A Fast Algorithm for Grasp Quality Evaluation Using the Object Wrench Space

Shuo Liu Stefano Carpin

Abstract— **Grasp quality evaluation is an important problem when multifingered robotic hands are used to restrain objects in automation and manufacturing. In this paper we present a new algorithm to greatly expedite the evaluation of grasp quality considering the so-called object wrench space, i.e., the set of all disturbance wrenches that may be generated by a disturbance force acting on the object being grasped. While this metric has been known since a while, its practical use inside grasp planners has been limited because its exact computation is time consuming and thus prevents its use when many grasps have to be repeatedly evaluated during the planning process. Building on some geometric insights related to convex hulls, our algorithm determines on the fly which subset of the input data needs to be processed, and stops the computation as soon as it is known that the exact value of the metric has been determined. We show that our new algorithm significantly decreases the time needed to compute the grasping quality measure thus enabling grasp planners to use this metric to search the space of possible grasps.**

I. Introduction

The ability to restrain an object with a robotic hand is a fundamental building block for numerous tasks related to robotics and automation. Grasp planning is the problem of determining where to establish contact points on the surface of an object in order to restrain it. It is a problem that has been extensively studied and can be seen as search problem in the space of feasible grasps.[1] The search is commonly informed by a grasp evaluation function. In fact, for a given grasp planning problem there exist multiple solutions, and the question of which one should be preferred naturally arises. In the context of force closure grasps, a quality metric proposed by Ferrari and Canny more than two decades ago has been widely embraced [1]. In essence their criterion favors grasps that can resist arbitrary disturbances (wrenches) with the least effort. While this idea makes sense in theory, it turns out to be often too conservative in practice. In fact, the torque component of the disturbance wrench can hardly appear alone and is almost always induced by a disturbance force. Building upon this observation, Strandberg and Wahlberg proposed a new method for grasp evaluation aiming at assessing the ability to reject disturbance forces [2]. Besides being more realistic and less conservative, their method enjoys additional desirable properties like scale invariance – an aspect missing in [1]. Despite these advantages, however, this method has not enjoyed wide use in the community. One of the reasons is that its computational requirements are quite onerous and this negatively affects the planning cycle where many tentative grasps have to be evaluated.

In this paper, building upon our recent results presented in [3], we present an algorithm that significantly expedites the computation of the metric proposed in [2], thus paving the way for its use in practice. Our method exploits the same principle we presented in [3], namely that in many practical situations a complete computation of the geometric objects defining the metric is unnecessary because the result can be inferred from a partial computation. The similarity however ends here, in the sense that the method we presented in [3] is tailored for the Ferrari-Canny metric, whereas we here tackle a significantly different measure. Our experimental evaluation shows that a basic implementation of our algorithm achieves a more than tenfold acceleration over the brute force method and is comparable with the optimized implementation we presented in [3] for the Ferrari-Canny metric.

The rest of the paper is organized as follows. Related work is presented in Section II and the necessary mathematical background is given in Section III. Our method is then presented in Section IV and experimentally evaluated in Section V. Finally, conclusions and future work are discussed in Section VI.

II. Related work

The problem of evaluating the quality of a force closure grasp has been extensively studied (see [**?**] for a survey). Ferrari and Canny introduced the most commonly used grasp metric (often called Q measure) built upon the associated concept of grasp wrench space (GWS), i.e., the set of grasps that can be resisted by a grasp [1]. From a computational point of view their method requires the computation of the convex hull of a set of points in 6 dimensions (elementary wrenches), and this is usually achieved using the QuickHull algorithm [4]. The Q measure is defined as the radius of the largest ball centered on the origin and fully contained in convex hull of the set of elementary wrenches. Recently [3], we have shown that the computation of this metric can be greatly increased using a partial quick hull computation, i.e., stopping the computation of the convex hull as soon

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.

[1]Throughout the paper we exclusively consider force closure grasps.

as enough information is available to compute the Q metric. Pollard introduced the concept of object wrench space (OWS) [5], i.e., the set of all wrenches that can be exerted on an object by an external disturbance force. Borst et al. [6] propose a method to approximate the object wrench space computing an enclosing ellipsoid rather than the exact shape. This ellipsoid is then transformed into a sphere using a linear transformation and then a grasp quality estimate is produced taking the radius of the largest sphere inside the grasp wrench space (as in the original definition of the Q measure). This method however is approximate because there can be quite some mismatch between the actual object wrench space and the enclosing ellipsoid. Strandberg and Wahlberg [2] proposed a method for the exact computation of a grasp metric based on the object wrench space. Their argument is that using the grasp wrench space to evaluate the quality of a grasp is too conservative, since this approach fails to consider that from a practical point of view disturbance wrenches are the consequence of disturbance forces, and it then makes sense to only consider those wrenches (i.e., the object wrench space), as opposed to all possible wrenches. A suitable metric is then defined considering how much the object wrench space can be inflated before it reaches the boundary of the grasp wrench space. This is in contrast to considering the radius of the largest ball, i.e., considering all possible wrenches. From a computational point of view, their method resembles a brute force approach, with only some minor improvements introduced through some bucketing. A similar method was proposed in [7]. Pushing even further the idea that a grasp should only be evaluated with regard to its ability to resist the wrenches it will encounter in practice, the task wrench space was defined in [8]. The task wrench space is the set of all external wrenches that can be generated during the execution of a specific task with an object. However, while this idea makes sense in theory, it has been scarcely used in practice because it is not easy to determine this set in general.

III. Background

In this section we provide a short recap on grasping metrics and introduce relevant notation used in the remainder of the paper. The reader is referred to [9] for a more thorough discussion.

A. Object Wrench Space and Grasp Wrench Space

A wrench $\mathbf{w}$ is a six dimensional vector in which the first three components are a force $\mathbf{f}$ and the last three components are the torque generated by the force. The wrench generated by a force $\mathbf{f}$ exerted at point $\mathbf{p}$ on the surface of an object is

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

The coordinates of point $\mathbf{p}$ are expressed with respect to a reference frame normally placed at the center of mass of the object. The object wrench space (OWS from now onwards) is the union of all wrenches that can be generated by a unit length disturbance force acting on the surface of the object being grasped. Let $\mathcal{B}$ be the object being grasped and let D be its surface. Let $\mathbf{e}_i$ be a unit length vector. The object wrench space is then defined as

$$\text{OWS} = \left\{ \bigcup_{\mathbf{a}_j, \mathbf{e}_i} \begin{bmatrix} \mathbf{e}_i \\ \mathbf{a}_j \times \mathbf{e}_i \end{bmatrix} \mathbf{a}_j \in D, \mathbf{e}_i \in \text{FC}(\mathbf{a}_j) \right\}.$$

Note that in the definition the contact point $\mathbf{a}_j$ spans the whole surface of the object, but we constrain the vector $\mathbf{e}_i$ to be inside the friction cone $\text{FC}(\mathbf{a}_j)$ at the contact point (to prevent slippage). In the sequel, it is convenient to assume the surface D is expressed as a collection of triangles. Indeed, this is a representation often used in digital design and often also assumed in grasp planning [10], [11]. Indicating with $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ the three vertices of a triangle, any contact point point $\mathbf{a}_j$ inside the triangle can then be written as a convex combination of the vertieces, i.e., $\mathbf{a}_j = \sum_{k=1}^{3} \alpha_k \mathbf{v}_k$ where $\sum_{k=1}^{3} \alpha_k = 1$ and $\alpha_k \geq 0$. The wrench generated by $\mathbf{e}_i$ acting at $\mathbf{a}_j$ can then be written as

$$\mathbf{q} = \begin{bmatrix} \mathbf{e}_i \\ \sum_{k=1}^{3} \alpha_k \mathbf{v}_i \times \mathbf{e}_i \end{bmatrix}.$$

Therefore the wrench generated by $\mathbf{e}_i$ acting on any point of the surface D can be expressed as a convex combination of the wrenches generated when the forces act on the vertices of the triangles. Assuming the surface D is composed of m triangles, OWS can then be alternatively defined as follows

$$\text{OWS} = \bigcup_{\mathbf{e}_i} \bigcup_{\mathbf{v}_{j,k}} \text{CH} \left(\begin{bmatrix} \mathbf{e}_i \\ \mathbf{v}_{j,k} \times \mathbf{e_i} \end{bmatrix} \right. \\ \left. j = 1 \dots m, k = 1, 2, 3 \ \mathbf{e}_i \in \text{FC}(\mathbf{v}_{j,k}) \right) \tag{1}$$

where CH indicates the convex hull and $\mathbf{v}_{j,k}$ is the $k-$th vertex of the $j-$th triangle on the triangle mesh.

Following a similar notation, the space of all wrenches that can be generated by a grasp with n contact points can be defined. To this end, let $\mathbf{e}_i$ be the unary length force acting at contact point $\mathbf{p}_i$. To prevent slippage the force must lie inside the friction cone at point $\mathbf{p}_i$. It is customary to discretize the friction cone with a pyramid with d edges, and then $\mathbf{e}_i$ can be written as

$$\mathbf{e}_i = \sum_{k=1}^{d} \beta_{ik} \mathbf{f}_i^k$$

where $\mathbf{f}_i^k$ is the kth component of the discretization of the friction cone (see Figure 1). The wrench generated by the i-th normalized contact force can then be written as

$$\mathbf{w}_i = \begin{bmatrix} \sum_{k=1}^{d} \beta_{ik} \mathbf{f}_i^k \\ \mathbf{p}_i \times \sum_{k=1}^{n} \beta_{ik} \mathbf{f}_i^k \end{bmatrix} = \sum_{k=1}^{d} \beta_{ik} \mathbf{w}_i^k.$$

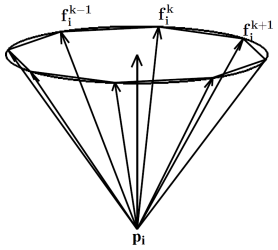

Fig. 1. When defining the grasp wrench space it is convenient to approximate the friction cone at the contact point with a regular pyramid with d edges.

Building upon this notation, assuming there are n contact points, the grasp wrench space (GWS) is then defined as follows:

$$\text{GWS} = \text{CH}\left(\bigcup_{i=1}^{n}\{\mathbf{w}_i^1, \dots \mathbf{w}_i^d\}\right).$$

The set of nd wrenches appearing in the definition GWS ($\mathbf{w}_i^j$) is commonly referred to as set of *elementary wrenches*.

B. Grasp Quality Measures

The most commonly used grasp quality metric, often indicated as Q measure is defined as the radius of the largest ball within GWS [1]. The Q measure formalizes the idea that we should prefer grasps capable of resisting arbitrary external disturbance wrenches using the smallest forces at the contact points. However, as pointed out in [2] this criteria is practically too conservative because it considers all possible external wrenches, but in practice wrenches are generated only from disturbance forces. Therefore GWS is formulated with respect to many wrenches that in practice may never occur (e.g., a wrench consisting of just a torque component without a force). Building upon this insight, a different quality measure was therefore proposed in [2], namely

$$Q_{\text{OWS}} = \max_{r>0}\{r \cdot \text{OWS} \subseteq \text{GWS}\}$$

Q_{OWS} by the definition is how much one can inflate OWS (through the parameter r) before reaching the boundary of GWS. Hence Q_{OWS} rectifies the definition of Q by considering only the wrenches that can be generated by disturbance forces, as per the definition of OWS. The following theorem, whose easy proof is omitted in the interest of space, offers an alternative and more efficient way to compute Q_{OWS}.

Theorem 1: Let OWS be an object wrench space and GWS be a grasp wrench space. Then,

$$\max_r\{r \cdot \text{CH(OWS)} \subseteq \text{GWS}\} = \max_r\{r \cdot \text{OWS} \subseteq \text{GWS}\}.$$

Theorem 1 is important because the convex hull of OWS includes much less vertices than OWS as defined in Eq. 1, and this will have important consequences in the following. If we consider the structure of a convex hull, it is clear that the largest value for r will be achieved when CH(OWS) intersects GWS, and the intersection point occur between a vertex of CH(OWS) and a point on one of the boundaries on the facets defining the boundary of GWS Therefore the expression to compute Q_{OWS} can be further rewritten as

$$Q_{\text{OWS}} = \min_r\left\{r \cdot \mathbf{v}_i \bigcap \text{GWS} \neq \emptyset, \mathbf{v}_i \in V(\text{CH(OWS)})\right\} \quad (2)$$

where V(CH(OWS)) is the set of vertices defining the convex hull of OWS.

Remark: while we here only consider Q_{OWS}, the same principles can be applied towards the task wrench space (TWS), i.e., the space of wrenches that can be generated during the execution of a given task [8]. In fact, TWS is a subset of OWS and hence the same inflation principles apply.

C. QuickHull algorithm

The QuickHull algorithm is commonly used to compute Q and we here introduce some of its terms that we will use in the following (the reader is referred to [4] or [3] for more details). Given n points in space, QuickHull iteratively grows the convex hull following an heuristic that works well in practice. The convex hull consists of a set of facets and each facets is supported by an hyperplane. For each facet the algorithm maintains a unit normal vector pointing outside the convex hull and the offset (distance) from the origin. Moreover, for each facet, the outside set of the facet is defined as the set of input points not yet included in the convex hull and located in the halfplane identified by the normal vector.

IV. Q_{OWS} Calculation

Building upon the definitions and observations given in the previous section, we here present an efficient method to compute Q_{OWS} based on partial convex hull computation. As per Eq. 2, we need to determine the maximum factor r by which we can inflate CH(OWS) until one of its vertices intersects GWS. It is useful to recall that GWS is by definition the convex hull of a set of points, and hence a convex set bounded by a set of hyperplanes in six dimensions. A brute force approach to determine r could therefore iterate over all possible vertices in OWS and hyperplanes supporting the facets of GWS and determine by how much each vertex could be expanded before touching the hyperplane. For the i-th vertex $\mathbf{v}_i$ and the j-th hyperplane H_j we define

$$r_{i,j} = \frac{o(H_j, 0)}{\mathbf{v}_i \cdot \mathbf{n}(H_j)} \quad (3)$$

where $o(H_j, 0)$ is the offset from the origin of hyperplane H_j, $\mathbf{n}(H_j)$ is the outward normal to H_j, and $\mathbf{v}_i \cdot \mathbf{n}(H_j)$ is the internal product between the two vectors. Based on these computations, the needed metric is then

$$Q_{\text{OWS}} = \min\{r_{i,j} | r_{i,j} > 0\}.$$

Note that in the following we will ensure that the set of positive $r_{i,j}$ values is never empty, so Q_{OWS} is well defined. While this method provides the needed result, it is too demanding because it runs in $\Theta(V_{\text{OWS}} H_{\text{GWS}})$ where V_{OWS} is the number of vertices defining the OWS convex

hull and H_{GWS} is the number of hyperplanes in the convex hull of GWS. This last term is particularly important because the number of hyperplanes can be as high as $\Theta(V_{\text{GWS}}^3)$ where V_{GWS} is the number of vertices in GWS[2]. The reader should also notice that when approximating the friction cone with a pyramid one has interest in increasing the number of edges d, and this further increases the size of of V_{GWS}. One should further notice in a grasping planning OWS can be pre-computed once and it does not change, but GWS is directly related to the grasp being evaluated and this repeatedly changes during the planning process, so that one has to defer most of the computation at run time.

In order to overcome this computational bottleneck, we present in the following a strategy to compute Q_{OWS} based on a partial computation of the convex hull of GWS. This expedient greatly reduces the computational time without introducing any approximation.

A. *Partial convex hull for Q_{OWS}*

The main idea of the algorithm we propose is to iteratively grow the convex hull leading to GWS similarly to what the QuickHull algorithm does, and to stop the growth as soon as the correct value for the Q_{OWS} metric can be determined. In the following, when we refer to GWS we are in fact referring to a partial convex hull because it is the convex hull of just a subset of the elementary wrenches and not of the full set. To determine when the computation of the convex hull can be stopped, we assign each vertex of OWS to one of the facets in the partial convex hull of GWS. The objective of the association is to compute the inflation ratio only between associated vertices and facets, thus avoiding a brute force search as in Eq. 3. The main challenge, of course, is in efficiently establishing and updating these associations while the convex hull is iteratively growing and changing with facets being added and deleted. The following lemma provides the foundation for our algorithm.

Lemma 1: Let V be the set of vertices in CH(OWS) and H be the set of hyperplanes supporting the facets of the convex hull GWS. If $\mathbf{v}_i \in V$ and $H_j \in H$ achieve the minimum in the expression given by Eq. 3, then $r_{i,j}\mathbf{v}_i$ lies on the boundary of the convex hull GWS, i.e., it lies in one of its facets.

The algorithm works as follows. Initially, as per Theorem 1, CH(OWS) is computed. As in Eq. 2, we indicate with $V(\text{CH(OWS)})$ the set of vertices of this convex hull. Next, a partial convex hull for the space of elementary wrenches is initialized with 7 vertices[3], as per the QuickHull [4] or PQH [3] algorithms. The initial convex hull is then iteratively expanded until it includes the origin. This precondition is necessary to ensure the correctness of the successive steps. Figure 2 shows the initialization phase.

[2]It is known that the convex hull of n points in $\mathbb{R}^d$ can include up to $\Theta(n^{\lfloor (d+1)/2 \rfloor})$ facets. Since our algorithm is applied in $\mathbb{R}^6$ our claim follows.

[3]In general, when computing a convex hull in $\mathbb{R}^n$ with QuickHull, one should initialize the convex hull with $n+1$ points. However, our algorithm is exclusively applied in the special case in which the input consists of wrenches, i.e., points in $\mathbb{R}^6$, so the initialization is completed with 7 points.

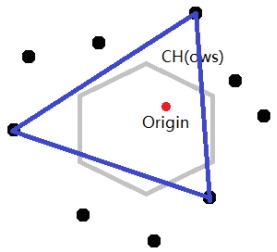

Fig. 2. Initialization step for the algorithm. CH(OWS) is computed (gray) as well as a partial convex hull over the set of elementary wrenches (black dots). The process stops when the origin is included the partial convex hull (blue). The figure shows a two dimensional depiction of a computation happening in $\mathbb{R}^6$.

At this point the initial association between vertices of OWS and facets of GWS takes place. Each vertex is associated with the facets with which it will intersect when inflated. The search for the correct face to associate is done brute force over the set of all hyperplanes supporting the facets in the current partial convex hull. Lemma 1 assures that taking the smallest value computed for each vertex will identify the correct facet. Note that even though this association requires to iterate over all facets of the partial convex hull, the effort is modest because the iteration happens over a set of facets significantly smaller than the whole GWS (see Figure 3). Note also that since the association starts after the origin has been included in the partial GWS, each vertex in OWS is associated with a positive inflation ratio.

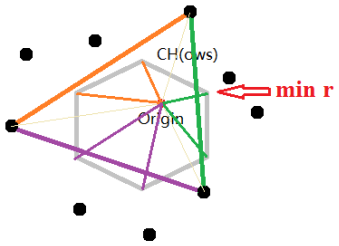

Fig. 3. Initial association between vertices OWS with facets in the partial GWS. Associations are shown by corresponding colors. Note that for some vertices (e.g., the purple ones) the actual inflation factor is smaller than 1. The figure also shows the facet associated with the vertex with the smallest inflation ratio.

Next, the algorithm proceeds expanding the partial convex hull adding a new vertex to to it. Since our objective is to determine the smallest inflation ratio, we expand the facet associated with the point with the smallest inflation ratio. The facet is expanded by adding the farthest point in its outside set (see Figure 4). During the expansion, the facet is replaced by a set of new facets, and each point in OWS formerly associated with the removed facet is associated with one of the new facets. It is important to notice that these points can only be associated with one of the new facets, and it is therefore not necessary to search the whole set of facts in the partial convex hull.

This expansion process is iterated until the hyperplane associated with the vertex with the smallest positive inflation

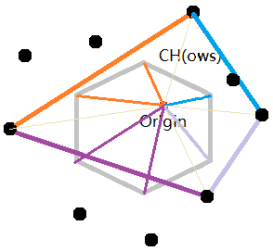

Fig. 4. Expansion set. The green facet in Figure 3 is removed to grow the convex hull towards the farthest vertex in its outside cell. Two new facets are created and the vertices of OWS formerly associated with the green face are now associated with the two new facets.

rate has an empty outside set. At that point the computation can be stopped because the smallest inflation rate has then been determined and this value is indeed Q_{OWS}.

Algorithm 1 sketches the pseudocode for the strategy we just described. In line 1 we create the initial simplex $\mathcal{C}$ as in the QuickHull algorithm. Next, (loop from line 3 to 13) we iteratively expand the partial convex hull until the origin is included. This condition is satisfied when the offset of the hyperplane supporting every facet is not negative. If this is not the case, the simplex is expanded by selecting among the facets with a non-empty outside set the one with the smallest offset (lines 11 and 12). This heuristic accelerates the process. The initial assignment of vertices of CH(OWS) to the facets is performed in lines 14-17 through a search over the set of facets in the partial convex hull. For each facet $\mathcal{F}$, $r_{min}(\mathcal{F})$ is the smallest inflation factor among all vertices associated with the facet. Then, in the final loop (lines 18-25) the convex hull is iteratively expanded. At each iteration the facet with the smallest value for $r_{min}(\mathcal{F})$ is expanded, if possible, (line 19-21) and after the expansion the assignment of vertices to facets is updated. If the facet with the smallest value for $r_{min}(\mathcal{F})$ cannot be expanded, then the algorithm terminates and returns the corresponding inflation value (line 25).

Algorithm 1 Partial Quick Hull for Q_{OWS} algorithm

1: Create intial simplex $\mathcal{C}$ with 7 points
2: OriginInside $\leftarrow$ **false**
3: **while not** OriginInside **do**
4: FCflag$\leftarrow 1$
5: **for all** $\mathcal{F} \in \mathcal{C}$ **do**
6: **if** $o((\mathcal{F}, 0) < 0$ **then**
7: FCflag$\leftarrow 0$
8: **if** FCflag=1 **then**
9: OriginInside $\leftarrow$ **true**
10: **else**
11: $\mathcal{F}_{set} = \{\mathcal{F} \mid outsideset(\mathcal{F}) \neq \emptyset\}$
12: $\mathcal{F}_e \leftarrow \arg \min_{\mathcal{F} \in \mathcal{F}_{set}} o(\mathcal{P}(\mathcal{F}), 0)$
13: Expand $\mathcal{F}_e$ and update $\mathcal{C}$
14: **for all** facets $\mathcal{F}_j \in \mathcal{C}$ **do**
15: **for all** $\mathbf{v_i} \in \mathbf{V}(\text{CH(OWS)})$ **do**
16: Compute $r_{i,j}$ as per Eq. 3.
17: Associate each $\mathbf{v}_i$ to the facet $\mathcal{F}_j$ with smallest $r_{i,j}$
18: **loop**
19: $\mathcal{F}_e \leftarrow \arg\min_{\mathcal{F}} r_{min}(\mathcal{F})$
20: **if** $outsideset(\mathcal{F}_e) \neq \emptyset$ **then**
21: Expand $\mathcal{F}_e$
22: Reassign vertices associated with $\mathcal{F}_e$ to the new facets $(\mathcal{F}_{new})$
23: Calculate r_{min} for the new facets $(\mathcal{F}_{new})$
24: **else**
25: **return** $r_{min}(\mathcal{F}_e)$

V. Evaluation

In this section we experimentally evaluate our algorithm for the computation of Q_{OWS}. Our code is based on the public available QuickHull implementation[4] and is freely available on our website[5] together with the datasets used to generate the results presented in this section.

We compare our algorithm against two alternative methods to compute Q_{OWS}. The first one implements the brute force strategy formerly illustrated. The method is evidently slow, but produces the exact result. The second method implements the principle described in [6] to approximate Q_{OWS}. Therein the authors propose to approximate OWS with an ellipsoid and to then transform it into a sphere through a linear transformation. After this transformation the grasp quality is determined as the maximum radius of the sphere so that it is fully contained in GWS, and then the complexity of the method is equivalent to computing the Q measure. It is important however to recall that this approach just provides and approximation and not the exact value for Q_{OWS}. Since this method ultimately requires to compute the Q measure, we consider two different (although equivalent) implementations, i.e., one based on the QuickHull library and the other based on our recent much faster algorithm PQH. Figure 5 shows the time comparison for the various methods over 100 different grasps for the same OWS. Note that every method requires the preliminary computation of OWS, so this time is excluded from the comparison because is done upfront once for all for all algorithms. Our method (blue line) largely outperforms the brute force exact method (red line) and is faster than the implementation using QuickHull. It is slightly slower than the approach based on PQH, but both provide an approximation and not the exact result. In addition, one should consider that to the best of our knowledge the quality of such approximation is not known. In conclusion, the figure demonstrates that the method we propose is clearly to be preferred because it provides the exact result and its computational requirements are comparable with the faster of the implementations for the approximate methods.

The next experiment is to show that grasps evaluated using Q or Q_{OWS} are indeed ranked differently and this has practical importance. Figure 6 shows two different grasps (in red and blue) each with four contact points. The grasp in blue is favored by Q_{OWS} whereas the one in red is preferred

[4] http://www.qhull.org/
[5] http://robotics.ucmerced.edu

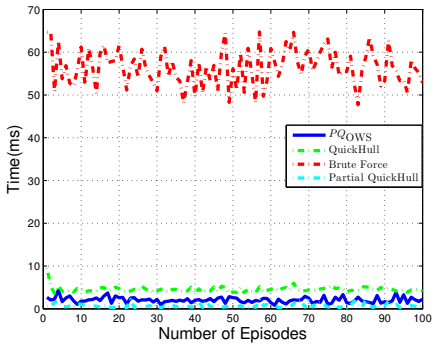

Fig. 5. Time comparison between our method, Quickhull and the brute force algorithm.

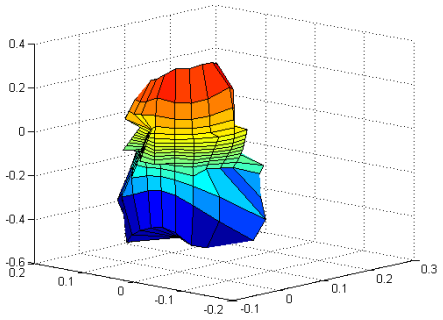

Fig. 8. Disturbance force surface for the grasp favored by Q^{-}.

by the Q measure. However, the intensity of the maximum disturbance that can be resisted by the blue grasp is 0.0604, while the scale is 0.0514 for the grasp in red. The quality of the two grasps can be seen in figures 7 and 8 where for every direction in three dimensions we plot the intensity of the maximum force that can be resisted by the blue grasp (Figure 7) and by the red grasp (Figure 8). The surfaces show that the blue grasp can resist a much larger set of forces, as indicated by the larger volume enclosed by the surface.

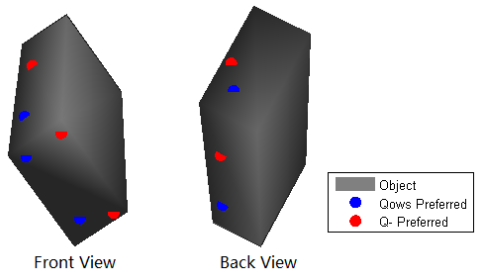

Fig. 6. Two grasps either favoring Q_{OWS} or Q^{-}.

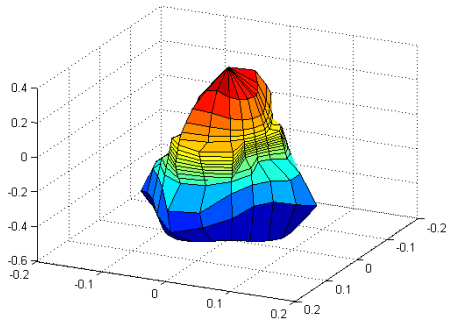

Fig. 7. Disturbance force surface for the grasp favored by Q_{OWS}.

VI. Conclusions

In this paper we have presented a novel algorithm to compute a grasp quality metric force based on the object wrench space. This metric Q_{OWS} considers only the disturbance wrenches that may be generated by disturbance forces, so it is less conservative than the more commonly used Q measure. However, since it is computationally more demanding, it is rarely used in practice. Our algorithm exploits some insights from computational geometry and allows to compute Q_{OWS} through a partial computation of a convex hull. The method does not introduce any approximation and its performance is comparable with the more advanced methods used to compute Q. We believe that this new algorithmic advancement will enable to integrate the Q_{OWS} into the grasp planning cycle, thus generating grasp configurations that will be more robust and useful in an automation environment.

References

[1] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295.

[2] M. Strandberg and B. Wahlberg, "A method for grasp evaluation based on disturbance force rejection," *IEEE Transactions on Robotics*, vol. 22, no. 3, pp. 461–469, 2006.

[3] 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, pp. 4279–4285.

[4] C. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 469–483, 1996.

[5] N. Pollard, "Parallel methods for synthesizing whole-hand grasps from generalized prototypes," Ph.D. dissertation, MIT, 1994.

[6] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: how to choose a suitable task wrench space," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006, pp. 319–325.

[7] H. Jeong and J. Cheong, "Evaluation of 3D grasps with physical interpretations using object wrench space," *Robotica*, vol. 30, no. 3, pp. 405–417, 2012.

[8] Z. Li and S. Sastry, "Task oriented optimal grasping by multifingered robot hands," *IEEE Journal of Robotics and Automation*, vol. 4, no. 1, pp. 32–44, 1988.

[9] D. Pratticchizzo and J. Trinkle, "Grasping," in *Handbook of robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, ch. 28, pp. 671–700.

[10] K. Hang, J. Stork, F. Pokorny, and D. Kragic, "Combinatorial optimization for hierarchical contact-level grasping," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 381–388.

[11] S. Liu and S. Carpin, "Global grasp planning using triangular meshes," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015, pp. 4904–4910.