Efficient Models for Grasp Planning With A Multi-fingered Hand

Jean-Philippe Saut, Daniel Sidobre
LAAS-CNRS, Universite de Toulouse, CNRS, UPS Toulouse, France
email daniel.sidobre, kevin.desormeaux@laas.fr

March 2012

Abstract

This paper presents a simple grasp planning method for a multi-fingered hand. Its purpose is to compute a context-independent and dense set or list of grasps, instead of just a small set of grasps regarded as optimal with respect to a given criterion. By context-independent, we mean that only the robot hand and the object to grasp are considered. The environment and the position of the robot base with respect to the object are considered in a further stage. Such a dense set can be computed offline and then used to let the robot quickly choose a grasp adapted to a specific situation. This can be useful for manipulation planning of pick-and-place tasks. Another application is human-robot interaction when the human and robot have to hand over objects to each other. If human and robot have to work together with a predefined set of objects, grasp lists can be employed to allow a fast interaction.

The proposed method uses a uniform sampling of the possible hand approaches. As this leads to many finger inverse kinematics tests, hierarchical data structures are employed to reduce the computation times. The data structures allow a fast determination of the points where the fingers can realize a contact with the object surface. The grasps are ranked according to a grasp quality criterion so that the robot will first parse the list from best to worse quality grasps, until it finds a grasp that is valid for a particular situation.

Keyword: Robotic Grasping, Multi-fingered Hand, Inverse Kinematics.

1 Introduction

Mobile manipulators are now common in robotics research laboratories, but planning the manipulation of objects with complex shapes is still a challenging task. Among the sub-tasks involved in manipulation planning, grasp planning is of first importance as grasping is the starting point of any manipulation task.

Grasp planning basically consists in finding where to place the fingers on the object the robot must grasp. If we consider a complete robotic platform, not only the grasp configuration is needed but also the configuration of the robot base and arm. Several aspects must be taken into account in order to find a configuration for the whole robot that is suitable for picking up the object:

- The configuration must be accessible to the robot, *i.e.* it must be compatible with its inverse kinematics (base, arm and finger kinematics);
- The grasp associated to the configuration must be stable according to a chosen relevant stability criterion;
- The grasp configuration must not lead to robot self-collision or collision against the environment.

The paper proposes a method to find such configurations and it is presented as follows.

Section 2 gives an overview of the existing works related to grasp planning. Section 3 presents the proposed method. Section 3.1 explains how relative hand/object poses (referred later as *grasp frame*) are computed. Section 3.2 details how a grasp configuration is computed from a grasp frame. Such a computation is based upon an approximation of the intersection between the object surface and the finger workspace. Section 3.5 explains how to compute the intersection from the models of object surface and finger workspace detailed in sections 3.3 and 3.4. The obtained grasps are then evaluated and sorted according to the quality score described in section 3.6.

2 Related Work

Most of the early grasp planning methods did not take into account finger nor arm kinematics and are often referred as contact-level techniques [1, 2, 3]. The contacts are regarded as freely-moving points with no link to any mechanical chain. Many grasp stability criteria have been introduced for this model of point/surface contact, the most common being certainly the force closure criterion [2, 4]. Force closure criterion is verified if a grasp can resist arbitrary force/torque perturbation exerted on the grasped object and is tested for a specific set of contacts (positions and normals). To integrate the notion of robustness of the grasp stability with respect to the contact positions, the concept of independent regions of contact has been introduced [1]. These regions are such that a grasp always verifies force closure as long as the contacts stay within the region. The computation of these regions has been solved for different object surface modelization (2D discrete surface [5], 2D polygonal surface [6], 3D polyhedral surface [7, 8, 9]).

All these contact-level techniques were not very well-suited for real applications. Therefore, many new methods appeared that integrate considerations on finger and/or arm kinematics. Miller *et al.* [10] proposed to decompose the object into a set of primitives (spheres, cylinders, cones and boxes). A pregrasp configuration of the hand is associated to each primitive. A set of parameters is sampled in order to test the different directions of approach of the hand. Then, for each pose, the fingers are closed on the object until collision. The quality of the obtained grasp is then computed according to the measure described in [2]. The idea of object decomposition was widely used and is still the base of many grasp planners. It provides a heuristic to reduce the possible relative palm/object poses to test. In [11], the authors decompose the object model into a superquadric *decomposition tree* employing a nonlinear fitting technique. Grasps are then planned for each superquadric using a heuristic approach close to the one in [10]. The grasps are then simulated on the original object model using the GraspIt! dynamics simulator [12], to sort them by quality. Huebner *et al.* [13] proposed a technique to build a hierarchy of minimum volume bounding boxes from 3D data points of the object envelop. This method offers an interesting robustness with respect to the quality of the object's 3D model, acquired from sensors (here laser scan). In [14], the object is decomposed into a set of boxes called *OCP* (Object Convex Polygon). Each box of the OCP is compared to a *GRC* (Grasping Rectangular Convex), which gives an estimation of the maximum size of the object that the hand can grasp. Different GRCs are defined corresponding to different grasping styles. Xue *et al.* [15] presented a method to optimize the quality of the grasp that takes into account the kinematics of the fingers during the optimization phase. They use a swept volume precomputation associated with a continuous collision detection technique to compute, for a given hand/object relative pose, all the possible contacts of each finger on the object surface. After obtaining an initial grasp provided by the GraspIt! software [12], they locally optimize the quality of the grasp in the finger configuration space.

Some works gave more focus on arm and/or robot base inverse kinematics issues. Berenson *et al.* [16] were interested in finding grasp configurations in cluttered environments, for a given robot base position in the object area. From different object approaches, the authors precompute a set of grasps, all verifying the force closure property. Instead of trying to solve the arm inverse kinematics and checking for collisions for each grasp of the set in an arbitrary order, the authors propose to compute a *grasp scoring* function for each grasp. The function is used to evaluate the grasps that are more likely to succeed the inverse kinematics and collision tests. It is based upon a force closure score, a relative object-robot position score and an environment clearance score. The authors of [17] focused on path planning for the robot base (or body) and arm and presented a planning algorithm called *BiSpace*. Like in [16], they first compute a set of grasp configurations for the hand alone. Once one or more collision free configurations for the hand are found, they become the start nodes of several RRTs (Rapidly Random-exploring Tree [18]) that will explore the hand workspace while another RRT is grown from the robot base start configuration, that explores the robot configuration space.

Some recent works were inspired by results in neuroscience [19, 20] which have shown that humans mainly realize grasping movement that are restricted in a configuration space of highly reduced dimensionality. From a large data set of human pregrasp configurations, Santello *et al.* [19] performed a principal component analysis revealing that the first two principal components account for more than 80% of the variance. Ciocarlie *et al.* [21] called the components *eigengrasps* and use them as a base to represent the reduced configuration space of the hand. They also add the six DOFs of the wrist pose. Then, they use a simulated-annealing-based optimization method, in eigengrasp space, to find the best grasp according to an energy function. The energy function takes into account two parameters. First, the distance between specified points on the hand and the object surface. Secondly, a quality metric based on the one in [2].

Other neuroscience results suggest a very close connection between visual perception and the choice of the hand approach to grasp the object. According to Goodale and Milner [22], the human visual system is organized into two separate parallel pathways: The ventral and dorsal streams, where the ventral stream plays a major role in perceptual identification of objects, while the dorsal stream plays a major role in visually-guided action. Although some recent studies [23] revealed that there is no evidence of the existence of two distinct pathways on healthy subjects, contrary to what the previous studies have suggested, nice results were obtained in robotics by combining vision and grasping action. In these vision-based grasping techniques [24, 25], the grasping points are directly extracted from image features, instead of using images to build or localize a 3D model and work with it. Once an image is divided into square patches, descriptors (*e.g.* the *shape context* descriptor [25] and the ones based on edge, texture, and color filters [24]) are computed on each patch, and a classifier trained by a supervised-learning method is used to identify patches that contain grasping points. The 3D poses of the grasping points are then computed from images obtained from different camera positions.

3 Grasp List Computation

Grasp planning of a complex object has been so far too computationally expensive to consider it can be performed in real-time. Therefore, in a real application, it is preferable to use precomputations as much as possible. In the proposed framework, a grasp list is computed off-line for the considered object in order to capture the best possible the variety of the possible grasps. This list will then be used to select, in an online application, the grasps that are currently reachable and, from them, the best one according to a scoring function described further.

In this paper, we consider only precision grasps *i.e.* contacts are made with fingertips only. This allows to reason with point contact only, that is the most common case in literature. It also gives more grasping possibilities as smaller parts can be grasped, at the cost of a weaker stability compared to power grasps and a bigger contact pressure, due to smaller contact regions. In the following, we illustrate the method with the Schunk Anthropomorphic Hand (SAH hand), depicted in Fig. 3. It has four fingers. Each finger has four joints. Only the three first joints are actuated, the last one being coupled with the third one. The thumb has an additional actuated joint to place it in opposition to the other fingers. The method however applies to other hand kinematic structures, after some minor adaptations.

For some applications, precomputing a reduced set of good grasps is of no interest because it is mainly the situation (positions of the environment obstacles) that will constrain the way the object can be grasped by the robot.

The grasp list has thus to be as general as possible. Consequently, we choose to uniformly sample the possible hand approaches. Each hand approach is characterized by a frame referred as *grasp frame*. A grasp frame is the transform matrix of the relative object/palm pose. The grasp frames are centered on the intersection of the finger workspaces so that they are roughly centered where the contacts may occur.

Fig. 1 shows the overall principle of the proposed method, that will be detailed in next sections. The inputs to the algorithm are the models of object

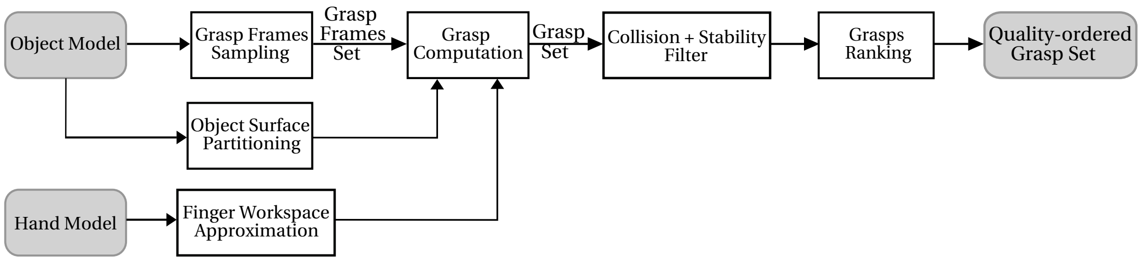
Object Model
Grasp Frames Sampling
Grasp Frames Set
Grasp Computation
Grasp Set
Collision + Stability Filter
Grasps Ranking
Quality-ordered Grasp Set
Object Surface Partitioning
Hand Model
Finger Workspace Approximation

Figure 1: Overview of the proposed method to compute a set of grasps from object's and hand's models.

and hand. The first step is to build a set of grasp frame (*i.e.* palm configurations) samples (section 3.1). Then for each grasp frame, one has to find the contacts that will realize a grasp. Two approaches are possible. The first one is to use finger forward kinematics *i.e.* to close the fingers until contact. This usually requires many collision tests and can not find contact in objects holes or loops (*e.g.* a mug handle). The second approach is to use finger inverse kinematics *i.e.* to compute the points on object's surface that are reachable by the fingers. Exact solutions are computationally very expensive if not impossible. Therefore we introduce models to quickly find an approximation of the accessible part of the objects surface (sections 3.3 and 3.4). The next steps of the algorithm consist in filtering the grasps to remove collision and instabilities (sections 3.6.1 and 3.6.2) before ranking them to obtain a set sorted by grasp quality (section 3.6.3).

3.1 Grasp frame sampling

The possible grasp frames are sampled by the mean of a grid. We set as an input the number of positions and the number of orientations. Each couple position-orientation defines a frame. The positions are uniformly sampled in the object axis-aligned bounding box with a step computed to fit the desired number of position samples. The orientations are computed with an incremental grid like the one in [26]. Such an incremental grid is a sequence of rotations guarantying that $\forall n \in \mathbb{N}$, the first n elements of the sequence will provide some optimality on the dispersion of these elements.

Next section explains how grasps are computed from grasp frames.

3.2 Grasp computation from grasp frame

To compute a grasp from a grasp frame, the most expensive computation – except for collision test– is the finger inverse kinematics. Therefore we make use of two data structures that let us find quickly the intersection between the finger workspaces and the object surface. In particular, it is crucial to be able to know the fastest possible if a finger can not reach the object surface for a given grasp frame because we do not want to test all the object surface for inverse kinematics if the object is not reachable at all.

3.3 Object surface model

In our planner, all the bodies are modeled as polyhedra, stored as triangle meshes. We propose to approximate the object surface with a contact point set, keeping trace of where it is on the object mesh to be able to get some local information (surface normal and curvature) later. The set is obtained by a uniform sampling of the object surface. The sampling step magnitude is chosen from the fingertip radius. A space-partitioning tree is built upon the point set in order to have a hierarchical space partition of the points (Fig. 2). It is similar to a kd-tree. Starting from the original set of points, we compute the minimal axis-aligned box containing all the points. Such a box is usually referred as Axis-Aligned Bounding Box or AABB. This first AABB is the tree root. The root AABB is then split in two along its larger dimension. This leads to two new nodes, children of the root, containing each a subset of the original point set. The splitting process is then recursively applied to each new node of the tree. The process ends when a AABB node contains only one point. We then need to find the intersection of each finger workspace with the object surface tree. So we introduce another data structure to approximate the finger workspace and compute the intersection quickly.

3.4 Finger workspace model

As spheres are invariant in rotation, it is interesting to build an approximation of the finger workspace as a sphere set. We build incrementally a set S, composed of spheres fitting inside the workspace, using Algorithm 1. The inputs to the algorithm are grid approximations W and E of the finger workspace interior volume and of the finger workspace envelope, respectively. These grids are depicted in left image of Fig. 3. The interior volume grid is obtained by sampling the three joint angles over their respective ranges, with values strictly inside their bounds. The envelope grid is obtained by blocking one of the joint angle to its

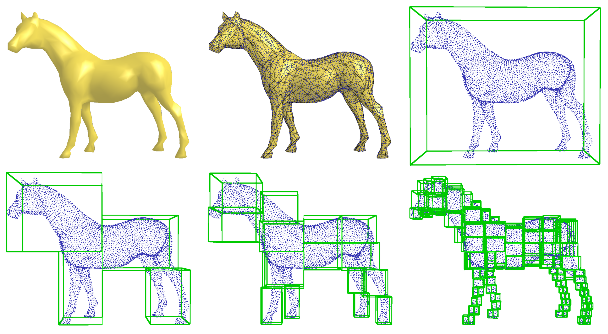

Figure 2: The object mesh is uniformly sampled with a point set (top images). The point set is then partitioned using a kind of kd-tree (bottom images).

limit values (lower then upper) and sampling the two others. The idea of the algorithm is just to find the interior point that maximizes the distance with the envelope and the previous spheres in S.

For each inner point p (*i.e.* $p \in W$), the smallest distance from p to the boundary points E is computed, noted d_{min}. The inner point having the biggest d_{min} is the center of the first sphere S_1, of radius d_{min}. For all the inner points that are not inside S_1, a new d_{min} is computed, that is the minimum of the old d_{min} and the minimal distance to S_1. The point that has the biggest d_{min} is the center of the second sphere S_2, of radius d_{min}. This process is repeated until the maximal desired sphere number k_{max} is reached or the last computed sphere has a radius less than a specified threshold r_{min}. Depending on k_{max} and r_{min}, the sphere set S will cover more or less the real workspace of the finger. Choosing a small k_{max} and big r_{min} will let volumes of the finger workspace not covered by spheres of S. As these volumes are the closest to the joint limits, it is worth discarding such volumes to reduce the sphere hierarchy size (Fig. 3).

We keep the ordering of the construction so that the sphere hierarchy starts from the biggest ones, corresponding to workspace parts that are the farthest to the finger joint bounds. These bounds were first slightly reduced in order to eliminate configurations where the fingers are almost completely stretched.

Once we have both the contacts tree and the workspace sphere hierarchy, it is very fast and easy to determine the intersection of the two sets and so the contact points.

3.5 Intersection between object surface and finger workspace

All the operations that have to be performed are sphere-box intersection tests. The intersection is tested from the biggest to the smallest sphere, guarantying that the "best" parts of the workspace will be tested first, *i.e.* the one farthest to the workspace singularities due to the joint bounds. Starting from the tree root, we test if there is a non null intersection between an AABB-node and

Algorithm 1: Finger workspace approximation

```
input  : $W$= a set of points strictly inside the finger workspace ;
         $E$= a set of points on the envelope of the finger workspace ;
         $k_{max}$= the desired maximal size of the sphere decomposition
         ($i.e.$ the number of spheres) ;
         $r_{min}$= the desired minimal sphere radius ;
output: S= a set of spheres $S_k$ ordered from the biggest to the smallest
         ;

$S = \emptyset$ ; $k = 1$ ;
while $k < k_{max}$ do
    foreach $p \in W$ do
        $d(p) = \min\limits_{p_i \in (E \cup S)} (\|p - p_i\|)$ ;
    $p_{best} = \{p \in W : d(p) = \max\limits_{p_i \in W}(d(p_i))\}$;
    $S_k$ = sphere(center = $p_{best}$, radius = $d(p_{best})$) ;
    $S = S \cup S_k$ ;
    $W = W - \{p \in W : p \subset S_k\}$ ;
    $k = k + 1$ ;
    if $d(p_{best}) < r_{min}$ then
        break ;
return S ;
```

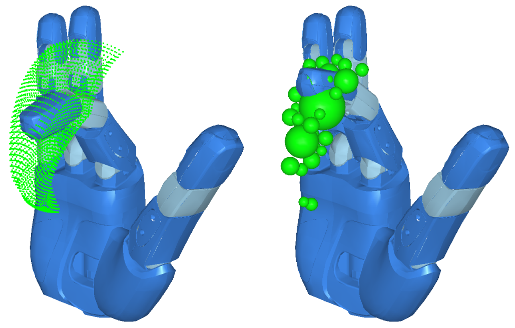

Figure 3: The finger workspace, discretized with a grid (forefinger workspace, left image). The grid is converted to a volumetric approximation as a set of spheres (right image).

the sphere. If not, we stop exploring this branch, otherwise we test the sphere against the two node children, until we arrive to a leaf node *i.e.* a single point. We then just have to test if the point is included in the sphere volume. Fig. 4 shows the different steps to compute candidate contact points from a given grasp frame, for the SAH hand. At this stage, we just know that the points will pass the finger inverse kinematics test. No collision tests have been performed yet.

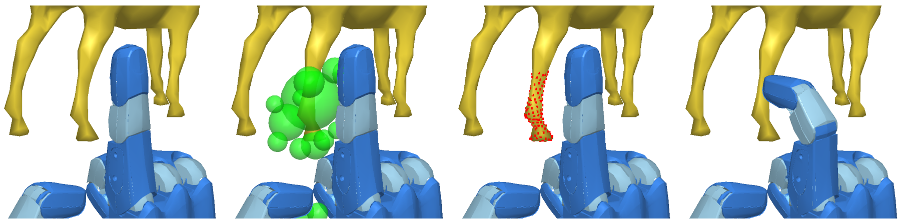

Figure 4: From the intersection between the finger workspace approximation (second image from the left), a set of reachable points is obtained (third image), before one of them is selected as a contact point (fourth image).

Algorithm 2 shows how the finger configurations are computed. For a given grasp frame, the grasp is computed finger by finger, that means that, if we have the contacts and configurations of the fingers 1 to $i-1$, we search for a contact point for finger i and test collision only with the fingers 1 to i as the other finger configurations are not yet known. We start from the thumb as no stable grasp can be obtained without it. If a finger can not establish a contact, it is left in a *rest* (stretched) configuration. If we have three contacts or more, we can proceed to the stability test. Note that, at this stage, we have a collision-free grasp *i.e.* no collision between the hand and the object but we do not yet consider collision with the environment or with the robot arms or body. Fig. 5 shows the candidate contacts for different objects and grasp frames.

Algorithm 2: Intersection between object surface and finger workspace.

```
foreach $finger_i \in [\![1; nbFingers]\!]$ do
    foreach $S_j \in [\![1; nbSpheres]\!]$ do
        point_set = intersect($S_j$, object_tree) ;
        foreach $p \in point\_set$ do
            set_finger_config_from_IK(p) ;
            collision_test(finger$_i$, object) ;
            collision_test(finger$_i$, palm) ;
            collision_test(finger$_i$, finger$_{1,\dots,i-1}$) ;
            if no collision then
                → next finger ;
```

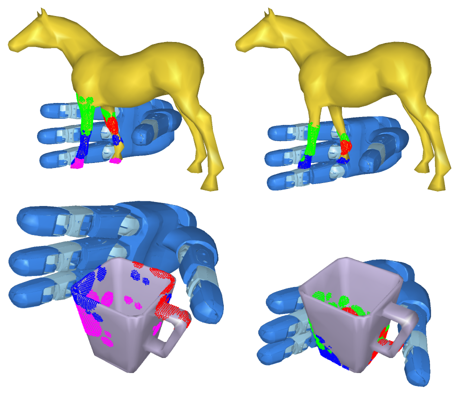

Figure 5: The potential finger contacts, drawn in red, green, blue and magenta for the thumb, forefinger, middle finger and ring finger respectively. On the right images, no contact can be found for the ring finger because of its limited workspace. Note that no collision tests are yet performed at this stage.

3.6 Grasp Filtering And Ranking

At this point, we have a list of hand configurations, such that some of the fingers contact the object with their tip. Next subsections describe a few more steps that are required to filter contacts (3.6.1), ensure grasp stability (3.6.2) and finally rank the grasps according to some quality score (3.6.3).

3.6.1 Contacts Filter

The contacts obtained from the previous step are points on object's surface that only verify finger's inverse kinematics while avoiding collisions –between object and finger phalanges and between the fingers themselves. These contacts may not be well-suited for realizing a grasp, so the next step is to remove such contacts from the grasps, *i.e.* the corresponding finger configurations will be kept but the contact will not be taken into account.

The goal of the first filter is to remove contacts such that the finger will not be able to exert a force in the opposite direction of the surface normal at the contact. For each contact, the angle between the inverted contact normal and the major axis of the force ellipsoid of the corresponding finger is computed. The force ellipsoid is obtained from the jacobian matrix of the finger. Its longer axis is simply the (normalized) eigen vector of the jacobian that has the biggest eigen value. This longer axis correspond to the direction that offers the best transmission ratio from the joint torques to the force exerted at the fingertip (more details can be found in [27]). If the angle is more than a threshold ($60°$) the contact is rejected. We have chosen to not take into account the abduction joint in the computation of the jacobian as it is not related to the finger closing motion. Fig. 6 shows the direction of the axis for the fingers with different configurations.

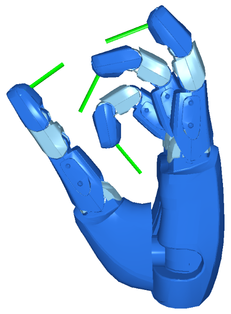

Figure 6: The major axes of each finger force ellipsoid are drawn as green lines. They are used to remove candidate contacts according to their normal.

Fig. 7 shows, for different contacts, the inverted contact normal in green and the direction of best force transmission in red. One can see as the smallest is the angle between the two lines, the best will be the contact in term of potential grasping force. Also the direction of best force transmission corresponds to the direction of the force exerted by the finger if it is closed by giving the same velocity to the finger joints. This is an easy way to control the hand closing that do not require specific computation or sensor unlike a more sophisticated technique relying on grasping force computation and control (*e.g.* [28]).

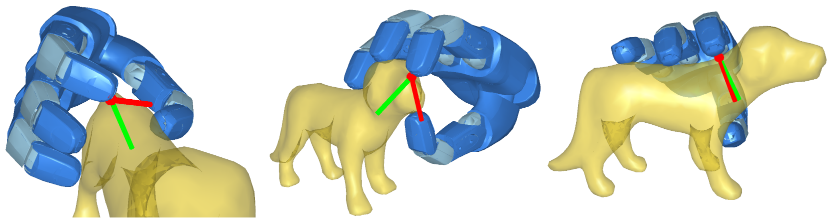

Figure 7: The major axis of the forefinger force ellipsoid for different contacts (red lines) and the normals of the contacts (green lines). It is preferable to have contact with a limited angle between these two lines (right image) rather that large one (left images) to reduce of contact slipping if the finger closing strategy does not use grasping force optimization and control.

The stability test only considers contact positions and normals to check if a grasp is stable or not. It does not take into account the local geometry around the contacts. For practical applications, the robustness of the grasp selection with respect to object's localization error is crucial. Indeed, as one should not expect a localization system with a perfect accuracy, the robot will not be able to place its fingers exactly where they are supposed to be to achieve the desired grasp. As the normal direction at a contact point strongly modifies the grasp stability, it is better to have contacts on a region where the local variation of the normal direction is small. This local variation can be encoded via surface curvature as proposed in subsection 3.6.3. However, at some locations, the discrete approximation of the surface curvature is too rough. Such locations correspond to points where the object's surface is not differentiable, such as sharp edges. Therefore, a second filter is used that considers contacts that are too close to sharp edges as non relevant and to be removed from the grasp. This requires to define two parameters: The angle used to classify edges as sharp and the smallest acceptable distance between a contact point and a sharp edge. We chose $80°$ for the angle and a distance equal to the width of the fingertips.

3.6.2 Stability Filter

The stability test is based on a *point contact with friction* model, that explains why at least three contacts are required. The inputs to the test are the contact positions and normals. The test is based on a force-closure test, solved as a linear programming problem [29]. All the grasps that do not verify force-closure are discarded.

3.6.3 Grasps Ranking

Several aspects can be taken into account to compute a grasp quality measure [27]. Moreover many scores can be considered if the grasp quality has to be measured with respect to a particular task (*e.g.* grasping the object to give it to a human, to place it on a support, *etc.*). A trade-off is often chosen with a score that is a weighted sum of several measures. This leads to the difficult problem of choosing weights that will be suitable in various situations. Therefore we chose a strategy of bad candidates removal in the previous step to only use a comparison function with one criterion at the end, as explained in the following.

As written in section 3.6.1, robustness with respect to contact locations can be improved if the contacts lie on regions where the local variation of the normal direction is small. Therefore, a contact where object's surface curvature is low is preferable (Fig. 8). A curvature score for each contact is thus introduced. To compute the curvature score, the object's surface mean curvature is first computed at each vertex of the triangle mesh. The curvature values are then normalized over all the computed values. At each contact point i, the curvature curv_i is obtained by interpolating the curvature values of the vertices of the triangle the point belongs to from its barycentric coordinates. The curvature score is computed as $c_i = 1 - \mathrm{curv}_i$.

We use a comparison function between two grasps instead of introducing a quantitative score, to be able to compare grasps with different number of contacts. If two grasps g_1 and g_2 have n_1 and n_2 contacts, respectively, with $n_1 \leq n_2$, then we compare the smallest curvature score of the contacts of g_1, noted sc_1, to the $(n_2 - n_1 + 1)$-th smallest curvature score of the contacts of g_2, noted sc_2. If sc_1 is bigger than sc_2, then g_1 is considered as having a better quality than g_2.

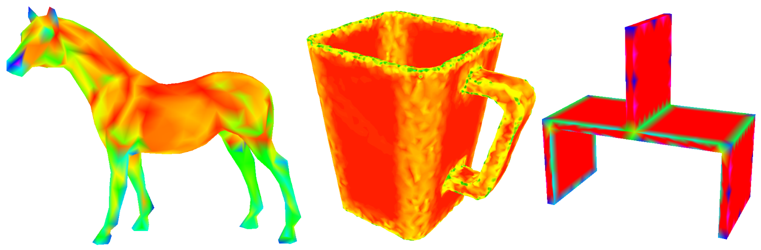

Figure 8: The mean curvature of the object surface is used as a quality criterion on the contact position. Surface color varies from red (low curvature, good location) to blue (high curvature, bad location), through green.

3.6.4 Examples

Fig. 9 shows some of the best quality grasps computed with our algorithm for objects with complex shape. For the horse statuette on the left, from about 5000 grasp frames sampled, we found 163 valid grasps in less than 5 minutes on a standard PC[1]. Similar results can be observed for the mug on the right, whose 3D model was obtained from a real object.

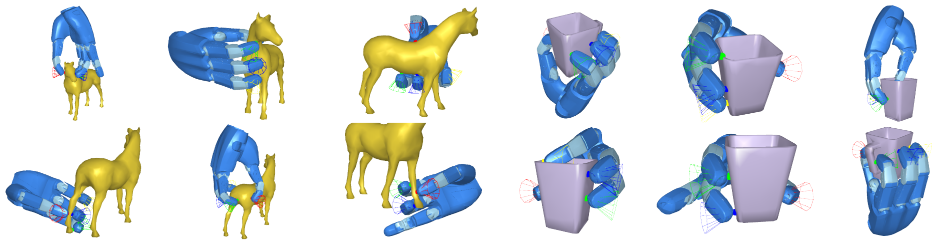

Figure 9: Some of the various grasps that were computed for objects with complex shape. Friction cones are also represented.

3.7 Selecting the grasp from the list

Once the grasp list of the specified object is computed, the online phase begins, whose first step is the selection of a grasp from the list. If there were no additional constraints, the best choice would be the grasp with the best ranking. However, the robot must consider the limited reachability of its arm. Ensuring the grasp is reachable just requires an inverse kinematics computation. Useful techniques exist that can precompute a model of the inverse kinematics of the robot arm and then use it to quickly find the best way to place the robot for a given end-effector placement [30]. In the case of a single-arm robot with only

[1]One of the cores of an Intel Core2Duo CPU T7250@ 2.00GHz, equipped with a 2.0 GigaBytes RAM

6 DOFs as shown in Fig. 10, randomly sampling the robot base configuration in a ring centered on the object and testing each grasp of the list for validity is fast enough.

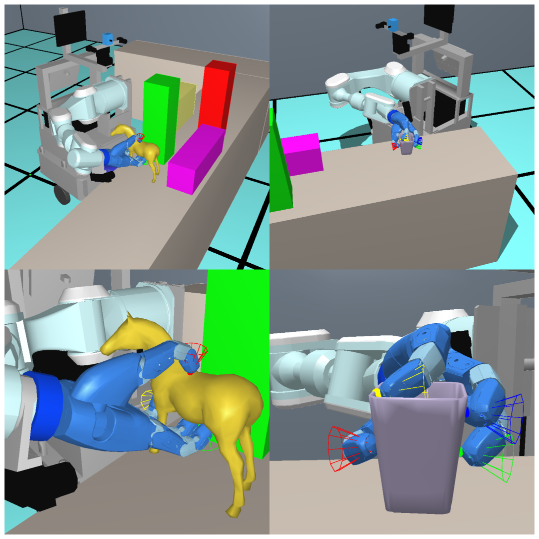

Figure 10: Examples of grasping configurations found for the whole robot: Far (top images) and close views (bottom images). Friction cones are also represented.

4 Conclusion

We have presented a method that can be used to compute a list of grasps for an object with any shape, for a multi-fingered hand. It is based on some approximation of the object surface and finger workspace volume, that can be used in combination with a dense sampling of the possible relative hand/object poses. The computed grasp list gives then a good overview of the variety of the different ways to grasp the object. Such a grasp set can be especially useful in the planning of manipulation tasks. A list could be built for each involved object before the motion planning phase, so that the planner can quickly search in the lists the grasps that are valid in a particular context.

Future work could concern the grasp quality evaluation. Indeed, in experiments with real platforms, object localization is often inaccurate, leading to real grasping configurations that can be far from the computed ones. Maximizing the robustness with respect to object localization error is of great interest. Favoring contacts where object curvature is low is a simple solution to improve robustness but better criteria could be implemented like [9].

Acknowledgment

The research has been funded by the EC Seventh Framework Programme (FP7) under grant agreement no. 216239 as part of the IP DEXMART. This work was also supported by the ANR project ASSIST.

References

[1] V.-D. Nguyen, Constructing force-closure grasps, Robotics and Automation. Proceedings. 1986 IEEE International Conference on 3 (1986) 1368–1373.

[2] C. Ferrari, J. Canny, Planning optimal grasps, Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on (1992) 2290–2295 vol.3http://dx.doi.org/10.1109/ROBOT.1992.219918

[3] D. Ding, Y.-H. Liu, S. Wang, The synthesis of 3d form-closure grasps, Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on 4 (2000) 3579–3584 vol.4. http://dx.doi.org/10.1109/ROBOT.2000.845289

[4] A. Bicchi, On the closure properties of robotic grasping, Int. J. Rob. Res. 14 (4) (1995) 319–334. http://dx.doi.org/http://dx.doi.org/10.1177/027836499501400402

[5] J. Cornelà, R. Suárez, Determining independent grasp regions on 2d discrete objects, Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on (2005) 2941–2946

[6] J. Cornellà, R. Suárez, Fast and flexible determination of force-closure independent regions to grasp polygonal objects, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on (2005) 766–771.

[7] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, J.-P. Merlet, On computing four-finger equilibrium and force-closure grasps of polyhedral objects, Int. J. Rob. Res. 16 (1) (1997) 11–35.

[8] M. Roa, R. Suárez, Independent contact regions for frictional grasps on 3d objects, Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (2008) 1622–1627

[9] R. Krug, D. Dimitrov, K. Charusta, B. Iliev, On the efficient computation of independent contact regions for force closure grasps, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, 2010, pp. 586 –591.

[10] A. Miller, S. Knoop, H. Christensen, P. Allen, Automatic grasp planning using shape primitives, Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on 2 (2003) 1824–1829 vol.2.

[11] C. Goldfeder, P. Allen, C. Lackner, R. Pelossof, Grasp planning via decomposition trees, Robotics and Automation, 2007 IEEE International Conference on (2007) 4679–4684

[12] A. Miller, P. Allen, Graspit! a versatile simulator for robotic grasping, Robotics & Automation Magazine, IEEE 11 (4) (2004) 110–122.

[13] K. Huebner, S. Ruthotto, D. Kragic, Minimum volume bounding box decomposition for shape approximation in robot grasping, Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (2008) 1628–1633

[14] K. Harada, K. Kaneko, F. Kanehiro, Fast grasp planning for hand/arm systems based on convex model, Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (2008) 1162–1168

[15] Z. Xue, J. Zoellner, R. Dillmann, Grasp planning: Find the contact points, Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on (2007) 835–840

[16] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, J. Kuffner, Grasp planning in complex scenes, in: Humanoid Robots, 2007 7th IEEE-RAS International Conference on, 2007, pp. 42 –48.

[17] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, J. Kuffner, Bispace planning: Concurrent multi-space exploration, in: Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland, 2008.

[18] S. M. LaValle, J. J. Kuffner, Rapidly-exploring random trees: Progress and prospects, in: Proceedings of Workshop on the Algorithmic Foundations of Robotics, 2000.

[19] M. Santello, M. Flanders, J. F. Soechting, http://www.jneurosci.org/cgi/content/abstract/18/23/10105Postural hand synergies for tool use, J. Neurosci. 18 (23) (1998) 10105–10115.

[20] E. Todorov, Z. Ghahramani, Analysis of the synergies underlying complex hand manipulation, Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE 2 (2004) 4637–4640.

[21] M. Ciocarlie, C. Goldfeder, P. Allen, Dimensionality reduction for hand-independent dexterous robotic grasping, Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on (2007) 3270–3275

[22] M. A. Goodale, A. Milner, Separate visual pathways for perception and action, Trends in Neurosciences 15 (1) (1992) 20–25.

[23] V. H. Franz, K. R. Gegenfurtner, H. H. Bülthoff, M. Fahle, Grasping visual illusions: No evidence for a dissociation between perception and action, Psychological Science 11 (1) (2000) 20–25.

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

[25] J. Bohg, D. Kragic, Learning grasping points with shape context, Robotics and Autonomous Systems 58 (4) (2010) 362–377.

[26] A. Yershova, S. LaValle, Deterministic sampling methods for spheres and so(3), in: Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Vol. 4, 2004, pp. 3974 – 3980 Vol.4.

[27] R. Suárez, M. Roa, J. Cornelà, Grasp quality measures, Tech. rep., Universitat Politecnica de Catalunya (UPC) (2006).

[28] G. Liu, Z. Li, Real-time grasping-force optimization for multifingered manipulation: theory and experiments, Mechatronics, IEEE/ASME Transactions on 9 (1) (2004) 65 –77.

[29] B. Bounab, A. Labed, D. Sidobre, Stochastic optimization-based approach for multifingered grasps synthesis, Robotica (28) (2010) 1021–1032.

[30] F. Zacharias, C. Borst, G. Hirzinger, Capturing robot workspace structure: representing robot capabilities., in: Intelligent Robots and Systems, 2007. (IROS 2007). 2007 IEEE/RSJ International Conference on, IEEE, 2007, pp. 3229–3236.