Unknown object grasping by using concavity

Lei, Qujiang; Wisse, Martijn

DOI
10.1109/ICARCV.2016.7838637

Publication date
2016

Document Version
Accepted author manuscript

Published in
Proceedings 2016 14th International Conference on Control, Automation, Robotics and Vision

Unknown Object Grasping by Using Concavity

Qujiang Lei
TU Delft Robotics Institute
Delft University of Technology
Delft, The Netherlands
q.lei@tudelft.nl

Martijn Wisse
TU Delft Robotics Institute
Delft University of Technology
Delft, The Netherlands
m.wisse@tudelft.nl

Abstract— **Reducing the grasp candidates for unknown object grasping while maintaining grasp stability is the goal of this paper. In this paper, we propose an efficient and straight forward unknown object grasping method by using concavities of the unknown objects to significantly reduce the grasp candidates. Shortest path concavity is first employed to work out the concavity value for every vertex of the unknown objects followed by concavity extraction to obtain the most salient concave areas. Grasp candidates are then generated on the most salient concave areas and evaluated by using force balance computation. Grasp candidates are ranked according to the result of force balance computation and the manipulability of every grasp candidate. The grasp with the best force balance and manipulability is chosen as the final grasp. In order to verify the effectiveness of our algorithm, some unknown objects commonly used by other papers about unknown object grasping are used to do simulations and favorable performance is obtained.**

Keywords-unknown object grasping; concavity; oriented bounding box; robot

I. INTRODUCTION

Grasping of unknown objects with neither appearance data nor object models given in advance is very important for robots that work in an unfamiliar environment. Vast research has been conducted on the problem of unknown object grasping and many achievements have been obtained in the previous years. However, unknown object grasping is still a challenging task that has not yet been solved in a general manner.

[1] gives a profound survey about unknown object grasping. The existing unknown object grasping algorithms can be divided into two main categories, that is, using partial model and using full model.

In order to accelerate the grasping process of unknown objects, partial information of an object may be used to realize the grasping. [2] uses partial object geometry to achieve a semantic grasp. This algorithm needs predefined example grasps and cannot deal with the grasping task of symmetric objects since multiple views of a symmetric object could have the same depth images. [3] proposes a data-driven grasp planner that requires partial sensor data. Matching and alignment methods were used for grasping after obtaining the Columbia Grasp Database. [4] uses local descriptors from several images to construct the 3D model of an object. Object registration was conducted by using a set of training images. [5] installs a 2D range sensor on the robot at an inclined angle to acquire partial shape information of the unknown objects. Two straight lines are extracted directly from this partial shape information as the two grasp sides for a parallel jaw gripper. [6] uses binocular vision to recover the partial 3D structure of unknown objects. Then process the incomplete 3D point clouds searching for good grasp candidate for a three finger robot hand according to a function that accounts for both the feasibility and the stability.

The second method is building a full 3D model using many images or point clouds of the target object. In [7], the full 3D model is fit and split into many minimum volume bounding boxes and a grasp is found on these bounding boxes. In [8], two flat, parallel surfaces are found on the 3D model to realize the grasping task with a gripper. In [9], the center of mass and axes of inertia of the target object are calculated from the 3D model, and then a grasp on the center or along the axes is found. [10] uses a genetic algorithm to search for grasping points on a 3D model of the target object. [11] uses a cost function to analyze the 3D model to obtain grasping points. In [12], the 3D model is simplified into some shape primitives (boxes or cylinders). Then grasping points which are assigned offline to these shape primitives are selected for the corresponding shape.

The best way to find a good grasp is said to use full 3D model to do grasp candidate simulation [13]. GraspIt which was first introduced in [13] established a benchmark for the object grasping community. However, there is a big problem we must face if we want to use GraspIt on unknown object grasping, that is how to deal with large number of grasp candidates promptly. After GraspIt, OpenGRASP [14] is invented on the base of the OpenRAVE [15], which made a progress comparing with GraspIt. OpenGRASP uses the normal of the object as the approaching vector of the robot hand, which can greatly reduce the number of grasp candidates. However, [16] states that depending on the choice of the parameters, the time of using OpenGRASP to simulate all the corresponding grasp candidates for a common object can vary from a few minutes to more than an hour. And then [16] uses [17] to do sampling to further reduce the grasp candidates. However, it still needs about one minute to find a good grasp for the unknown object, which is pretty time consuming. That is why the above 3D model based grasping algorithms try to use shape primitive or boxes to simplify the unknown objects.

The reason that using partial information for unknown object grasping is becoming popular is that it requires less data comparing with using full 3D model. In another word, using partial model can be quicker than using full 3D model. However using full 3D model can achieve a better stable grasp, especially when GraspIt or OpenGRASP is used even though it is time consuming. Can we combine the merits of these two methods together? What if we have a 3D model first and then we just use part of the 3D model to synthesize a stable grasp.

In our previous works [18, 19], we employed the principal axis of the unknown object to accelerate the grasp searching process and good results are obtained. In this paper, inspired by using curvature on unknown object grasping in [20] and using downsampling to reduce the number of grasp candidates in [16], we propose to use concavity of the unknown object to reduce grasp candidates to accelerate the generation of grasp candidates for the unknown objects. The concavity we said is different from the 2D curvature used in [20]. There is a significant difference between 2D curvature and 3D curvature regarding geometry properties. As can be seen from Fig.1 (a), the two blue points mean the maximum curvatures of the green silhouette of the spray bottle. Apparently, 2D curvature has a good performance to help searching a good grasp for robots. However, the situation for 3D curvature is different. As can be seen from Fig.1 (b), the blue in the red circles stands for the Gaussian curvature (3D curvature), which actually cannot give much clue for robot grasping. Fortunately, [21] and [22] give us inspiration about using concavity on unknown object grasping. Fig.1 (c) shows an example of the concavity that we mentioned. Specifically, the red area of the bunny represents the most concave area. Obviously, the concavities of the unknown object have a higher possibility to form a more stable force closure grasp. Our motivation is to generate grasp candidates on concavities and choose the best grasp by evaluating every grasp candidate.

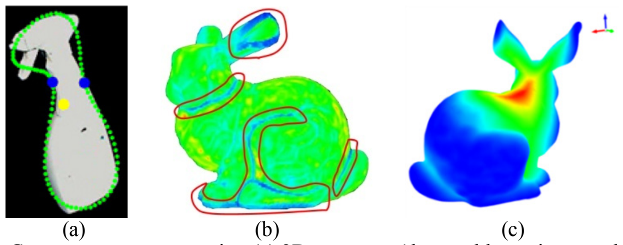


Fig. 1 Curvature versus concavity: (a) 2D curvature (the two blue points stands for the maximum curvature of the silhouette of the spray bottle), (b) 3D curvature (the blue in the red circles mean the Gaussian curvature of the bunny), (c) an example of concavity (the red represents the most salient concavity).

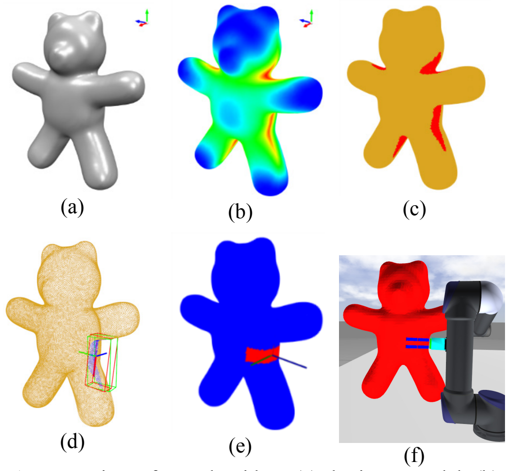


Fig. 2 An over view of our algorithm: (a) the input model, (b) concavity computation, (c) the most salient concavities are extracted (the red stands for the most salient concavities), (d) concavity analysis, (e) the returned best grasp, (f) the execution of the best grasp.

The overview of our grasping algorithm is as follows. First, the 3D model (point cloud or meshed model) is input and the concavity of the unknown object is worked out as Fig.2 shows. Then the most salient concavities are extracted according to the concavity intensity. After that, grasp candidates are generated followed by the evaluation of every grasp candidate. Finally, the grasp with best force balance and manipulability will be chosen as the final grasp to execute.

This paper is organized as follows. Section II contains a detailed explanation of our algorithm, Sections III shows the simulation results, and section IV is the conclusion of this paper.

II. DETAILED ALGORITHM

A. Concavity calculation

Concavity calculation is carried out directly on the meshed model of the target object. The meshed model can be obtained by fast triangulating the full point cloud of the target object. Take the bear as example, Fig.3 (a) is its full point cloud and Fig.3 (b) is its meshed model. Then a proper outer convex hull is computed (shown as the Fig.3 (c)). The free space between the meshed model and the proper outer convex hull is used to compute the concavity. First, the stable Constrained Delaunay Tetrahedralization is used to discretize the free space. The corresponding discretized space is shown as Fig.3 (d). Then Fast Marching Method is employed to compute the shortest path distance by using the discretized free space. The shortest path distance between the meshed model and the convex hull is returned as the concavity value of every vertex of the meshed model. Fig.3 (e) shows the concavity computation result, as can be seen, the original concavity value is in disorder. In order to have a better view of the concavity computation result, the concavity value is rendered by color according to its intensity from the maximum concavity value to the minimum concavity value (the red means the maximum concavity and the blue means the minimum concavity). Fig.3 (f) shows the result of the rendered concavity.

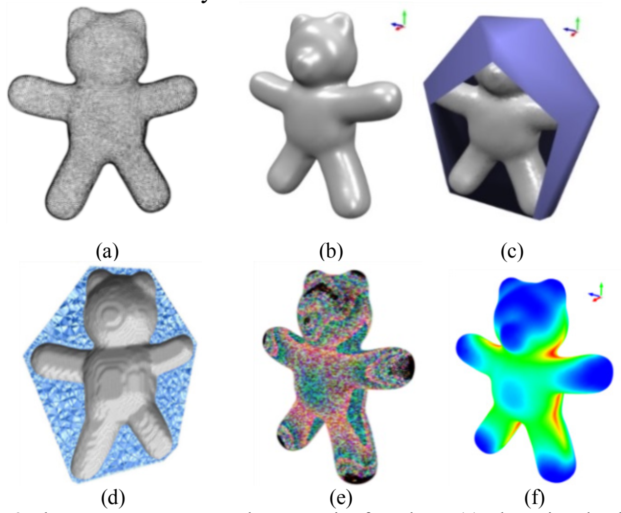


Fig. 3 The process to compute the concavity for a bear, (a) The point cloud of the bear, (b) the meshed model of the bear, (c) the convex hull of the bear, (d) discretization of the free space between the bear and the convex hull, (e) the original concavity computation result for every point in (a), (f) the result of the rendered concavity.

B. Concavity extraction

As can be seen from Fig.3 (f), the red stands for the most concave areas and the blue represents the least concave areas. In order to make it convenient to do further analysis on the point cloud with the concavity value of every point, the most concave areas are extracted. The red in Fig.4 (a) means the most concave areas. It is extracted out from the whole point cloud to be shown as the Fig.4 (b). After that, the Euclidean cluster extraction is employed to separate the concavity with each other (shown as the Fig.4 (c)). Further grasping analysis will carried out on the separated concavity point clouds. Specifically, the separated point clouds of the concavity will be used to generate grasp candidates.

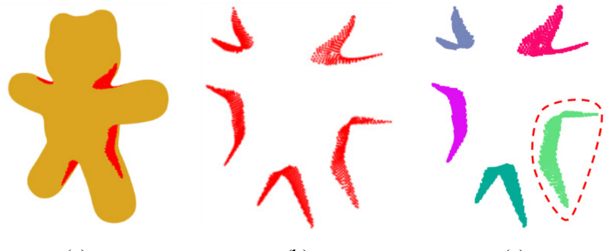


Fig. 4 The extraction and separation of the bear concavity. (a) and (b) demonstrate the extraction of the most salient concavities (the red areas), (c) the most salient concavities are separated into every single concavity.

C. Construct the concavity coordinate system

The concavity in the red dashed circle in Fig.4 (c)) is used to explain our following algorithm. After we get the point cloud of the every single concavity, we can use the principle of OpenGRASP. The grasp candidates can be generated by using the normal of the point cloud. But one important fact we must notice is that there are still a lot of grasp candidates even though we downsampled the normal of the point cloud. The reason is that the normal can only decide the approaching direction of the robot hand. The robot hand can rotate around the normal, which means a lot of grasp candidates can be generated. In order to effectively decrease the number of grasp candidates. We propose to use oriented bounding box to reduce the grasp searching, which will be explained later.

Let's first look at what the oriented bounding box is and how to get the oriented bounding box. There are two common ways to obtain a bounding box, that is the axis-aligned bounding box (AABB) and the oriented bounding box (OBB). The axis-aligned bounding box for a given point set is its bounding box subject to the constraint that the edges of the box are parallel to the Cartesian coordinate axes. It is simply the Cartesian product of N intervals each of which is defined by the minimal and maximal value of the corresponding coordinate for the points. The oriented bounding box is the bounding box calculated subject to no constraints as to the orientation of the result. By using the eccentricity and moment of inertia, a position vector and a rotation transform matrix can be obtained. And then, each vertex of the given AABB must be rotated with the given rotation transform matrix and then positioned to get the OBB. Therefore, the OBB is much more generous and better suitable for concavities analysis. The green and red rectangular parallelepiped frames in Fig.5 respectively demonstrate the axis-aligned bounding box and the oriented bounding box. The blue, green and red straight lines in Fig.5 (b) respectively stand for the X, Y and Z axis of the concavity coordinate system.

After the oriented bounding box is obtained, the algorithm will analyze which two parallel planes of the rectangular parallelepiped frames have the higher possibility to be grasped by robot hand without collision with the object. For the concavity shown in Fig.5 (a), an oriented bounding box is obtained as Fig.5 (b) shows. The oriented bounding box can be divided into three pairs of parallel planes. The pair of the front and the back planes has the less possibility to collide with the bear. Therefore, the following grasp analysis will be carried out in XOY plan of the concavity coordinate system.

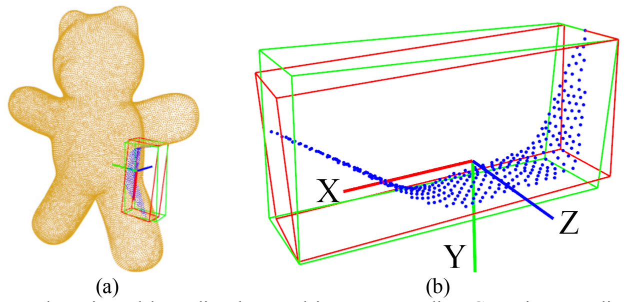


Fig. 5 The oriented bounding box and its corresponding Cartesian coordinate system. The blue and the red rectangular parallelepiped frames respectively stand for the axis-aligned bounding box and the oriented bounding box.

D. Analyze concavity and generate grasp candidates

In the above section, the oriented bounding box and its corresponding Cartesian coordinate system have been obtained. Actually, the three axis of the Cartesian coordinate system respectively sand for the major eigenvector, the middle eigenvector and the minor eigenvector. According to the analysis in section C, the major and middle vector will be used to generate the grasp candidates. First, the point cloud of the concavity is projected to the XOY plane of the OBB coordinate system (shown as the Fig. 6).

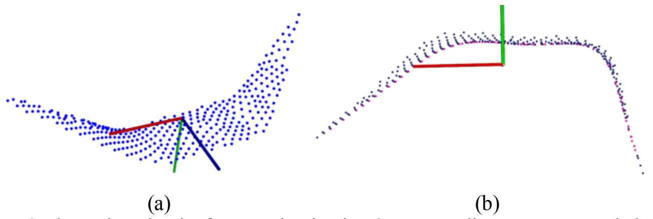


Fig. 6 The point cloud of concavity in the OBB coordinate system and the projected point cloud, (a) the point cloud of the concavity in the OBB coordinate system, (b) the projected point cloud of the concavity.

After that, we need to extract the concavity outer layer on the projected point cloud. The outer layer boundary (shown as the purple points in Fig.6 (b)) of the projected point cloud will be used to configure the grasp candidates. We first obtain the concave hull of the projected point cloud (shown as the Fig.7). The concave hull can be divided into two parts, one part is inside the object, and the other part is on the boundary. Usually, the point density is employed to judge whether a two dimensional point is on the boundary or not, therefore, an enlarged point cloud is extracted by adding more less concave areas (shown as the Fig.8 (a)). Then the enlarged point cloud is downsampled shown as the red points in the Fig. 8 (b).

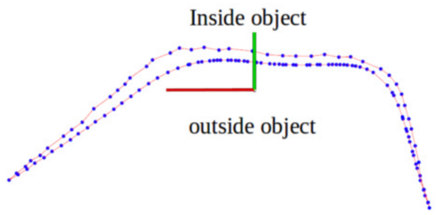


Fig. 7 Concave hull of the projected point cloud of the concavity.

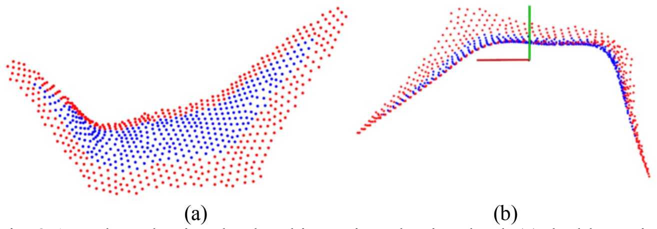


Fig. 8 An enlarged point cloud and its projected point cloud, (a) the blue points mean the concavity point cloud shown in Fig. 6 (a), the red points mean the enlarged point cloud by add more less concavity area, (b) the enlarged point cloud is projected and downsampled shown as the red points.

Originally, we tried to compare the point density of every point (the blue points in Fig. 9) on the concave hull of the concavity point cloud. By introducing a circle with the radius of R (the green and the black circle in Fig. 9), we can get the number of the points within the circle. Apparently, the point on the boundary will have a low point density. However, all the blue points are pretty close to the boundary and the downsampled enlarged point cloud is sparse, comparing the density of every point on the concave hull of the projected concavity point cloud is not robust, especially for the blue points in the green rectangular.

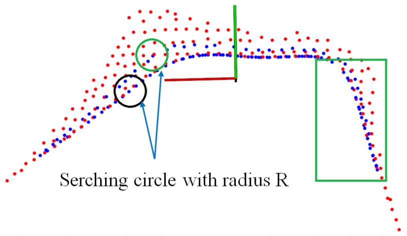


Fig. 9 Extract the boundary points by compare the point density of every blue point.

Considering the instability of comparing point density, we come up with a method to extract boundary point more stably. Specifically, two concave hulls of the concavity point cloud and the enlarged point cloud are used. By observing the two concave hulls shown in Fig.10 (a), it is pretty obvious that the boundary points on the concave hull of the concavity point cloud are almost the same or pretty near to the points on the concave hull of the enlarged point cloud. Therefore, the distance between every blue point and the green line (shown as the Fig.10 (b)) is used to judge whether a point is on the boundary or not. One important fact we can use is that points on concave hull is generated in sequential order, as can be seen, the blue point on the upper red line of Fig.10 (b) is in order as n-1, n and n+1. A vector can be used to store all the straight lines between the two adjacent red points of the concave hull of the enlarged point cloud. For every blue point, A vertical straight line to the straight lines constructed by the two adjacent red points can be obtained. If the intersection point between the vertical line and the straight line is lying between the two red points, then that intersection point is recorded. The orange point in Fig.10 (b) is an example of the intersection point and it satisfies the equation (1). A distance threshold ($d_{threshold}$) is given by the system to decide whether the blue point (point n) is on the boundary or not, if the distance $d < d_{threshold}$, the point is considered as the boundary point. Using above method to go through all blue points can get the boundary point cloud shown as the orange points in Fig.11.

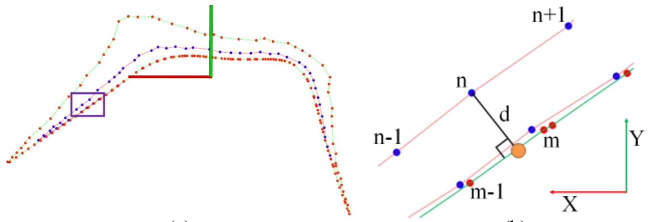


Fig. 10 (a) is the two concave hulls of the projected concavity point cloud and the enlarged point cloud, (b) is the enlarged image of the points in the purple rectangular in (a).

$$\begin{cases} x_m < x_{point\_orange} < x_{m-1} \\ y_{m-1} < y_{point\_orange} < y_m \end{cases} \tag{1}$$

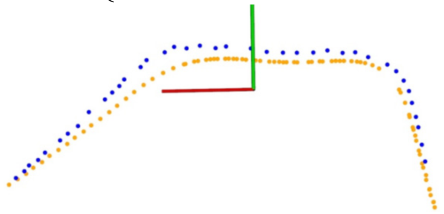


Fig. 11 Orange points mean the boundary points of the concavity point cloud.

E. generate grasp candidates

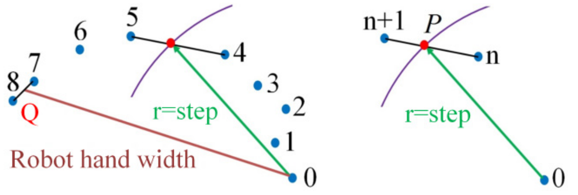
After we get the boundary point cloud, the points are not in order. The first thing to do is to make all the boundary points into order. We first find the point with $x_{\min}$ and store it into a vector ($V_{boundary\_in\_order}$). Then sequentially find the closest point and add it into the vector. Fig.12 shows an example of the boundary points, which are in sequential order as 0,1,2,3,4,5, till 8.

After the boundary point cloud in order is obtained, the next to do is to calculate a series step points. Step points are used to configure the grasp candidates. It means the robot will search along the boundary with a step. Searching process of the step points will start from the point 0 ($V_{boundary\_in\_order}$[0]). And then search along the boundary to find any two adjacent points, which can construct a straight line and have intersection point with a circle with a radius equal the searching step. If the step is give as r and the two adjacent points are point n and point $n+1$, the distance between the point 0 and point n is defined as d_{0_n} and the distance between the point 0 and point $n+1$is defined as d_{0_n+1}.

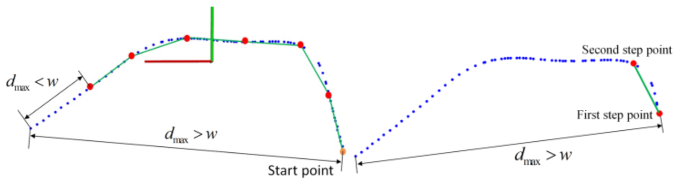
If the distance between the start point and the two adjacent points satisfies anyone of the equation (2), there must be an intersection point between the boundary and the step circle. Fig.12 shows an example of searching the step points. The purple curve stands for the searching circle with the radius equaling the searching step. The purple curve has an intersection point with the straight line between point 4 and point 5. Giving the coordinate value of point 0, point n, point $n+1$ and the step length (r), using equation (3) can get the coordinate value of the intersection point P (shown as the red point in Fig.12 (b)). After we get the first step point, all the points from the point 0 to point n will be removed from the current boundary point cloud to form a new point cloud. The new point cloud is shown as the Fig.13 (b). Then, the $d_{\max}$ is checked to see whether it is bigger than the robot hand width. If yes, then the algorithm will repeat the way of finding the first step point to find the second step point. The above steps are repeated until $d_{\max}$ is smaller than the robot hand width (shown as the Fig.13 (a)).

$$\begin{cases} d_{0\_n} < r < d_{0\_n+1} \\ d_{0\_n+1} < r < d_{0\_n} \end{cases} \tag{2}$$

$$\begin{cases} m = \dfrac{p_n.x - p_{n+1}.x}{p_n.y - p_{n+1}.y} \\ w = p_{n+1}.x - mp_{n+1}.y \\ p.y = \dfrac{2p_0.y + 2mw \pm \sqrt{(2p_0.y + 2mw)^2 - 4(m^2+1)(p_0.y^2 + w^2 - r^2)}}{2(m^2+1)} \\ p.x = my + p_{n+1}.x - mp_{n+1}.y \end{cases} \tag{3}$$

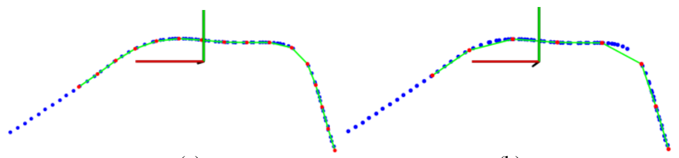


(a) (b)
Fig. 12 The step point searching

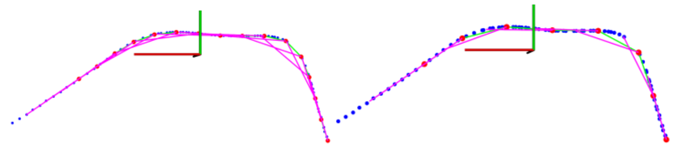


(a) (b)
Fig. 13 The way to find all the step points, if a step point is found, the points out of the searching circle will form a new point cloud as (b).

Fig.14 shows the step searching result with different step length. The blue points stands for the boundary point cloud of the concavity. The red points represent all the step points. Fig.14 (a) and (b) respectively show the result of step searching with a small and a big step length.


(a) (b)
Fig. 14 The step points obtained by using different steps length. The blue points are the boundary points of the concavity. Every green line stands for a step. Every red point represents a step point. (a) shows the result of step searching with a small step. (b) demonstrates the result of step searching with a big step.

After all the step points are obtained, the way to work out the step point can be used to work out the hand configuration. In Fig.12, if point 0 is a step point and the length of the brown line equals the robot hand width, using method shown in Fig.12 (b) can work out the intersection point Q. A grasp candidate can be configured between point 0 and point Q (in Fig.12 (a)). The purple lines in Fig.15 stand for all the grasp candidates. Fig.15 (a) and (b) respectively show all the grasp candidates corresponding to Fig.14 (a) and (b).


(a) (b)
Fig. 15 The grasp candidates obtained by using a small step and a big step. The purple lines stand for the grasp candidates. Blue points, green lines and red points are the same as Fig.14.

After all the grasp candidates are obtained, middle vertical lines to every grasp candidate are work out shown as blue lines in the Fig.16. On every grasp candidate, a local Cartesian coordinate system is established by using purple lines as the X axis and the blue lines as the Z axis. Using cross product of the Z axis and the X axis can get the Y axis. Then point cloud for every grasp candidate is extracted and transformed to the local coordinate system. Fig.17 shows 3 example grasp candidates in their own local coordinate system.

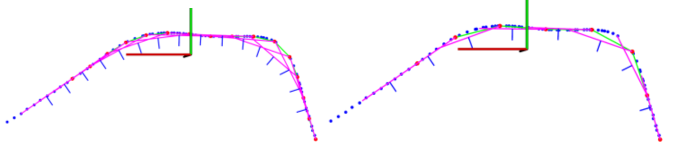


Fig. 16 The construction of local coordinate system, the purple lines represent the grasp candidates and work as the X axis of the local coordinate system. The middle vertical lines (blue lines) work as the Z axis.

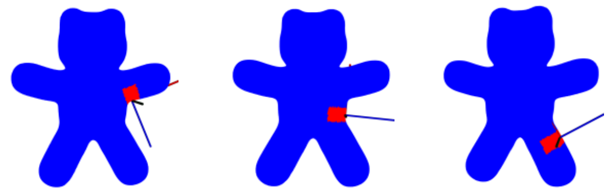


Fig. 17 Three grasp candidates and their coordinate system

F. Force balance computation and manipulability analysis

Force balance analysis is carried out on every grasp candidate to evaluate the stability of the grasp candidate. Fig. 18 is one grasp candidate from the 16 grasp candidates shown in Fig.15 (a). This grasp candidate will be used to explain the way of doing force balance analysis.

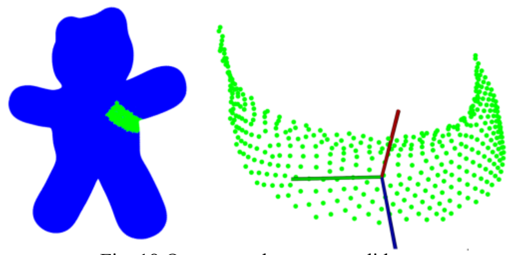
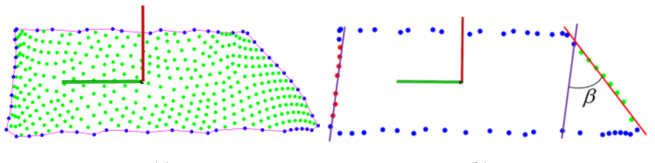


Fig. 18 One example grasp candidate

At first, the point cloud of the grasp candidate is projected to the XOY plane (made of the red and green lines in Fig.18). The projected point cloud can be seen as the green points in Fig.19 (a). Then the concave hull (the blue points in Fig. 19(a)) of the projected point cloud is abstracted. The two grasp sides are extracted shown as the red points and the green points in the Fig.19 (b). After we get the points of the two grasp sides, a straight line fitting method is employed to do line fitting for the two grasp sides. If a straight line is defined as $y = kx + b$, equation (4) can be used to work out k and b. The purple line on the left in Fig.19 (b) stands for the fitting line for the red points. Correspondingly, the red line on the right in Fig.19 (b) represents the fitting line for the green points. The angle (β) between the purple line and the red line is used to evaluate the grasping stability of this grasp candidate. Fig.20 shows the result of force balance computation of the 16 grasp candidates.



(a) (b)

Fig. 19 (a) shows the projected point cloud (green points) of the grasp candidate, the blue points mean the concave hull of the projected point cloud. (b) is force balance analysis of the two grasp sides by using line fitting.

$$\begin{cases} k = \dfrac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i} \\ b = \dfrac{1}{n}\sum_{i=1}^{n} y_i - k\dfrac{1}{n}\sum_{i=1}^{n} x_i \end{cases} \qquad (4)$$

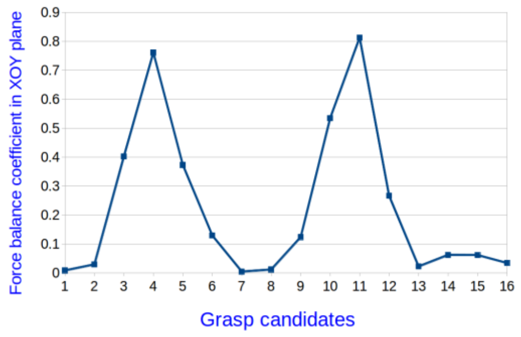
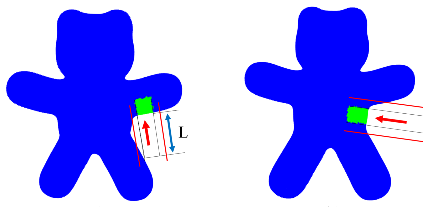


Fig. 20 The result of force balance computation of the 16 grasp candidates shown in Fig.15 (a)

As can be seen from Fig.20, both grasp candidate 1 and 7 have good force performance. Because all grasp candidates are generated on concavity of the target object. That means when the robot tries to execute the grasp action, the robot may collide with the object. Therefore, it is necessary to analyze the manipulability of every grasp candidate which has good force balance. Fig.21 shows the grasp 1 and grasp 7. When the robot executes the grasp, it will approach the object along the red arrow. If the length of the end-effector of the robot is longer than L shown in Fig.21, the robot should not grasp in this configuration. The width of the end-effector should also take into consideration. If the two red lines in Fig.21 means the width of the end-effector, the collision between the red lines should be considered. If the collision is OK with robot, then the robot will choose the grasp with best force balance. Here, grasp 7 is chosen as the best grasp for this concavity. After finishing the grasping analysis of the example concavity shown in dashed circle in Fig.4 (c), the algorithm will repeat the above concavity analysis on other four concavities shown in Fig.4 (c). A best grasp is returned for every concavity; then, the best grasp from every concavity is compared with each other in the aspects of force balance and manipulability, and then the best one is chosen as the final grasp execution.



(a) (b)

Fig. 21 The comparison of two grasps with good force balance.

III. Simulation

In order to verify our grasping algorithm, several objects in different geometry shapes are chosen to do simulations. All the tested objects can be seen in the first column in Fig.22. The second and third column shows the results of concavity computation and concavity extraction. The fourth column is the results of force balance computation of grasp candidates on one concavity of the object. The best grasps for the objects of the first column is shown in the fifth column. Fig.23 shows the execution of the best grasp returned from the algorithm. The returned best grasp for each object has good force balance and manipulability. The concavity computation for these objects can be finished within ten seconds. Concavity analysis, grasp candidate generation and force balance analysis can be completed within 2 seconds. Therefore, the whole grasping computation from computing the concavity to obtaining the best grasp is within 12 seconds, which is much faster than [16]. [16] uses OpenGRASP and downsampling of the grasp candidates, but the time for an common object still needs about one minute. In summary, the simulations demonstrated our improvement over other grasping algorithms which also use full 3D model.

IV. Conclusion

In this paper, a novel grasping algorithm for unknown objects is presented. Concavity is first introduced to unknown object grasping in this paper. Oriented bounding box is used to construct the coordinate system for every concavity followed by grasp candidate generation on every concavity. Force balance analysis and manipulability analysis are employed to evaluate every grasp candidate and the grasp with best force balance and manipulability is returned as the final grasp. In order to verify the effectiveness of our algorithm, several objects commonly used by other grasping algorithms with different geometric shapes are used to do simulations and successful results are obtained. Our algorithm can quickly finish the whole grasping computation from calculating concavity to obtaining the best grasp within 12 seconds, which is much faster than [16]. [16] uses OpenGRASP and downsampling of the grasp candidates, however, the time for an common object grasping needs about one minute. In summary, our algorithm shows improvement over other grasping algorithms which also use full 3D model. Our algorithm has a much better performance in time efficiency and grasping stability.

ACKNOWLEDGMENT

The work leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 609206.

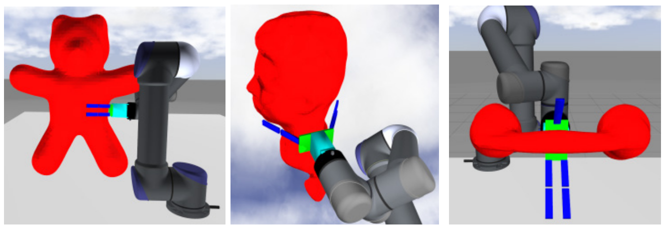


Fig. 23 Execution of the best grasp returned from the algorithm.

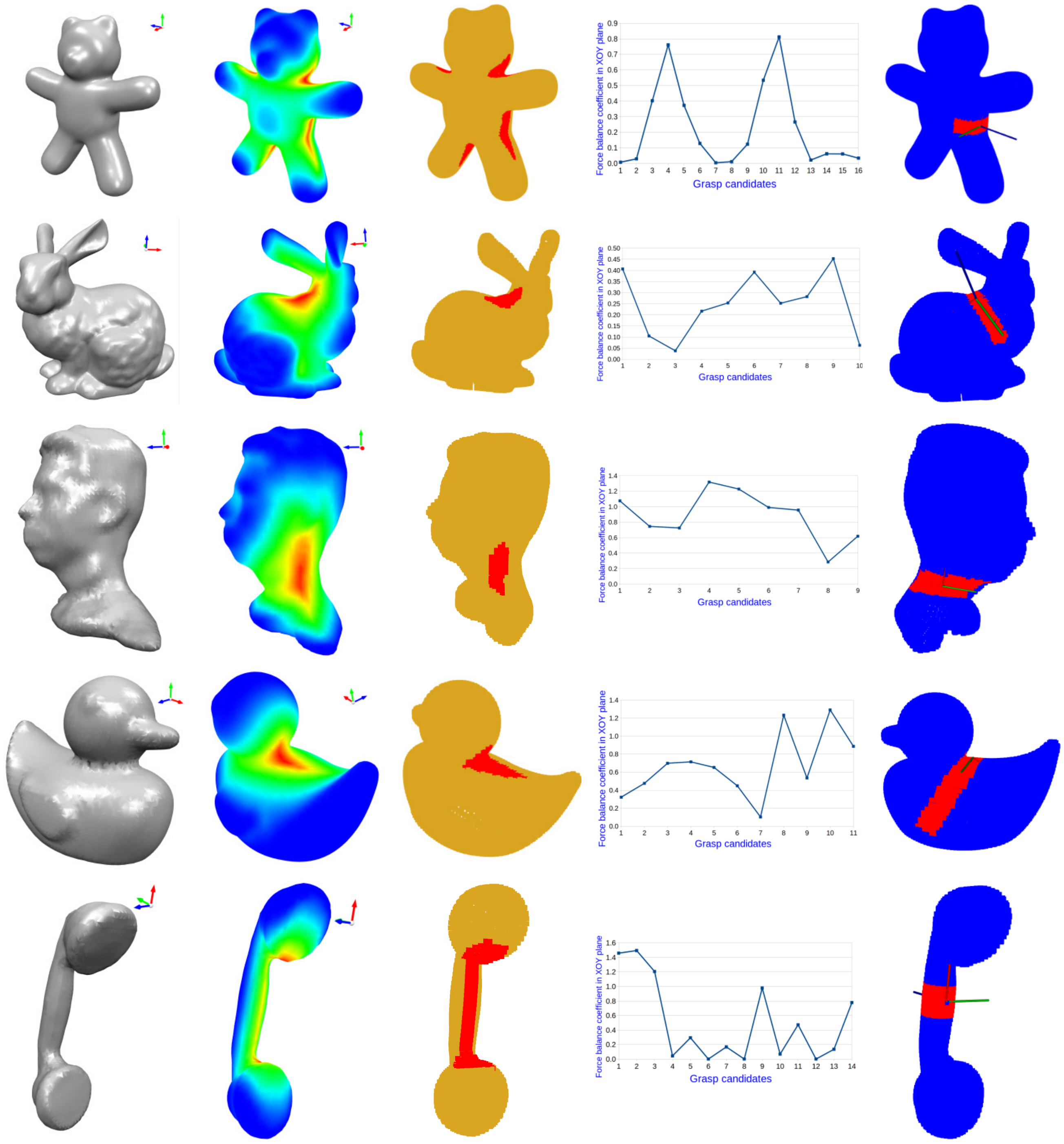


Fig.22 simulation results

REFERENCE

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven graspsynthesis—a survey,” IEEE Transactions on Robotics, VOL. 30, NO. 2, pp. 289–309, 2014.

[2] Hao Dang and Peter K. Allen, “Semantic Grasping: Planning Robotic rasps Functionally Suitable for An Object Manipulation Task,” in IROS, pp. 1311-1317, 2012.

[3] Goldfeder C, Ciocarlie M, Peretzman J, Dang H, Allen PK, “Data-driven grasping with partial sensor data,” in IROS, pp. 1278–1283, 2009.

[4] Collet A, Berenson D, Srinivasa SS, Ferguson D, “Object recognition and full pose registration from a single image for robotic manipulation,” in ICRA, pp. 3534–3541, 2009.

[5] Zhaojia Liu, Lounell B. Gueta, and Jun Ota, “Feature Extraction from Partial Shape Information for Fast Grasping of Unknown Objects,” in ROBIO, pp. 1332–1337, 2011.

[6] I. Gori, U. Pattacini, V. Tikhanoff, and G. Metta, “Three-finger precision grasp on incomplete 3D point clouds,” in ICRA, pp. 5366–5373, 2014.

[7] Hubner K, Kragic D, “Selection of robot pre-grasps using box-based shape approximation,” in IROS, pp. 1765–1770, 2008.

[8] Bone GM, Lambert A, Edwards M, “Automated modeling and robotic grasping of unknown three dimensional objects,” in ICRA, pp. 292–298, 2008.

[9] E. Lopez-Damian, D. Sidobre, and R. Alami, “A grasp planner based on inertial properties,” in ICRA, pp. 754–759, 2005.

[10] H.-K. Lee, M.-H. Kim, and S.-R. Lee, “3D optimal determination of grasping points with whole geometrical modeling for unknown objects,” Sensors and Actuators, vol. 107, pp. 146–151, 2003.

[11] K. Yamazaki, M. Tomono, and T. Tsubouchi, “Picking up an Unknown Object through Autonomous Modeling and Grasp Planning by a Mobile Manipulator,” Field and Service Robotics, Springer Tracts in Advanced Robotics, vol. 42, pp. 563–571, 2008.

[12] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, “Automatic grasp planning using shape primitives,” in ICRA, pp. 1824–1829, 2003.

[13] Andrew Miller and Peter K. Allen, “Graspit!: A Versatile Simulator for Robotic Grasping,” IEEE Robotics and Automation Magazine, vol. 11, no. 4, pp.110–122, 2004.

[14] B. Le´on, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales,T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and R. Dillmann, “OpenGRASP: A toolkit for robot grasping simulation,” in SIMPAR ’10:Proceedings of the 2st International Conference on Simulation, Modeling, and Programming for Autonomous Robots, pp. 109–120, 2010.

[15] R. Diankov and J. Kuffner, “Openrave: A planning architecture forautonomous robotics,” Robotics Institute, Pittsburgh, PA, Tech. Rep.CMU-RI-TR-08-34, 2008.

[16] J. Bohg, M. Johnson-Roberson, B. Leon, J. Felip, X. Gratal, N. Bergstrom, D. Kragic, and A. Morales, “Mind the gap–robotic grasping under incomplete observation,” in ICRA, pp. 686–693, 2011.

[17] M. Richtsfeld and M. Vincze, “Grasping of Unknown Objects from a Table Top,” in *ECCV Workshop on ’Vision in Action: Efficientstrategies for cognitive agents in complex environments’*, Marseille,France, 2008.

[18] Qujiang Lei, Martijn Wisse, “Fast grasping of unknown objects using force balance optimization”, in IROS, 2014, pp. 2454–2460.

[19] Qujiang Lei, Martijn Wisse, “Unknown object grasping using force balance exploration on a partial point cloud”, in AIM, 2015, pp. 7–14.

[20] B. Calli, M. Wisse, and P. Jonker, “Grasping of unknown objects viacurvature maximization using active vision,” in IROS, pp. 995–1001, 2011.

[21] J. ming Lien and N. M. Amato, “Approximate convex decomposition of polygons,” in SoCG, pp. 17–26, 2004.

[22] H. Zimmer, M. Campen, L. Kobbelt, “Efficient computation of shortest path-concavity for 3D meshes,” in CVPR, pp. 2155–2162, 2013.