OCOG: A Common Grasp Computation Algorithm for a Set of Planar Objects

Avishai Sintov[1], Roland J. Menassa[2] and Amir Shapiro[1]

Abstract— This paper addresses the problem of defining a simple End-Effector design for a robotic arm that is able to grasp a given set of planar objects. The OCOG (Objects COmmon Grasp search) algorithm proposed in this paper searches for a common grasp over the set of objects mapping all possible grasps for each object that satisfy force closure and quality criteria by taking into account the external wrenches (forces and torque) applied to the object. The mapped grasps are represented by feature vectors in a high-dimensional space. This feature vector describes the design of the gripper. A database is generated for all possible grasps for each object in the feature vector space. A search algorithm is then used for intersecting all possible grasps over all parts and finding a common grasp suitable for all objects. The search algorithm utilizes the *kd*-tree index structure for representing the database of the sets of feature vectors. The *kd*-tree structure enables an efficient and low cost nearest-neighbor search for common vectors between the sets. Each common vector found (feature vector) is the grasp configuration for a group of objects, which implies the future end-effector design. The final step classifies the grasps found to subsets of the objects, according to the common vectors found. Simulations and experiments are presented for four objects to validate the feasibility of the proposed algorithm. The algorithm will be useful for standardization of end-effector design and reducing its engineering time.

Keywords—Algorithm; Common; Grasp; Search; End-Effector.

1. Introduction

In today's automotive assembly manufacturing plants robots are the foundational element of high volume automation and are used extensively in stamping, body assembly, and paint operations. Today's manufacturing plants have to juggle two critical and often conflicting requirements: the greater demand and the higher quality. This requires manufacturing high quality products while meeting a very high production rate. Robots are used in body shops and other areas of an assembly plant for performing a myriad of tasks such as material handling of components for assembly (Fig. 1) and performing various tasks. The robots interact with the objects by using an arm and an end-effector that is a key interface between the robot and the component or product that needs to be handled. The robot arms are general purpose and built for multi-purpose tasks. While many end-effectors look similar to one another; they are designed, built, and optimized for a specific task (e.g., assembling one component to another) and part geometry (e.g., front door).

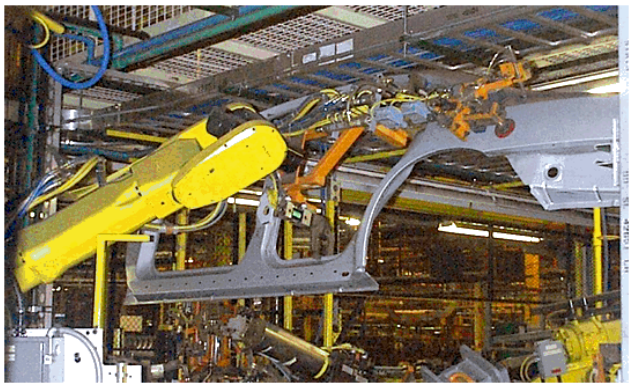


Fig. 1 - End-effector for part handling components in assembly lines.

[1]A. Sintov and A. Shapiro are with the Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (Corresponding author: e-mail sintova@post.bgu.ac.il, phone +972-54-556-2555).
[2]R. Menassa is with Global General Motors R&D, Manufacturing Systems Research Lab, Warren, MI 48090, USA.

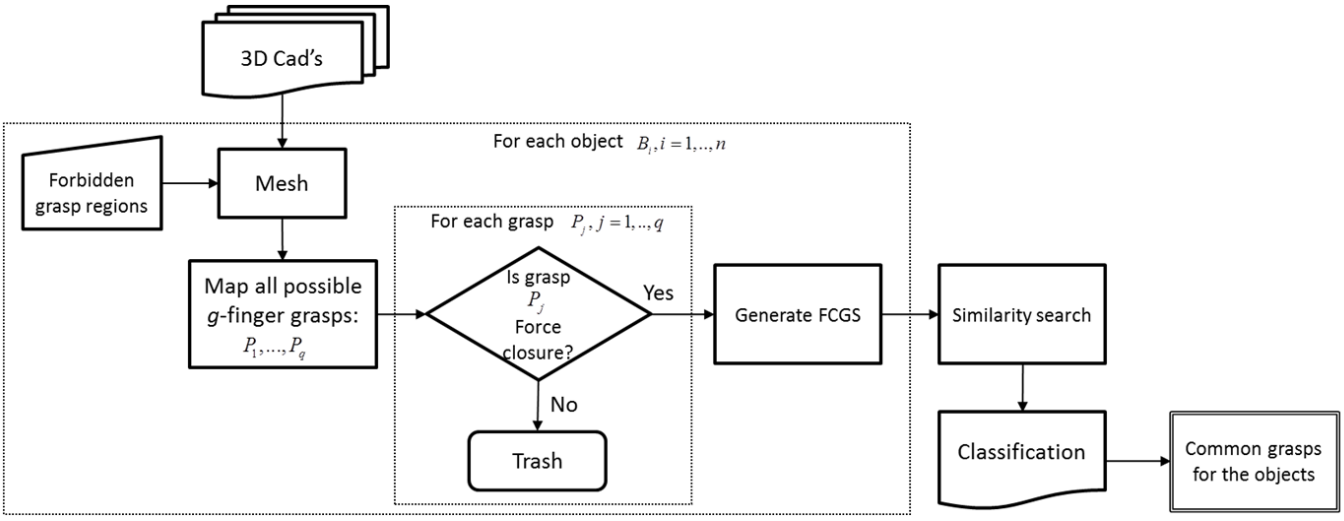


Fig. 2 - OCOG algorithm approach.

Grasping (locating and clamping) a component is the primary function of an end-effector regardless of its application. However end-effectors usually are designed and built for grasping a specific component or for a specific task, which makes them very inflexible in handling variations in component shape or in task. Current automatic solutions to flexibility involve adding either active elements to the end-effectors in order to adjust to different part geometries (typically limited to two variations per end-effector) or by simply adding end-effectors adjacent to the robot that can be replaced via a robot-interface plate. This requires more factory space and thus is more costly. The design, build, and testing phase of a typical end-effector consume a considerable amount of engineering time and add extra cost to the final product.

We propose a novel solution for designing simple and flexible end-effectors that can handle a class of objects rather than a single instance of it. A key task in this is to identify the common grasping locations of different sets of objects, which can lead to multiple sets of end-effectors, each able to handle one class of objects. Specifically, this work is to develop an algorithm for computing the configuration of a universal end-effector for grasping a set of objects. Given a set of all CAD models of the parts, the idea is to design an end-effector that is universal in the sense of being able to hold a wide set of components for multiple manipulation tasks. The algorithm will be able to characterize the object's geometries by discretizing its surface, defining forbidden grasping zones (according to operational demands), and finding a configuration for grasping it. As we discuss the design of an industrial end-effector, the final configuration has to be simple and low cost; thus, it has to be with minimal degrees of freedom and optimal. An optimal configuration is the one that can stably grasp the object even under the application of external forces or torques due to the task being done, i.e., we require force-closure grasp. The OCOG (Object COmmon Grasp) algorithm proposed in this paper is presented in Fig. 2. The concept behind this algorithm is the parameterization of the force-closure grasps of each object and using it for classification of the objects with respect to these grasps. In the first stage of the algorithm a Force Closure Grasp Set (FCGS) is constructed for every object by sampling all force-closure grasps with a lower bound on a grasp quality measure, and representing the possible grasps as feature vectors in high-dimensional space. Each feature vector injectively defines the grasp and implies the end-effectors' configuration.

The next step of the algorithm is the similarity join, in which the FCGS of each object is mapped to a *kd*-tree (*k*-dimensional tree) index structure database. Such a tree is a multidimensional data structure version of the well-known binary search tree and enables organizing vectors in the *k*-dimensional space. In this tree every non-leaf node is a partition in one dimension, splitting the space into two subspaces. Hence, repeatedly every level represents a single dimension partitioning of the FCGS to two subspaces. This database structure enables the finding of feature vector pairs in the FCGS of all objects. Our similarity search algorithm is based on the nearest-neighbor search algorithm for finding for every two FCGS sets, all pairs of vectors that are within a predefined distance. The mean vector of every pair found is then inserted into a registry set and associated to the FCGS from where it originated. The *kd*-tree structure has many advantages for this process by reducing the number of inspections and by keeping the feature vectors sorted at every instance. Finally, classification is done in order to find the minimal set of feature vectors that covers the whole set of objects. The classification is in fact an intersection between all vectors in the registry set, to find a single vector that exists in all of the

FCGS sets. If it fails to find a single vector, the algorithm tries to find the minimal number of vectors, where each vector is the grasp of a subset of the objects. Thus classifying the objects set into subsets of objects, with a compatible grasp for each subset.

In this work, we consider the geometry of the contact locations, which is the output of the algorithm, in order to design the end-effector. Based on the contact locations the end-effector will have only a single degree of freedom. That is, opening and closing the gripper. Hence, we address a low dexterity end-effector for industrial use and the design of the end-effector is done specifically to reach the contact points. The reason for minimal degrees of freedom is to improve accuracy, which is of high priority. However, the detailed design of the end-effector and the accuracy analysis are out of scope of this paper.

The paper is organized as follows. The next section is a summary of related work. Section 3 gives an overview of grasping fundamentals used in this work. The structure of the grasp feature vector and the FCGS generation algorithm is described in section 4. Section 5 presents the main algorithm for similarity join and classification of the common feature vectors. Section 6 describes the simulations implementing the proposed algorithms. Section 7 presents the experiments done to validate the concept and simulations. Finally, Section 8 contains a summary and proposes future work.

2. Related Work

This work uses familiar grasp properties known in the literature for synthesis and evaluation of a grasp. A force closure grasp is its capability to resist external forces and torques applied on the object with forces exerted at the contact points by the fingers. The notion of force-closure for determining feasible grasps is widely studied in literature. It is typically defined using a generalized vector called a *wrench*, combining force and torque, and the wrench space derived from it. In Murray et al. [1], Niparnan and Sudsang [2], and Shapiro et al. [3], the force-closure criterion is well defined. Murray et al.'s force-closure grasp criteria used the notion of convexity conditions for a defined grasp map. Niparnan and Sudsang used the notion of the wrench space as a 3- or 6-dimensional (for planar or spatial cases, respectively) convex-hull, previously introduced by Ferrari and Canny [4], to define geometrical conditions for force closure criteria. Several algorithms for synthesis of optimal grasps were presented. Roa and Suarez [5] proposed a grasp optimization algorithm based on the convex hull criterion, meaning, increasing the largest perturbation wrench a grasp can resist independent of its direction. Wang [6] introduced a greedy algorithm for fixture synthesis of frictionless grasps on a discrete point set, minimizing the workpiece positioning errors. Ponce and Faverjon [7] and Liu [8] introduced computation methods for synthesis of frictional grasps. Ponce and Faverjon proposed a computation method for the three finger grasps of 2D polygonal objects with frictional point contact, by using linear sufficient conditions for force closure and equilibrium. Liu addresses the same problem; however, a new sufficient condition for force-closure was introduced, based on dimensionality reduction of the convex-hull.

Several grasp optimization methods using different grasp quality measures have been presented in the literature; Ferrari and Canny [4] and Li and Sastry [9] introduced a quality measure (used in this work) based on the external wrench to be resisted, where they first introduced a general measure based on the largest wrench that the grasp can resist; Li and Sastry used a task-oriented quality measure defined by the specific wrenches applied during execution. Schulman et al. [10] and Roa and Suarez [5] used the general measure approach for finding the set of contact points optimizing the grasp quality. Lin et al. [11] presented a frame-invariant quality measure for compliant grasps and fixtures, and presented its application to the planning of grasps and fixtures. Li and Sastry also introduced a quality criterion that measures how far the grasp configuration is from reaching singularity. The quality measures mentioned are based on the position of the fingers. Other quality measures are based on geometric criteria where the distribution of the fingers on the objects is maximized. Chinellato et al. [12] introduced quality measures criteria for 3-finger planar objects, one based on the area of the triangle to assess the distribution of the contact points. Kim et al. [13] presented the stability grasp index, which defines a polygon created of the contact points and measures the deviation of the polygon’s angles from a regular polygon; this implies the distribution of the polygon on the object.

Several heuristic approaches for force closure and quality measure analysis were introduced. Niparnan et al. [14] proposed force closure criteria of an *n*-finger grasp, where a heuristic condition was used for initial filtering of grasps, thus reducing running time. Prado and Suarez [15] introduced a heuristic approach for generating and measuring a 3-finger grasp, based on geometrical conditions considering the relative orientation and position of the three contact faces.

To the best of our knowledge, no previous work has been done for searching common grasps in the design of end-effectors for a set of objects. However, the work of Rodriguez and Mason [16] has similarities to ours. In their work, a finger design that can hold an object invariant of its scale or pose is searched. However, in our work we focus on grasping objects of various geometries but of the same scale. Much work has been done in the area of 3D shape similarity comparison, such as

the work by Novotni and Klein [17] and Osada et al. [18]. These algorithms are used for Internet and local database search, face recognition, image processing, and parts identification. Ohbuchi et al.'s [19] work on shape similarity search uses a generalized feature vector of a 3D polygonal mesh constructed of the moment of inertia, average distance of the surface from the model's axis, and its variance. However, such methods deal with mean parameterization of the geometry (such as volume, shape distribution, moment of inertia) of the objects and cannot be applied for grasping. The work of Li and Pollard [20] is based on shape matching for finding the best grasp of a set of objects. The best grasp is found by matching hand poses from a database of objects. This is done by using a predefined parameterization of the object surface and the hand poses. This shape matching method inspired this work.

The common approach for supporting similarity search is the use of multi-dimensional index structures. Space partitioning methods, which are relevant to this work, like *kd*-tree [21], grid-file, quad-tree [22], or ε-tree [23], divide the data space along predefined hyper-planes regardless of the point clusters. These methods allow low-cost data search in a high-dimensional space (such as the one done by Moore [24]), a feature that is essential for this work.

Some work was done in the field of end-effector design. Hong and Payandeh [25] introduced a design of an end-effector for grasping a variety of objects. The end-effector is a parallel jaw gripper with spring-loaded pegs that adapt to the object's shape and cage it within the pegs. Amend et al. [26] introduced a universal gripper based on a bag with granular material that in positive and negative pressure can grasp and release a variety of objects. Such a grasping method is very effective in manipulating objects rapidly but has difficulties with large objects, executing complicated tasks, and energy consumption. Dollar and Howe [27] have done much work in the area of end-effector design in an unknown environment, i.e., where the object geometry and orientation is not fully known. They have applied a joint coupling design to under-actuated end-effectors through compliance in the manipulator, which allows the end-effector to passively conform to a wide range of objects while minimizing contact forces.

3. Background

3.1. Grasping Model

Forces and torques can be represented as a *wrench* vector in a vector space denoted as the *wrench* space. A wrench is a *k*-dimensional vector in the *k*-dimensional wrench space, where *k* equals 3 for the planar case and 6 for the spatial (3D) object case, and is denoted as $\boldsymbol{w} = \begin{pmatrix} \boldsymbol{f} & \boldsymbol{\tau} \end{pmatrix}^T \in \mathbb{R}^k$, where $\boldsymbol{f}$ is a force vector and τ is a torque vector. We set the object's reference frame at its center of mass. Therefore, a wrench applied at the contact point, $\boldsymbol{p_i}$, can be described as $\boldsymbol{w_i} = \begin{pmatrix} \boldsymbol{f_i} & \boldsymbol{p_i} \times \boldsymbol{f_i} \end{pmatrix}^T$, where $\boldsymbol{p_i}$ is the position vector of the contact point represented in the object's reference frame (Fig. 3).

Friction exists between the fingertips of the end-effector and the object's surface and can be represented by the simple Coulomb friction model $\left| f_i^T \right| \le \mu f_i^N$ where f_i^N and f_i^T are the normal and tangential forces at the contact point, respectively, μ is the coefficient of friction. In this model, forces exerted at the contact point must lie within a cone centered about the surface normal. This is known as the *Friction Cone* (FC), and in the planar case can be defined by $\boldsymbol{f_i}^+$ and $\boldsymbol{f_i}^-$. The angle between them equals $2\tan^{-1}\mu$. If the force lies within the FC, the force $\boldsymbol{f_i}$ can be represented as a linear combination of $\boldsymbol{f_i}^+$ and $\boldsymbol{f_i}^-$ given by

$$\boldsymbol{f_i} = \alpha_i^+ \boldsymbol{f_i}^+ + \alpha_i^- \boldsymbol{f_i}^- \tag{1}$$

where α_i^+, α_i^- are nonnegative constants [8]. The associated wrenches can be expressed by the primitive forces as

$$\boldsymbol{w_i}^+ = \begin{pmatrix} \boldsymbol{f_i}^+ \\ \boldsymbol{p_i} \times \boldsymbol{f_i}^+ \end{pmatrix}, \boldsymbol{w_i}^- = \begin{pmatrix} \boldsymbol{f_i}^- \\ \boldsymbol{p_i} \times \boldsymbol{f_i}^- \end{pmatrix} \tag{2}$$

A *g*-finger grasp can be represented by the position vectors of all contact points $P = (\boldsymbol{p_1}, \ldots, \boldsymbol{p_g})$. We can represent the grasp using the matching wrenches applied at the contact points represented in the object's reference frame as $W = (\boldsymbol{w_1}, \ldots, \boldsymbol{w_g})$. If we consider the friction cones, the wrench set can be expressed using the primitive wrenches as $W = (\boldsymbol{w_1}^+, \boldsymbol{w_1}^-, \ldots, \boldsymbol{w_g}^+, \boldsymbol{w_g}^-)$.

Moreover, a linear mapping called the grasp map *G*, maps all forces applied at the contact points to a net wrench applied at the origin of the object's reference frame. The map is given by

$$\boldsymbol{w_o} = \mathrm{G}\boldsymbol{f_c} \tag{3}$$

where $\boldsymbol{f_c}$ is a generalized column vector of all forces applied to the object while maintaining the friction constraint $\boldsymbol{f_c} \in FC$. It can be shown that the columns $\{G_i\}$ of the grasp map *G* are the wrenches applied to the object with respect to the object's reference frame. For more details of the grasp map, refer to [1].

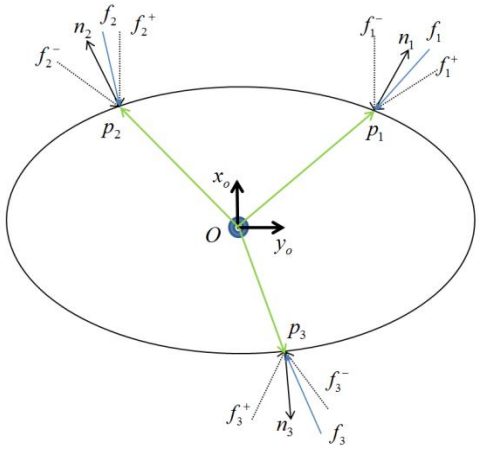


Fig. 3 - An ellipse planar object.

3.2. Force Closure Grasp

A grasp is said to be Force Closure if it is possible to apply wrenches at the contacts such that any external force and torques acting on the object can be counter-balanced. A grasp is force-closure when any external load can be balanced by a non-negative (assuming fingers can only push and not pull) combination of the wrenches [7]. Therefore, the notion of the Convex-Hull is introduced.

Definition 1. *The Convex Hull (CH) of system S of vectors $s_1,\ldots,s_n$ is the set of all non-negative convex combinations of the subsets of points from S. In other words, the CH is the minimal convex set containing S and is defined as*

$$CH(S) = \left\{ \sum_{i=1}^{n} a_i \boldsymbol{s_i} : \boldsymbol{s_i} \in S, \sum_{i=1}^{n} a_i = 1, a_i \geq 0 \right\} \tag{4}$$

where a_i is the linear combination coefficient bounded to be positive in order to ensure positive grip (non-sticky fingers).

With system *W* of contact wrenches $w_1,\ldots,w_g$, *CH(W)* is called the *grasp wrench set* (GWS) and represents the set of all the net wrenches that can be applied to the object through the contact points. The GWS is a mean for analyzing the grasp of an object regardless of the ability of the end-effector; we build the wrenches constructing it based on unit forces. Wrenches constructed on unit forces are used for the quality measure computation, which will be presented in the next section. In this way, the quality of the grasp can be compared to others regardless of the end-effector abilities to develop the magnitude of the forces, but only their directions. Moreover, we need to scale the torques in a way relating unit force to unit torque and keeping it independent of the object's size. Therefore, the wrenches constructing the convex hull are defined as

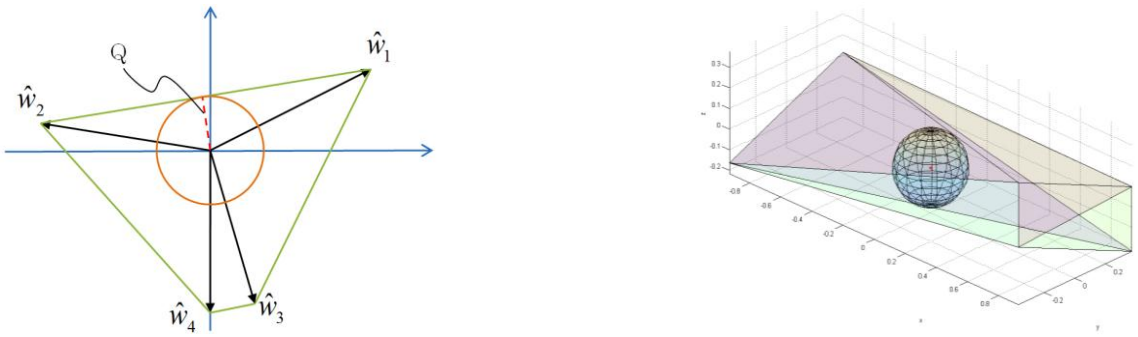
$$\boldsymbol{w_i^+} = \begin{pmatrix} \boldsymbol{n_i^+} \\ \dfrac{\boldsymbol{p_i}}{p_{\max}} \times \boldsymbol{n_i^+} \end{pmatrix}, \boldsymbol{w_i^-} = \begin{pmatrix} \boldsymbol{n_i^-} \\ \dfrac{\boldsymbol{p_i}}{p_{\max}} \times \boldsymbol{n_i^-} \end{pmatrix} \tag{5}$$

where $p_{\max} = \max_k \left(\| \boldsymbol{p}_k \| \right)$ and represents the maximum radius of the bounding circles of all objects centered in the object's reference frame (as presented in [28]). This normalization relates torque units to force units and ensures that the quality of

the grasp will be independent of object size. This allows comparison of quality metrics between grasps. $\boldsymbol{n}_i^+$ and $\boldsymbol{n}_i^-$ are the unit vectors in direction of the primitive forces $\boldsymbol{f}_i^+$ and $\boldsymbol{f}_i^-$ constructing the friction cone.

Theorem 1 (Force closure grasp [1],[7],[29])**.** *A necessary and sufficient condition for a system of g wrenches $w_1,\ldots,w_g$ to be force-closure is that the origin O of $\mathbb{R}^k$ lies in the interior of the convex hull of the contact wrenches. Meaning,*

$$O \in interior(CH(W)) \tag{6}$$



(a) (b)

Fig. 4 - The Grasp Wrench Set (GWS) and thelargest ball (TWS) in (a) 2D and (b) 3D cases.

Equivalently, it can be said that a grasp is force-closure if the convex hull of the columns {G_i} of the grasp matrix G, positively spans $\mathbb{R}^k$.

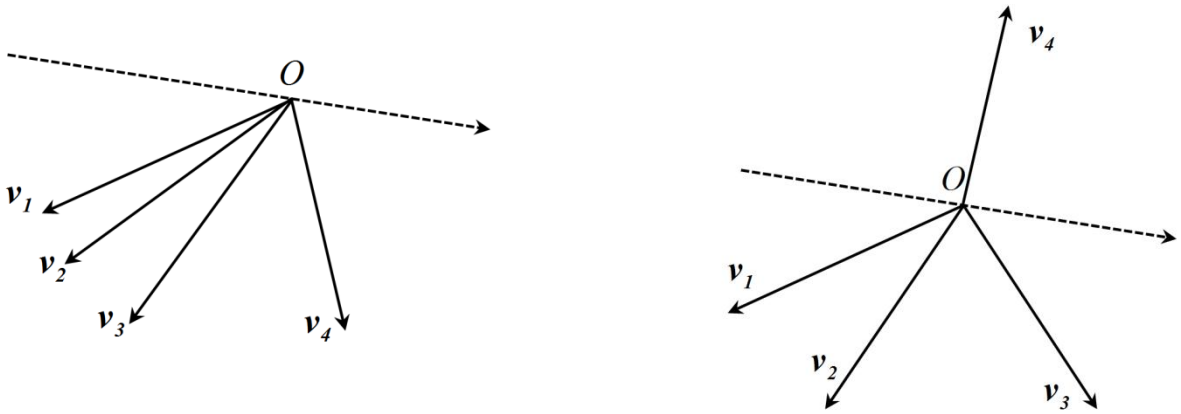
Lemma 1 ***[30].*** *Let G be a grasp with an associate set of wrenches W and H_k be a hyper-plane of CH(W). The origin O of the wrench space satisfies $O \in \text{interior}(CH(W))$ if and only if $\forall k$ any point $\boldsymbol{r} \in \text{interior}(CH(W))$ and O lie in the same half-space defined by H_k.*

By selecting ***r*** as a linear combination of the grasp wrenches we can guarantee that it will constantly be in the interior of CH(*W*). Thus, Lemma 1 is used in this work for force closure verification by checking if the interior point ***r*** and the origin *O* are on the same side of each of the convex hull's facets.

3.3. Grasp quality measure

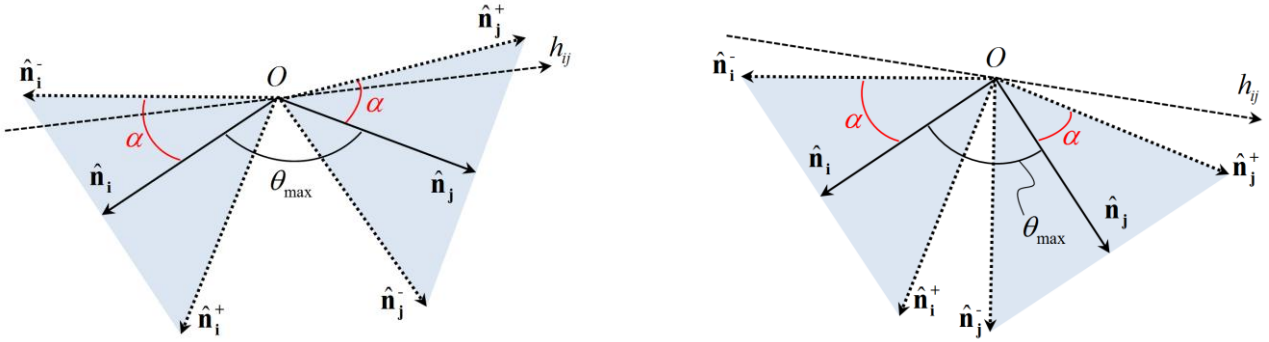
A quality measure is defined enabling quantification and as a comparison tool for a grasp. The quality measure is defined such that it can measure how much a grasp can resist an external wrench without the fingers losing contact or starting to slip [31]. Increasing the quality of a grasp reduces the magnitude of the contact forces that are required to counter-balance an external wrench. Thus, a higher quality measure reduces object deformations and actuator resources. The quality measure will be used as a grasp criterion for the algorithm presented later in this paper.

There are several quality measures, most of them based on the *task wrench set* (TWS). The TWS is a wrench set of all external wrenches that need to be applied during execution of a prescribed task. In general, the quality measure is the relation between which wrenches need to be applied (denoted by the TWS) for the tasks to be done and what wrenches can be applied (denoted by the GWS) based on the position of the contact points and the friction model chosen. The most common quality measure is the largest ball criterion that will be used in this work. This measure is based on a general TWS and is used when there is no prior knowledge of the task forces. In this method, the grasp quality is equivalent to the radius of the largest ball centered at the origin of the GWS and fully contained in the Convex-Hull of *W* [5]. The shortest length between the center of the ball and the boundary of the convex hull is the radius of the largest ball and denotes the direction of the wrench where the convex hull is the weakest. This direction is the weakest because the wrenches formed by the contact points do not have large projections in this direction. In order to resist an external wrench applied in this direction, larger forces at the contact point would have to be applied relative to other directions. Therefore, we define the quality to be the radius of the ball. In other words, the grasp quality measure is defined as the distance from the origin of the GWS to the closest facet of *CH(W)*.



(a) (b)

Fig. 5 - A set of vectors that (a) does not span and (b) does span the plane.



(a) (b)

Fig. 6 - The cone edge vectors constructing the friction cones (a) span and (b) do not span the subspace $\mathbb{R}^2$.

Mathematically we can say that the quality measure Q is defined as

$$Q = \min_{w \in \partial W} \|\boldsymbol{w}\| \tag{7}$$

where ∂W is the boundary of $CH(W)$ [4]. The quality measure in this method denotes the weakest net wrench that can be applied to counter external wrenches on its direction. Fig. 4 illustrates the GWS and the largest ball contained in it. The quality measure Q is the radius of the ball. This means that large contact forces would have to be applied when an external wrench is applied in the weakest direction, defined by the vector from the origin to the point where the Q-sized ball is tangent to the boundary of $CH(W)$.

3.4. Sufficient non-force closure condition

Due to high complexity of the generation of a convex hull and computing the largest ball criteria, a preliminary condition has to be added in order to filter out non-force-closure grasps prior to the "expensive" computation of the convex-hull. This can prevent excessive convex-hull computation and decrease the CPU time. The following Lemma is a necessary condition for a set of vectors to positively span the entire space and was proposed by Niparnan et al. [14].

Lemma 2. *A necessary condition for a set of vectors to positively span $\mathbb{R}^n$ is that the projection of the vectors on any subspace $\mathbb{R}^{t<n}$ must positively span the subspace.*

The following Lemma is a sufficient condition for a set of vectors $\boldsymbol{v_1},...,\boldsymbol{v_g}$ not to span a plane and a proof for the opposite condition is presented in the work of Niparnan and Sudsang [2].

Lemma 3. *A set of vectors does **not** span $\mathbb{R}^2$ if there exists a vector $\boldsymbol{v_i}$ $(i = 1,...,g)$ that satisfies the condition*

$$\boldsymbol{v_i} \cdot \boldsymbol{v_j} > 0, \ \forall j = 1,...,g, \ i \neq j \tag{8}$$

Proof. Given *g* vectors $\boldsymbol{v_1},...,\boldsymbol{v_g}$, if a vector $\boldsymbol{v_i}$ exists where its dot products with all other vectors are all positive, it means that its angle with all of them is less than 90°, i.e., all vectors are at the same plane and do not span $\mathbb{R}^2$ (as illustrated in Fig. 5a). However, if such vector does not exist, there is a possibility that the set of vectors span $\mathbb{R}^2$ (example of such is illustrated in Fig. 5b) and the vectors should be further investigated. ■

Lemma 3 is a sufficient condition for the normals at the contact points not to span $\mathbb{R}^2$. However, even if the normals do not span $\mathbb{R}^2$, the cone edge vectors constructing the friction cones may span. The following Lemma is a sufficient condition for the cone edge vectors **not** to span $\mathbb{R}^2$. This condition was inspired by the one described by Prado and Suarez [15].

Lemma 4. *Let* θ_{ij} *be the angle between* $\boldsymbol{n_i}$ *and* $\boldsymbol{n_j}$ *for all* $i, j = 1,...,g$ *and* $i \neq j$. *A sufficient condition for a set of 2g cone edge vectors* $\boldsymbol{n_k^+}, \boldsymbol{n_k^-}$ $(k = 1,...,g)$ ***not*** *to span* $\mathbb{R}^2$ *is for the associated normals* $\boldsymbol{n_1},...,\boldsymbol{n_g}$ *to satisfy Lemma 3 and for all possible pairs* $\boldsymbol{n_i}, \boldsymbol{n_j}$ *to satisfy the condition*

$$\theta_{\max} < 180^\circ - 2\tan^{-1}\mu \tag{9}$$

where $\theta_{\max} = \max_{ij} \theta_{ij}$ *for all* $i, j = 1,...,g$ *and* $i \neq j$.

Proof. The requirement for force-closure is for the normals at the contact points to positively span the entire wrench space, that is $\mathbb{R}^3$ (in the 2D case). According to Lemma 2, a necessary condition for this is for the normals to positively span a subspace of $\mathbb{R}^3$ as well. Therefore, we can analyze the normal directions to positively span the force space. If all normals and their friction cones are at one side of a half space, the subspace $\mathbb{R}^2$ cannot be positively spanned and therefore the grasp is definitely not force-closure. If Lemma 3 is satisfied all normals are at one side of the half space and it is required that the friction cones be at the same side as well. Denote the angle between the normals $\mathbf{n_k}$ and $\mathbf{n_h}$ as $\theta_{kh} = \cos^{-1}(\boldsymbol{n_k} \cdot \boldsymbol{n_h})$, and the friction angle $\alpha = \tan^{-1}\mu$ (Fig. 6). Denote $\mathbf{n_i}$ and $\mathbf{n_j}$ to be two normals out of the *g* normals that form the maximum angle $\theta_{\max} = \max_{ij} \theta_{ij}$ for $i, j = 1,...,g$ and $i \neq j$. Let $\boldsymbol{h_{ij}}$ be a supporting line separating $\mathbb{R}^2$ to two half spaces where one half space contains the normals $\boldsymbol{n_i}, \boldsymbol{n_j}$. The condition for the cone edge vectors $\boldsymbol{n_i^+}, \boldsymbol{n_i^-}, \boldsymbol{n_j^+}, \boldsymbol{n_j^-}$ not to positively span the entire $\mathbb{R}^2$ (including the other side of $\boldsymbol{h_{ij}}$) is for the angle between two corresponding edge vectors, for example $\boldsymbol{n_j^+}$ and $\boldsymbol{n_i^-}$ in Fig. 6, to be smaller than 180°, thus satisfying the condition $\theta_{\max} + 2\alpha < 180°$. This is equivalent to condition (9) and is a sufficient condition for a grasp to be non-force-closure (Fig. 6a for a positive span example and Fig. 6b for a non-positive span example). ■

The following theorem is the final sufficient condition for a grasp to be non-force-closure and is directly deduced from Lemmas 2 to 4.

Theorem 2. (Non-force closure grasp) *A sufficient condition for a set of g frictional contacts* ***not*** *to be force closure is that the g normals at the contact points do not span* $\mathbb{R}^2$*, thus satisfying the conditions of Lemma 3 and Lemma 4.*

Proof. According to Lemma 2, a set of vectors that do not span $\mathbb{R}^2$, do not span $\mathbb{R}^3$ as well. If Lemma 3 and Lemma 4 are satisfied, the set of *g* normals at the contact points and their friction cones do not span $\mathbb{R}^2$ and the grasp is non-force-closure. ■

The described theorem may be used to filter out some candidate grasps and the ones **not** satisfying the condition are to be further investigated using the convex-hull approach.

4. FCGS generation algorithm

We generate the set of all possible grasps for each object. The force closure grasps are represented as a *D*-dimensional vector of numerical features called feature vectors. These feature vectors construct a *D*-dimensional set called the *Force*

Closure Grasp Set (FCGS). This section presents the proposed structure of the grasp feature vector and the method for constructing the FCGS.

4.1. Grasp feature vector

In this paper we only consider planar objects. Assume an object to be grasped by *g* fingers (Fig. 3). A *g*-finger grasp of an object *B* can be defined by a set of contact points, $P = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_g)$, on the object, and the normal to the object surface at each point. This grasp definition can be mapped into a set of parameters injectively representing the grasp. A parameter set for a grasp can be written as a *D*-dimensional feature vector $\boldsymbol{e} = (u_1 \quad \ldots \quad u_D)^T$.

Theorem 3. (Polygon parameterization) *A parameterization of a k-sided polygon requires 2k-3 constraints, which determine its shape and size.*

Proof. Any *k*-sided polygon has as many angles as the number of edges and their sum (sum of the inner angles) is $180(k-2)$. Therefore, defining *k*-1 angles summed to $\alpha_{k-1} = \sum_{i=1}^{k-1} \gamma_i < 180(k-2)$, necessarily constrains the remaining one to be $\gamma_k = 180(k-2) - \alpha_{k-1}$. Moreover, any *k*-sided polygon can be constructed by *k*-2 triangles. And therefore, after constraining the angles, one edge of each triangle (in this case the one on the outer boundary of the polygon) defines its size, giving additional k-2 constraints to the polygon. This, with a total of *2k-3* constraints, is needed to define a polygon's shape and size. ■

For a general case of a *g*-finger grasp, the grasp can be represented by a *g*-sided polygon, where each vertex is positioned in the contact point. The *g*-sided polygon describing a specific grasp can be created by constructing concave or convex polygons from a set of points in a plane according to polygonization algorithms (see [32]). As mentioned, 2*g*-3 constraints are needed to define the polygon (Fig. 7). The internal angles of the polygon are given by

$$\gamma_i = \begin{cases} \cos^{-1}\left[\dfrac{(\boldsymbol{p}_2 - \boldsymbol{p}_1)\bullet(\boldsymbol{p}_g - \boldsymbol{p}_1)}{\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|\|\boldsymbol{p}_g - \boldsymbol{p}_1\|}\right], & i = 1 \\ \cos^{-1}\left[\dfrac{(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)\bullet(\boldsymbol{p}_{i-1} - \boldsymbol{p}_i)}{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|\|\boldsymbol{p}_{i-1} - \boldsymbol{p}_i\|}\right], & i = 2, \ldots, g-1 \end{cases} \tag{10}$$

and they provide *g*-1 constraints. The length of the edges are given by

$$d_i = \|\boldsymbol{p}_i - \boldsymbol{p}_{i+1}\|, \; i = 1, \ldots, g-2 \tag{11}$$

and they provide *g*-2 constraints, a total of 2*g*-3 shape constraints defining the grasp polygon.

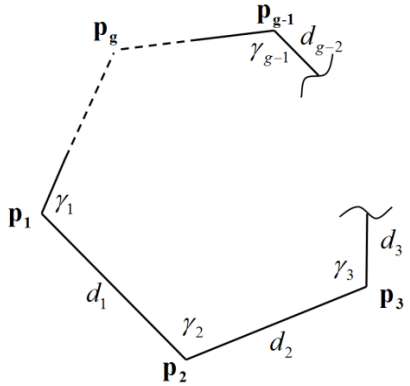


Fig. 7 - g-sided polygon.

At each contact point, the normal has to be defined relative to the grasp polygon. The normal's directions can be defined by its angles relative to the adjacent edges and is calculated using

$$\theta_i = \begin{cases} \cos^{-1}\left[\dfrac{\hat{\boldsymbol{n}}_1 \bullet (\boldsymbol{p}_g - \boldsymbol{p}_1)}{\|\boldsymbol{p}_g - \boldsymbol{p}_1\|}\right], & i = 1 \\ \cos^{-1}\left[\dfrac{\hat{\boldsymbol{n}}_i \bullet (\boldsymbol{p}_{i-1} - \boldsymbol{p}_i)}{\|\boldsymbol{p}_{i-1} - \boldsymbol{p}_i\|}\right], & i = 2,\ldots,g \end{cases} \tag{12}$$

There are g angles, one for each contact point. From these 3 sets of parameters defining the g-finger grasp polygon's shape and size and the direction of the contact normal, a D-dimensional feature vector is constructed and represented as:

$$\boldsymbol{e} = \begin{pmatrix} \gamma_1 & \cdots & \gamma_{g-1} & d_1 & \cdots & d_{g-2} & \theta_1 & \cdots & \theta_g \end{pmatrix}^T . \tag{13}$$

If we sum the number of parameters in feature vector $\boldsymbol{e}$, its dimension in the general case is given by

$$D = (2g-3) + g = 3g - 3 \ . \tag{14}$$

Example 1. Assume an elliptical object to be grasped by a 3-finger grasp. Such a grasp can be represented by a triangle (3-sided polygon). Fig. 8 presents a possible 3-finger grasp. The position of the 3 contacts relative to each other can be injectively represented as a triangle by two angles γ_1, γ_2 and the edge length between them d_1. The normal at each contact point can be represented relative to the triangle by the angle θ_i (i=1,2,3) it makes with the adjacent edge of the triangle. Therefore, the 3-finger grasp can be injectively defined by a 6-dimensional feature vector $\boldsymbol{e}$,

$$e = \begin{pmatrix} \gamma_1 & \gamma_2 & d_1 & \theta_1 & \theta_2 & \theta_3 \end{pmatrix}^T \tag{15}$$

where the parameters of the feature vector are given by equations (10)-(12). This representation will be used for the simulations and experiments presented in this paper. A triangle can be injectively described by two angles and the edge length between them. However, it should be mentioned that there are three different representations for a triangle, depending on which edge is picked for defining d_1. Therefore, in our implementation, we always pick the longest edge of the triangle to represent d_1. This ensures that the algorithm will always observe the triangles from the same point of view. This selection is a general one, and the largest edge will always be picked to be d_1 for any g-sided polygon; in such case, the next edge length d_2 will be adjacent to d_1 through γ_2 and so on.

The advantage of this feature vector representation is that it defines a grasp independent of the object and its reference frame, and therefore can be compared to other feature vectors of other objects.

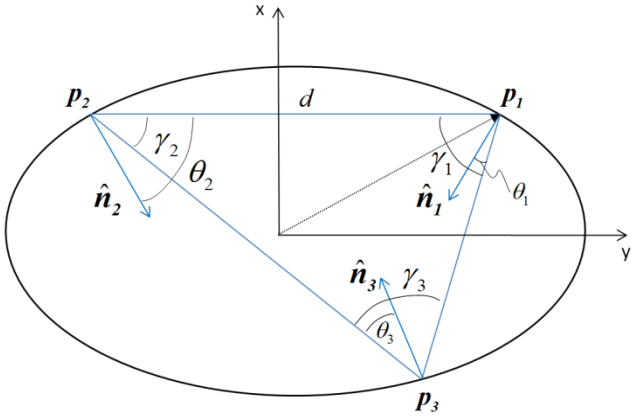


Fig. 8 - Six parameters representing a 3-finger grasp.

4.2. Force-closure check

In section 3 the force-closure criterion was presented. After sampling and generating all grasp feature vectors of a certain object, force-closure check has to be done for each possible grasp in order to omit those that do not meet the force-closure criterion. The force-closure criterion checks whether the grasp is force closure (according to theorem 1) and its quality measure Q is greater than a predefined value Q_d, measured according to the GWS and TWS method proposed previously. The value of Q_d is for now defined manually according to the desired size of the FCGS; in future work an algorithm for defining it may be applied. Algorithm 1 presents the method of doing so.

4.3. Grasp set generation

Generation of the FCGS is done by going over all possible grasp feature vectors of object *B*, omitting those that are not force-closure or have low quality measure (using algorithm 1). Algorithm 2 receives as an input a uniform mesh of a CAD model. The 2D mesh can be generated manually, using discrete functions or using image processing techniques.

Algorithm 1 Force-Closure check

Input: A *g*-finger grasp set $P_j = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_g)$ defining grasp *j* of object *B*, and the minimal desired grasp quality Q_d.

Output: True – if grasp is force-closure, False – if not.

1: Map[1] P_j to a system of *2g* wrenches W_j.
2: **If** Theorem 2 for grasp P_j is false **then**
3: Compute the convex hull of W_j: $CH(W_j)$
4: **If** $O \in \text{interior}\left(CH(W_j)\right)$ **then** /* using Lemma 1.
5: Compute the quality measure *Q*.
6: If $Q \geq Q_d$
7: **Return** True /* grasp *j* is force-closure.
8: **end if**
9: **end if**
10: **end if**
11: **Return** False /* grasp *j* is non-force-closure.

The *k*-sized mesh consists of the set of points $K = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k)$ representing the boundary of the object and its matching set of normals at each point $N = (\boldsymbol{n}_1, \ldots, \boldsymbol{n}_k)$. The algorithm samples possible grasp feature vectors for the object and computes a set $E \in \mathbb{R}^D$ representing all combinations of *g*-points that achieve force-closure under the frictional contact assumption. Thus, the output of the algorithm will be a set of feature vectors $E = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_v) \in \mathbb{R}^D$ representing the set of all force closure grasps of object *B*. The algorithm is based on the one proposed in [33].

Algorithm 2 FCGS generation

Input: Mesh of object *B* to be grasped given by *K* and *N*.

Output: FCGS $E = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_v)$ of object *B*.

1: Generate grasp *j* defined by $P_j = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_g) \in K$.
2: **If** P_j is force-closure **do** /* according to algorithm 1.
3: Map grasp *j* to feature vector $\boldsymbol{e}_j = \begin{pmatrix} u_1 & \ldots & u_D \end{pmatrix}^T$.
4: Label $\boldsymbol{e}_j$ as force-closure and add to set *E*.
5: Store link between $\boldsymbol{e}_j$ and P_j.
6: **Else**
7: Label $\boldsymbol{e}_j$ as non-force-closure.
8: **end if**
9: **If** all possible grasps of object *B* are not fully labeled, **then goto** step 1.
10: **Else Return** grasp set $E = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_v)$.

[1] Mapping is done by using eq. (2) considering unit force applied at each contact.

5. Multi-sets common vectors search

Given *n* sets of vectors in *D*-dimensional space $E_1,\ldots,E_n \in \mathbb{R}^D$ representing the FCGS of each object and a set of parameter tolerances for each dimension $\varepsilon_1,\ldots,\varepsilon_D \in \mathbb{R}$, a similarity algorithm will take the sets and output a set of vectors $Z \in \mathbb{R}^D$ that are common to two or more of the sets $E_1,\ldots,E_n$. Two vectors are considered to be common if the projected distance between them maintains the following condition,

$$\left|\boldsymbol{e_i}(k)-\boldsymbol{e_j}(k)\right| \le \varepsilon_k \text{ for } \boldsymbol{e_i} \in E_i, \boldsymbol{e_j} \in E_j, \forall k = 1,...,D \tag{16}$$

where $\boldsymbol{e_i}(k)$ is the i^{th} component of $\boldsymbol{e_i}$. This distance criterion will be reviewed later in the paper. Every vector in *Z* will be labeled with the sets from which it originated.

5.1. *kd*-tree database & nearest neighbor search

The data representation used in this work is the well-known *kd*-tree [24]. Each FCGS set is represented as a high-dimensional binary tree enabling efficient search. The *kd*-tree data representation is based on dimension partitioning. All nodes in the tree are vectors in the *k*-dimensional space and are used to mark the hyper-plane position that partitions the current level's dimension. For every node, vectors that are larger than the node value (in the current dimension) will go to the right sub-node and the smaller ones will go to the left sub-node. Hence, every level in the tree is a partition of one dimension in the *k*-dimensional space and they are repeatedly split until one vector is left in the formed subspaces. Fig. 9 is an example of a *kd*-tree in a 2-dimensional space partitioning the x-y plane shown in Fig. 10.

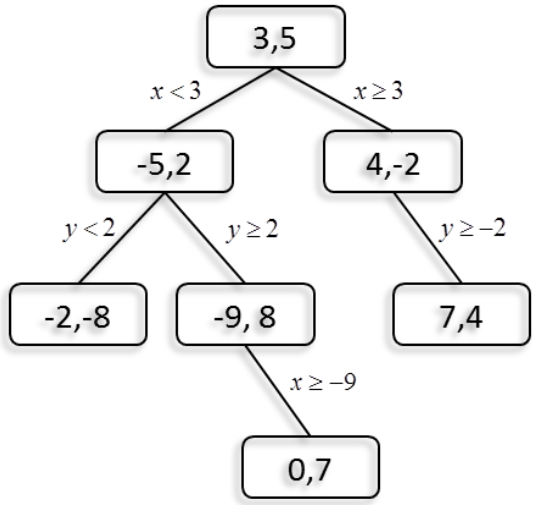


Fig. 9 - The form of a 2D-tree (*k*=2).

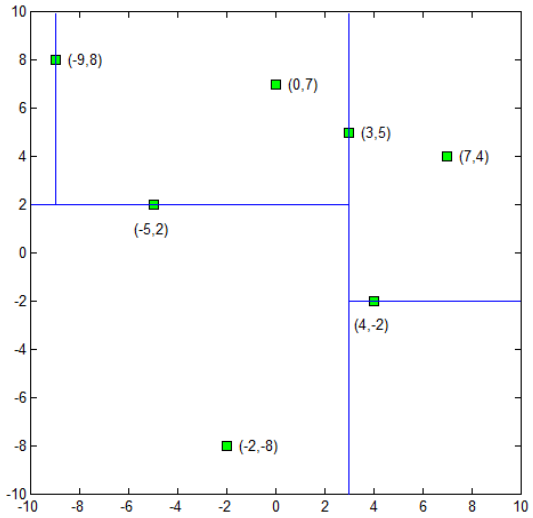


Fig. 10 - Partitioning of the 2D space by the *kd*-tree.

Several algorithms for nearest neighbor search, which operates on *kd*-tree, have been widely surveyed in literature [21]. They take advantage of the *kd*-tree structure, which enables efficient search of the tree. In most cases, during the search only a few leaf nodes are required to be checked. It should be noted that the nearest neighbor search algorithm uses a Euclidean distance criterion; we will present another custom distance criterion that will be used based on our grasping demands.

5.2. Similarity join

The main idea of the search algorithm is: for every two high-dimensional sets E_i, E_j, represented in the *kd*-trees, find vectors that are common to the two sets. A nearest-neighbor search algorithm is used for finding for every two FCGS sets, all pairs of vectors that are within a predefined distance. Every two vectors found are close enough to be considered as one. Every vector found is checked to determine whether it is already in the registry set *Z*. If it exists in *Z*, it is labeled to be in E_i and E_j. If not, it is added to *Z* and then labeled to exist in E_i and E_j.

The ***Join FCGS*** procedure receives all FCGS data sets $E_1,\ldots,E_n \in \mathbb{R}^D$ and, for every two sets, using the ***Nearest Neighbor*** function, pairs vectors in one set to the nearest one in the other. The ***Nearest Neighbor*** function is the algorithm mentioned in subsection 5 of this section. The ***Nearest Neighbor*** search is further implemented in simulations using the Matlab *knnsearch* function. The *knnsearch* function receives two FCGS sets E_i, E_j and outputs two parallel sets *F and G* (size of *G* equals the size of *F*) stored as a list, where the vector $\boldsymbol{e_i} \in F, E_i$ positioned in the *k* place in *F* is the closest one to $\boldsymbol{e_j} \in G, E_j$, which is positioned in the *k* place in *G*. After all the pairs are found, further inspection of whether they are close enough (in the grasping meaning) to be considered as the same grasp should be done. Therefore, function ***IsSimilarVector***($\boldsymbol{e_i}, \boldsymbol{e_j}$) calculates the projection distances between the two vectors along the major coordinate axes and checks whether they are both inside a hyper-rectangle with edge lengths of the predefined tolerances $\varepsilon_1,\ldots,\varepsilon_D$. If they are, then the vectors are considered to be the same. It should be noted that the nearest neighbor search is applied to find the 1st nearest neighbor. If solutions found are insufficient, a wider search to the 2nd, 3rd, or *k*th nearest neighbor can be done to find more potential pairs that satisfy the distance criterion and could provide more grasp solutions.

Algorithm 3 Function **Join_FCGS(** $E_1,\ldots,E_n$ **)**

Input: FCGS of each object.
Output: Registry set *Z* containing vectors that are common to 2 or more FCGS.

```
1:  For i = 1 to n-1 do
2:    For j = i+1 to n do
3:        [F, G] = Nearest_Neighbor( E_i, E_j )
4:        For k = 1 to size(F) do
5:            If IsSimilarPoint(e_k^F ∈ F, e_k^G ∈ G) = true then
6:                    Z = Insert_to_Z( e_k^F , e_k^G ,i, j).
7:            end if
8:        end for
9:    end for
10: end for
11: Return Z.
```

A pair of vectors found that are considered to be the same is the input to procedure ***Insert_to_Z***, which takes the mean vector of the pair (they are considered to be close enough) and checks whether it exists in the registry set *Z*. As mentioned, the set *Z* is a *D*-dimensional database of the vectors that are common in two or more sets of $E_1,\ldots,E_n$. The set *Z* is also represented in a *kd*-tree database. This way, nearest-neighbor search is easily possible.

Algorithm 4 Function **Insert_to_Z(** $\boldsymbol{e_i}, \boldsymbol{e_j}$ **, i, j)**

Input: Vector $\boldsymbol{e_i}$ in FCGS *i* and vector $\boldsymbol{e_j}$ in FCGS *j*.

Output: Updates registry set *Z* to contain $\boldsymbol{e_i}$ and $\boldsymbol{e_j}$.

```
1: v=mean( e_i , e_j )
2: u = Nearest_Neighbor( Z, v )   /* Search in tree Z for nearest vector u to v.
3: If IsSimilarPoint(v, u) = true  then /* Vector v exists in Z
4:       set ũ_i = 1 ; ũ_j = 1
5: Else   /* Vector v doesn't exist in Z, Add new vector v to Z.
6:       Add vector v to Z.
7:       set ṽ_i = 1 ; ṽ_j = 1
8: end if
9: Return Z.
```

Let U_n be an *n*-dimensional vector space of binary values, i.e., a vector with *n* components consisting of 0's and 1's. Each common vector added to *Z* is a *D*-dimensional vector $\boldsymbol{v}_i \in \mathbb{R}^D$, denoting the position of the vector in the *D*-dimensional space. A compatible vector $\tilde{\boldsymbol{v}}_i \in U_n$ is coupled to $\boldsymbol{v}_i$. Component *k* of vector $\tilde{\boldsymbol{v}}_i$ denotes whether the vector is in the set E_k if labeled "1" and "0" if not.

Algorithm 5 Function **IsSimilarVector(** $\boldsymbol{e}_i, \boldsymbol{e}_j$ **)**

Input: Vectors $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$.

Output: True if vectors in predefined tolerances and false otherwise.

1: **For** *k* = 1 to *D* **do**
2: **If** $|\boldsymbol{e}_i(k) - \boldsymbol{e}_i(k)| > \varepsilon_i$
3: **Return** False.
4: **end if**
5: **end for**
6: **Return** True.

Algorithm 6 is the main procedure that matches all vector sets with each other to generate the registry set *Z*. After running algorithm 6, the high-dimensional registry vector set *Z* is generated. This registry is a cluster of vectors consisting of the set of common grasp feature vectors that exist in two or more primitive sets. Each feature vector within *Z* is marked with the primitive set it originated from. The classification will be described next.

Algorithm 6 Main algorithm

Input: 2D CADs of *n* objects $B_1, \ldots, B_n$ to be grasped.

Output: Registry set *Z*.

1: **For** *j* = 1 to *n* **do**
2: Mesh object B_j.
3: Label forbidden grasp regions on mesh of object B_j.
4: Generate set E_j = FCGS for B_j /* Using algorithm 2.
5: **end for**
6: ***Z*** **= Join_ FCGS(** $E_1, \ldots, E_n$ **)**

5.3. Classification

We now have a *D*-dimensional registry set *Z* of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in Z$, where each position vector $\boldsymbol{v}_i$ is coupled to a binary vector $\tilde{\boldsymbol{v}}_i$, which denotes its existence in any of the primitive sets $E_1, \ldots, E_n$. The next step is to classify all vectors in the set *Z* so as to find the minimum subset of vectors $H \subseteq Z$ that exists in all of the primitive vector sets. Such a subset is said to cover all the primitive sets $E_1, \ldots, E_n$.

Let a vector $\boldsymbol{u}_i \in Z$. Then its compatible vector $\tilde{\boldsymbol{u}}_i \in U_n$ is a binary vector consisting ones or zeroes. It can be said that a subset $H \subseteq Z$ where $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p \in H$, covers $E_1, \ldots, E_n$ if

$$\bigcup_{i=1}^{p} \tilde{\boldsymbol{u}}_i \Big|_{\min(p)} = \left(\vec{1}\right)_{n\times 1} \tag{17}$$

As mentioned, we want the size *p* of the subset *H* to be minimal while seeking for 1s. The class search algorithm will start by searching for a single vector $\boldsymbol{u}_1$ within *Z* that covers the whole primitive set. This implies that all components of the compatible vector $\tilde{\boldsymbol{u}}_1$ equal 1. If they fail to do so, it will search for two vectors that cover the primitive sets. The algorithm will continue to do so until success. At the end of the process, a minimal set *H* of feature vectors is found, where each of

them is in fact a grasp configuration of the matched subset of the objects. If several solutions are found, the algorithm will produce the one with the largest quality measure.

5.4. Algorithm Computation Complexity analysis

Given *n* shapes, they are each discretized to a *k* size mesh of vectors. For the FCGS part we generate *n*-dimensional sets of *g*-finger grasps. Let *N* be the number of all possible combinations of *g*-finger grasps and its size is given by $N = \frac{k!}{g!(k-g)!}$. We generate a convex hull for a *g*-finger grasp using *2g* wrenches (2 bounding wrenches for each friction cone). The Matlab[2] integrated Convex hull generation function uses the Quickhull algorithm =[34]=[34]. Generation of a convex hull for each grasp configuration in the worst case is done in $O(2g\log(2g))$ time if $d \le 3$. Therefore, generation of *n* FCGS sets is done in $O(n \cdot N \cdot 2g\log(2g))$ time. Nearest neighbor search of a target vector in the *kd*-tree takes an average of $O(\log N)$ inspections. In order to join each pair of sets to find common vector pairs, it takes up to $O(\log^2 N)$ inspections. Therefore, nearest neighbor search between every two sets is done in $O\left(\frac{n!}{2!(n-2)!}\log^2 N\right)$ time. We find $m \le N$ vectors that are common to two or more sets in up to $O(\log m)$ time for every pair found. Therefore, generation of the Z set takes up to $O\left(\frac{n!}{2!(n-2)!}\log^2 N \log m\right)$ time. An additional $O(m)$ time is required for classification, yielding an overall runtime of $O\left(2gn\left(\frac{k!}{g!(k-g)!}\right)\log(2g) + 0.5n^2\log^2\left(\frac{k!}{g!(k-g)!}\right)\log m + m\right)$. In the worst case, $m = N$ and assuming $g \ll k$ we obtain a polynomial runtime of $O(nk^g)$. Therefore, the algorithm proposed will produce a solution if it exists in polynomial time of the order of $O(k^g)$. However, this is the worst case complexity possible, because the high-complexity comes from the analysis of the convex-hull and by adding the condition in Theorem 2 we filter out a large amount of non-force-closure grasps without generating the convex-hull. Moreover, the FCGS for each object can be computed in parallel, which significantly decreases runtime. Parallel computation is implemented in the following simulations.

6. Simulations

For simulations of the proposed method, the algorithm was implemented in MATLAB on an Intel-Core i7-2620M 2.5 GHz computer with 8 GB of RAM. The following simulations present examples of the algorithm operation on 3-finger frictional grasps of four and five simple planar shapes.

6.1. Implementation and results

The performance of the proposed algorithm is illustrated using the four 2D shapes shown in Fig. 11. The shapes are described with a mesh of *k*=158 vectors uniformly distributed along their boundary. For such mesh there are $\frac{158!}{3!(158-3)!} = 644{,}956$ possible grasps. According to algorithm 2, the FCGS is generated for every shape. Each candidate grasp is checked for force-closure and only grasps with quality measure greater than 0.1 are approved (for convenience, the quality measure *Q* is normalized by the maximum quality measure of all sampled grasps and is bounded by $0 < Q \le 1$). Fig. 12 shows one generated FCGS of rectangular shape. Due to the high-dimensionality of the space, for illustration, each space is projected to two 3-dimensional spaces.

[2] Matlab® is a registered trademark of The Mathworks, Inc.

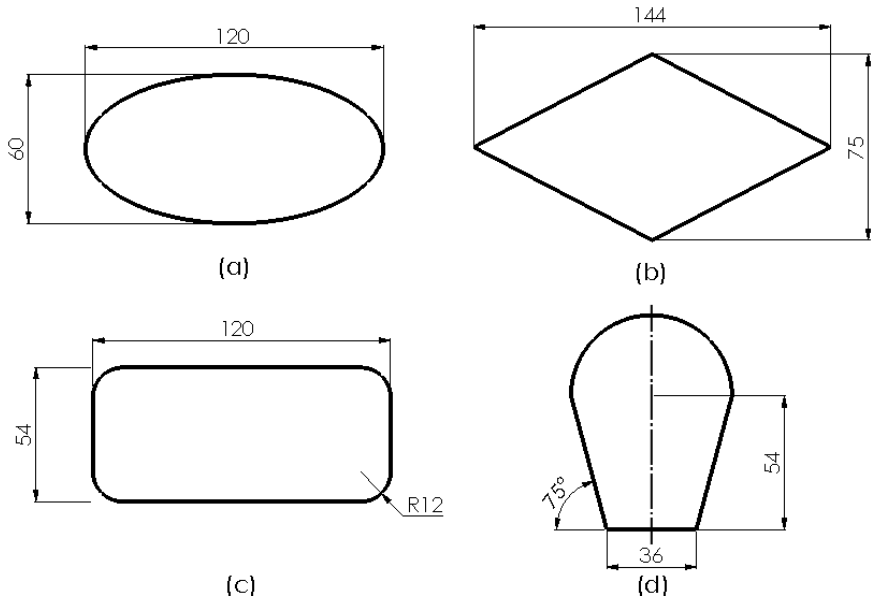


Fig. 11 - Four planar shapes to be grasped: (a) ellipse, (b) dalton, (c) rectangle, and (d) an ice cream cone.

The edge tolerances of the similarity search $\varepsilon_1[°], \varepsilon_2[°], \varepsilon_3[mm]$ are not constant numbers and they were chosen in such a way that the edges of the triangle will not extend or shorten by more than 6% of their original length due to changes in the angles of the triangle (see appendix for the corresponding criteria). These tolerances are continuously computed during the simulation execution. Similarly, the angular tolerances $\varepsilon_4, \varepsilon_5, \varepsilon_6[°]$ are defined to be 70% of the friction cone's angle (the friction coefficient was chosen to be 0.6). Under these conditions, the algorithm's outputs are 10 solutions of common grasps for all shapes. Fig. 13 shows 6194 vectors/grasps in registry set *Z* that are common to 2 or more shapes (the blue dots). The 10 solutions that are marked with red squares are the common grasps for all shapes.

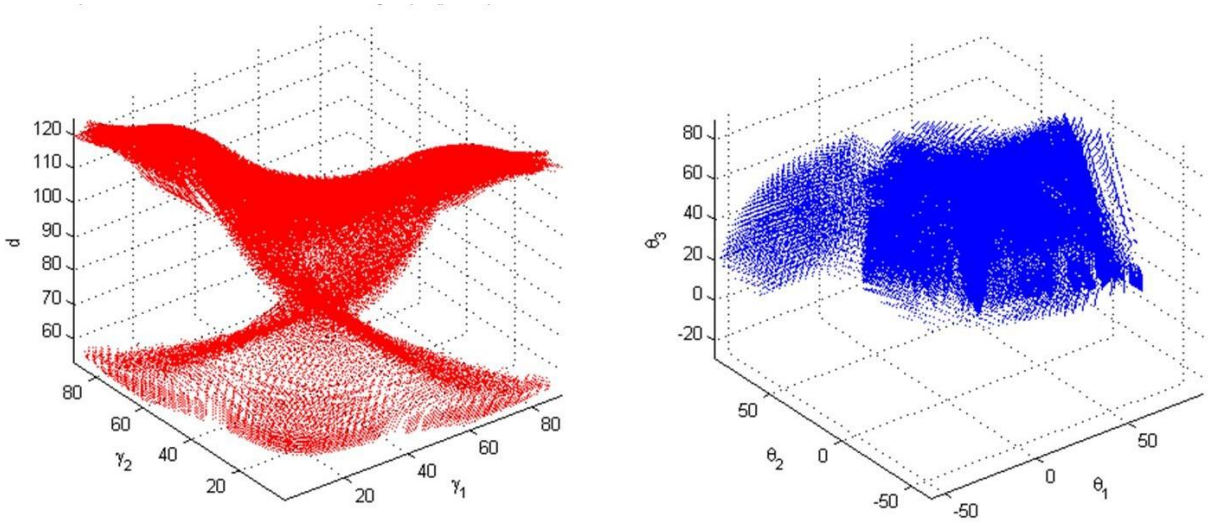


Fig. 12 - FCGS for rectangular shape.

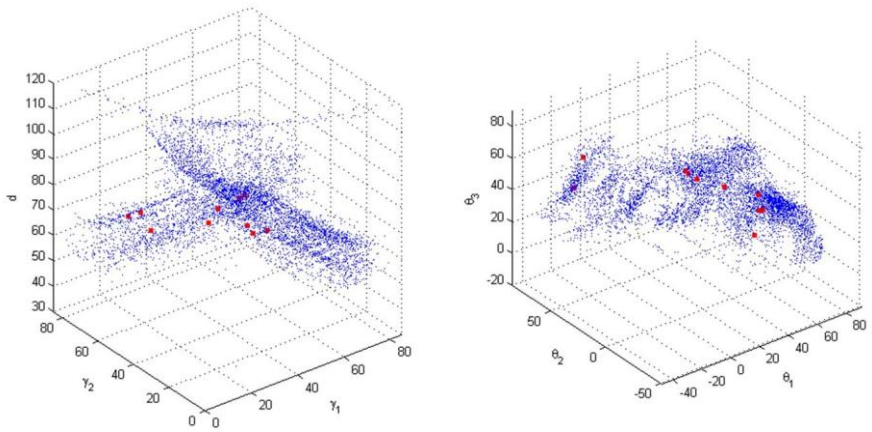


Fig. 13 - Registry set Z and the 12 solutions (red squares).

In the implementation of the algorithm, for comparison reasons, we have normalized the quality measures by the maximum radius of the largest ball of all feasible grasps of all objects. The highest quality measure solution (Q = 0.51 after normalization) is shown in Fig. 14 and its configuration is illustrated in Fig. 15. The convex hull and the largest ball of this configuration are shown in Fig. 16. The radius of the ball is shown before the mentioned normalization, while in Fig. 15 the quality measure (Q=0.51) is the radius of the ball divided by the maximal ball of all grasps of all objects. This solution will be used for experimental validation and will be shown in the next section. The presented configuration is the mean configuration of the four grasps. Because of the defined tolerances, $\varepsilon_4, \varepsilon_5, \varepsilon_6$, the normals of the mean configuration are within the friction cones defined by the normals of each grasp at the contact points. In this result, some of the contact points are marginal, i.e., near a vertex. However, there are other results such as Fig. 17 that do not have contact points near a vertex. As mentioned, if contact points near a vertex are unwanted, their area should be added to the forbidden zones by the operator prior to the computation.

Simulations were performed to four- and five-finger grasps as well. Fig. 18 presents a solution for a 4-finger grasp for the four objects when the mesh size for the calculations was k=150. Fig. 19 presents the simulation output for a 5-finger grasp of the four objects with mesh size of k=150. For both computations, the tolerance demands were the same as presented for the 3-fingers grasps. Slight differences can be seen between the grasps as the tolerances allowed. With higher mesh resolution, more accurate solutions could be achieved; this of course is based on the grasp accuracy demands.

Some of the grasps presented may be seen to be unable to counterbalance an external torque in one direction according to the normals direction. However, the normals are not the ones that positively span the wrench space, but rather the forces within the friction cones centered on the normals.

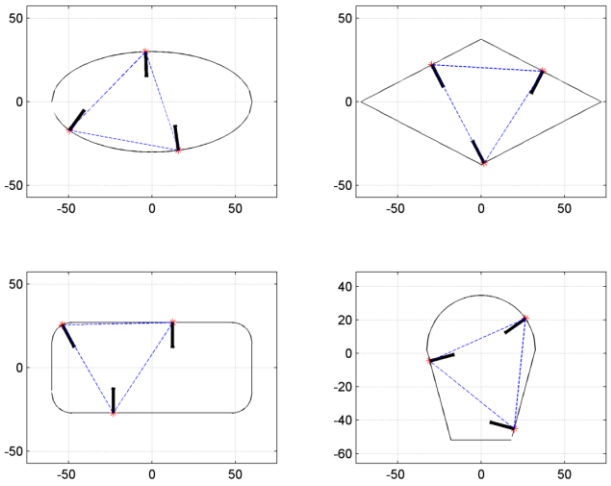


Fig. 14 - One of the grasp solutions.

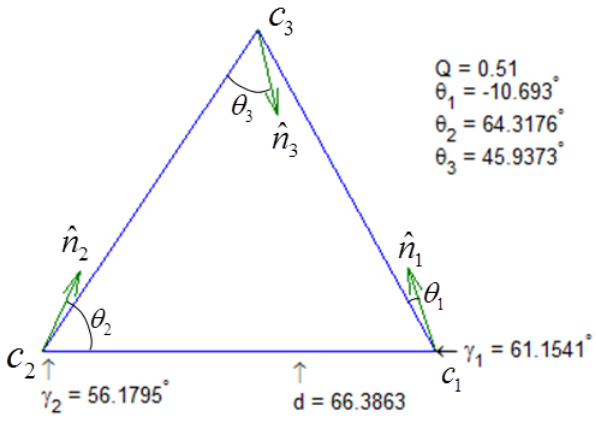


Fig. 15 - The common grasp configuration.

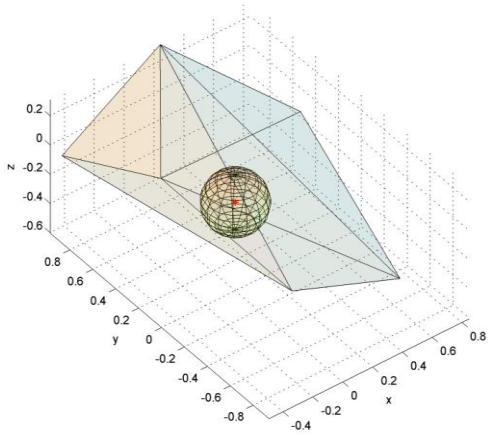


Fig. 16 - Solutions for convex hull and grasp quality.

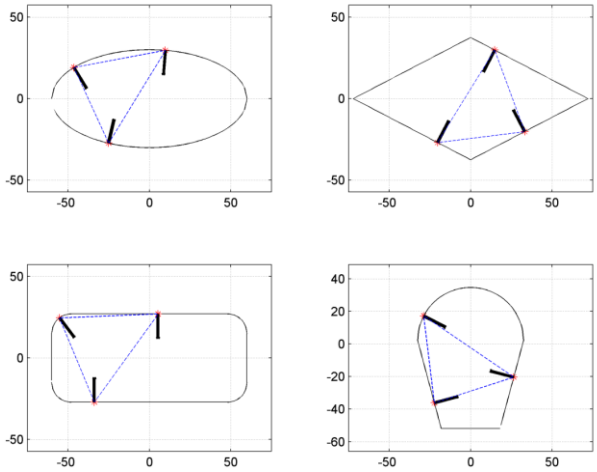


Fig. 17 - Another solution of a 3-finger common grasp.

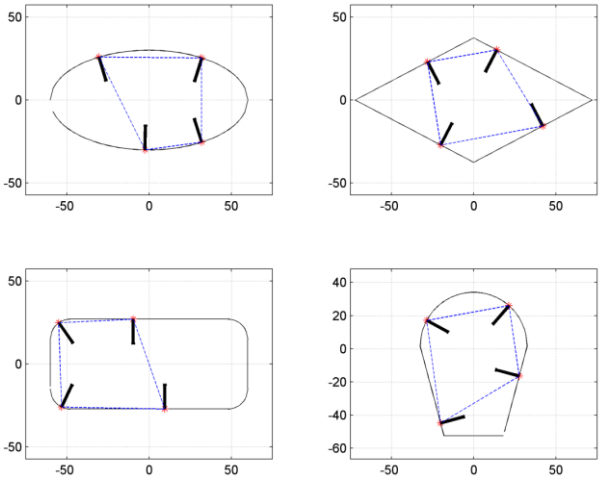


Fig. 18 - Solution of a 4-finger common grasp.

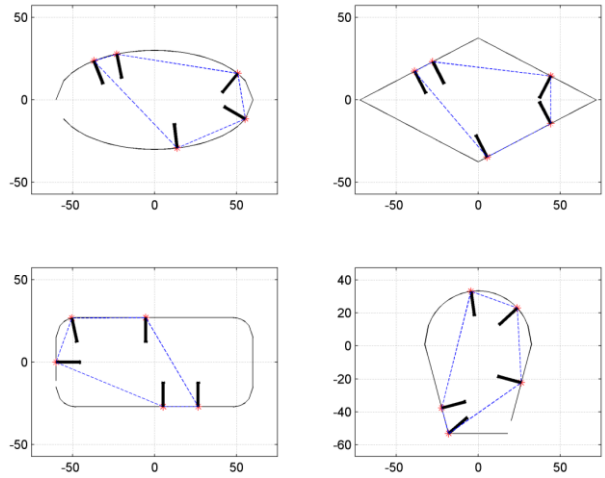


Fig. 19 - Solution of a 5-finger common grasp.

6.2. Performance

Fig. 20 shows performance measures as a function of the mesh size chosen. Fig. 20a is the time required for the generation of the FCGS. We have seen significant improvements of the runtime using Theorem 2. Running the algorithm using Theorem 2 decreased CPU runtime up to 25%.

Fig. 20b shows the time required for the similarity search and classification until outputting the common grasp. Exponential behavior can be seen in both plots. Fig. 20c shows the total number of possible grasps and the force closure grasps. Fig. 20d shows the size of registry set Z, which is the number of common grasps for 2 or more shapes. Figure 16e is the number of solutions found, meaning the number of common grasps for all shapes. Fig. 20f presents the number of solutions found relative to the number of objects tested; this was calculated using mesh of k=118. We have also added a fifth test case (the crescent concave shape shown in Fig. 21). The two objects were the ellipse and the dalton and the three objects were the ellipse, Dalton, and rectangle. It can be seen that as the number of objects increases, it is harder to extract common grasps.

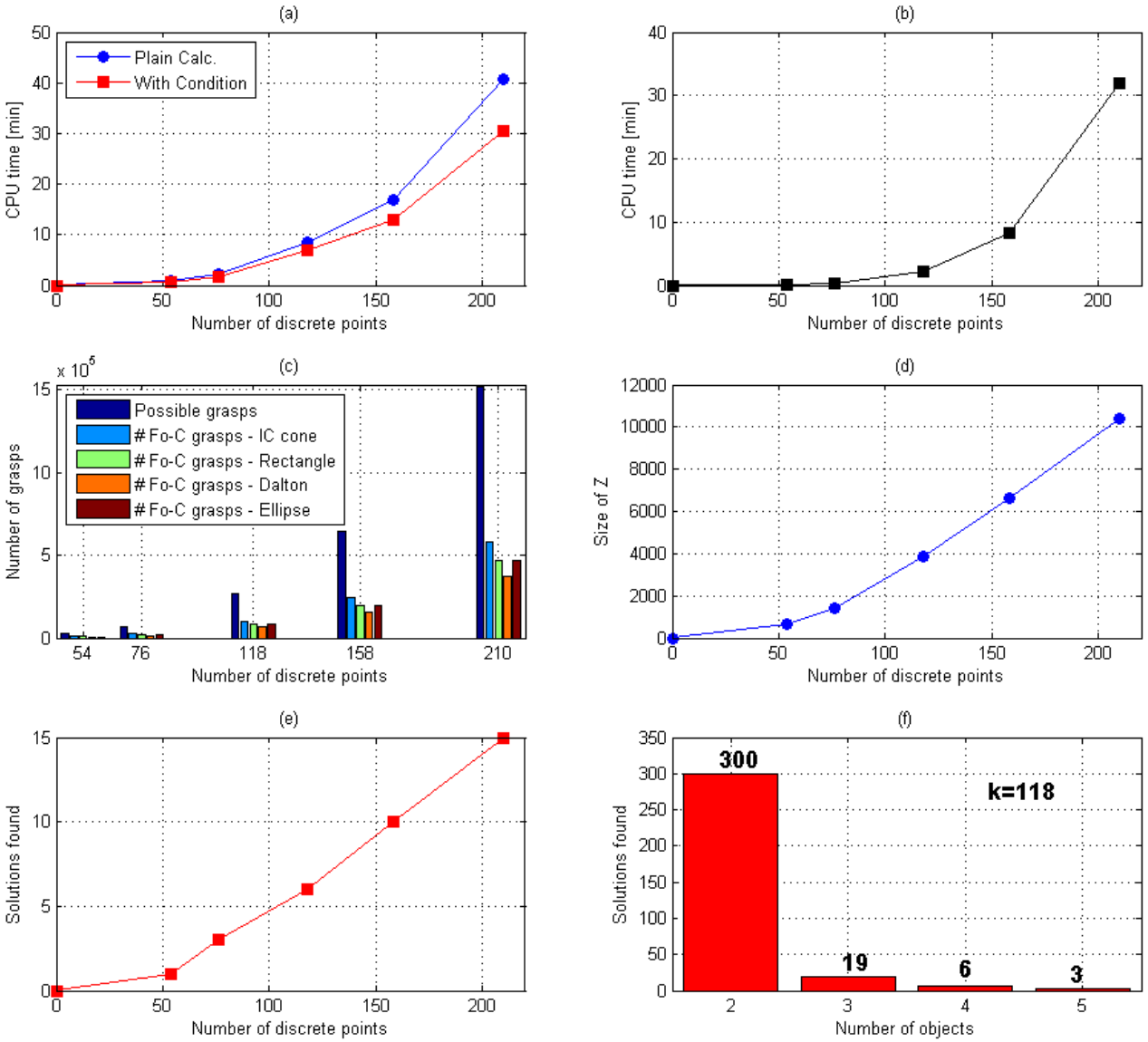


Fig. 20 - Performance parameters in the search algorithm relative to the mesh number: (a) FCGS generation solution time, (b) Similarity search and classification time, (c) Total number of grasps and the number of force closure ones in every shape, (d) Number of grasps common to 2 or more shapes, (e) Number of solutions found, and (f) number of solutions found relative to the number of tested objects (including the fifth crescent shape in Fig. 21).

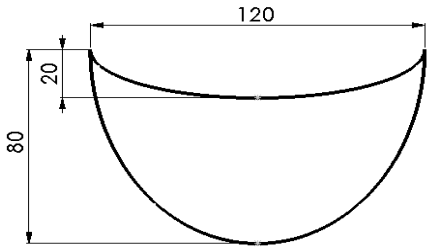


Fig. 21 - The fifth cresent object.

Fig. 22 presents the runtime of the algorithm with reference to the number of fingers *g*, shown here with mesh size *k*=150. Exponential behavior of the number of contact points is shown, as anticipated from the calculated complexity.

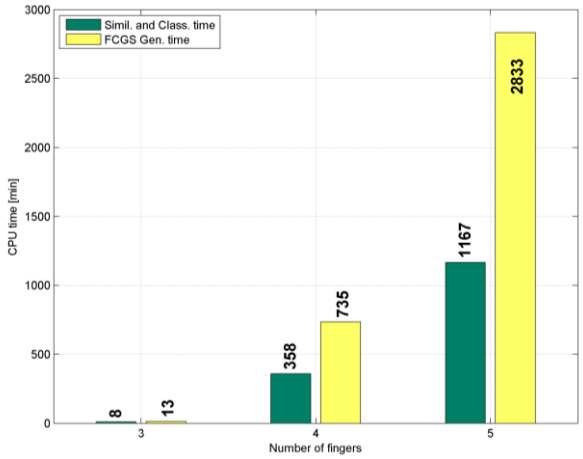


Fig. 22 - CPU runtime as function of the number of fingers (*k*=150).

Sensitivity analysis was done to examine the number of solutions output with the change in the tolerances. The data were calculated on a 3-fingers grasp with mesh size of *k*=75. The tolerances presented in the simulations were multiplied by a factor ξ so that the new tolerances are $\left(\varepsilon_1^{new} \quad \cdots \quad \varepsilon_6^{new}\right)^T = \xi \cdot \left(\varepsilon_1 \quad \cdots \quad \varepsilon_6\right)^T$. Fig. 23 presents the change in the number of solutions as a function of factor ξ. Great sensitivity can be seen with change of the tolerances, which concludes that we can acquire many more solutions if we are willing to accept reduced accuracy.

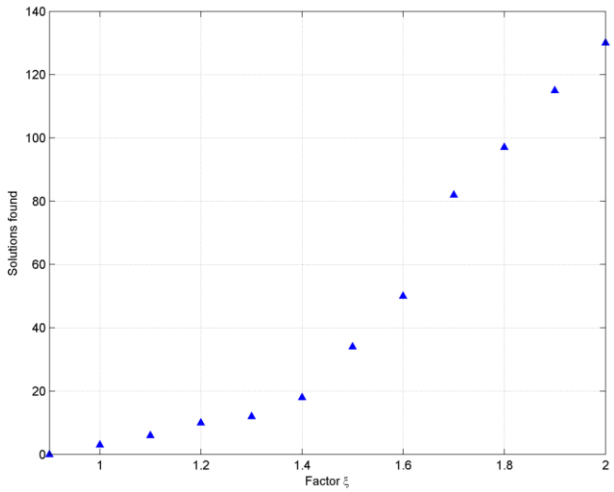


Fig. 23 - Sensitivity analysis with change of the tolerances.

7. Experimental Results

In order to validate the simulations presented, we have designed an experimental system to test the proposed grasp configuration (Fig. 24). The system is constructed of three fixture elements (fixels), each equipped with an ATI Nano25 force/torque transducer. Each fixel is located according to the configuration illustrated in Fig. 15 and is capable of applying force by fastening each fixel's bolt. These fixels are the implementation of the 3-finger grasps and the force transducer is used for acquiring contact force data at each fixel. A fourth fixel is used to apply external load to the test object and is equipped with an ATI Mini40 force/torque transducer. This fourth fixel is used to simulate a disturbance force. All fixels have hemispheric caps on their tips to simulate the point contact model. As we are dealing with planar models, we placed the test object on a sliding platform that neutralizes undesired and un-modeled frictional forces. The grasp of the four tested objects can be seen in Fig. 24 and Fig. 25.

All of the force/torque transducers were connected to an NI Data Acquisition (DAQ) board. Force data from the transducers were acquired and processed using the Matlab DAQ toolbox. The force data of the transducers are acquired in their own

coordinate frames and therefore they were all rotated to a central coordinate frame *O*. The coordinate frames of the transducers shown in Fig. 24 are the factory ones, and the measurements are transformed to the central coordinate frame.

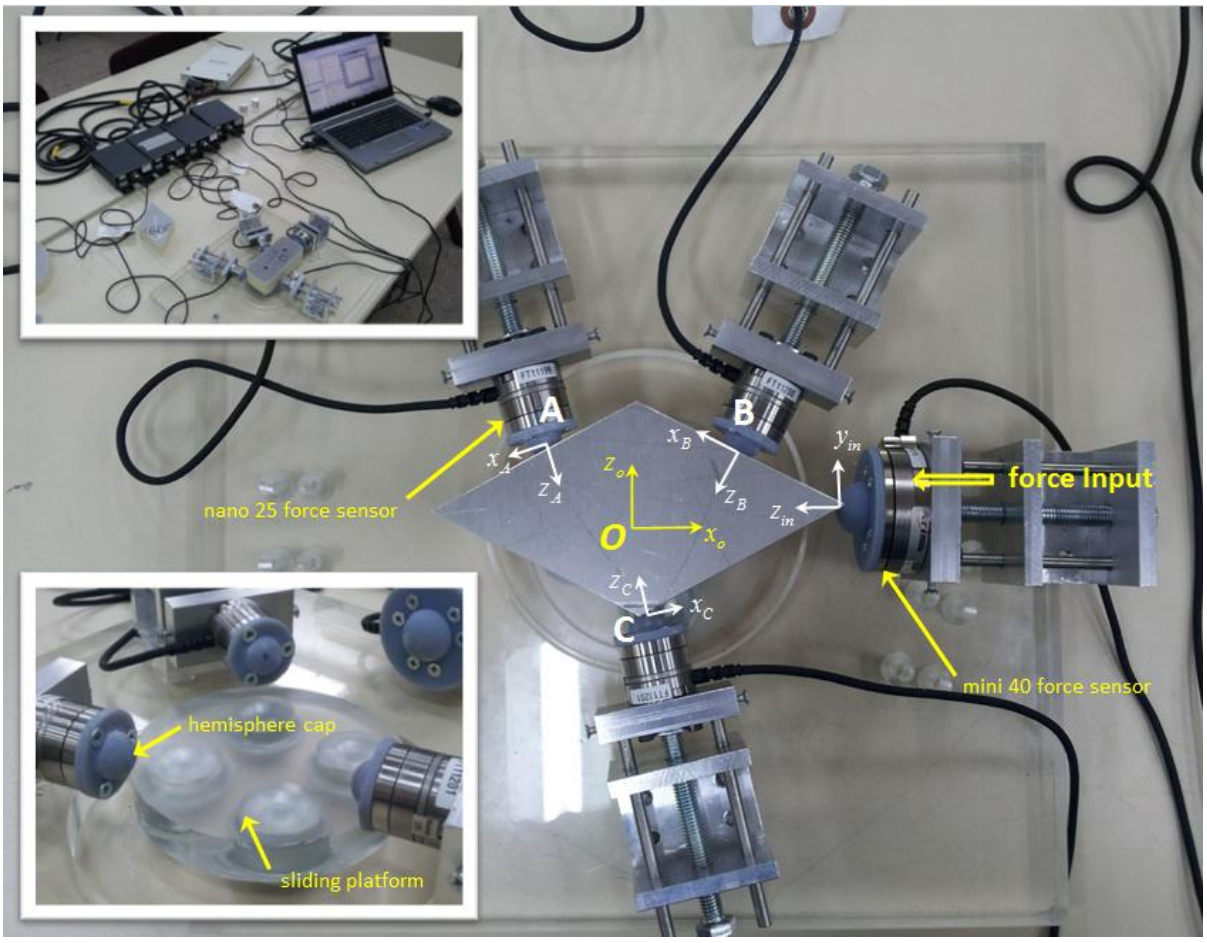


Fig. 24 - The 2D experimental setup.

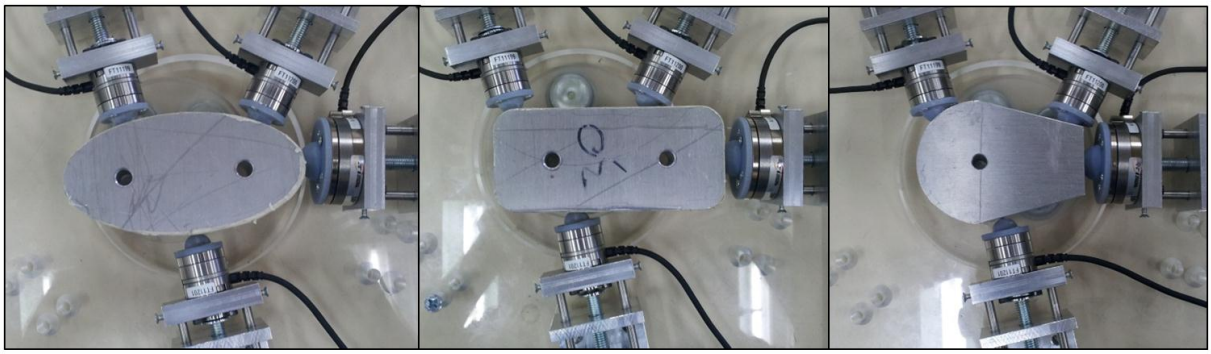


Fig. 25 - The objects loaded into the experimental system.

The force results for all tested shapes represented in the *O* coordinate frame are shown in Fig. 26 to Fig. 29. As we increase the external input force, changes in the fingertip forces can be seen in the x_o and z_o directions (y_o is irrelevant as we are dealing with planar grasp). However, equilibrium of all forces on the object is maintained and can be seen in the figures as the F^{eq} (black) curves around the zero force line. There are minor deviations of the equilibrium curve from the zero line due to minor measurement errors of the transducers. Despite the small deviations, equilibrium is maintained for the grasp of all objects and equilibrium directly

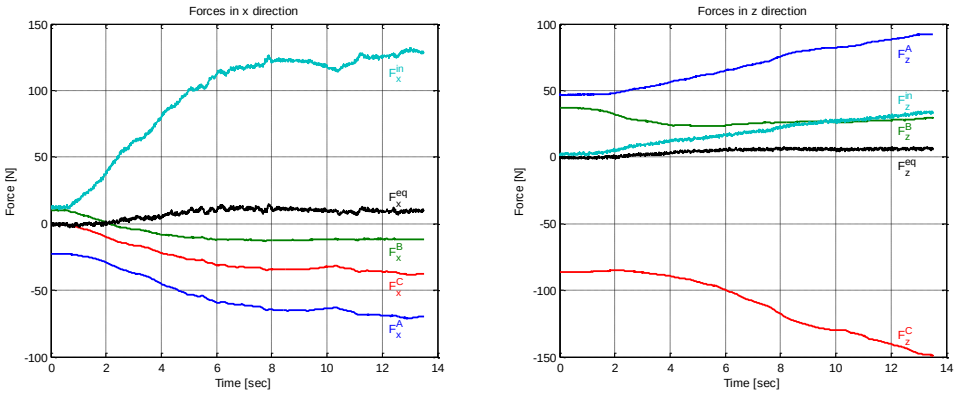


Fig. 26 - Forces in x and z directions for the ellipse shape.

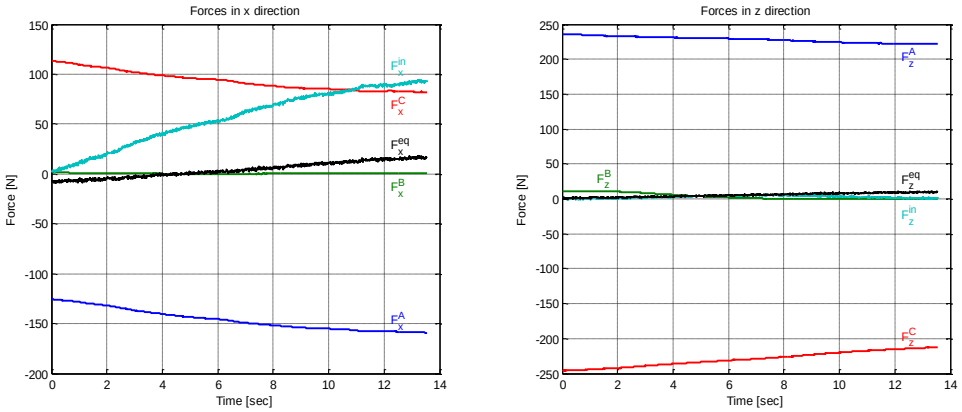


Fig. 27 - Forces in x and z directions for the dalton shape.

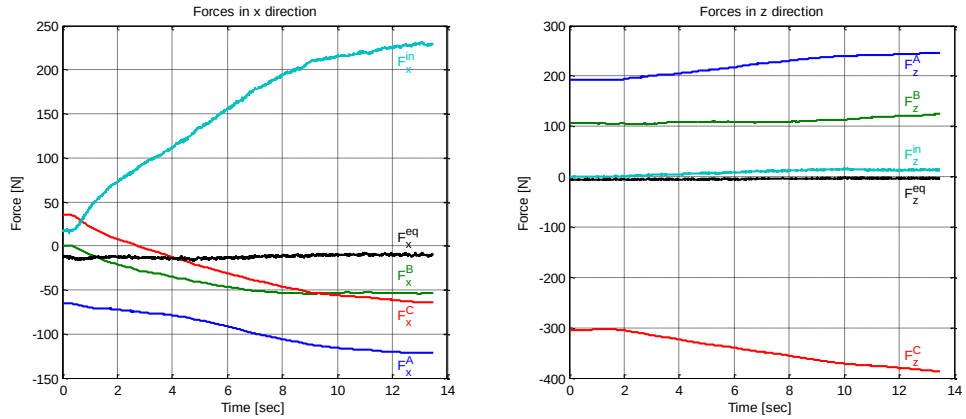


Fig. 28 - Forces in x and z directions for the rectangular shape.

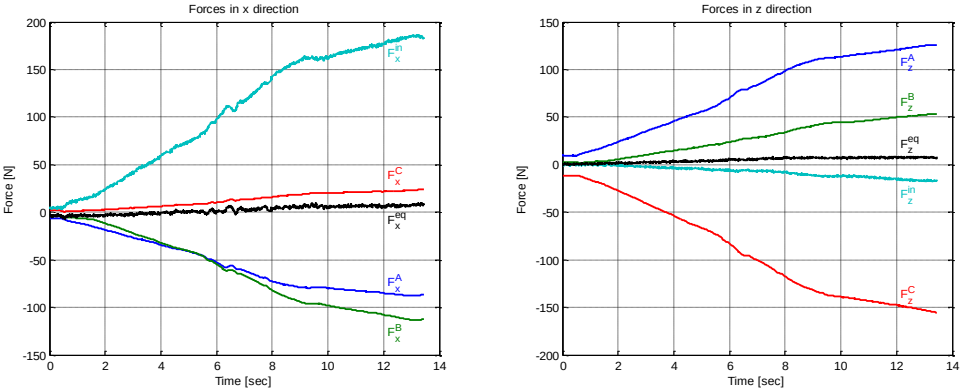


Fig. 29 - Forces in x and z directions for the ice-cream cone shape.

implies no slippage of the objects from this specific grasp configuration. Several other external loading directions and magnitudes were tested as well with similar results. Therefore we can say that this specific common grasp configuration is feasible and stable for the four tested objects.

8. Conclusion

The main idea of the OCOG algorithm presented in this paper is an efficient searching of high dimensional sets denoting all force-closure grasps of several objects. A method for generating the FCGS containing a cluster of vectors representing grasp feature vectors was presented. Each FCGS was then converted into a *kd*-tree index structure. Such a representation enables the efficient similarity join search algorithm described in the paper and the simple classification algorithm which followed.

Finally we try to find the minimum set of grasp-feature vectors that exists in all of the primitive sets. The result is a set of grasp configurations where each grasp can hold a subset of the objects. Every grasp found is in fact a configuration of an end-effector used to grasp the compatible subset of objects. Simulation results from a Matlab implementation were used to generate grasp solutions for all shapes and were followed by experimental verification. The simulations and experiments proved the feasibility of the proposed algorithm. It should be mentioned that the sizes of all analyzed shapes need to be in the same scale in order to compute the solutions. The performance analysis from the simulations shows that larger meshes take a lot more time. However, the proposed algorithm is to be executed offline and therefore the running time is not a strong constraint. Moreover, the nature of the algorithm enables distributed computation of n objects separately (computation for each object on a separate computer), which enables reduction of CPU time by approximately $1/n$.

After validating the proposed algorithm on the planar case, we have modified and implemented the algorithm for the 3D case. Moreover, simulations and experiments were conducted to validate the algorithm. The algorithm for a 3D object has the same complexity of $O(k^g)$. A paper discussing the 3D problem and algorithm is currently in preparation.

If more than one grasp (feature vector) was found, overlap of grasps of the same object is very likely, i.e., multiple grasp configurations that can grasp the same object. Future work will deal with reduction of the number of end-effectors needed to grasp a set of objects by adding degrees of freedom (DOF) to an end-effector. The DOFs added will be calculated according to the distance between the two compatible grasp feature vectors in the high-dimensional space. As we have seen, the sensitivity of the number of solutions varies according to the chosen tolerances. Further work should systematically define the tolerances according to the task's accuracy and demands. Further work will involve optimization of the end-effector design in the sense of DOF, clamping constraints, object deformations, etc. We also need to consider a form-closure-based solution and compare it with the proposed force-closure-based approach.

Appendix

In our simulations, during the similarity search, the tolerances $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are used to determine whether two feature vectors, $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, are considered to be similar. The tolerances are chosen such that the edges of the grasp triangle will not extend or shorten by more than 6%. Tolerance ε_3 corresponds to parameter d_1 (base edge length of the triangle) of the grasp feature vector. While comparing two feature vectors, $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, the tolerance will be $\varepsilon_3 = 0.06 \cdot max(d_1^j, d_1^i)$ and parameter d_1 of the two will be compared and checked to be

$$\left| d_1^i - d_1^i \right| \le \varepsilon_3 \tag{18}$$

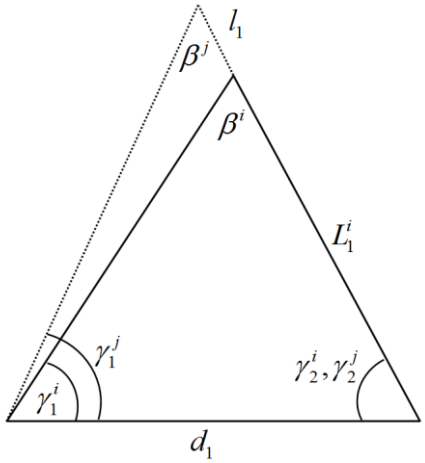


Fig. 30 - Enlargement of the triangles angle.

The tolerances $\varepsilon_1, \varepsilon_2$ correspond to the triangle's angles γ_1^j, γ_2^j of feature vector $\boldsymbol{e}_j$ and γ_1^i, γ_2^i of feature vector $\boldsymbol{e}_i$ (Fig. 30). The tolerances are chosen as follows. By using the law of sines, we extract the connections

$$\frac{d_1}{\sin\beta^i} = \frac{L_k^i}{\sin\gamma_k^i},\ \beta^i = 180^\circ - \gamma_1^i - \gamma_2^i \tag{19}$$

and

$$\frac{d_1}{\sin\beta^j} = \frac{L_k^i + l_k}{\sin\gamma_k^j},\ \beta^j = 180^\circ - \gamma_1^j - \gamma_2^j \tag{20}$$

where $l_k = L_k^j - L_k^i$ and $k = 1, 2$. Therefore, with the two equations, the difference between the two angles will be

$$\gamma_k^j - \gamma_k^i = \sin^{-1}\left(\frac{\left(L_k^i + l_k\right)\sin\beta^j}{d_1}\right) - \sin^{-1}\left(\frac{L_k^i \sin\beta^i}{d_1}\right) \tag{21}$$

If we demand that l_k will be not more than λL_k^i, where λ is a number bounded to be $0 < \lambda < 1$, we can define the angle tolerance

$$\varepsilon_k = \sin^{-1}\left(\frac{\left(L_k^i + \lambda L_k^i\right)\sin\beta^j}{d_1}\right) - \sin^{-1}\left(\frac{L_1^i \sin\beta^i}{d_1}\right) \tag{22}$$

and demand that

$$\gamma_k^j - \gamma_k^i \le \varepsilon_k,\ k = 1, 2 \quad . \tag{23}$$

Acknowledgment

This work was supported by General Motors Ltd. under PO No. IMEUS-NA-UNIV-11-025, and was partially supported by the Paul Ivanier Center for Robotics Research and Production Management, and the Pearlstone Center for Aeronautical Engineering Studies.

References

[1] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. CRC Press, Mar. 1994. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/-0849379814

[2] N. Niparnan and A. Sudsang, "Computing all force-closure grasps of 2d objects from contact point set." in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 1599–1604.

[3] A. Shapiro, E. Rimon, and S. Shoval, "On the passive force closure set of planar grasps and fixtures," *The International Journal of Robotics Research*, vol. 29, no. 11, pp. 1435–1454, 2010. [Online]. Available: http://-ijr.sagepub.com/content/29/11/1435.abstract

[4] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1992, pp. 2290–2295. [Online]. Available: http://dx.doi.org/10.1109/-ROBOT.1992.219918

[5] M. Roa and R. Suarez, "Geometrical approach for grasp synthesis on discretized 3d objects," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2007, pp. 3283–3288. [Online]. Available: http://dx.doi.org/10.1109/IROS.2007.4399440

[6] M. Wang, "An optimum design for 3-d fixture synthesis in a point set domain," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 839 –846, Dec 2000.

[7] J. Ponce and B. Faverjon, "On computing three-finger force-closure grasps of polygonal objects," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 6, pp. 868 –881, dec 1995.

[8] Y. H. Liu, "Computing n-Finger Form-Closure grasps on polygonal objects," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 149–158, Feb. 2000. [Online]. Available: http://dx.doi.org/10.1177/-02783640022066798

[9] Z. Li and S. Sastry, "Task oriented optimal grasping by multifingered robot hands," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, Mar. 1987, pp. 389–394. [Online]. Available: http://-ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087852

[10] J. D. Schulman, K. Goldberg, and P. Abbeel, "Grasping and fixturing as submodular coverage problems," in *Proceedings of the 15th International Symposium on Robotics Research (ISRR)*, 2011.

[11] Q. Lin, J. W. Burdick, and E. Rimon, "A stiffness-based quality measure for compliant grasps and fixtures," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 675–688, 2000. [Online]. Available: http://dx.doi.org/-10.1109/70.897779

[12] E. Chinellato, R. Fisher, A. Morales, and A. del Pobil, "Ranking planar grasp configurations for a three-finger hand," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, Sept. 2003, pp. 1133 – 1138 vol.1.

[13] B.-H. Kim, S.-R. Oh, B.-J. Yi, and I. H. Suh, "Optimal grasping based on non-dimensionalized performance indices," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, 2001, pp. 949 –956 vol.2.

[14] N. Niparnan, T. Phoka, and A. Sudsang, "Heuristic approach for multiple queries of 3d n-finger frictional force closure grasp." in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1817–1822.

[15] R. Prado and R. Suarez, "Heuristic grasp planning with three frictional contacts on two or three faces of a polyhedron," in *Proceedings of the 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing*, July 2005, pp. 112 –118.

[16] A. Rodriguez and M. T. Mason, "Grasp invariance," *Int. Journal of Robotic Research*, vol. 31, no. 2, pp. 236–248, 2012.

[17] M. Novotni and R. Klein, "A geometric approach to 3d object comparison," *International Conference on Shape Modeling and Applications*, pp. 154–166, 2001.

[18] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, Oct. 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=571648

[19] R. Ohbuchi, T. Otagiri, M. Ibato, and T. Takei, "Shape-similarity search of three-dimensional models using parameterized statistics," *Proceedings 10th Pacific Conference on Computer Graphics and Applications*, pp. 265–274, 2002. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1167870

[20] Y. Li and N. S. Pollard, "A shape matching algorithm for synthesizing humanlike enveloping grasps," in *Proceedings of the 5th IEEE-RAS International Conference on Humanoid Robots*, 2005, pp. 442–449. [Online]. Available: http://dx.doi.org/10.1109/ICHR.2005.1573607

[21] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, Sep. 1977. [Online]. Available: http://dx.doi.org/10.1145/355744.355745

[22] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*, 1st ed. Prentice Hall, Oct. 2001. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/-0130319953

[23] K. Shim, R. Srikant, and R. Agrawal, "High-dimensional similarity joins," in *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, W. A. Gray and P.-Å. Larson, Eds. IEEE Computer Society, 1997, pp. 301–311.

[24] A. W. Moore, "Efficient memory-based learning for robot control," Ph.D. dissertation, Cambridge, UK, 1990. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.2654

[25] M. Hong and S. Payandeh, "Design and planning of a novel modular end-effector for agile assembly," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, Apr. 1997, pp. 1529 –1535 vol.2.

[26] J. R. Amend, E. Brown, N. Rodenberg, H. M. Jaeger, and H. Lipson, "A positive pressure universal gripper based on the jamming of granular material," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 341–350, 2012.

[27] A. M. Dollar, "Joint coupling design of underactuated grippers," in *Proceedings the 30th Annual ASME Mechanisms and Robotics Conference*, 2006, pp. 10–13.

[28] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/-abs_all.jsp?arnumber=1371616

[29] B. Mishra, J. T. Schwartz, and M. Sharir, "On the existence and synthesis of multifinger positive grips," *Algorithmica*, vol. 2, pp. 541–558, 1987. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/-summary?doi=10.1.1.38.7726

[30] M. Roa and R. Suarez, "Regrasp planning in the grasp space using independent regions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, oct. 2009, pp. 1823 –1829.

[31] C. Borst, M. Fischer, and G. Hirzinger, "A fast and robust grasp planner for arbitrary 3d objects," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, 1999, pp. 1890 –1896 vol.3.

[32] L. Deneen and G. Shute, "Polygonizations of point sets in the plane," *Discrete & Computational Geometry*, vol. 3, pp. 77–87, 1988, 10.1007/BF02187898. [Online]. Available: http://dx.doi.org/10.1007/BF02187898

[33] M. A. Roa, R. Suárez, and J. Rosell, "Grasp space generation using sampling and computation of independent regions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2258–2263. [Online]. Available: http://dx.doi.org/10.1109/IROS.2008.4651102

[34] C. Bradford-Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996. [Online]. Available: http://dx.doi.org/10.1145/235815.235821