Object grasping by combining caging and force closure

Lei, Qujiang; Wisse, Martijn

DOI
10.1109/ICARCV.2016.7838638

Publication date
2016

Document Version
Accepted author manuscript

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

Object Grasping by Combining Caging and Force Closure

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*— The current research trends of object grasping can be summarized as caging grasping and force closure grasping. The motivation of this paper is to combine the advantage of caging grasping and force closure grasping to enable under-actuated grippers like the Lacquey gripper and the parallel grippers like the PR2 gripper to quickly grasp the flat unknown objects. Inspired by the idea that caging grasping generates finger points along the object's boundary and considering the geometry property of the grippers, we propose to allocate a discrete set of finger candidates along the object's boundary. Any two of the finger candidates can form a grasp candidate, which is analyzed by using force closure to choose the best grasp candidate as the final grasp execution. The grasp quality during the manipulation of the object is guaranteed by considering the gravity of the object. Simulations and experiments on an Universal arm UR5 and an under-actuated Lacquey Fetch gripper are used to examine the performance of this algorithm, and successful results are obtained.**

Keywords-object grasping; caging; force closure; robot

I. Motivation

The motivation of this paper is to quickly find suitable grasp for flat objects (shown as the Fig.1 (a)), specifically, this grasping algorithm is specially designed for under actuated grippers like the Lacquey gripper (shown as the Fig.1 (b)) or parallel grippers like the PR2 gripper (shown as the Fig.1 (c)). In order to enhance grasping stability, force balance and torque balance are taken into consideration. The stability is divided into two parts: one is the stability when the grasp action is being executed; the other is the stability while the object is being transported. These two parts of stability can ensure that the object is securely grasped during the whole process when the object is being grasped and manipulated. Inspired by [1] and [2], a novel grasping algorithm is proposed for flat unknown objects. [1] and [2] only concentrate on the objects themselves without considering the geometry property of the gripper. We are illuminated to combine the force closure grasping and caging grasping. In this paper, we propose to consider both force and torque balance, as well as the geometry property of the robot gripper, for example, hand width and grasping range, when the robot tries to execute the grasp. Then the gravity of the object is considered when the robot tries to manipulate the object after it is grasped. This grasping algorithm has several advantages. First, it is simple to implement, which can lead to sound computational efficiency. Second, considering both force balance and torque balance and the geometry property of the gripper can ensure the grasp is executed successfully. Third, the grasping quality during the manipulation of the object is also guaranteed by considering the gravity of the object.

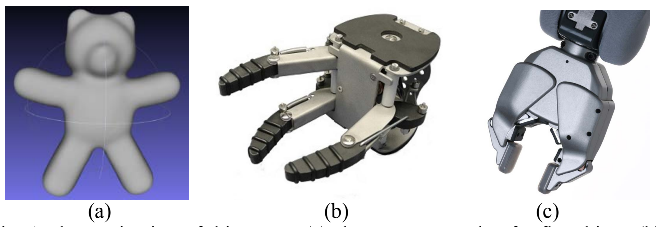
Fig. 1 The motivation of this paper, (a) shows an example of a flat object, (b) and (c) show the Lacquey gripper and the PR2 gripper respectively. The motivation of this paper is to quickly find suitable grasp on flat objects for under-actuated grippers like the Lacquey gripper or parallel grippers like the PR2 gripper.

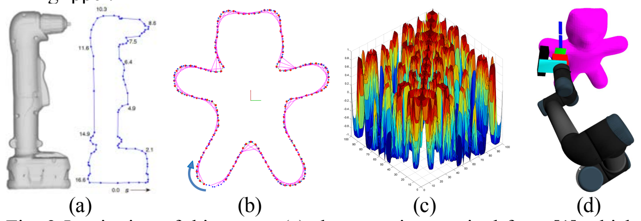
Fig. 2 Inspiration of this paper, (a) shows an image cited from [1] which uses caging method by introducing a discrete set of finger points allocating along the object's boundary to grasp the target object. (b) shows our inspiration. Inspiration from [1] promotes us to generate finger candidates (the purple lines) along the object boundary. (c) shows the result of force balance computation for all grasp candidates (one grasp candidate can be obtained by combining any two finger candidates in (b)). (d) shows the final grasp execution.

II. Introduction

Caging grasping is becoming increasingly popular in recent years. Since caging grasping was first introduced by [3] and [4], the analysis and synthesis of caging grasps has become an active research area. The basic idea of caging grasping is that the manipulators or fingers constitute a set of constraints in the object's configuration space that prevent it from escaping arbitrarily far. [5] proposed the first two-finger caging grasping for polygonal objects. Since the early works, many algorithms have been invented for finding two or three finger caging grasp for polygonal planar objects. [6] and [7] present comprehensive two-finger caging synthesis algorithms by formulating the caging grasping problem in the four dimensional configuration space of the two-finger hand. Afterwards, [1] formulates the caging set synthesis problem in two dimensional contact space which parameterizes the finger locations along the object's boundary. Several papers go further to consider the problem of planning and controlling the caging manipulation of an object by a team of mobile disc robots [8], [9], [10]. All above caging grasping algorithms by two/three-finger hand or by a team of mobile disc robots illuminate us to use a discrete of finger candidates allocating along the object's boundary to generate the grasp candidates. Fig.2 (a) and (b) show our inspiration of using caging grasping to generate grasp candidates.

Force closure grasping is a popular approach in the field of robotic grasping. Vast amount of research has been conducted in the domain of force closure grasping [11], [12]. Given the 3D meshed model of the target object and the friction coefficients, force closure grasping employs a grasp quality scoring function defined in terms of contact points and surface normal on the object to generate force stable grasp candidates [13], [14]. Force closure grasping confirms well with human's grasping synthesis. If given the 3D model of the target object, human can synthesize suitable grasp candidates by using the geometry information of the 3D model and the force closure requirement. Therefore, force closure grasping is a very promising method to solve the problem of unknown object grasping. Our previous works [15, 16] create a new method to compute force balance grasp directly on the partial point cloud of the target object. [15, 16] do not require the 3D meshed model and the friction coefficients of the target object, which makes it more practical for unknown object grasping. The advantage of force closure grasping sheds illumination on using force balance and torque balance on robot grasping. Fig.2 (c) shows the inspiration of using force balance analysis on object grasping, and the idea of force balance searching from [15, 16] will be used in this paper.

Inspired by the advantage of caging grasping and force closure grasping, we propose to use the method that caging grasping adopts to generate finger candidates along the object's boundary. After that, force closure analysis is employed to do force balance and torque balance computation. Specifically, force and torque balance computation is divided into two parts: one is the balance during the grasping execution; the other is the balance during the object manipulation after it is grasped. The purpose of the force balance and torque balance during the grasp execution is to assure that big movement and rotation will not occur (exact explanation is given in section E). The aim of considering the force balance and torque balance during the manipulation of the object is to ensure that the possibility of the object sliding from the gripper is minimized. Grasping quality during the manipulation of the object is guaranteed by considering the gravity of the object.

This paper is organized as following: section III contains a detailed explanation of our algorithm, section IV shows the simulation results, section V demonstrates the experiment results, section VI discusses the comparison between our algorithm and [1], section VII is a conclusion of this paper.

III. Detailed algorithm

This section contains a detailed explanation of the whole grasp algorithm. Part A shows how to borrow the idea from caging grasping to generate finger candidates. Part B demonstrates the details about how to use force closure analysis to work out good grasp candidates. Part C shows the gravity analysis.

A. Grasp candidates generation

The existing work about flat polygonal object grasping usually have a hypothesis, that is, the polygonal object is on the desk. In this paper, we make a progress to enable the robot to find the main plain by employing the oriented bounding box (OBB), which can ensure that the robot finds the main plane on which to project the point cloud of the object. Even if the object is held in the air, the robot can find the polygonal contour of the object.

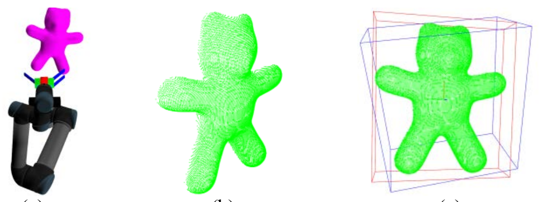

Fig.3 Construction of the oriented bounding box (OBB), (a) shows the virtual setup in simulation environment, an Asus Xtion sensor is installed at the end of the robot arm. (b) shows the point cloud acquired by the Asus Xtion sensor. (c) shows the OBB box, the red rectangular frame stands for the OBB box, the blue rectangular frame represents the axis-aligned bounding box (AABB).

A.1 Construct the Oriented Bounding Box

A teddy bear is used to explain our algorithm. Fig.3 (a) shows a virtual setup in a simulation environment. An eye-in-hand system is established by installing an Asus Xtion sensor at the end of the robot arm, which is used to capture the point cloud of the target object. After the point cloud of the target object (shown as Fig.3 (b)) is obtained, the Oriented Bounding Box (OBB) algorithm is used to find the main plane to project the point cloud.

There are two 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. 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. The blue and the red rectangular frames in Fig.3 (c) respectively stand for the Oriented Bounding Box and the axis-aligned bounding box. We can easily find that the oriented bounding box is more generous and better suitable for the grasping purpose.

A.2 Project the point cloud to the main plane of the OBB

A local object coordinate system can be established by using the Oriented Bounding Box shown in the Fig.4 (a). The red, green and blue lines respectively stand for the X, Y and Z axis of the local object coordinate system. Then the point cloud is projected to the XOY plane (the main plane) to obtain the main silhouette of the target object shown as Fig.4 (b). The concave hull (Fig.4 (c)) of the projected point cloud is extracted to work as the main silhouette of the target object. The points making up the concave hull are conveniently stored in serial order for later processing. As can be seen from Fig.5 (b), which is an enlarged image of the red rectangle in Fig.5 (a), the points are stored in serial order.

A.3 Finger candidates generation

After the main silhouette of the target object is obtained, finger candidates need to be first generated to do further analysis. The specific procedures to generate finger candidates are as follows.

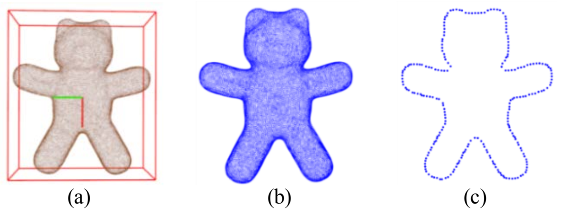
Fig.4 Abstraction of the object contour. (a) shows the point cloud in the OBB box. (b) shows the point cloud projected to the main plane of the OBB. (c) shows the object boundary acquired by abstracting the concave hull of (b).

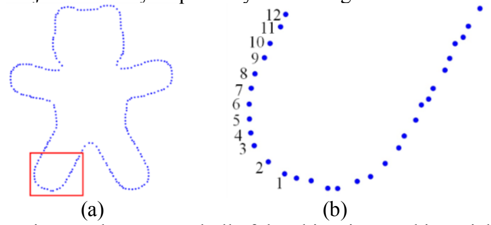
Fig.5 The points on the concave hull of the object is stored in serial order.

A.3.1 Step points searching

Employing the property that all the boundary points are in serial order, two adjacent boundary points can be connected to form a polygon. Fig.6 (a) shows the corresponding partial polygon for the boundary points in Fig.5 (b). Two adjacent points in Fig.6 (a) are connected by an orange line. As it is can be seen from Fig.6 (b), a step distance (r) is used to work as the search radius. The distance between point 1 to point n is defined as d_{1_n}, the distance between point 1 to point n+1 is defined as d_{1_n+1}. if d_{1_n} and d_{1_n+1} satisfy one of the equation (1), an intersection point can be found to work as the step point. If several intersection points are found at the same time, the point with the minimum serial number is chosen as the step point. In another word, if there are m intersection points, the $\min(n_1, n_2 \dots n_m)$ will be chosen as the step point. Fig.7 shows the result of step point searching. The blue and red points respectively stand for the boundary points and step points.

$$\begin{cases} d_{1\_n} < r < d_{1\_n+1} \\ d_{1\_n+1} < r < d_{1\_n} \end{cases} \quad (1)$$

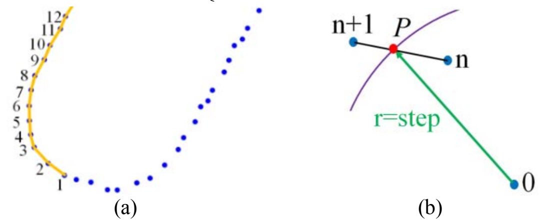

Fig.6 The step point searching process. (a) shows the orange polygon formed by connecting two adjacent boundary points. (b) shows how to compute the step points.

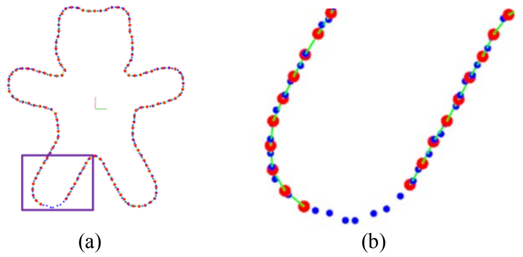
Fig.7 The result of step point searching. The blue points are the boundary points and the red points are the step points. (a) shows all the step points and (b) is an enlarged image of (a).

A.3.2 Obtain the finger candidates

After all the step points are obtained, the finger width is taken into consideration. In last section, if the step point is located on the line between point n (P_n) and point n+1 (P_{n+1}), a point cloud Ω can be constructed by adding P_n, P_{n+1} until the last point of the concave hull boundary in Fig.5 (a). w_f is used to describe the width of the finger, and the method of finding step points in Fig.6 can be employed to find the end point of the finger candidate. The step point works as the start point of the finger candidate and w_f works as the searching radius. An intersection point can be found on Ω by using the start point and the searching radius w_f. The line between the start point and the end point stands for a finger candidate. Fig.8 shows all the finger candidates. Every purple line in Fig.8 represents a finger candidate.

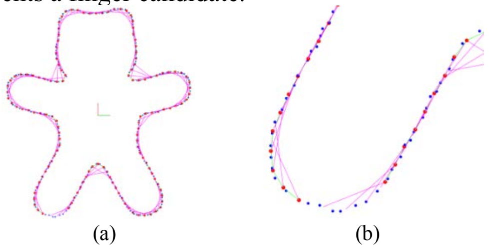
Fig.8 The result of finger candidate computation, every purple line stands for a finger candidate. (a) shows all the finger candidates, (b) is an enlarged image of (a).

A.3.3 Obtain the grasping direction for finger candidates

After the finger candidates are obtained, the first thing need to be done is to find the grasping direction. For every finger candidate, there are two possible grasping directions shown as Fig.9 (a). The blue line and the orange line respectively demonstrate the inside and outside grasping direction. Fig.9 (b) shows random grasping directions for all finger candidates, some lines are toward inside the object, some others are toward outside the object. How to find all the inside grasping direction?

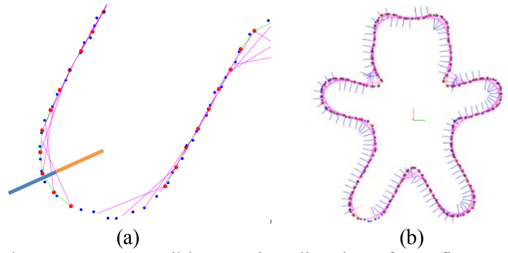
Fig.9 There are two possible grasping directions for a finger candidate. (a) shows two grasping direction (the orange line and the blue line) for a finger candidate. (b) shows all the random grasping direction for all the finger candidates.

In order to solve the problem of finding correct grasping direction, let's first look at how to judge whether a given point is inside or outside the contour of the object. First, the contour of the target object is used to construct a polygon (Φ). An effective way to find whether a given point is inside or outside a polygon is to cast many random rays from the given point to any direction. The intersects between the casting ray and the polygon are used to judge whether the given point is inside or outside the polygon. If the number of intersects is an odd number, the given point lies inside the polygon, otherwise, it lies outside the polygon. Fig.10 shows the idea of how to judge whether a give point is inside or outside of a polygon. The red point and the orange point are two given points. The green and the purple points are two random points. The line between the red point and the green point has two (even number) intersects with the polygon. The line between the orange point and the purple point has three (odd number) intersects with the polygon. Specifically, a given point (P_g) is first given, and then, a controlled number of random points (P_{rn} , $n = 1, 2...m$. m is the total number of the random points) are generated by system, A straight line (l_{g_r1}) can be constructed by connecting the given point (P_g) and the first random point (P_{r1}). l_Φ is used to represent one side of the polygon (Φ). The intersect point between l_{g_r1} and the l_Φ will be found. If the intersect point is on the polygon, it means there is a real intersect point. A for-loop is used to go through all the sides of the polygon (Φ) to find all the intersection points. The number of the intersects between l_{g_r1} and the polygon (Φ) is defined as n_1 . Then, the second line l_{g_r2} can be constructed by connecting another random point (P_{r2}) and the given point (P_g). The number of intersects between l_{g_r2} and the polygon (Φ) is defined as n_2 . The line between P_g and P_{rm} is defined as l_{g_rm} , the number of intersects between l_{g_rm} and the polygon (Φ) is defined as n_m . If all the numbers (n_1 , n_1 … n_m) are odd numbers (that is, $(n_1, n_2 ... n_m)\%2 = 1$), the given point is inside the polygon (Φ), otherwise it is outside the polygon.

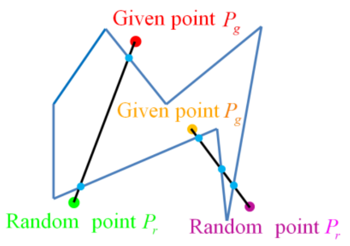

Fig.10 The method to judge whether a given point is inside or outside a polygon. If a given point is outside a polygon, the line between the given point and the random point has even number of intersects with the polygon. If a given point is inside a polygon, the line between the given point and the random point has odd number of intersects with the polygon.

After we known how to judge whether a given point is inside or outside of a polygon, we can use it to find the grasping direction for the first finger candidate. As can be seen from Fig.11 (a), a step value (δ) is used to do step searching along the middle vertical line of the first finger candidate. A pair of step points are set along the green arrow and the black arrow with the step of δ . Then, above method can be used to judges whether the two step points are inside or outside of the object contour. If the two step points are both inside the object boundary, then the algorithm continues to search along the direction of the green arrow and the black arrow. The step searching process stops until one step point is inside the object boundary and the other step point is outside the object boundary. The direction from the middle point of the first finger candidate to the step point inside the object boundary is used to work as the grasping direction.

After the grasping direction of the first finger candidate is obtained, a coordinate system can be established (seen as Fig.11 (b)). The middle point of the finger candidate works as the origin, the direction from the start point of the finger candidate to the end point of the finger candidate works as the X axis. The Z axis is vertical to the main plane. If Y_1 is the grasping direction, the grasping direction is the cross product of X and Z, that is, $Y_1 = X \times Z$. If Y_2 is the grasping direction, the grasping direction is the cross product of Z and X, that is, $Y_2 = Z \times X$. The grasping direction of other finger candidates can be worked out by using the same cross product. Fig.11 (c) shows inside grasping directions for all finger candidates.

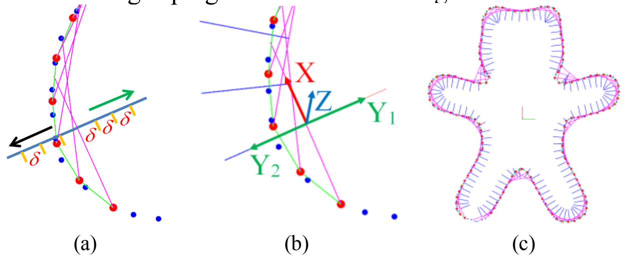

Fig.11 Grasping direction searching process. (a) shows how to find the grasping direction for the first finger candidate. (b) shows how to use the grasping direction of the first finger candidate to build a cross product, which can be used to work out the grasping direction of the rest finger candidates. (c) shows grasping direction for all the finger candidates.

B. Force closure analysis

After the grasping directions for all finger candidates are worked out, any two finger candidates can form a grasp candidate. Force closure analysis is used to do further analysis. Specifically, force balance and torque balance are used to do balance computation to choose the stable grasp candidates. Then, grasping range is considered to remove grasp candidates of which the distances between the two grasp sides are bigger than grasping range. Afterwards, the local geometry property of the grasp candidates is considered to remove those grasp candidates with big variance, which may lead to grasp failure. Then, the operability analysis is used to remove those grasp candidates of which the robot gripper may collide with the object when the robot tries to grasp it.

B.1 Force balance computation

After the grasping directions for all finger candidates are worked out, any two finger candidates can form a grasp candidate. For every grasp candidate, force balance computation is used to analyze the resultant force applied on the object. If the total number of the finger candidates is m , f_i ($i = 1, 2, ..., m$) is used to represent the i_{th} finger candidate and F_i is used to stand for the force applied on the object by f_i . If the force along the grasping direction for every finger candidate is a unite force, the angle between the two unite forces can represent the intensity of the resultant force. In Fig.12, the orange line and the red line respectively stand for the grasping direction for two example finger candidates (f_i and f_j). The angle (γ_{ij}) is used to describe the intensity of the resultant force of F_i and F_j . γ_{ij} is used to evaluate the stability of the grasp candidate consisting of f_i and f_j . If

i and j go from 1 to m, the result of force balance computation for every grasp candidate can be obtained (shown as Fig.13 (a)). Specifically, this figure shows all the resultant force. The red areas mean the maximum resultant force and the blue areas stand for the minim resultant force. Fig.13 (b) is the projected image of Fig.13 (a), we can clearly see that the resultant force is maximum when it satisfies $i = j$, that is the area between the two green parallel lines. The bigger the resultant force is, the more unstable the grasp is. The centers of the blue circles in Fig.13 (b) mean the minimum resultant force. At that point, the resultant force is almost zero, which means the grasp is the most stable.

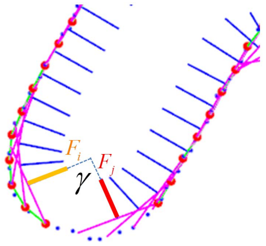

Fig. 12 Force balance computation, the orange line and the red line respectively stand for the grasping direction for the finger candidate f_i and f_j. The angle (γ_{ij}) is used to describe the intensity of the resultant force of F_i and F_j. The smaller γ_{ij} is, the more stable the grasp is.

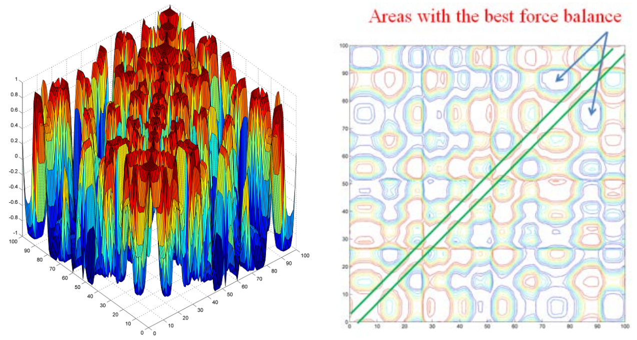

Fig.13 The result of force balance computation. (a) shows the result of force balance computation, the red areas mean the maximum resultant force and the blue areas stand for the minim resultant force. (b) is the projected image of (a).

B.2 Torque balance computation

After the above steps, the grasps candidates $g_{i,j}$ (consisting of f_i and f_j) satisfying the force balance requirement set by the system are chosen out. However, only use of force balance cannot make sure the grasp stability. Fig.14 shows an example grasp on the bear's head, which satisfies the force balance requirement. However, if the robot tries to grasp the bear using this grasp configuration, the bear would rotate around the green point, which may lead to grasp failure. Therefore, the torque of every grasp $g_{i,j}$ should be taken into consideration. $T_{i,j}$ is used to stand for the torque of a grasp candidate $g_{i,j}$. A function is used to represent the relation between $T_{i,j}$ and $g_{i,j}$, that is $T_{i,j} = f(g_{i,j})$. If $T_{i,j}$ is bigger than the threshold (T_s) set by the system, then the grasp $g_{i,j}$ is removed, otherwise, $g_{i,j}$ is kept. All the grasp candidates left are used to do following analysis.

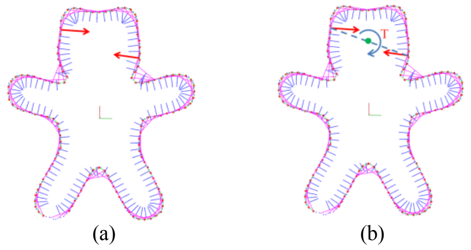

Fig.14 Torque balance analysis. (a) shows a possible grasp candidate satisfying the force balance requirement. (b) shows the torque balance analysis. Big torque may lead to big rotation of the object, which may result in grasping failure.

B.3 grasping range computation

After the force balance and torque balance computation, grasping range of robot hand should be considered. If the distance $d_{i,j}$ between the two grasp sides of a grasp candidate $g_{i,j}$ is bigger than the grasp range, the robot cannot grasp the object. Therefore, if the distance $d_{i,j}$ of the two grasp sides of every grasp $g_{i,j}$ is smaller than the grasping range, the grasp is remained, otherwise it is removed.

B.4 Variance analysis

After finishing all steps mentioned above, the grasps left satisfy the force requirement, torque requirement and grasping range requirement. However, the local geometry property of grasp candidates is not yet considered. Fig.15 shows a grasp candidate, right up and right down are the enlarged images of the two grasp sides (the green points). The distance between one green point to the purple line is defined as d_i, $0 < i \le n$, n is the total number of the green points. All the distances are added together to get the variance v of the grasp, $v = \sum_{i=1}^{i=n} d_i$. The bigger the variance is, the larger possibility of grasp failure is. If the variance is smaller than the threshold set by the system, the grasp is saved, otherwise, it is removed.

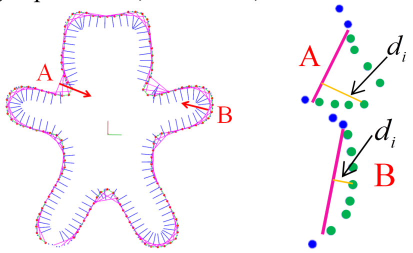

Fig.15 Variance analysis. The left image shows a possible grasp candidate. The right images are enlarged images of the two grasp sides. Variance of the points of the two grasp sides is used to evaluate the grapping quality.

B.5 Operability analysis

Operability in this section means whether the grasps found in above section can be executed or not. Not all grasp candidates obtained by above steps can be executed successfully, Fig.16 shows an example, the two purples lines stand for a grasp candidate, the two orange lines represent the biggest open width of the robot hand. For the example grasp candidate, the robot finger will collide with the bear at the red circle. Therefore, it is necessary to analyze the operability of grasp candidates. In order to simplify computation, a local coordinate system is established and the concave hull boundary of the object is transferred into the local coordinate system. The biggest open width of the robot hand is defined as w_o and the hand width is defined as w_h, the distance between the two grasp sides is defined as d. If the points on the concave hull boundary satisfy equation (2), then it means there is collision when the robot try to execute this grasp, otherwise, there is no collision. Using the above steps repeatedly, we can find all the grasp candidates satisfying the operability requirement.

$$\begin{cases} -0.5 * w_h \le x \le 0.5 * w_h \\ -w_o \le y \le -0.5 * d \,||\, 0.5 * d \le y \le w_o \end{cases} \tag{2}$$

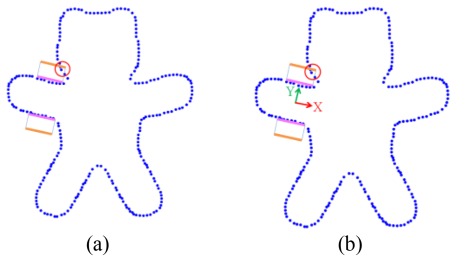

Fig.16 Grasping operability analysis. (a) shows an example grasp candidate that the robot finger will collide with the object. (b) shows the local coordinate system which is used to do point cloud transformation.

C. Gravity analysis

When an object is under manipulation after it is grasped, its gravity inevitably brings instability to the grasp. How to take the gravity of the object into consideration is a key problem which can decide whether a grasp action is reliable or not. In this paper, we propose to use the distance between the gravity center and the grasping line (between the two grasping points) to evaluate the grasp candidates. For example, if the robot already grasped the teddy bear and the robot wants to move the bear in the air. At this moment, the gravity needs to be considered to prevent the object falling from the robot gripper. Fig.17 (a) shows an example grasp, specifically, the two purple lines stand for the grasp and the red point represents the gravity center. If the robot grasps the teddy bear and moves in the air, the object may rotate around the red line, the corresponding torque is defined as T, T can lead to instability which may result in grasp failure. The distance (d) between the gravity center and the grasping line is used to evaluate the grasp quality after the object is grasped. The shorter the distance is, the smaller the torque is. The smaller the torque is, the more stable the grasp is. Fig.17 (b) shows the result of gravity analysis, the grasp with the smallest gravity torque is chosen as the final grasp (shown as the two bold red lines in Fig.17 (b)).

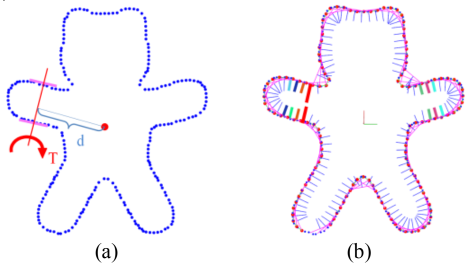

Fig.17 Gravity analysis. (a) the distance (d) between the gravity center and the grasping line is used to evaluate the effect of the object's gravity. (b) the final grasp obtained.

IV. Simulation

In order to test the algorithm, various objects are chosen to conduct simulation to determine the grasping performance. The simulation system consists of Robot Operating System (ROS), Gazebo (a Standalone Open Dynamics Engine based simulator) and MoveIt! (a state of art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation). In the Gazebo simulation environment, A Lacquey under-actuated gripper and an Asus Pro Live sensor are installed at the end of the Universal arm (UR5). The Asus Pro Live sensor is used to acquire the point cloud of the target object in the simulation environment. The Lacquey under-actuated gripper is used to execute the final grasp found by the algorithm.

Five objects with different geometry shapes are used to do simulations. These objects are a teddy bear, an electric drill, a pistol, a spray bottle and a pan. Fig.18 shows the simulation results. The first column shows the simulation setup. The second column shows the OBB box to process the point cloud. The third column shows the finger candidates and the grasping directions. The fourth column shows the final grasp found by the algorithm. The two bold red lines stand for the final grasp. The fifth column shows the grasp area on the point cloud of the target object. The sixth column shows the grasp execution. The algorithm can find good grasp for all these tested objects, which proved the effectiveness of this algorithm.

V. Experiment

The experiments are conducted using a six degrees of freedom Universal arm UR5 and an underactuated Lacquey Fetch gripper. An Xtion pro live sensor is installed on the tool tip of the robot. The whole experiment setup can be seen in Fig.19. Five objects with different geometry shapes are used to do experiment. These objects include an electric drill, a spray bottle, a hammer, a pan and a juice box. Fig.20 shows some snapshots of the grasping process of these objects. The first column is the initial state of the robot and the target objects. The second column is result of grasping computation, the two red lines stand for the final grasp found by this algorithm. The third column shows the grasp area on the point cloud of the target object. The fourth column shows the gripper arriving at grasping point. The fifth column shows objects grasped by the gripper.

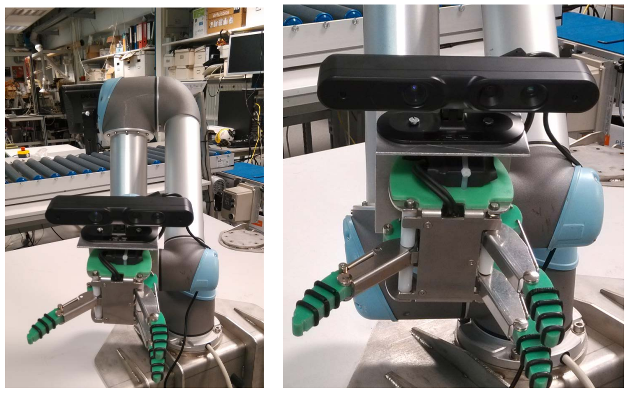

Fig. 19 Experiment setup. A Lacquey under-actuated gripper and an Asus Pro Live sensor are installed at the end of the Universal arm (UR5).

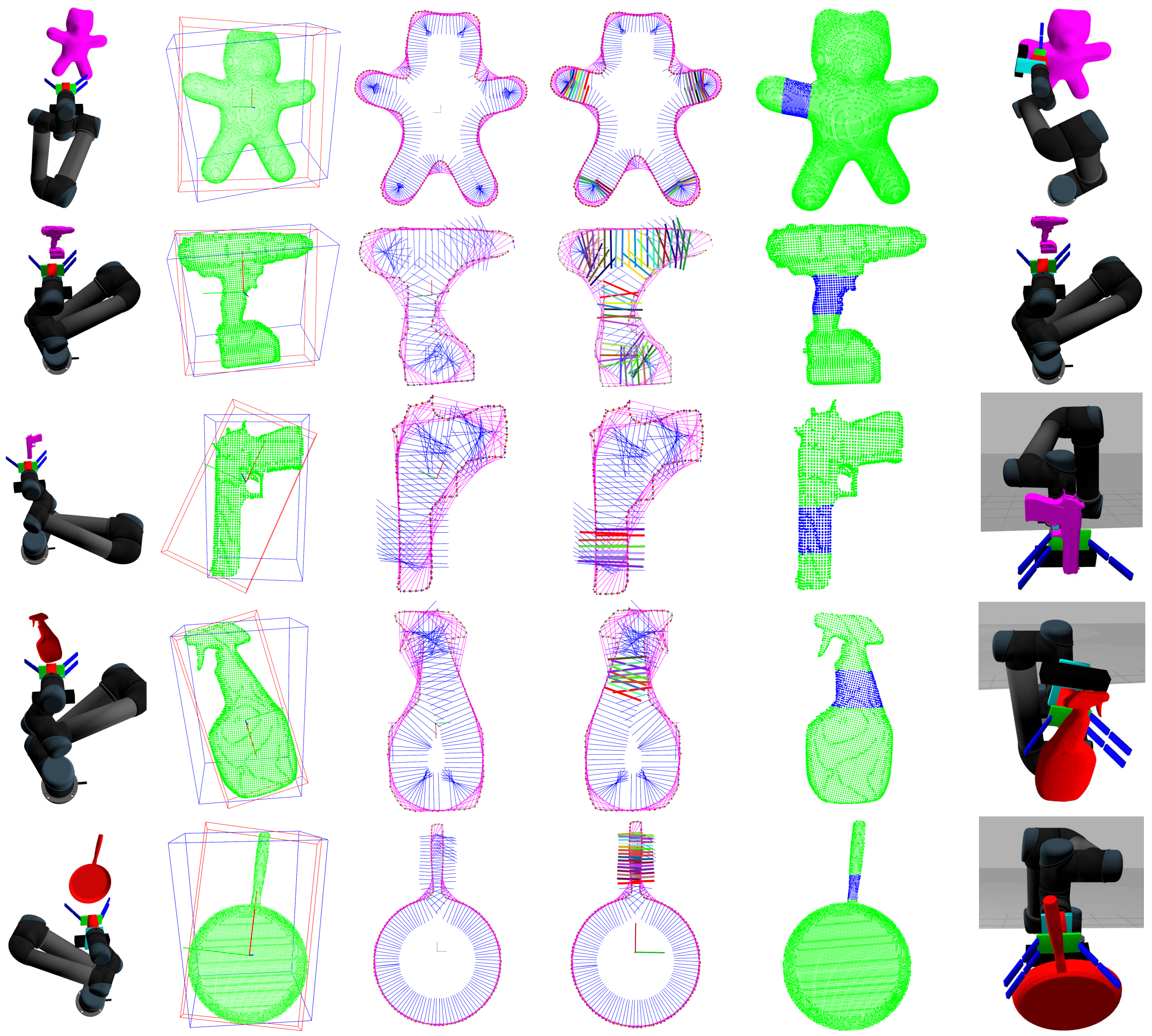

Fig. 18 Simulation results.

From this experiment, authors can safely draw three conclusions. The first is that this grasping algorithm is very fast. Grasping computation of the these objects can finish within one second. The second one is that this grasping algorithm is reliable. All the grasps found for these objects have good force balance and torque balance, as well as the gravity optimization. The third is that this grasping algorithm has a good tolerance. Point clouds of the electric drill, the spray bottle and the pan missed a lot of pixels because of the restriction of the Asus Xtion pro live sensor. However, the grasping algorithm can still work out good grasps for the target objects. The experiment also proved the effectiveness of our algorithm

VI. Comparison

As mentioned in the first part of motivation, inspiration of this paper comes from [1]. Let's look at the outcomes of [1] and our algorithm. Fig.21 shows the comparison between [1] and our algorithm. We made several improvements over [1]. The first one is [1] did not tell how they get the boundary of the object. We propose to use Oriented Bounding Box to obtain the boundary of the object, which is proved to be efficient in our experiments. The second is [1] did not consider the geometry property of the gripper. Actually, the grasps found by [1] in the red circle are impossible to be executed by grippers like the PR2 gripper, because [1] did not consider the geometry property of the gripper. [1] just uses one single point to represent the finger. On the contrary, we consider the geometry shape of the finger from the beginning of our algorithm. The third is [1] did not consider gravity of the object, which plays an important role in object grasping. On the contrary, we choose the nearest grasp to the gravity center to work as the final grasp. This grasp can not only make sure the grasp can be executed successfully, but also ensure the grasp quality during the manipulation of the object after it is grasped. To sum up, our algorithm combines the advantage of caging grasping and force closure grasping, which is much more practical for flat object grasping than [1].

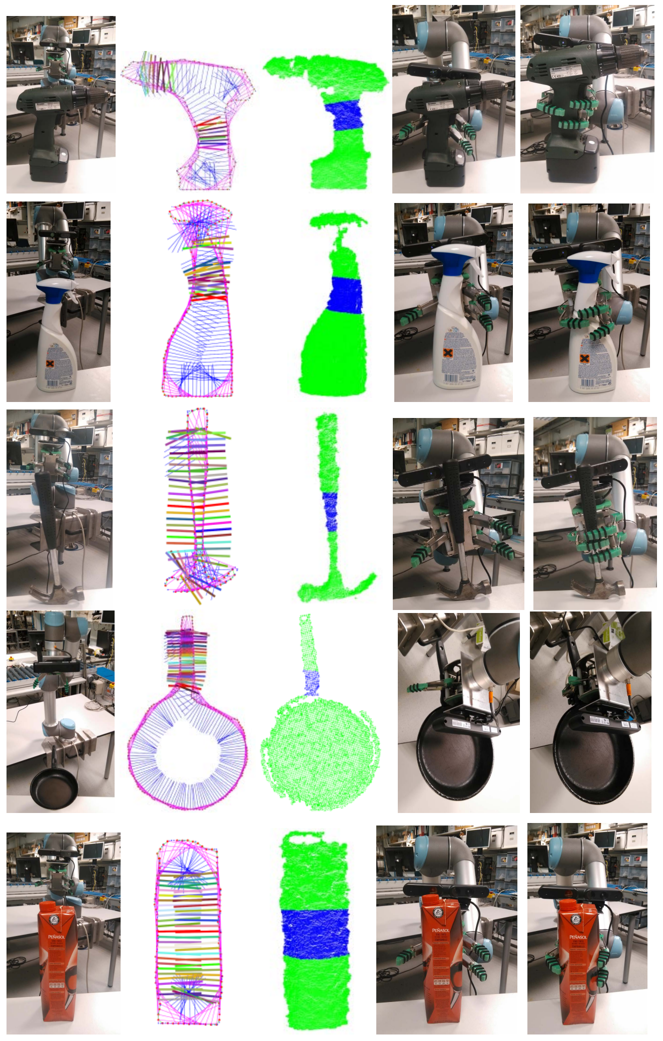

Fig. 20 Snapshots from the experiments: Fist column is the initial state of the robot and the target objects. Second column is the result of grasping computation. Third column shows the grasp area on the point cloud of the target object. Fourth column shows the gripper arriving at grasping point. Fifth column shows objects grasped by gripper.

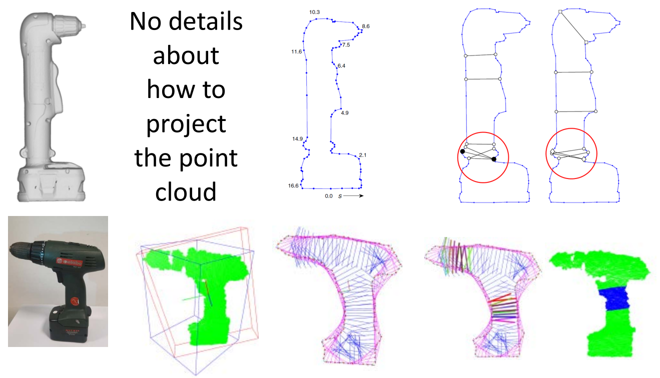

Fig. 21 Comparison between [1] and our algorithm. The top is the algorithm form [1]. The bottom is our algorithm. Three improvements are made. The first is that [1] did not give details about how to project the point cloud. We use to OBB to find main plane to project point cloud. The second is that some grasps found by [1] are not practical for two finger gripper because [1] did not consider the geometry property of the robot hand. We consider the geometry properry of robot hand like the hand width, grasp range and the local geometry of every finger candidate on the boundary. The third is that [1] did not consider gravity of the object. Our algrithm consider graveity to make it more reliable.

VII. Conclusion

In this paper, a novel grasping algorithm is presented for flat object grasping by combining the merits of caging grasping and force balance grasping. The idea of caging points is borrowed to generating grasp candidates. After that, force balance computation is carried out to find out suitable grasps by considering the gripper geometry properties, for example, the grasping range and the hand width. Gravity of the target object is also considered to ensure the grasping quality during the manipulation of the object after it is grasped. This algorithm can quickly work out the best grasp with good force balance and torque balance. In order to prove the validity of our grasping algorithm, several objects with different geometry shapes are used to do simulations and experiments. And good results are obtained.

ACKNOWLEDGEMENT

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.

References

[1] T. Allen, J. Burdick, and E. Rimon, "Two-fingered caging of polygons via contact-space graph search, " in ICRA, pp. 4183-4189, 2012.

[2] Jianhua Su, Hong Qiao, Zhicai Ou, Zhiyong Liu, "Vision-Based Caging Grasps of Polyhedron-Like Workpieces With a Binary Industrial Gripper," IEEE Transactions on Automation Science and Engineering, vol. 12, no. 3, pp. 1033–1046, 2015.

[3] E. Rimon and A. Blake, "Caging 2D bodies by 1-parameter two-fingered gripping systems," in ICRA, vol. 2, pp. 1458-1464, 1996.

[4] Elon Rimon, Andrew Blake, "Caging Planar Bodies by One-Parameter by Two-Fingered Gripping Systems," The Int. J. of Robotics Research, vol.18, no.3, pp. 299-318, 1999.

[5] A. Sudsang and J. Ponce, "On grasping and manipulating polygonal objects with disc-shaped robots in the plane," in ICRA, vol. 3, pp. 2740-2746, 1998.

[6] P. Pipattanasomporn and A. Sudsang, "Two-finger caging of nonconvex polytopes," IEEE Trans. Robot., vol. 27, no. 2, pp. 324-333, 2011.

[7] M. Vahedi and A. van der Stappen, "Caging polygons with two and three fingers," Int. J. Robot. Res., vol. 27, no. 11-12, pp. 1308-1324, 2008.

[8] G. Pereira, M. Campos, and V. Kumar, "Decentralized algorithms for multi-robot manipulation via caging," Int. J. Robot. Res., vol. 23, no. 7-8, pp. 783-795, 2004.

[9] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in ICRA, pp. 1471-1476, 2008.

[10] Yanyan Dai, Yoon-Gu Kim, Dong-Ha Lee, and SukGyu Lee, "Symmetric caging formation for convex polygon object transportation by multiple mobile robots," in IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 595-600, 2015.

[11] H. Kruger, E. Rimon, and A.F. van der Stappen, "Local force closure," in ICRA, pp. 4176-4182, 2012.

[12] J. Li and J. Xiao, "Progressive generation of force-closure grasps for an n-section continuum manipulator," in ICRA, pp. 4016-4022, 2013.

[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, pp. 109–120, 2010.

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

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