editors, *Handbook of robotics*, chapter 28, pages 671–700. Springer, 2008.

[16] A. Saxena, J. Driemeyer, and A.Y. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(2):157–173, 2008.

[17] R. Suárez, M. Roa, and J. Cornella. Grasp quality measures. Technical Report IOC-DT-P-2006-10, Universitat politècnica de Catalunya, March 2006.

[18] X. Zhu and J. Wang. Synthesis of force-clousure graspos on 3-D objects based on Q distance. *IEEE Transactions on Robotics and Automation*, 19(4):669–679, 2003.