Partial Convex Hull Algorithms for Efficient Grasp Quality Evaluation

Shuo Liu[a], Stefano Carpin[a,*]

[a]*School of Engineering, University of California, Merced*
5200 N Lake Rd, Merced, CA, 95343, USA

Abstract

We present two algorithms to efficiently determine the value of two grasp quality metrics formerly proposed in literature. The first one is heavily used in practice but has some drawbacks, e.g., it is not scale invariant and it does not focus on the disturbance forces that will occur in practice when the robot grasps an object. The second one overcomes these limitations, but is rarely used because it is computationally too demanding. The two algorithms we propose are based on the common intuition that both metrics can be efficiently computed through a modified version of the QuickHull algorithm that is commonly used to compute convex hulls. In both cases it is possible to establish when enough information has been generated to determine the desired value, and then stop the construction of a suitably defined convex hull. Extensive numerical evaluations demonstrate that our algorithms provide substantial computational gains when compared with the state of the art. The speedup provides an immediate benefit to planners using grasp quality metrics to guide the search through the space of possible grasps.

Keywords: Grasp quality metrics; grasp planning; computational geometry.

1. Introduction

The ability to restrain an object with a multifingered robotic hand is a critical component in numerous robotics and automation processes [1–4]. When a geometrical model of the object is available, a grasp planning algorithm can be used to determine suitable contact points and forces. In general, for a given object and robotic hand there exist multiple configurations that can achieve a successful grasp, and therefore a quality metric is often used to select which one should be used, or to inform the search through the space of possible grasps. Algorithms capable of efficiently evaluating the quality of a candidate grasp are therefore essential to accelerate the planning process. The grasp metrics we consider in this paper evaluate the ability to resist an external disturbance to the object being grasped, i.e., we consider so-called force closure grasps. Various grasp quality evaluation functions have been proposed in literature, but in practice just a few have been extensively used.

In this sequel we present two algorithms that significantly expedite the computation of two formerly proposed quality metrics. The first was introduced by Ferrari and Canny [5]. It relies on the construction of a convex hull in six dimensions and it is extensively used in practice – in fact it is perhaps the most used. Notwithstanding, this metric has multiple drawbacks. In particular, it is not scale invariant, it does not consider the shape of the object being grasped, and it is somehow too conservative because it considers the set of all possible disturbances, including those that will hardly occur in practice. In the following we refer to this measure as the grasp wrench space (GWS) metric. The second metric we examine was proposed by Strandberg and Wahlberg [6] and, at least in principle, it offers various advantages when compared to [5]. In particular, it considers the shape of the object, it is scale invariant, and it focuses on disturbances that may happen in practice. However, it has been so far rarely used because it is computationally demanding. In the following we refer to this measure as the object wrench space (OWS) metric because it considers only

*Corresponding author.

the disturbance wrenches that can occur on the object being restrained. Our algorithms are built upon a common observation, i.e., that both metrics can be determined by computing a partial convex hull. That is to say that both algorithms start building a six dimensional convex hull, but exploiting some computational geometry insights they opportunistically identify when the computation can be stopped because enough information has been generated to compute the needed values. We dub this approach "partial quick hull computation."

The contributions of this paper are four. First: we present an algorithm (PQHGWS – Partial Quick Hull for Grasp Wrench Space) to efficiently compute the metric proposed in [5]. The algorithm can be applied both for force closure and non-force closure grasps. In the former case it returns the grasp quality measure, whereas in the latter case it returns the distance of the convex hull from the origin – a value that can be used to guide the planning process. The algorithm presented in this paper extends and improves our former contribution [7], where the speedup occurred only for the force closure case. Second: we present an algorithm (PQHOWS – Partial Quick Hull for Object Wrench Space) to efficiently compute the metric proposed in [6]. PQHOWS is orders of magnitude faster than the naive brute force method. If the grasp being evaluated does not achieve force closure, the metric is not defined and PQHOWS quickly identifies this case returning an appropriate flag. Thanks to these improvements, the metric proposed in [6] can now be used in lieu of [5] during grasp planning. Third: we provide the theoretical foundations sustaining the correctness of our methods. Fourth: we show with extensive simulations that the algorithms we developed offer significant advantages when compared with the state of the art. The code implementing our findings is made freely available to the scientific community, together with the datasets used to generate the results presented in this paper[1].

The remainder of this manuscript is organized as follows. Related work is presented in Section 2. Background material and notation regarding convex hulls is given in Section 3, whereas in Section 4 we formally define the metrics we are targeting. Our two algorithms are presented in Section 5 and 6. Simulations substantiating the computational advantages of our methods are given in Section 7, and conclusions are presented in Section 8.

2. Related Work

We provide a short recap of relevant literature in grasp quality metrics and convex hulls. Throughout this paper we exclusively consider force closure grasps (as formalized in Section 4).

2.1. Grasp Quality Metrics

Grasping has a rich history dating back to the dawn of robotics. Everyday experience suggests that objects can be restrained in different ways and it is therefore natural asking which grasping configuration should be preferred. Being able to quickly evaluate the goodness of a metric impacts also grasp planning efficiency, because planning can be seen as a search in the space of possible grasps driven by the quality metric [8, 9].

With these motivations, various grasp quality metrics have been proposed and the reader is referred to [10] for a recent survey. Given two force closure grasps, one would intuitively prefer the one that can balance disturbances with the minimal effort, i.e., applying the smallest forces. This idea was originally introduced by Kirkpatrick et al. [11] and later on refined by Ferrari and Canny [5]. This metric is obtained computing the convex hull of a set of wrenches, i.e., six-dimensional vectors (see section 4.1 for more details). This grasp quality measure has de-facto become the most frequently used grasp metric, and the QuickHull algorithm [12] is commonly used to compute the corresponding convex hull. Despite its popularity, this quality measure has some notable drawbacks. To be specific, it is not scale invariant, it does not consider the geometry of the object being grasped, and its value depends on an arbitrary choice for the point about which torques are computed. Moreover, it is too conservative, in the sense that it considers the set of all possible disturbance wrenches without instead considering just those that may occur in practice. Finally,

[1] Code and data are available for download at `https://github.com/ucmrobotics/PQH`

its computation relies on an approximation of the friction cone, and in order to expedite the computation there is then an incentive to use coarse approximations. Despite these limitations, it nevertheless continues to be the most commonly used.

In an effort to overcome some of these drawbacks, alternative approaches have been introduced. Zheng proposed to replace QuickHull with an algorithm that iteratively approximates the convex hull by growing a polytope guaranteed to be inside the convex hull [13]. Differently from PQHGWS, this algorithm cannot be applied when evaluating grasps that do not achieve force closure, and this is a limitation because when a grasp is not force closure the distance between the hull and the origin is still useful to guide the planning process [14]. The algorithm presented in [13] provides an approximation of the convex hull and the approximation error converges to zero only asymptotically with an unknown convergence rate. Pokorny and Kragic [15] provide the first accurate study of the analytical properties of the GWS metric proposed by Ferrari and Canny and of its approximations. Starting from this study, they propose an efficient algorithm to recognize and reject non-force closure grasps, but this method does not rank grasps achieving force closure, i.e., it does not compute a grasp quality metric. Importantly, [15] presents an analytic bound for the approximation error introduced by the discretization of the friction cone, and we exploit their result to quantify the quality of our findings. Various methods have been proposed with the objective of removing the approximation introduced discretizing the friction cone [14, 16]. None of these has however gained much traction, mostly because of the associated computational costs.

With the objective of considering not all disturbance wrenches, but only those occurring in practice, the object wrench space (OWS) was introduced [17]. Borst et al. [18] proposed a method to approximate the object wrench space computing an enclosing ellipsoid rather than its exact shape. This ellipsoid is then transformed into a sphere using a linear transformation and then the GWS metric is used to assess the quality of the grasp. This method however is approximate and there can be a significant mismatch between the actual object wrench space and the enclosing ellipsoid. Strandberg and Wahlberg [6] introduce a method for the direct computation of the OWS metric. The key observation is that a disturbance wrench is almost invariably generated by a disturbance force, and they formulate a metric considering how much the object wrench space can be inflated before it reaches the boundary of the grasp wrench space (see Section 4.2 for more details.) This metric however has enjoyed limited use because it is costly to compute and the authors provide a method only slightly better than a brute force approach. A similar method was proposed in [19]. Li and Sastry defined the concept of task wrench space (TWS), i.e., the set of wrenches that can be generated while executing a specific task [20]. This wrench space is even more specific than the OWS (i.e., it is a subset), but it has found limited applications because it is difficult to determine TWS for an arbitrary task [21].

2.2. Convex Hull Computation

The problem of computing the convex hull of n points in $\mathbb{R}^d$ is one of the most studied problems in computational geometry [22, 23]. The computational complexity for convex hull algorithms can be described as a function of three parameters, i.e., the number of points n, the dimensionality of the space d, and the number of facets in the convex hull h. For the planar case, an $\Omega(n \log n)$ lower bound is easy to prove, and there exist various optimal algorithms matching this bound [24, 25]. Vertices in the convex hull are a subset of the set of input points. This has motivated the development of algorithms with *output sensitive* complexity. In [26, 27] two algorithms with complexity $\mathcal{O}(n \log h)$ have been proposed, where h is the number of vertices in the resulting convex hull, i.e., the size of the output. Note that in $\mathbb{R}^2$ the number of vertices and facets in the convex hull are the same. For the $\mathbb{R}^2$ case, $\mathcal{O}(n \log h)$ algorithms are known to be optimal [26]. In $\mathbb{R}^3$, the convex hull of n points can still be computed in $\mathcal{O}(n \log n)$ [28] and the number of facets is still linear in the number of vertices in the hull (see e.g., [22], chapter 11). For $d > 3$ the number of vertices and facets in the resulting convex hull is no longer linear in n. Therefore output sensitive algorithms are not necessarily the best option. If d is constant, then the convex hull can be deterministically computed in $\mathcal{O}(n \log n + n^{\lfloor d/2 \rfloor})$ by an optimal algorithm proposed by Chazelle [29]. Seidel instead proposed an output sensitive algorithm [30] with complexity $\mathcal{O}(n^2 + h \log n)$. To compare these two alternative approaches, it is useful to recall that in $\mathbb{R}^d$ the number of facets h can grow as $\Theta(n^{\lfloor d/2 \rfloor})$ (see e.g., [23]). Randomized incremental algorithms have also been proposed, e.g., [31], and they are optimal in expectation.

The QuickHull algorithm [12] has become the de-facto standard when it comes to computing convex hulls in higher dimensional spaces. QuickHull improves the randomized algorithm presented in [31] by introducing some heuristics that significantly improve its performance. So far, an analytical characterization of its computational performance has remained elusive, but its observed empirical performance suggests an $\mathcal{O}(n \log s)$ complexity, where s is the number of processed vertices. Among the reasons of QuickHull's popularity is the availability of a freely available, highly optimized implementation.[2] Section 3 provides a short recap of its principles.

3. Background in Convex Hulls

We start by summarizing some facts about computational geometry and convex sets. The reader is referred to [22] for more details. As numerous symbols are introduced in the following, the reader is referred to the appendix for a table summarizing them.

Definition 1. *Let $\mathcal{S}$ be a subset of $\mathbb{R}^d$. $\mathcal{S}$ is a* convex *set if for each $x, y \in \mathcal{S}$ and each $\lambda \in [0, 1]$ the point $\lambda x + (1 - \lambda)y$ is an element of $\mathcal{S}$.*

Definition 2. *Let $\mathcal{N}$ be a set of n points in $\mathbb{R}^d$. The* convex hull *of $\mathcal{N}$ is the smallest convex set including $\mathcal{N}$. The convex hull of $\mathcal{N}$ will be indicated as* $\mathrm{CH}(\mathcal{N})$.

Given a set $\mathcal{N}$ of n points in $\mathbb{R}^d$, its convex null $\mathrm{CH}(\mathcal{N})$ is a convex polytope in $\mathbb{R}^d$ that can be represented by its vertices and its facets. Each of its facets $\mathcal{F}_i$ is a convex subset of a $(d-1)$-dimensional hyperplane $\mathcal{H}_i$ and each vertex of $\mathrm{CH}(\mathcal{N})$ is an element of $\mathcal{N}$. The hyperplane $\mathcal{H}_i$ including facet $\mathcal{F}_i$ is also called the hyperplane *supporting* $\mathcal{F}_i$. In the following, we will also write $\mathcal{H}(\mathcal{F})$ to indicate the hyperplane supporting facet $\mathcal{F}$. The hyperplane $\mathcal{H}_i$ splits $\mathbb{R}^d$ in two halfspaces, one of which contains $\mathrm{CH}(\mathcal{N})$. The *inside set* of hyperplane $\mathcal{H}_i$ is defined as the half space including $\mathrm{CH}(\mathcal{N})$, and it will be indicated as $\mathcal{I}(\mathcal{H}_i)$. Similarly, the *outside set* of $\mathcal{H}_i$ is the half space not including $\mathrm{CH}(\mathcal{N})$ and it is indicated as $\mathcal{O}(\mathcal{H}_i)$. We associate to $\mathcal{H}_i$ a unit normal vector $\mathbf{n}_i$ directed towards its outside set $\mathcal{O}(\mathcal{H}_i)$.

Definition 3. *A simplex in d dimensions is a polytope that is the convex hull of its $d+1$ vertices.*

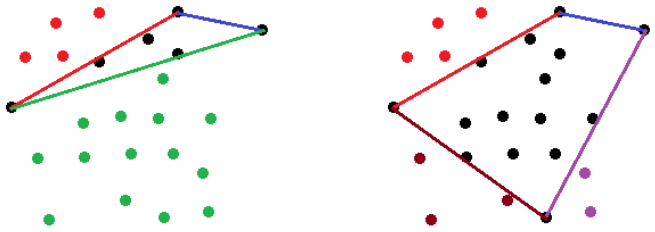
It is known that a simplex in d dimensions has $d+1$ facets.

Next, we shortly recap how the QuickHull algorithm works and we refer the reader to [12] for a more thorough discussion. Assume the n points in $\mathcal{N}$ do not all lie on the same $(d-1)$-dimensional hyperplane. QuickHull computes $\mathrm{CH}(\mathcal{N})$ as follows. An initial simplex is created selecting $d+1$ points from $\mathcal{N}$. If possible, QuickHull picks points with an extreme coordinate, but this is not necessary for the correctness of the algorithm. Every facet $\mathcal{F}_1, \ldots, \mathcal{F}_d, \mathcal{F}_{d+1}$ in the initial simplex is supported by an hyperplane. For every facet $\mathcal{F}_i$, its outside set[3] is defined as the set of all points in $\mathcal{N}$ located in the outside set of the supporting hyperplane $\mathcal{H}_i$. Following the same notation introduced for the supporting hyperplanes, the inside and outside sets of facet $\mathcal{F}_i$ will be indicated as $\mathcal{I}(\mathcal{F}_i)$ and $\mathcal{O}(\mathcal{F}_i)$, respectively. Note that in general a point in $\mathcal{N}$ may belong to more than one outside set, and an outside set can also be empty (see Figure 1.a).

From the initial simplex, the convex hull is iteratively grown as follows. If there exists a facet with a non empty outside set, the convex hull is expanded by growing the convex hull to include the point in its outside set that is the farthest from the facet. This step is called *expansion*. Expansion eliminates one facet, and generates a set of new facets for which their respective outside sets are created (see Figure 1.b). Once the outside sets of all facets are empty, the process terminates. By *greedily* growing the convex hull towards the farthest point in outside set, it was empirically shown that QuickHull outperforms other algorithms utilizing different criteria to expand the current hull, e.g., [31].

[2] `http://www.qhull.org`

[3] Note that in QuickHull the outside set of a facet is given by a set of points, whereas in our general definition $\mathcal{O}(\mathcal{H}_i)$ is an half space. With a slight abuse of notation we use the same term to avoid introducing additional symbols.



(a) (b)

Figure 1: a) QuickHull initialization. In $\mathbb{R}^2$ the initial simplex includes 3 points. The outside set of each facet is displayed using the same color of the facet. Black points are inside the simplex and do not belong to any outside set. b) QuickHull expansion. The green facet is expanded and replaced by two new facets (brown and purple), and their respective outside sets are updated.

Throughout its computation, QuickHull maintains a list of the vertices and facets of the convex hull being built. Moreover, for each facet $\mathcal{F}_i$ it maintains also the normal $\mathbf{n}_i$ associated with the supporting plane $\mathcal{H}_i$.

4. Background in Grasping and Grasp Metrics

To explain our algorithm, we introduce basic facts concerning grasping and grasp metrics, and refer the reader to [32, 33] for more detailed discussions. A rigid body $\mathcal{B}$ is grasped using a multifingered hand with m fingers. A grasp $\mathcal{G}$ is specified by the m contact points $\mathbf{p}_1, \ldots, \mathbf{p}_m$ on the surface of $\mathcal{B}$ together with m contact forces $\mathbf{f}_1, \ldots, \mathbf{f}_m$. It is customary to express the coordinates of the contact points with respect to a frame whose origin O coincides with the center of mass of $\mathcal{B}$. Each force generates a *wrench*, i.e., a six dimensional vector including both the force and the torque with respect to O, i.e., $\mathbf{w}_i = [\mathbf{f}_i \quad \mathbf{p}_i \times \mathbf{f}_i]^T$. To prevent slippage at the contact point, forces are constrained in a given range defined by the assumed friction model. As common in literature, in the following we consider the contact with friction model (also known as hard finger model). More complex contact models can be expressed as multiple contacts of this type, whereas the simpler frictionless model is too restrictive and usually not utilized in practice. Therefore our assumptions is general enough to be useful in many practical scenarios. The hard finger assumption is built upon Coulomb's friction model, i.e., to prevent slippage at the contact point each force must lie inside the friction cone defined at the contact point. More formally, at every contact point $\mathbf{p}_i$ we establish an orthonormal reference frame with vectors $\mathbf{t}_i, \mathbf{u}_i, \mathbf{v}_i$ where $\mathbf{t}_i$ is inward and $\mathbf{u}_i, \mathbf{v}_i$ lie on the plane tangent to $\mathcal{B}$ in $\mathbf{p}_i$. The contact force $\mathbf{f}_i$ can then be written as $\mathbf{f}_i = [f_{i1} \; f_{i2} \; f_{i3}]^T$ where f_{i1} is the component along $\mathbf{t}_i$ and f_{i2}, f_{i3} are the components along $\mathbf{u}_i$ and $\mathbf{v}_i$. The set of forces that do not cause slippage at $\mathbf{p}_i$ is then

$$F(\mathbf{p}_i) = \left\{ \mathbf{f}_i \in \mathbb{R}^3 \mid f_{i1} \geq 0 \wedge \sqrt{f_{i2}^2 + f_{i3}^2} \leq \mu f_{i1} \right\}$$

where μ is the friction coefficient between the object and the finger, and for simplicity we assume it is the same for all fingers. The condition $f_{i1} \geq 0$ implies no separation, whereas the second condition follows Coulomb's friction law. Geometrically, $F(\mathbf{p}_i)$ defines a cone, hence the name friction cone at $\mathbf{p}_i$. From a practical point of view, it is common to discretize the cone as a regular pyramid with k edges (see Figure 2).

Using this approximation each contact force lying inside the cone can be written as a non-negative combination of forces along the boundary of the friction cone, i.e., $\mathbf{f}_i = \sum_{j=1}^{k} \alpha_{i,j} \mathbf{f}_{i,j}$ with $\alpha_{i,j} \geq 0$. Based on this discretization the wrench generated by the ith contact force can then be written as

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{p}_i \times \mathbf{f}_i \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{k} \alpha_{i,j} \mathbf{f}_{i,j} \\ \mathbf{p}_i \times \sum_{j=1}^{k} \alpha_{i,j} \mathbf{f}_{i,j} \end{bmatrix} =$$

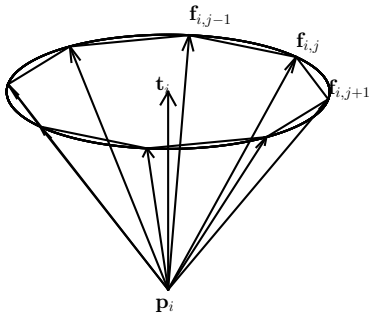


Figure 2: The friction cone $F(\mathbf{p}_i)$ is commonly approximated by a regular pyramid with k edges.

$$= \sum_{j=1}^{k} \alpha_{i,j} \begin{bmatrix} \mathbf{f}_{i,j} \\ \mathbf{p}_i \times \mathbf{f}_{i,j} \end{bmatrix} = \sum_{j=1}^{k} \alpha_{i,j} \mathbf{w}_{i,j}$$

where each of the $\mathbf{w}_{i,j}$ is called *elementary wrench*. The km elementary wrenches $\mathbf{w}_{i,j}$ can be arranged in a $6 \times km$ matrix $\mathbf{G}$ called the *grasp matrix*. The grasp achieves force closure if for each disturbance wrench $\mathbf{w}$ there exists a vector $\mathbf{x} \in \mathbb{R}^{km}$ such that

$$\mathbf{w} = \mathbf{G}\mathbf{x} \quad x_i \geq 0, \quad i = 1, \ldots, km. \tag{1}$$

That is to say, the grasp matrix $\mathbf{G}$ positively spans $\mathbb{R}^6$, and in such case $\mathbf{x}$ contains the $\alpha_{i,j}$ factors for all contact points. The set of wrenches obtained through the grasp matrix $\mathbf{G}$ when the sum of the elements in $\mathbf{x}$ is one is known as the *unit grasp wrench space* (UGWS), and this assumption is often made in practice [5, 6]. In this case it is known that an equivalent condition for force closure is that the convex hull of all the km elementary wrenches $\mathbf{w}_{i,j}$ includes the origin (see e.g., [33]).

4.1. Grasp Wrench Space Metric

Ferrari and Canny formalized the idea that among two different grasps achieving force closure preference should be given to the one capable of resisting an arbitrary external wrench with minimum effort [5]. This intuition is formalized through the following geometric interpretation. Any wrench applied on $\mathcal{B}$ through the m contacts can be scaled to belong to the set

$$W_{L_\infty} = \mathrm{CH}\left(\bigoplus_{i=1}^{m} \{\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,k}\}\right) \tag{2}$$

where $\oplus$ represents the Minkovski sum. From an operative point of view it is worth observing that the inner Minkovski sum in Eq. (2) gives a set of finite elements, and therefore W_{L_∞} is the convex hull of a finite set of elements in $\mathbb{R}^6$. The grasp metric Q_∞ is then defined as the distance of the closest facet of W_{L_∞} from the origin, i.e.,

$$Q_\infty = \inf_{\mathbf{x} \in \partial W_{L_\infty}} ||\mathbf{x}||_2 \tag{3}$$

where ∂W_{L_∞} indicates the boundary of W_{L_∞}, i.e., the union of its facets. Physically, Q_∞ aims at minimizing the maximum force exerted by any of the m fingers to resist an arbitrary external wrench. Alternatively, one could aim at minimizing the sum of forces exerted by all fingers. To this end, consider

$$W_{L_1} = \mathrm{CH}\left(\bigcup_{i=1}^{m} \{\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,k}\}\right). \tag{4}$$

Similarly to Q_∞, the grasp metric Q_1 is then defined as the distance of the closest facet of W_{L_1} from the origin

$$Q_1 = \inf_{\mathbf{x} \in \partial W_{L_1}} ||\mathbf{x}||_2. \tag{5}$$

Note that when $\sum_{i,j} \alpha_{i,j} = 1$, then W_{L_1} is the formerly defined unit grasp wrench space. The reader is referred to [5] for a thorough explanation of the relationship between the geometrical and physical interpretation of Q_∞ and Q_1. It is also important to remark that the two measures are not equivalent, i.e., a grasp that is optimal according to Q_1 may not be optimal according to Q_∞ and viceversa. Hence, from a practical standpoint they are both needed. This metric is related to the ability to resist an arbitrary wrench, and is therefore referred to GWS metric. As evident from Eq. (2) and Eq. (4), both Q_∞ and Q_1 rely on the computation of the convex hull of a set of vectors in $\mathbb{R}^6$, and the connection between grasp quality measures and convex hulls is therefore evident. From a practical standpoint, QuickHull is the algorithm most commonly used to compute these metrics.

4.2. Object Wrench Space Metric

The metric described in the previous subsection is somehow too conservative because it considers the set of all possible disturbance wrenches. But from a practical point of view a disturbance wrench is almost invariably caused by a disturbance force, and it would therefore make sense to accordingly restrict the set of disturbance wrenches to be considered. Starting from this observation, Strandberg and Wahlberg defined a method to evaluate grasps based on the ability to reject a disturbance force [6]. The construction is related to the object wrench space (OWS) introduced by Pollard [17], i.e., the set of wrenches that can be generated by an arbitrary unit force $\mathbf{e}_i$ acting on the surface of the object. Assuming that the surface $\partial\mathcal{B}$ of the object is composed by l triangles, OWS can then be defined as follows

$$\text{OWS} = \bigcup_{\mathbf{e}_i} \bigcup_{\mathbf{v}_{j,c}} \text{CH}\left(\begin{bmatrix} \mathbf{e}_i \\ \mathbf{v}_{j,c} \times \mathbf{e}_i \end{bmatrix} \; j = 1 \dots l, \; c = 1, 2, 3, \; \mathbf{e}_i \in F(\mathbf{v}_{j,c}) \right) \tag{6}$$

where CH indicates the convex hull, $\mathbf{v}_{j,c}$ is the $c-$th vertex of the $j-$th triangle on the triangle mesh, and $\mathbf{e}_i$ is a unit length force vector. Note that in the definition the force $\mathbf{e}_i$ is constrained to be inside the friction cone $F(\mathbf{v}_{j,c})$ at the contact point, and that under the hypothesis that the surface is represented with a triangulation the computation can be limited to the vertices of the mesh [6]. The OWS leads to a different grasp quality metric defined as follows. Assume a grasp $\mathcal{G}$ is given together with its associated grasp matrix $\mathbf{G}$. Let $\mathbf{e}$ be a unit vector specifying the direction of a disturbance force. All disturbance forces in that direction can be written as $f\mathbf{e}$ with $f \geq 0$ to avoid tractional forces. By *sweeping* $\mathbf{e}$ over the surface of $\mathcal{B}$ it is possible to characterize all disturbance wrenches generated by forces parallel to $\mathbf{e}$. Let f^* be the smallest value such that the associated wrench is exactly on the border of the unit grasp wrench space. Varying $\mathbf{e}$, one gets to the following min-max formulation

$$\begin{aligned} f^* = \min_{\mathbf{p}} \max_{\mathbf{x} \in \mathbb{R}^{km}} \Bigg\{ & f \in \mathbb{R}^+ : -f \begin{bmatrix} \mathbf{e} \\ \mathbf{p} \times \mathbf{e} \end{bmatrix} = \mathbf{G}\mathbf{x} \\ & \mathbf{e} \in F(\mathbf{p}), \; \mathbf{p} \in \partial\mathcal{B}, \; x_i \geq 0, \; \sum_{i=1}^{km} x_i = 1 \Bigg\} \end{aligned}$$

where the contact point $\mathbf{p}$ is constrained to be on the surface $\partial\mathcal{B}$ of the object being grasped, and the force $\mathbf{e}$ is constrained to be inside the friction cone $F(\mathbf{p})$ at the contact point. f^* is then the intensity of the maximum disturbance force that can be resisted by the unit grasp $\mathcal{G}$ associated with $\mathbf{G}$ and can be used to define the grasp quality metric

$$Q_{\text{OWS}} = \max\{r > 0 \mid r \cdot \text{OWS} \subseteq \text{GWS}\} \tag{7}$$

where $r \cdot \text{OWS}$ is the set obtained multiplying all elements of OWS by r. In the following, r is also referred to as *inflation factor* because its role is to expand or shrink OWS. From a computational standpoint, to consider all possible directions for the disturbance force one expresses $\mathbf{e}$ in spherical coordinates and then accordingly samples the associated space of angles φ, θ (see [6] for details.) Note that in the original formulation an additional wrench $\mathbf{w}_0$ accounting for the gravity force is included, but without loss of generality, we do not consider it here.

The advantages of this metric are manifold. It is scale and reference invariant, it explicitly includes the shape of the object being grasped, and it considers just the wrenches that appear in practice. Moreover, one can determine the optimal f value varying $\mathbf{e}$ over the unit sphere, thus leading to a graphical interpretation in three dimensions that is easy to visualize and understand (see [6] for examples.) The major downside is that the method is computationally demanding and therefore it is not practical to use it to guide a search over the space of possible grasps.

5. Grasp Wrench Space Metric with Partial Convex Hulls

In this section we show how the computation of the grasp metric presented in section 4.1 can be greatly accelerated. When considering a force closure grasp, both Q_1 and Q_∞ are defined as the radius of the largest ball centered in the origin and fully inside a suitably defined convex hull. If instead the grasp is not force closure, the origin is outside the hull and the grasp quality measure is not defined. However, the distance between the origin and the convex hull still provides valuable information because it indicates how far a grasp is from achieving force closure [14]. Therefore, for both force and non-force closure grasps, Eq. (3) and (5) provide an informative value. In both cases the computation of the entire convex hull is not necessary, since the distance, and then the metric, is exclusively determined by the closest facet of the hull. The algorithm we propose, dubbed Partial Quick Hull for grasp wrench space (PQHGWS from now onwards) builds upon this observation. From a practical standpoint, the convex hull is iteratively grown using the same principles used in QuickHull, but its computation is stopped when the closest facet is determined. The following lemmas prove the correctness of the proposed approach and introduce relevant quantities we will use when describing the algorithm.

Definition 4. *Let $\mathcal{H}$ be a hyperplane in $\mathbb{R}^d$ and $\mathbf{y}$ a point in $\mathbb{R}^d$. The Euclidean distance between $\mathcal{H}$ and $\mathbf{y}$ is indicated as $d(\mathcal{H}, \mathbf{y})$.*

Definition 5. *An oriented hyperplane is an hyperplane $\mathcal{H}$ associated with a unary vector $\mathbf{n}_\mathcal{H}$ orthogonal to $\mathcal{H}$.*

The role of $\mathbf{n}_\mathcal{H}$ is to identify one of the two halfspaces defined by $\mathcal{H}$. As QuickHull and PQHGWS incrementally build the resulting convex hull, both algorithms maintain for each facet $\mathcal{F}$ an orthogonal unit-length vector pointing into the outside set of the supporting hyperplane $\mathcal{H}$.
Assumption: from now onwards, when considering a convex hull, the hyperplanes supporting its facets will always assumed to be oriented with a unit vector pointing into the outside set.

Definition 6. *Let $\mathcal{H}$ be an oriented hyperplane in $\mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^d$. We define $o(\mathcal{H}, \mathbf{y})$ (offset from $\mathbf{y}$ to hyperplane $\mathcal{H}$) as*

$$o(\mathcal{H}, \mathbf{y}) = d(\mathcal{H}, \mathbf{y})\text{sgn}((\mathbf{p}_\mathcal{H} - \mathbf{y}) \cdot \mathbf{n}_\mathcal{H})$$

where $\mathbf{p}_\mathcal{H}$ is any point on the hyperplane $\mathcal{H}$, $\cdot$ is the dot product, and sgn *is the signum function.*

From the above definitions it follows that $d(\mathcal{H}, \mathbf{y})$ is always non-negative, whereas $o(\mathcal{H}, \mathbf{y})$ can be positive, negative, or zero. The offset $o(\mathcal{H}, \mathbf{y})$ can be interpreted as a *signed distance*, i.e., it is $d(\mathcal{H}, \mathbf{y})$ when $\mathbf{y}$ is in the inside set, whereas it is $-d(\mathcal{H}, \mathbf{y})$ when $\mathbf{y}$ is in the outside set.

Definition 7. *Let $\mathcal{F}$ be a facet of a convex hull and $\mathbf{y}$ be a point in $\mathbb{R}^n$. The distance between $\mathbf{y}$ and $\mathcal{F}$ is*

$$d(\mathcal{F}, \mathbf{y}) = \min_{\mathbf{x} \in \mathcal{F}} ||\mathbf{y} - \mathbf{x}||_2.$$

Definition 8. *Let $\mathcal{F}$ be a facet on the convex hull, $\mathcal{H}$ be the hyperplane supporting $\mathcal{F}$, and $\mathcal{H}(\mathbf{y})$ the projection[4] of $\mathbf{y}$ onto $\mathcal{H}$. We define $o(\mathcal{F}, \mathbf{y})$ (offset from point $\mathbf{y}$ to facet $\mathcal{F}$) as*

$$o(\mathcal{F}, \mathbf{y}) = \begin{cases} o(\mathcal{H}, \mathbf{y}) & \text{if } \mathcal{H}(\mathbf{y}) \in \mathcal{F} \\ \text{sgn}(o(\mathcal{H}, \mathbf{y})) \cdot d(\mathcal{F}, \mathbf{y}) & \text{otherwise.} \end{cases}$$

[4]The projection of $\mathbf{y}$ into $\mathcal{H}$ is the point obtained by the intersection between $\mathcal{H}$ and the line through $\mathbf{y}$ orthogonal to $\mathcal{H}$.

Starting from these definitions we can state the two lemmas supporting the PQHGWS algorithm.

Lemma 1. *Let* $\mathrm{CH}(\mathcal{N})$ *be the convex hull of* n *points in* $\mathbb{R}^d$ *and* $\mathbf{x}$ *be a point in* $\mathbb{R}^d$*. If* $\mathbf{x}$ *is inside* $\mathrm{CH}(\mathcal{N})$*, then the offset from point* $\mathbf{x}$ *to all the oriented hyperplanes supporting all facets of* $\mathrm{CH}(\mathcal{N})$ *is larger than 0. If* $\mathbf{x}$ *is outside* $\mathrm{CH}(\mathcal{N})$*, there exists at least one facet of* $\mathrm{CH}(\mathcal{N})$ *for which* $\mathbf{x}$ *has negative offset from the supporting hyperplane.*

Proof. $\mathrm{CH}(\mathcal{N})$ is the intersection of all the inside sets of the hyperplanes supporting its facets (see e.g., [22]), i.e.,

$$\mathrm{CH}(\mathcal{N}) = \bigcap_i \mathcal{I}(\mathcal{H}_i)$$

where $\mathcal{H}_i$ is the supporting plane for the i-th facet and i varies over all the facets of $\mathrm{CH}(\mathcal{N})$. If $\mathbf{x}$ is inside $\mathrm{CH}(\mathcal{N})$ then it is in the inside set of all the hyperplanes supporting its facets. As per definition 6, $o(\mathcal{H}_i, \mathbf{x})$ is then positive. If $\mathbf{x}$ is outside $\mathrm{CH}(\mathcal{N})$, then there exists at least one facet $\mathcal{F}_i$ such that $\mathbf{x} \notin \mathcal{I}(\mathcal{H}_i)$. Therefore $\mathbf{x}$ is in the outside set of $\mathcal{H}_i$ and then its offset $o(\mathcal{H}_i, \mathbf{x})$ is negative, as per definition 6. □

Lemma 2. *Let* $\mathbf{x}$ *be inside* $\mathrm{CH}(\mathcal{N})$ *and let* $\mathcal{F}$ *be a facet of* $\mathrm{CH}(\mathcal{N})$ *with the smallest value of* $d(\mathcal{F}, \mathbf{x})$*. Then* $d(\mathcal{F}, \mathbf{x}) = o(\mathcal{F}, \mathbf{x})$*, i.e., the distance of* $\mathbf{x}$ *from* $\mathcal{F}$ *is equal to the offset of* $\mathbf{x}$ *from* $\mathcal{F}$*.*

Proof. Let $\mathcal{H}$ be the plane supporting $\mathcal{F}$. Observe that it must be $\mathcal{H}(\mathbf{x}) \in \mathcal{F}$, i.e., the projection of $\mathbf{x}$ onto the plane supporting $\mathcal{F}$ must be inside $\mathcal{F}$. If this were not the case, then the line through $\mathbf{x}$ and orthogonal to $\mathcal{H}$ would intersect another supporting plane before intersecting $\mathcal{H}$ outside $\mathcal{F}$. However, this would contradict the hypothesis that $\mathcal{F}$ is the closest facet. Then the claim follows as per the first case in definition 8. □

Algorithm 1 sketches the details of PQHGWS. The algorithm creates the initial simplex as in QuickHull (line 1). Inside the main loop the algorithm looks for the next facet to expand, $\mathcal{F}_e$. The first for loop (line 4 to 6) establishes if the grasp can be determined to be force closure or not. If the grasp is force closure, as indicated by the boolean variable FCflag, the facet to be expanded is the one with the smallest offset from the origin (line 8). Note that in this case all facets have positive offset, so the search for the facet to expand happens of the whole set of facets. Instead, if the grasp cannot be yet determined to be force closure, the next facet to expand is the one with the largest offset among those with negative offset (line 10). In both cases, if the outside set of the facet to expand is empty, then the closest facet has been computed and the computation terminates (line 12). Otherwise, the facet is expanded (line 14), and a new loop starts.

Algorithm 1 PQHGWS algorithm

```
1:  Create initial simplex CH with d + 1 points
2:  loop
3:    FCflag ← true
4:    for all F ∈ CH do
5:      if o(H(F), 0) < 0 then
6:        FCflag ← false
7:    if FCflag then
8:      F_e ← arg min o(H(F), 0)
9:    else
10:     F_e ← arg max{o(F, 0) | o(F, 0) < 0}
11:   if O(F_e) = ∅ then
12:     return o(F_e, 0)
13:   else
14:     Expand F_e
```

It is immediate to observe that the algorithm is guaranteed to terminate, because at every additional iteration it expands one facet, so eventually all facets will have an empty set.

5.1. Computational Complexity

QuickHull is observed to be competitive in practice, but its computational complexity has not been formally determined. To date, its performance characterization is mostly empirical. Along the same lines, no accurate computational complexity analysis is available for PQHGWS, although one can say that by construction QuickHull's (conjectured) computational complexity $\mathcal{O}(n \log s)$ is necessarily an upper bound for PQHGWS.

6. Object Wrench Space Metric with Partial Convex Hulls

In this section we present an algorithm to efficiently compute the grasp metric based on the object wrench space described in Section 4.2. Before starting, it is important to recall that Q_{OWS} is defined only for force closure grasps, therefore when computing its value one has to verify whether the GWS associated with the grasp being evaluated includes the origin or not, otherwise the metric is not defined. Our algorithm includes a greedy initialization step aiming at determining whether the grasp is force closure or not. In the latter case, the computation terminates indicating that the metric is undefined. As per Eq. (7), to compute Q_{OWS} one needs to determine the inflation factor r, such that the boundary of OWS touches the boundary of GWS. If Q_{OWS} is used to guide a grasp planner, the shape of the object is fixed, so OWS can be precomputed once for all and does not change. On the contrary, every time a new grasp is evaluated a new GWS is needed. The following theorem provides a first step towards simplifying the computation of the metric.

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

$$\max\{r \mid r \cdot CH(OWS) \subseteq GWS\} = \max\{r \mid r \cdot OWS \subseteq GWS\}.$$

Proof. By definition, OWS $\subseteq$ CH(OWS). Therefore

$$\max\{r | r \cdot \text{CH(OWS)} \subseteq \text{GWS}\} \leq \max\{r | r \cdot \text{OWS} \subseteq \text{GWS}\}.$$

To see that the inequality cannot be strict, we have to remember that each vertex of CH(OWS) is also a vertex of OWS. Let

$$r^* = \arg\max\{r | r \cdot \text{CH(OWS)} \subseteq \text{GWS}\}.$$

Since GWS and CH(OWS) are both convex sets by construction, the intersection between $r^* \cdot$ CH(OWS) and GWS will always occur at a vertex[5] of CH(OWS). Since such vertex is also a vertex of OWS, that is also the optimal inflation rate for OWS, and then the inequality cannot be strict.

□

According to Theorem 1 we can conclude that the largest inflation factor r is obtained when one of the vertices of $r \cdot$ CH(OWS) intersects one of the facets of GWS. Note that this statement is true in general, i.e., also when the intersection happens between two vertices, or between two facets. Therefore the expression to compute Q_{OWS} can be further rewritten as

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

where $\mathcal{V}$(CH(OWS)) is the set of vertices defining the convex hull of OWS. This last expression can be used to derive a brute force algorithm to compute Q_{OWS}, i.e., for each vertex $\mathbf{v}_i \in \mathcal{V}$(CH(OWS)) and each facet $\mathcal{F}_j$ in GWS we can compute

$$r_{i,j} = \frac{o(\mathcal{H}_j, 0)}{\mathbf{v}_i \cdot \mathbf{n}_{\mathcal{H}_j}} \tag{8}$$

[5] In the special case in which the intersection happens between two parallel facets, the reasoning still holds considering one vertex of the facet of CH(OWS).

where $o(\mathcal{H}_j, 0)$ is the offset from the origin of the hyperplane $\mathcal{H}_j$ supporting $\mathcal{F}_j$ and $\mathbf{n}_{\mathcal{H}_j}$ is the outward normal to $\mathcal{H}_j$. $r_{i,j}$ is the factor by which which we need inflate vertex $\mathbf{v}_i$ so that it lies on hyperplane $\mathcal{H}_j$ (note that this value may also be negative). Based on these values, the needed metric is then

$$Q_{\text{OWS}} = \min\{r_{i,j} | r_{i,j} > 0\}. \tag{9}$$

While correct, the brute force method is not practical because it has complexity $\Theta(V_{\text{OWS}} H_{\text{GWS}})$ 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. To fully comprehend the limitations of this approach, it is useful to recall [23] that the convex hull of n points in $\mathbb{R}^d$ can include up to $\Theta(n^{\lfloor d/2 \rfloor})$ hyperplanes and since we are operating in $\mathbb{R}^6$ it follows that H_{GWS} can grow as $\Theta(V_{\text{GWS}}^3)$ where V_{GWS} is the number of vertices in GWS. Another computational challenge comes from the discretization of the friction cones. To reduce the impact of the approximation, one has interest in increasing k, i.e., the number of edges in the pyramid approximating the cone (see Figure 2). But increasing k further increases V_{GWS}. Due to these computational costs, the Q_{OWS} metric has been so far rarely used in practice during the planning stage. In the following we propose an efficient algorithm to compute Q_{OWS} that is based on principles similar to the partial convex hull computation we introduced in Section 5. This expedient greatly accelerates the computation.

Starting from the observation that CH(OWS) can be precomputed upfront, the idea is to iteratively grow GWS and to stop the process as soon as the correct value of Q_{OWS} can be determined. Recall that by definition GWS is a convex hull, so this iterative growing step has some aspects in common with the algorithm for PQHGWS. To determine when the computation can be stopped, each vertex of OWS is assigned to one of the facets in the partial convex hull of GWS being computed. The association is done so that the inflation factor in Eq. (8) is computed only between associated vertices and facets, and not among all possible vertices and facets. The main challenge, of course, is in efficiently establishing and updating these associations while the convex hull is iteratively growing with facets being added and deleted. The following lemma provides the foundation for our algorithm.

Lemma 3. *Let $\mathcal{V}$ be the set of vertices in* CH(OWS) *and H be the set of hyperplanes supporting the facets of the convex hull* GWS. *If $\mathbf{v}_i \in \mathcal{V}$ and $\mathcal{H}_j \in H$ achieve the minimum in the expression given by Eq. 9, then $r_{i,j}\mathbf{v}_i$ lies in the facet supported by $\mathcal{H}_j$.*

Proof. According to the definition of $r_{i,j}$ and elementary geometry, $r_{i,j}\mathbf{v}_i$ lies on $\mathcal{H}_j$ for every point $\mathbf{v}_i$ and every hyperplane $\mathcal{H}_j$. Let $\mathbf{v}'_i = r_{i,j}\mathbf{v}_i$ be such point, and by contradiction assume $\mathbf{v}'_i$ is not on a facet of GWS. Then there must exist a positive $r'_{i,j} < r_{i,j}$ such that $r'_{i,j}\mathbf{v}_i$ intersects a facet $\mathcal{F}_k$ supported by hyperplane $\mathcal{H}_k$. Then by definition $r'_{i,j}$ must be the inflation factor between vertex $\mathbf{v}_i$ and hyperplane $\mathcal{H}_k$, i.e., $r'_{i,j} = r_{i,k}$. But this contradicts the hypothesis that $r_{i,j}$ achieves the minimum in Eq. (9) and the claim follows. □

The algorithm works as follows. As first step CH(OWS) is computed. Let $\mathcal{V}$(CH(OWS)) be the set of vertices of this convex hull. Then, a simplex for the space of elementary wrenches is initialized with $d+1$ vertices, using the same initialization procedure used by QuickHull. Since we work in $\mathbb{R}^6$ the initial simplex features 7 points. The initial convex hull is then greedily expanded with the facet that has the smallest negative offset until it includes the origin. This precondition is necessary to ensure the correctness of the successive steps. If this expansion fails, i.e., the facet being expanded has an empty outside set, then it is not possible to include the origin, and the grasp is not force closure. Therefore the algorithm terminates because the grasp quality metric is not defined. Figure 3 shows the initialization phase for the simpler $\mathbb{R}^2$ case. The following lemma is a special case of Lemma 3 for the case where just a partial convex hull has been computed. The two proofs are similar and we therefore skip it.

Lemma 4. *Let $\mathcal{V}$ be the set of vertices in* CH(OWS) *and H be the set of hyperplanes supporting the facets of the current convex hull. If $\mathbf{v}_i \in V$ and $\mathcal{H}_j \in H$ achieve the minimum in the expression given by Eq.* (9) *over all hyperplanes in H, then $r_{i,j}\mathbf{v}_i$ lies on the boundary of the current convex hull, i.e., it lies in the facet supported by $\mathcal{H}_j$ and do not intersect with any other hyperplane.*

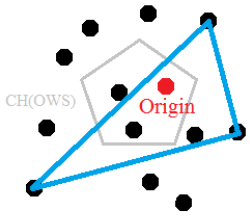


Figure 3: In the initialization stage, CH(OWS) is computed (gray) as well as a partial convex hull over the set of elementary wrenches (black dots). The process stops when the origin is included the partial convex hull (cyan).

Next, associations between vertices of OWS and facets of GWS are determined. Each vertex is associated with the facet 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 4 assures that taking the smallest value computed for each vertex will identify the correct facet. Although this initial association is performed trying all possible vertex-facet combinations, it is in general not time consuming because at this stage the convex hull has just a small number of facets (see Figure 4). 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 factor.

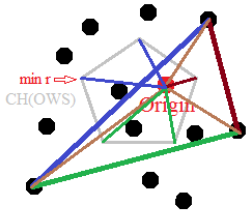


Figure 4: Preliminary association between vertices of OWS with facets of the partial GWS (associations are shown by colors). For some vertices (e.g., the blue ones) the actual inflation factor is smaller than 1 because they lie outside the partial GWS. The figure also shows the facet associated with the vertex with the smallest inflation factor.

In the main cycle, the algorithm iteratively expands the partial convex hull adding a new vertex at every iteration. As the objective is to determine the smallest inflation factor, the algorithm always expands the facet associated with the point with the smallest inflation factor. The expansion step follows the same heuristic of QuickHull and GWSPQH, i.e., the partial GWS is expanded by adding the farthest point in the outside set of the face being expanded (see Figure 5). During this step 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 essential to observe that these points can only be associated with one of the new facets, because the directions these points represent remain unchanged and the old facet is replaced by the new ones. Therefore, it is not necessary to search the whole set of facets in the partial convex hull.

This iterative expansion process continues until the facet associated with the vertex with the smallest positive inflation 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 the needed metric Q_{OWS}.

Algorithm 2 sketches the pseudocode for the strategy we just described. In line 1 the initial simplex CH is initialized as in the QuickHull algorithm. Next, (loop from line 3 to 15) CH is iteratively expanded until the origin is included. This condition is satisfied when the offset of the hyperplane supporting every facet is not negative, as per Lemma 1. 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 15). This heuristic accelerates the process of including the origin. If the origin cannot be included in CH the metric Q_{OWS} cannot be

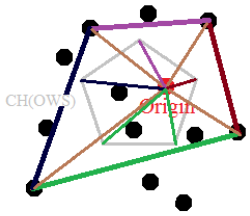


Figure 5: Expansion step. A facet in Figure 4 (blue) is removed to expand the convex hull towards the farthest vertex in its outside set. Two new facets are created and the vertices of OWS formerly associated with the green face are now associated with the two new facets.

computed, and the algorithm terminates (line 13). The initial assignment of vertices of CH(OWS) to the facets is performed in lines 16-19 through a search over the set of all 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 20-27) 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 21-23) 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 27).

6.1. Computational Complexity

As for PQHGWS (and QuickHull), the computational complexity of PQHOWS cannot be easily determined, and therefore its complexity analysis has to rely on approximations and conjectures. Note that the following discussion does not include the complexity for the upfront computation of the convex hull of the object wrench space CH(OWS). This preliminary computation can be done with QuickHull. The complexity of PQHOWS (Algorithm 2) is determined by the sequential execution of three phases, each associated with a loop. The first (line 3 to 15) expands the initial simplex until the origin is inside and it is based on the QuickHull algorithm. Therefore it inherits its conjectured $\mathcal{O}(n \log s)$ complexity, where n is the number of vertices and s the number of processed vertices. The second loop (line 16 to 18) has complexity $\mathcal{O}(V_{OWS}H_{CH})$ where V_{OWS} is the number of vertices of CH(OWS) and H_{CH} is the number of hyperplanes in the convex hull CH determined in the first loop. The third and last loop (line 20 to 27) continues to expand CH and at each iteration it reassigns some of the vertices in CH(OWS) associated with the facet in CH being expanded. Therefore its complexity is $\mathcal{O}(V_{OWS}n \log s)$ where we used again the conjectured complexity for QuickHull. Therefore the conjectured complexity for PQHOWS can be stated to be $\mathcal{O}(V_{OWS}H_{CH} + V_{OWS}n \log s)$.

7. Experimental Evaluation

In this section we experimentally show that the two algorithms presented in Section 5 and 6 provide significant computational improvements when compared with the state of the art.[6] Our code is obtained modifying the freely available QuickHull implementation and therefore benefits from a solid code base. All tests were run on a standard desktop running Linux with a 2.8GHz Intel i7 processor and 8 Gb RAM.

7.1. Grasp Wrench Space Metric

QuickHull is the algorithm most commonly used to compute the GWS based grasp quality metric, hence we compared PQHGWS with QuickHull over a set of 200 grasps with 4 contact points and an approximation of the friction cone with 32 edges. To put the following results into perspective, it is worth recalling that

[6]The code implementing the algorithms described in this paper is freely available in the authors' website, together with the datasets used to generate the results presented in this section.

Algorithm 2 PQHOWS algorithm

Create intial simplex CH with 7 points
OriginInside ← **false**
while not OriginInside **do**
  FCflag← 1
  for all $\mathcal{F} \in$ CH **do**
    if $o((\mathcal{F}, 0) < 0$ **then**
      FCflag← 0
  if FCflag=1 **then**
    OriginInside ← **true**
  else
    $\mathcal{F}_e \leftarrow \arg\min_{\mathcal{F}} o(\mathcal{H}(\mathcal{F}), 0)$
    if $\mathcal{O}(\mathcal{F}_e) = \varnothing$ **then**
      return "Grasp is not force closure"
    else
      Expand $\mathcal{F}_e$ and update CH
for all facets $\mathcal{F}_j \in$ CH **do**
  for all $\mathbf{v}_i \in \mathcal{V}$(CH(OWS)) **do**
    Compute $r_{i,j}$ as per Eq. 8.
Associate each $\mathbf{v}_i$ to the facet $\mathcal{F}_j$ with smallest $r_{i,j}$
loop
  $\mathcal{F}_e \leftarrow \arg\min_{\mathcal{F}} r_{min}(\mathcal{F})$
  if $\mathcal{O}(\mathcal{F}_e) \neq \varnothing$ **then**
    Expand $\mathcal{F}_e$
    Reassign vertices associated with $\mathcal{F}_e$ to the new facets ($\mathcal{F}_{new}$)
    Calculate r_{min} for the new facets ($\mathcal{F}_{new}$)
  else
    return $r_{min}(\mathcal{F}_e)$

PQHGWS does not approximate the value of the grasp quality metric, but rather returns exactly the same value obtained using QuickHull. The object being grasped is a bar. Since the grasps are randomly generated, some of them are force closure and some are not. Figure 6 shows the results. We contrast the two algorithms using three performance measures, namely the number of processed points, the number of generated hyperplanes, and the time spent to compute the metric. Note that the number of points and the number of hyperplanes are the measures used in literature to assess the scalability of QuickHull [12]. The three plots on the left contrast the absolute performance, with the blue line showing QuickHull's performance and the the green line showing PQHGWS' performance. Since the gap between the two is large, the three plots on the right display the ratio between them (the larger the better). It is interesting to note that the average speedup for the number of processed points is only about 6, whereas the average speedup in terms of number of processed hyperplanes exceeds 70 and the average speedup in terms of computational time is slightly below 40. With regard to this last number, it shall be outlined that our current implementation is not optimized, and therefore this gain can could be further improved.

In practical scenarios the friction cone is approximated using a regular pyramid. Hence to reduce the approximation error one would increase the number of edges in the pyramid. There is a linear relationship between the number of edges and the number of points for which the convex hull will be computed. Figure 7 shows the ratio between QuickHull's and PQHGWS' performance as a function of the number of edges approximating the friction cone. The figure shows that as the number of edges k increases the performance gap between QuickHull and PQHGWS grows as well. Therefore, PQHGWS allows to approximate the friction cone with a pyramid with a large number of edges without introducing a significant computational burden. This would not be possible with QuickHull. Figures 7 is particularly important in light of the findings

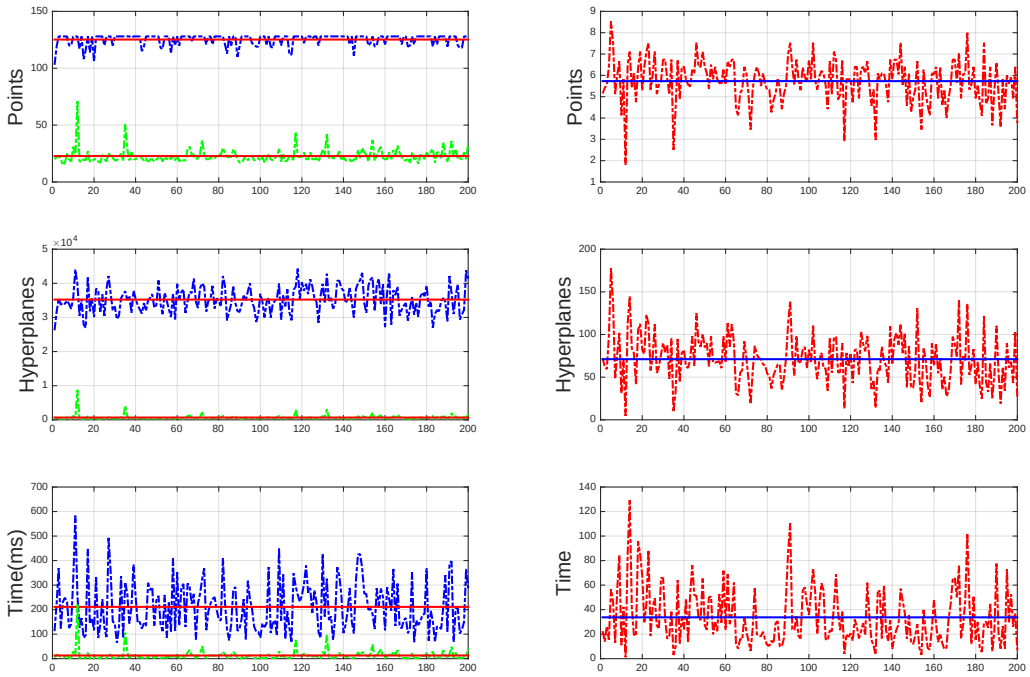


Figure 6: Left: Performance of QuickHull (blue) and PQH (green) for randomly generated grasps that may or may not be force closure. Right: Ratio between the two performances. In all figures the horizontal colored line indicates the average.

presented in [15]. Pokorny and Kragic have shown that when the grasp is force closure the approximation error introduced by discretizing the friction cone with k edges is $M(1-\cos(\pi/k))$ where M is a constant accounting for geometric aspects unrelated to k. Using PQHGWS one can afford to use large k values to approximate the cone. For example, for $k = 64$ PQHGWS is still extremely fast and $(1-\cos(\pi/k)) \approx 0.0012$.

7.2. Object Wrench Space Metric

We next evaluate the performance of the PQHOWS algorithm. In this case there is no widely used implementation available for comparison, so we contrast our method with two alternatives. The first considers the brute force approach obtained by applying Eq. (8) for every vertex $\mathbf{v}_i$ and every facet $\mathcal{F}_j$. This approach is slow but computes the correct result without introducing any approximation. The second method was proposed in [18] and is approximate. The idea is to enclose the OWS with the smallest ellipsoid, and to then transform the ellipsoid into a sphere with a linear transformation applied to both OWS and GWS. After this transformation the metric is obtained computing the radius of the largest sphere enclosed in the transformed GWS. This computation can be then performed using QuickHull or PQHGWS. Note that besides being approximated, this method also does not provide an explicit bound on the approximation error.

Figure 8 shows a time comparison of the various methods over 200 different grasps for the same OWS. Every method requires the preliminary computation of OWS, so this time is excluded from the chart because it is determined upfront once for all algorithms. We consider two comparisons. First, we compare the two methods computing the exact value for the OWS metric, i.e., the brute force method (red line) and PQHOWS (blue line). The chart clearly shows that PQHOWS largely outperforms the brute force method. Next, we compare PQHOWS with two different implementations for the approximate method proposed in [18]. The two implementations differ in how they compute the convex hull, i.e., the first uses QuickHull (green line), whereas the second uses PQHGWS (cyan line). The chart shows that the three methods are

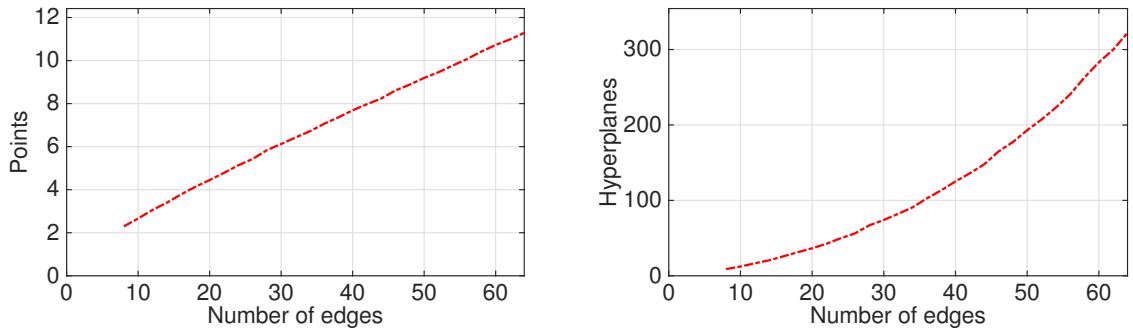


Figure 7: Ratio between the performance of QuickHull and PQH as a function of the number of edges used to approximate the friction cone.

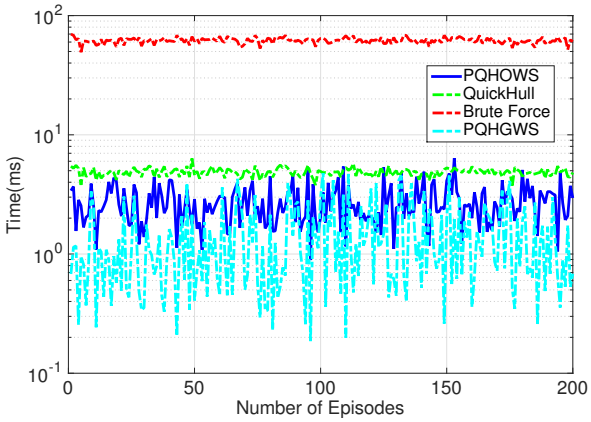


Figure 8: Time comparison for different methods computing the metric Q_{OWS}. Note the logarithmic scale on the y axis.

more or less comparable in terms of time. However, the main difference is that PQHOWS computes the exact result, while the other two methods provide only an approximation, and the next experiment shows that the approximation error can be at times significant.

Figure 9 shows the relative approximation error, that is for each of the 200 grasps evaluated we plot $\frac{|Q_{\text{OWS}}-\tilde{Q}_{\text{OWS}}|}{Q_{\text{OWS}}}$ where Q_{OWS} is the exact value and $\tilde{Q}_{\text{OWS}}$ is the approximate value computed with [18]. The figure shows that the approximation error can be at times very large and therefore the method in [18], albeit slightly faster, should be avoided. One should add that to the best of our knowledge the quality of the approximation proposed in [18] has not been analytically studied, and therefore an experimental evaluation is the best one can aim for at the moment. The analysis 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. Finally it is important to observe that a direct comparison of PQHOWS and PQHGWS is ill posed because the two algorithms compute two different metrics. For a given grasp PQHGWS is in general faster, but the metric it computes is different (and in a sense less desirable or accurate) than the one computed by PQHOWS.

7.3. Impact on Planning

While expedite grasp quality evaluation is of independent interest, efficient implementations are particularly valuable to improve the performance of grasp planners relying on grasp metrics to search the space of possible grasps. In this section we substantiate this claim considering two different planners and showing how they are affected by the choice of the algorithms used for grasp evaluation. First, for qualitative pur-

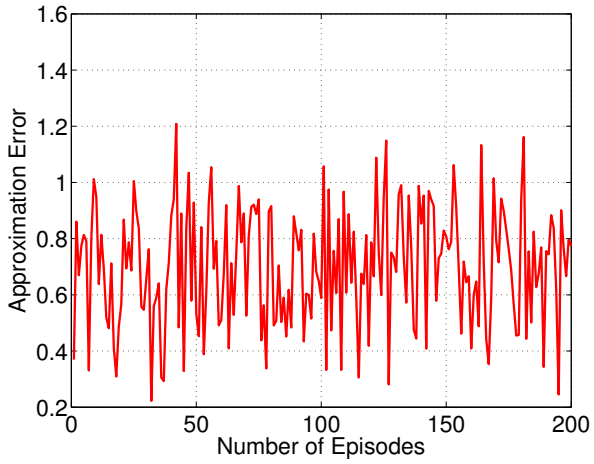


Figure 9: Approximation error introduced by the method proposed in [18].

poses consider Figure 10, where we display how the choice of the grasp metric is significant. For the object being grasped, 1000 force closure grasps were determined upfront. The grasp wrench space metric and the object wrench space metrics were then used to select the best one among them. The top three figures show the best grasp determined using the first metric and the bottom three figures the best grasp obtained using the second metric.

The difference is evident, and similar discrepancies were observed in all the experiments we performed, outlining that the choice of the metric may have a dramatic impact on the grasp selected by a planner, and it is then important to be able to integrate more than one quality evaluation algorithm into the planner. To put this statement into perspective, it is important to recall that to date the object wrench space metric is rarely used in practice because of its computational complexity. As we will show next, PQHOWS provides and algorithm whose performance is comparable to the current implementations for the grasp wrench space metric, and thus enables the use of this more powerful tool for grasp planning.

We next consider two planners to determine force closure grasps on some objects, three of which are displayed in figure 11. The other two objects we consider (not displayed in the figure) are a sphere and a cube.

The first planner we consider works like the GraspIt! planner [34], i.e., it places the hand at a random position in proximity of the object and then closes the fingers to determine if the resulting contact points produce a force closure grasp or not. With this algorithm we determined 1000 grasps and we evaluated each of them using both the grasp wrench space metric and the object wrench space metric. For both algorithms we used the current implementation (QuickHull, BruteForce) and our algorithms PQHGWS and PQHOWS to compare the time. Figure 12 shows the results (displayed time is averaged over 1000 grasps, and the top figures display standard deviations too.) Figures 12.a and 12.b confirm that PQHGWS and PQHOWS largely outperform their counterparts. Figure 12.c compares instead the performance of PQHOWS with the algorithm computing the grasp wrench space metric with QuickHull. The figure shows that PQHOWS is comparable[7] to QuickHull in terms of the time spent to evaluate a grasp, and it can therefore be readily integrated into existing planners without altering their performance while providing a significantly better grasp quality measure. For completeness Figure 12.d also compares the the performance of PQHGWS with PQHOWS. To put this chart into perspective, it is important to recall that the two algorithms compute two different metrics, with the object wrench space metric being more complex. Therefore it is not surprising that PQHGWS outperforms PQHOWS.

Next, we incorporated PQHGWS into a grasp planner we recently developed [35]. In essence the algo-

[7]The reader should consider that our PQHOWS implementation is not optimized yet, so small differences in performance can be eliminated.

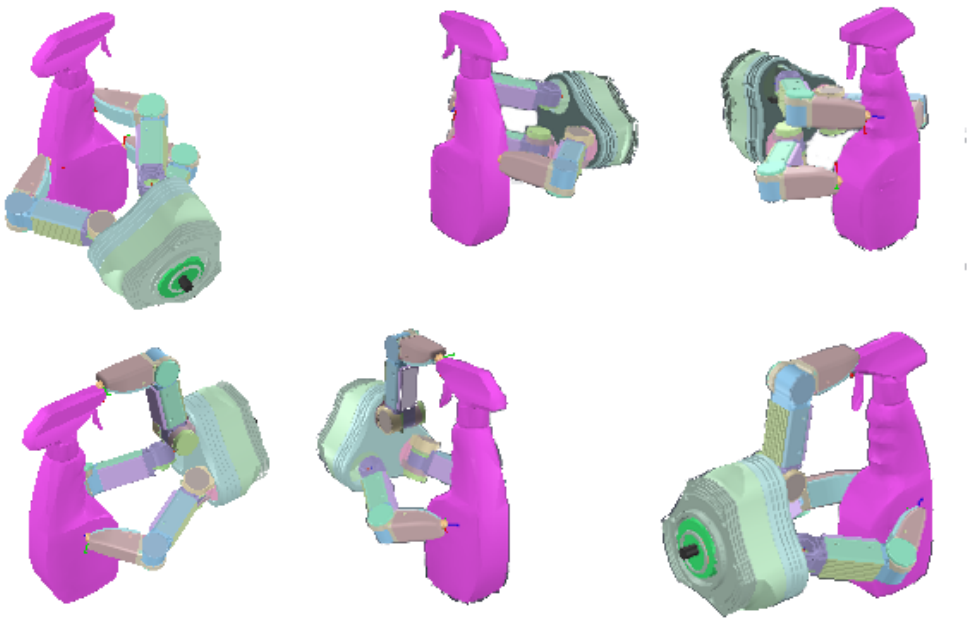


Figure 10: Top figures: best grasp determined using the grasp wrench space quality measure. Bottom figure: best grasp determined using the object wrench space quality measure. Each grasp is shown from three different view points to outline the differences.

rithm performs a search for the best grasp exploring the surface of the object and guiding its search using a gradient-descent like approach informed by the quality metric (the details of the planner are provided in [35]). Our original implementation relied on QuickHull to compute Q_{GWS}, and to perform this comparison we replaced it with PQHGWS. Table 1 illustrates the results obtained when the planner seeks a grasp with four contact points and the friction cone is approximated with $k = 24$ edges. Each test case refers to a different object, and averages over 100 different planned grasps are displayed.

The second and third column show the overall planning time spent when QuickHull or PQHGWS are used, respectively. The fourth column shows the ratio between the two. Similarly, the fifth and sixth columns show only the time spent for grasp quality evaluation and the seventh column displays their ratio. As expected, this second ratio is higher since it considers only the time spent for grasp quality evaluation, whereas in the previous case the overall planning time is shown. The table shows that by just substituting QuickHull with PQHGWS, speedups larger than 20 can be immediately obtained.

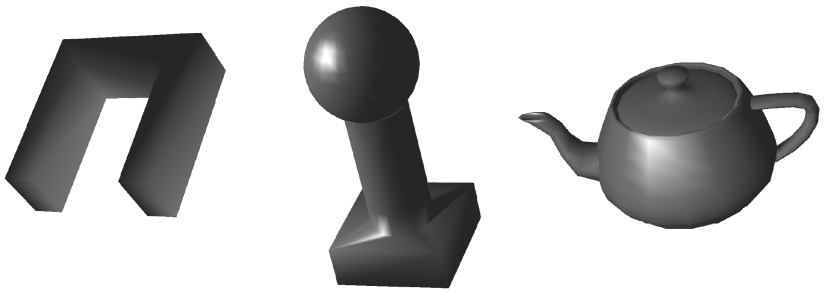


Figure 11: Three of the objects used to assess the impact of PQHGWS and PQHOWS on grasp planning. Left to right: C-shape, joystick and teapot.

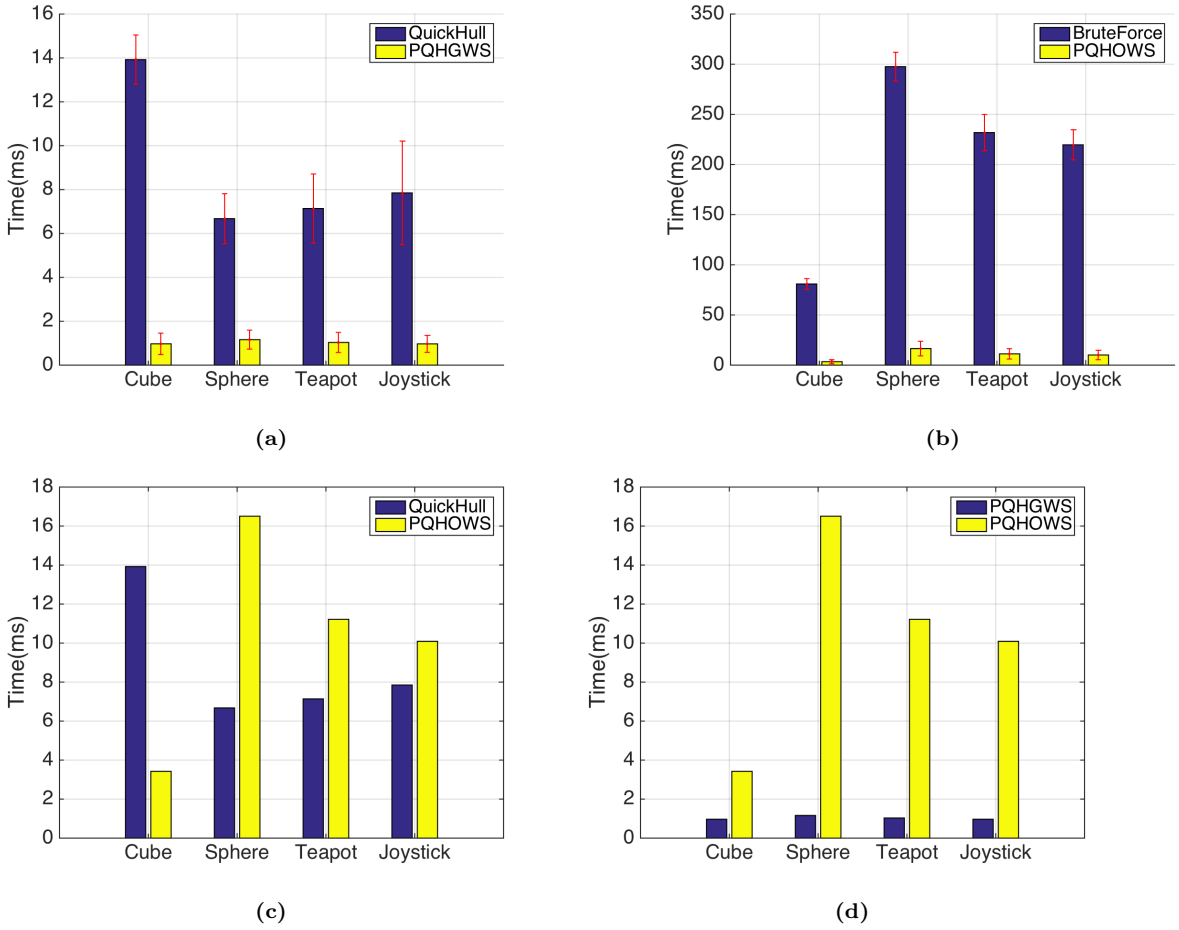


Figure 12: Performance comparison between different grasp quality evaluation algorithms.

8. Conclusions

In this paper we have presented two algorithms to efficiently compute two grasp quality evaluation metrics proposed in literature. Both algorithms build upon the common intuition that the metrics can be obtained computing a partial quick hull in the six dimensional space of wrenches. The first algorithm, PQHGWS, computes the widely used metric proposed by Ferrari and Canny more than twenty years ago. It is based on a modified criterion for expansion for the QuickHull algorithm and we have experimentally shown that is substantially faster than the original. The second algorithm considers a metric based on the object wrench space that was proposed in the past but rarely used in practice. This metric is defined only for force closure grasps, and we demonstrated that our method yields a performance comparable to the one obtained for the Ferrari-Canny metric. Therefore, thanks to our algorithm this metric can now be used in practice, whereas in the past it was deemed too demanding for being practically useful. Implementations for both algorithms are freely available.

In this paper we have focused on the grasp wrench space and the object wrench space, but a third paradigm was proposed in literature, namely the task wrench space. We have not considered the TWS metric because to date there is no efficient and practical way to determine the TWS in a general setting. However, it should be noticed that the algorithm presented in Section 6 could be equally applicable if the TWS is provided. In fact, one can just substitute OWS with TWS in algorithm 2 without making further changes.

	Planning Time			Grasp Quality Evaluation		
Testcase	QuickHull	PQHGWS	Speedup	QuickHull	PQHGWS	Speedup
Cube	219.713	10.326	21.28	218.034	8.647	25.26
Sphere	163.625	7.678	21.31	162.182	6.235	26.01
C shape	270.357	15.921	16.98	268.439	14.003	19.17
Joystick	309.247	13.895	22.26	306.965	11.613	26.43
Teapot	309.629	10.376	29.86	307.204	7.951	38.64

Table 1: For five different test cases, the table displays the planning time spent when QuickHull or PQHGWS are used. The last three columns display just the time for grasp quality evaluation. Time is expressed in seconds.

Appendix

In this appending we provide a table with the list of symbols used in the paper.

Symbol	Meaning
$\mathrm{CH}(A)$	Convex Hull of set A
$\mathcal{F}_i$	ith facet of a convex hull
$\mathcal{H}_i$	Hyperplane supporting facet $\mathcal{F}_i$
$\mathcal{I}(\mathcal{F})$	Inside set of facet $\mathcal{F}$
$\mathcal{O}(\mathcal{F})$	Outside set of facet $\mathcal{F}$
$\mathbf{n}_i$	Normal to hyperplane $\mathcal{H}_i$
$\mathcal{B}$	Rigid body being grasped
$\mathbf{p}_i$	ith contact point
$\mathbf{f}_i$	ith force applied
$\mathbf{w}_i$	Wrench generated by the ith force
$\mathbf{e}$	Unary force vector
$F(\mathbf{p}_i)$	Friction cone at the ith contact point
$\mathbf{G}$	Grasp matrix

Acknowledgments

This paper extends preliminary findings discussed in [7, 36]. This work is partially 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.

References

[1] J.-S. Cheong, H. Kruger, A. van der Stappen, Output-sensitive computation of force-closure grasps of a semi-algebraic object, IEEE Transactions on Automation Science and Engineering 8 (3) (2011) 495–505.

[2] T. Watanabe, T. Yoshikawa, Grasping optimization using a required external force set, IEEE Transactions on Automation Science and Engineering 4 (1) (2007) 52–66.

[3] B. Kehoe, D. Warrier, S. Patil, K. Goldberg, Cloud-based grasp analysis and planning for toleranced parts using parallelized monte carlo sampling, IEEE Transactions on Automation Science and Engineering 12 (2) (2015) 455–470.

[4] L. Odhner, R. Ma, A. Dollar, Open-loop precision grasping with underactuated hands inspired by a human manipulation strategy, IEEE Transactions on Automation Science and Engineering 10 (3) (2013) 625–633.

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

[6] M. Strandberg, B. Wahlberg, A method for grasp evaluation based on disturbance force rejection, IEEE Transactions on Robotics 22 (3) (2006) 461–469.

[7] S. Liu, 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.
[8] A. Sahbani, E. El-Khoury, P. Bidaud, An overview of 3D object grasp synthesis algorithms, Robotics and Autonomous Systems 60 (3) (2012) 326–336.
[9] M. Roa, R. Suárez, Computation of independent contact regions for grasping 3-d objects, IEEE Transactions on Robotics 25 (4) (2009) 839–850.
[10] M. A. Roa, R. Suárez, Grasp quality measures: review and performance, Autonomous Robots 38 (1) (2015) 65–88.
[11] D. Kirkpatrick, B. Mishra, C. Yap, Quantitative Steintz's theorems with applications to multifingered grasping, in: Proceedings of the ACM Symposium on Theory of Computing, 1990, pp. 341–351.
[12] C. Barber, D. P. Dobkin, H. Huhdanpaa, The Quickhull algorithm for convex hulls, ACM Transactions on Mathematical Software 22 (4) (1996) 469–483.
[13] T. Zheng, An efficient algorithm for a grasp quality measure, IEEE Transactions on Robotics 29 (2) (2013) 579–585.
[14] Y. Zheng, W.-H. Qian, Improving grasp quality evaluation, Robotics and Autonomous Systems 57 (2009) 665–673.
[15] F. Pokorny, D. Kragic, Classical grasp quality evaluation: New theory and algorithms, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 3493– 3500.
[16] G. Liu, J. Xu, X. Wang, Z. Li, On quality functions for grasp synthesis, fixture planning, and coordinated manipulation, IEEE Transactions on Automation Science and Engineering 1 (2) (2004) 146–162.
[17] N. Pollard, Parallel methods for synthesizing whole-hand grasps from generalized prototypes, Ph.D. thesis, MIT (1994).
[18] C. Borst, M. Fischer, 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.
[19] H. Jeong, J. Cheong, Evaluation of 3D grasps with physical interpretations using object wrench space, Robotica 30 (3) (2012) 405–417.
[20] Z. Li, S. Sastry, Task oriented optimal grasping by multifingered robot hands, IEEE Journal of Robotics and Automation 4 (1) (1988) 32–44.
[21] Y. Lin, Y. Sun, Grasp planning to maximize task coverage, International Journal of Robotics Research 34 (9) (2015) 1195–1210.
[22] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational Geometry, 3rd Edition, Springer, 2008.
[23] R. Seidel, Convex hull computations, in: J. E. Goodman, J. O. Rourke (Eds.), Handbook of discrete and computational geometry, Chapman & Hall/CRC, 2004, Ch. 22, pp. 495–511.
[24] A. C. Yao, A lower bound to finding convex hulls, Journal of the ACM 28 (780-787).
[25] R. L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Information processing letters 1 (132-133).
[26] D. G. Kirkpatrick, R. Seidel, The ultimate planar convex hull algorithm?, SIAM Journal of Computation 15 (1986) 287–299.
[27] T. M. Chan, Output-sensitive results on convex hulls, extreme points and related problems, Discrete Computational Geometry 16 (1996) 369–387.
[28] F. P. Preparata, S. J. Hong, Convex hulls of finite sets of points in two and three dimensions, Communications of the ACM 20 (1977) 87–93.
[29] B. Chazelle, An optimal convex hull algorithm in any fixed dimension, Discrete Computational Geometry 10 (1993) 377–409.
[30] R. Seidel, Constructing high-dimensional convex hulls at logarithmic cost per face, in: Proceedings of the ACM Symposium on Theory of Computing, 1986, pp. 404–413.
[31] K. Clarkson, P. Shor, Applications of random sampling in computational geometry, Discrete Computational Geometry 4 (1989) 387–421.
[32] D. Pratticchizzo, J. Trinkle, Grasping, in: B. Siciliano, O. Khatib (Eds.), Handbook of robotics, Springer, 2008, Ch. 28, pp. 671–700.
[33] R. Murray, Z. Li, S. Sastry, A mathematical introduction to robotic manipulation, CRC Press, 1994.
[34] A. Miller, P. Allen, Graspit! a versatile simulator for robotic grasping, IEEE Robotics Automation Magazine 11 (4) (2004) 110–122. doi:10.1109/MRA.2004.1371616.
[35] S. Liu, S. Carpin, Global grasp planning using triangular meshes, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2015, pp. 4904–4910.
[36] S. Liu, S. Carpin, A fast algorithm for grasp quality evaluation using the object wrench space, in: Proceedings of the IEEE Conference on Automation Science and Engineering, 2015, pp. 558–563.