Advanced Robotics

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tadr20

Robust grasp detection with incomplete point cloud and complex background

Xixun Wang, Sajid Nisar & Fumitoshi Matsuno

To cite this article: Xixun Wang, Sajid Nisar & Fumitoshi Matsuno (2021): Robust grasp detection with incomplete point cloud and complex background, Advanced Robotics, DOI: 10.1080/01691864.2021.1897674

To link to this article: https://doi.org/10.1080/01691864.2021.1897674

ADVANCED ROBOTICS
https://doi.org/10.1080/01691864.2021.1897674

FULL PAPER

Robust grasp detection with incomplete point cloud and complex background

Xixun Wang[a], Sajid Nisar[b] and Fumitoshi Matsuno[a]

[a]Department of Mechanical Engineering and Science, Kyoto University, Kyoto, Japan; [b]Department of Mechanical and Electrical Systems Engineering, Kyoto University of Advanced Science, Kyoto, Japan

ABSTRACT

Point cloud information is a convenient means to accomplish grasp detection in autonomous robotic grasping. However, the performance of various state-of-the-art grasp detection methods that use point cloud decreases significantly when the available point cloud information is incomplete and the background is complex, i.e. heterogeneous. To solve this problem, we propose a robust grasp detection method that demonstrates higher performance, especially with incomplete point clouds and complex backgrounds. We introduce a novel technique named '*visible point-cloud*' – generated using the point cloud and pose (position and orientation) information of the sensor(s) – that helps to eliminate unsafe grasp candidates quickly and efficiently. The remaining grasp candidates are then classified and the best candidate is determined using a cost function. The effectiveness of the proposed method is shown experimentally using a 6-DoF robot arm equipped with a two-finger gripper with three different background settings: (a) steps, (b) pillars, and (c) table top. The results show that the proposed method is 1.20 times faster and has a 20% higher grasp success rate than a state-of-the-art method for a single point cloud camera. The results demonstrate that the proposed method significantly improves the performance of autonomous grasping even with incomplete point cloud and is robust for different backgrounds.

ARTICLE HISTORY
Received 6 August 2020
Revised 18 January 2021
Accepted 14 February 2021

KEYWORDS
Autonomous grasping; perception; point cloud; occlusion; grasp visibility

1. Introduction

The action of autonomous robotic grasping can be divided into two major steps: (a) perception, and (b) planning. In the perception step, the goal is to find a suitable grasp configuration(s) for the object(s) of interest in a given scene. The strategy to move the robotic arm and grasp the desired object is determined in the planning step. The suitability of a grasping configuration typically means the chances of achieving a successful grasp, i.e. the object is picked up by the robot gripper without a failure.

To this effect, the majority of traditional grasping methods use a model-based approach, which involves estimation of objects' pose (position and orientation) using point cloud (or RGB-D information) [1,2]. The model-based approach needs predetermined grasp configuration(s) [3,4] and 6-D pose information of the object(s). This information is typically obtained using 3-D CAD (computer-aided design) models of the objects. The real limitation of this approach is the requirement of prior knowledge about objects' 3-D CAD models and as well the assignment of predetermined grasp configuration(s).

On the other hand, a recent approach – called model-free approach – tries to accomplish the perception task without knowing the objects' 3-D CAD models and predetermined grasp configurations. Several methods under this approach have made use of machine learning techniques, such as Deep Learning [5–8] and Reinforcement Learning [9]. However, these methods – as is the case with most deep neural network applications – require a lot of computational resources, setup time, and a big curated data set for training. Additionally, the robustness of grasp detection using these approaches may differ from scene to scene depending on the heterogeneity of the training data.

To avoid the issues faced by model-based approach and model-free methods discussed above, several model-free perception methods [10–15] have been proposed recently. This new model-free approach makes use of the point cloud (or a depth image) information, instead of the machine learning techniques, to assign the grasp candidates. It then determines the best possible grasp configuration(s) using simple classifiers, such as light neural networks like the LeNet, or Support Vector Machines (SVMs). The main advantage of this new approach is that

CONTACT Xixun Wang wang.xun.33a@st.kyoto-u.ac.jp; ojijin93@gmail.com Department of Mechanical Engineering and Science, Kyoto University, Kyoto 615-8540, Japan

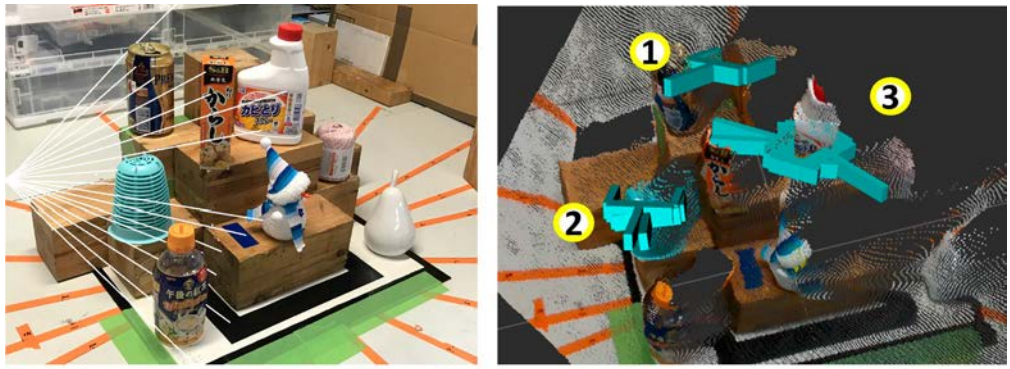


Figure 1. A state-of-the-art existing method (reference method [11]) when applied to a test scene (a) with a single depth-camera results in various incorrect but high-scored grasps, as shown by label ①, ②, and ③ in (b). (a) An example test scene. (b) Incorrect but high-scored grasps.

it does not need machine learning in the first step, i.e. grasp assignments.

However, the methods based on this new model-free approach so far suffer from another issue that is related to the success of grasp in scenes with multiple objects and incomplete point cloud information. One reason for this is that these methods do not consider the occlusion of the objects and, therefore, often result in incorrect grasp assignments. For example, when a state-of-the-art method based on this model-free approach, proposed by Pas et al. [11], is applied using a single depth-camera to a scene with heterogeneous background (Figure 1(a)), it often assigned high scores to partially-occluded (i.e. potentially unsafe) grasp candidates. ① in Figure 1(b) indicates an incorrect grasp because most of the wooden block is occluded by other objects in the scene, and those objects may cause a grasp failure by blocking the gripper-fingers. Similarly, ② shows incorrect grasps where the thickness of the object to be grasped is larger than the gripper's grasp-width. Some recent methods based on the model-free approach, for example [12–15], show an even higher grasp success rate than Pas et al.'s method [11]. However, they also did not take the occlusion of objects into account and are prone to similar errors mentioned above. In addition, their suitability for grasp detection of multiple objects with complex (i.e. heterogeneous) backgrounds is yet to be determined.

To achieve a successful and safer grasp for scenes with a complex background, we believe it is important to consider the object's visibility/occlusion, which so far has not been considered in the prior research. To elaborate on the importance of objects' visibility (or occlusion), the grasp handle ③ in Figure 1(b) shows a potentially unsafe grasp configuration because the right-limb of the gripper has to grasp the object from invisible (occluded) region. This could, for example, topple other objects present in the occluded region and/or result in gripper jamming. To solve this problem, we propose a grasp detection method that completely solves the problem of having incorrect but high-scored grasp candidates. The originality of the proposed method is that it ignores all the possible grasp locations that are occluded from the sensor. Such partially-visible object locations are considered risky and, thus, unsafe for the grasping action.

To identify the occluded portion of the objects, we define the region between the objects' point cloud and the camera position as the *visible region* (Figure 2(a)). Similarly, any region behind the point cloud is defined as the *occluded region*. Using the visible region and the gripper's size information, we define a region of interest which is a subset of the visible region close to the objects, as shown in Figure 2(b). Based on the point cloud information from this *visible region of interest*, the proposed method generates a new but highly dense point cloud, which we name as the *visible point cloud (v-cloud)*. With the help of v-cloud, we can efficiently filter out all the grasp handles that lie outside of the 'visible region of interest' and, therefore, grasp the objects only from their visible portion. The remaining grasp candidates are then classified using a light neural network and are assigned scores based on their grasp suitability determined using a cost function which takes into account the visibility of the grasp location, state of the robot, and the scores assigned by the neural network classifier. Finally, the best grasp candidate, based on the highest assigned scores, is selected. As the proposed method removes all the potentially incorrect (i.e. unsafe) grasps, such as ① ② ③ in Figure 1, at the beginning of the perception phase, it leads to a higher grasp success rate even with complex backgrounds and incomplete point cloud information. The proposed method additionally reduces the overall

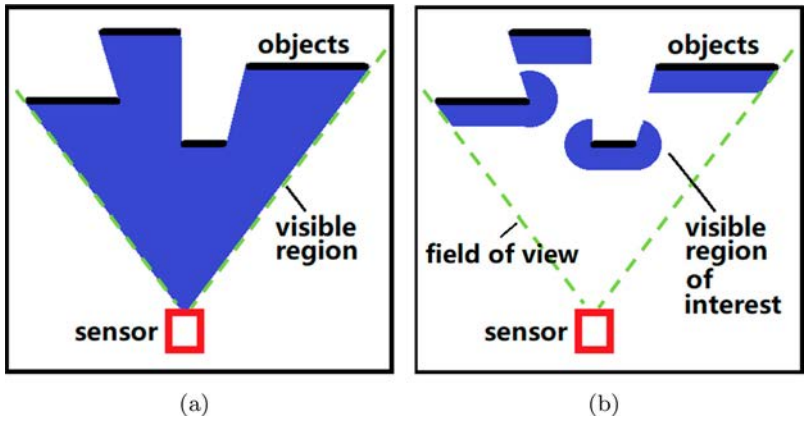


Figure 2. The visible region required for perception illustrated in 2-D: (a) the visible region generated by typical methods like ray-traversal algorithm [16], (b) the *visible region of interest* proposed in this research.

computational time as all of the incorrect grasp candidates are removed before applying the neural network classifier.

The effectiveness of the proposed method is demonstrated by performing grasp experiments (Section 4) and the performance is compared against a reference method [11]. The results show that the proposed method has a higher success rate and a slightly faster average calculation time than the reference method [11]. The experiments are also conducted with incomplete point cloud obtained through (i) a single, and then (ii) using two depth-cameras, with three different backgrounds: (a) steps, (b) pillars, and (c) table top. The results also show that the proposed method, contrary to the reference method, is robust enough to maintain a similar success rate across different scenes and levels of the point cloud (in)completeness.

The originality and main contributions of this research are:

- A new grasp detection method based on a novel concept of 'visible point cloud' is proposed that offers a higher grasp success rate and reduced computational time.
- The proposed method increases the grasp safety by considering the occlusion of the objects during the perception phase.
- The 'visible point cloud' technique is computationally inexpensive and easy to adapt to any grasp detection methods for filtering out unfeasible grasp candidates as it does not use machine learning.
- The robustness of the proposed method has been proved experimentally with incomplete point cloud information and complex backgrounds.

To explain the proposed method and its comparison to the state-of-the-art, this paper has been divided into five sections. Section 2 describes the algorithmic structure and generation of the v-cloud. Section 3 details the classification and evaluation using the v-cloud. Section 4 presents the experimental results. Finally, the conclusion and discussion about the future work are given in Section 5.

2. Proposed method

2.1. Algorithm

The algorithm of the proposed method consists of nine steps, as depicted in Figure 3.

(I) *Acquire the point-cloud data:* In the first step, the point cloud information of the physical objects present in the scene is obtained using one or multiple depth-cameras (sensors).

(II) *Generate the v-cloud:* Using the sensor(s) pose (position and orientation) and the point cloud information, a highly dense point cloud is generated. The purpose of this newly generated point cloud (called v-cloud) is to encompass the 'visible region of interest' between the objects and the camera view point. The 'visible region of interest' and the method of v-cloud generation are explained in Sections 2.2 and 2.3, respectively.

(III) *Map point-cloud and v-cloud:* In this step, both the point cloud and v-cloud are mapped to a single reference grid using the Octo-tree [17] data structure. Octo-tree provides convenient means to remove the sensing noise when multiple cameras are used. For this purpose, first, the Octo-tree discretizes the workspace into a voxel grid of resolution r, and each voxel in the Octo-tree is assigned an initial probability of 0.5. Then, the sensor(s) point cloud (Step I.) and the v-cloud

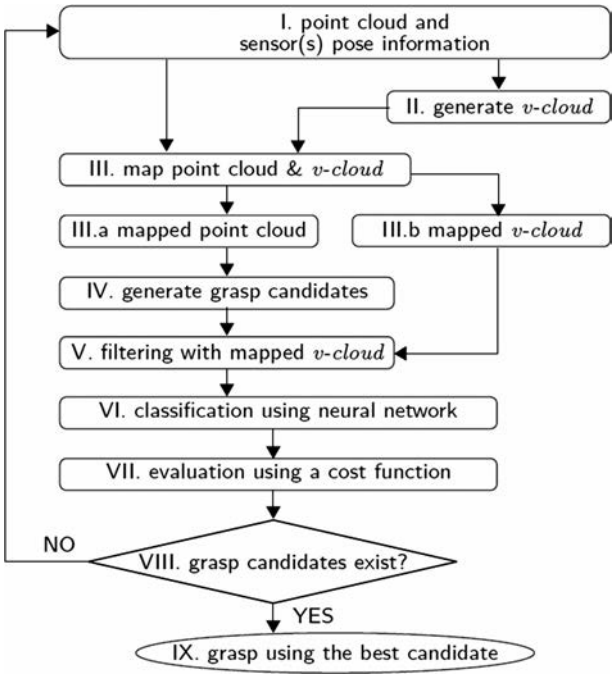


Figure 3. The algorithmic structure of the proposed method: (I) The point cloud and sensor (camera) pose information are obtained, (II) a visible point cloud (v-cloud) is generated, (III) the point cloud and v-cloud are then mapped using the Octomap technique, (IV) the grasp candidates are generated, (V) all the unsafe grasp candidates are removed, (VI) neural network is applied on the remaining candidates to classify them, (VII) a cost function is applied to assign scores, and (IX) the grasp action is performed using the best grasp with the highest assigned-score. If there remains no suitable grasp after (VII), the sensor pose is changed (VIII) and the perception task is started again.

(Step II.) are inserted into the voxels of the Octo-tree. Each point of the sensor point cloud is assigned a probability of 1. Each v-cloud point is assigned a probability of 0. Then each voxel in the Octo-tree updates its probability using the probabilistic sensor fusion technique used in [17], which takes into account the probabilities of the incoming v-cloud and sensor point cloud. Finally, voxels with $0.5 < probability \leq 1$ are treated as the mapped point-cloud (III.a) and with $0 \leq probability < 0.5$ as the mapped v-cloud (III.b).

(IV) *Generate the grasp candidates:* In this step, we generate N grasp candidates (i.e. 6-DoF hand poses) in the reference frame – known as the Darboux frame [18] – by sampling uniformly at random N points of the mapped point cloud using the method described in [11]. Each grasp candidate is assigned to its corresponding mapped point cloud on an object. The size of each grasp handle is kept equal to the size of the robotic gripper to be used.

(V) *Removal of potentially unsafe grasp candidates:* In this step, the grasp candidates that are not completely covered by the mapped v-cloud are removed as they are considered unsafe to carry out a successful grasp due to the lack of complete visibility (i.e., presence of occluded regions). The detail of the filtering process is given in Section 3.1 and Section 3.2.

(VI) *Classification of the grasp candidates:* The 2-D grasp images of the remaining grasp candidates from the preceding step are generated. We then classify these grasp images using a pre-trained LeNet [19] neural network. LeNet returns a score between range [-1,1], to show how closely a grasp image resembles to the prior training samples. The grasp candidates that are scored by the classifier equal to or below zero are removed.

(VII) *Determining the best grasp:* The classified grasp candidates are then sorted using a cost function. The cost function considers the visibility of the grasp handles and the ability of the robotic arm to reach a particular configuration using the manipulator's inverse kinematics (IK). Details of the cost function are given in Section 3.3.

(VIII) *Check grasp availability:* After assigning the scores, we check whether any grasp candidates are available. If yes, the algorithm moves to the next step. Otherwise, the camera pose is changed and the process restarts from step I.

(IX) *Perform the grasping action:* Among the available grasp candidates, the best one is selected based on the highest assigned scores. Once the best grasp is determined, the robot arm is commanded to grasp the object.

To further elaborate the steps explained above, we now describe the key concepts that define the structure of the proposed method and ensure its effectiveness in solving the problems faced by the state of the art methods.

2.2. The 'visible region of interest'

As mentioned earlier, we propose a novel concept called v-cloud to efficiently filter out the potentially incorrect and unsafe grasp candidates in an efficient manner. To generate the v-cloud, we first define a region of interest in the visible region. The concept of *visible region* has long been used for robot navigation and exploration. A typical technique to generate the visible region is the ray-traversal algorithm [16]. A well-known method based on the ray-traversal algorithm is the OctoMap [17]. A 2-D example of the visible region using OctoMap methods is shown in Figure 2(a).

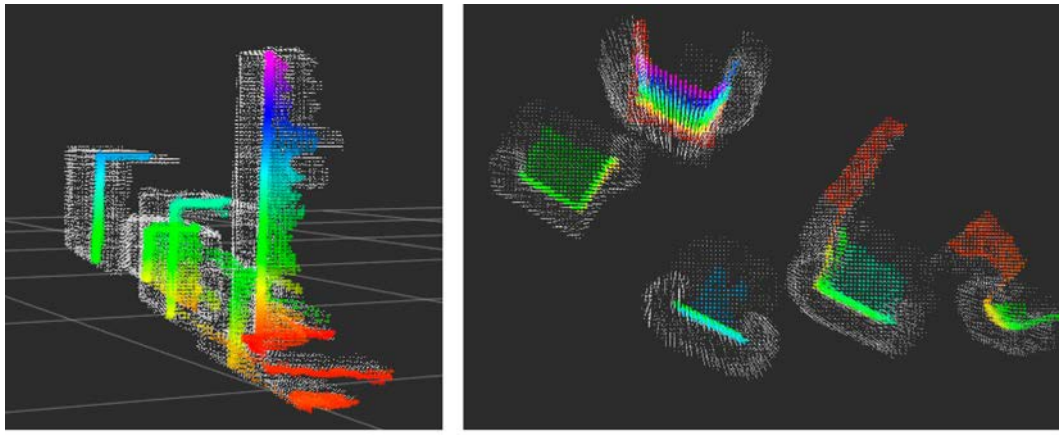


Figure 4. An example v-cloud in 3-D: White points represent the v-cloud (i.e. 'visible region of interest'). The dark color indicates all regions outside of the 'visible region of interest'. The colored points represent the objects' geometry. (a) Side view. (b) Top view.

However, OctoMap is not a suitable choice for grasp detection applications in our opinion. To determine the best grasp configuration, we are mostly concerned about the visible region which is very close to the objects of interest. The OctoMap method, however, considers all the visible regions between the objects and the sensor and, therefore, generates a lot of data in each frame which is not important to determine a good grasp configuration. This makes the OctoMap technique less efficient (took up to 1.2 seconds when tested with a 3.8 GHz Intel Core i7-7700HQ CPU and a leaf-node resolution of 0.003 m) for real-time grasp detection.

To avoid searching in the whole visible region, we define a visible region of interest which is a subset of the total visible region. Figure 2(b) illustrates the concept of the visible region of interest. The thickness and shape (around the objects' corners/edges) of the visible region of interest is defined such that the robot gripper can be fully covered in it when performing a grasping action. The advantage of defining this region is that we reduce the amount of data that needs to be processed in each frame significantly without losing the critical information required to determine the best grasp configuration(s).

2.3. The visible point cloud (v-cloud) and its generation

To represent the visible region of interest (explained in the previous section) in a suitable way for the perception task, we create a new but highly dense point cloud. As this point cloud represents the visible region of interest, we name it as the visible point cloud (v-cloud). The generation of v-cloud in 2-D is shown in Figure 5. Let us define $O_c - X_c Y_c Z_c$ as a coordinate system fixed on the camera (sensor), l_i as a line corresponding to a ray from the sensor, O_i as a surface, or an edge, point on l_i, and Σ_i as a coordinate frame $O_i - X_i Y_i Z_i$ with its origin on O_i and Z_i axis aligned with l_i, as shown in Figure 5. Green points denote the cloud points on the object's surface while the yellow points denote points that lie on the object's corners or sharp edges. The v-cloud is generated using information from green and yellow points such that its shape is similar to that of the visible region of interest. As Figure 5 shows, several short lines from the green points are generated that carry blue points. Similarly using the yellow points, a set of blue points with semi-circular shape is created. The v-cloud comprises all the blue points.

An example creation of the v-cloud in 3-D is shown in Figure 4. The 'visible region of interest' (v-cloud) is

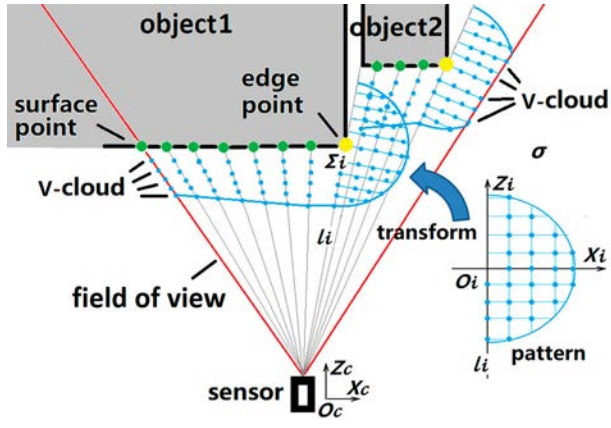


Figure 5. V-cloud generation: The green and yellow points represent the surface and edge points of an object. The v-cloud (blue points) is generated by transforming different patterns, such as the straight-lines or the half-circle, to the position of the object point cloud (green and yellow points), as shown in the figure.

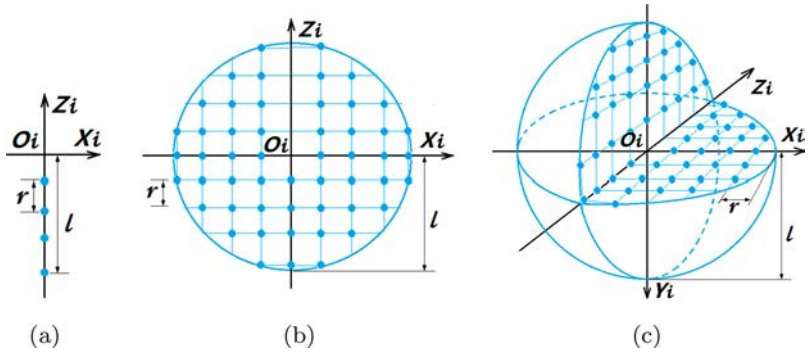


Figure 6. The three basic shape-patterns for generating the v-cloud: (a) short-line type, (b) circle type, and (c) sphere type. The resolution *r* of the v-cloud represents the distance between the two adjacent blue points. The thickness *l* is the length in the case of (a) or the radius of the circle in case of (b) and (c). There are no v-cloud points on origin and *z*-axis as they both lie outside of the 'visible region of interest'.

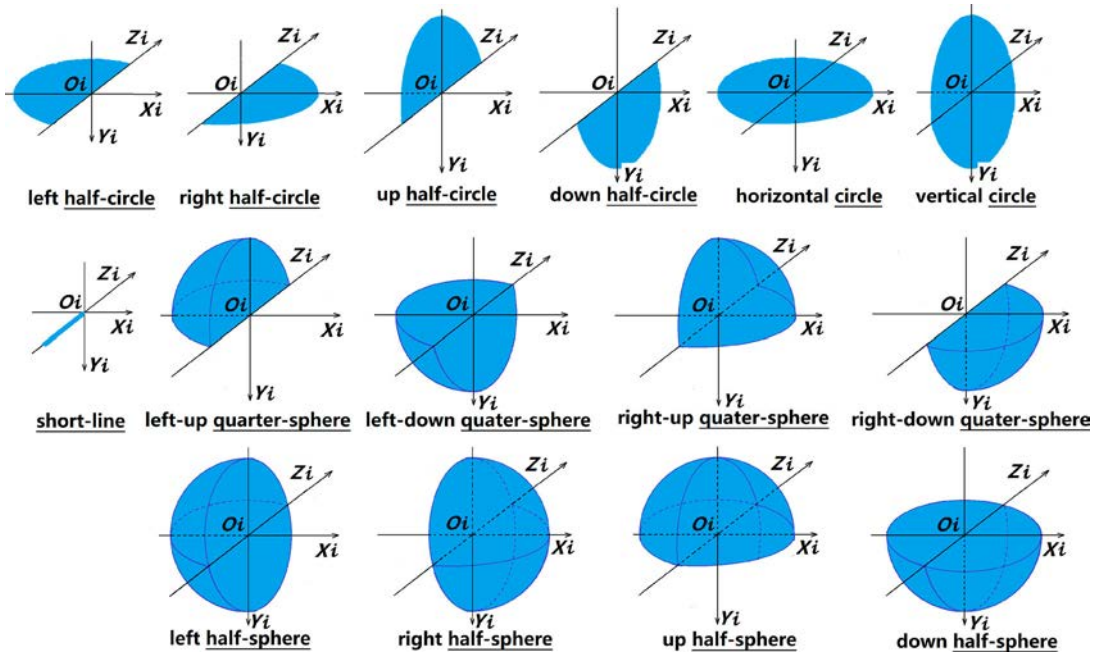


Figure 7. Fifteen different types of cloud-point patterns are used to generate the v-cloud around objects (*e.g.* left edge, right edge, left corner, right corner, and so on). The blue color represents the v-cloud shapes. All of these shapes originate from the three basic shape-patterns shown in Figure 6.

indicated by the white points and the objects are shown by the colored points. The black color represents all the remaining areas.

2.4. Patterns for the v-cloud

The process of creating v-cloud corresponding to the green points is straight-forward, as it uses the short-line pattern shown in Figure 6(a). However, the generation of v-cloud around edge points (yellow points) needs further consideration. As the visible region on the edges has (in most cases) a semi-circular shape, we create the v-cloud in that area by projecting semi-circular shapes pivoting at the edge-points, as shown in Figure 5.

The point cloud from the sensor can have 15 different types of points representing different locations on an object (details of each type are given in Section 2.5). To generate the v-cloud to cover the *visible region of interest*, we use fifteen different shape-patterns as shown in Figure 7; one for each type of the object points in the 3-D space. Using these patterns is not only efficient in terms of calculation time, but it also simplifies the process of v-cloud generation around the corners and edges of the objects. All the patterns originate from three basic shapes, shown in Figure 6. The short-line v-cloud is a set of 1-D points on a line, shown in Figure 6(a). The line starts at the origin (0,0,0) which is always placed at an object point (yellow or green in Figure 5). No v-cloud

point is generated at the origin itself because the objects' point cloud lies outside of the '*visible region of interest*'. Similarly, the z-axis does not contain any v-cloud points because it also lies in the *occluded region*, as can be seen in Figure 5.

The parameters l and r are determined automatically using the size of the robot gripper and the resolution of the grasp image coming from the classifier (explained in Section 3.2). The thickness l of the v-cloud is determined by considering the radius of a hypothetical sphere that covers the whole gripper. In our experiments, we used a two-finger parallel physical gripper (length = 0.12 m, width = 0.12 m, and height = 0.02 m, when fully opened) for which l turned out to be $(\sqrt{2} \times 0.12)/2 \approx 0.07$ m. The resolution r of the v-cloud was decided in relation to the resolution of the grasp image, coming from the LeNet classifier. The goal is to keep r such that the v-cloud points, covered by the gripper size (mentioned earlier) in any particular grasp configuration, fill every pixel when projected to its grasp image. Based on this, we calculated the value of $r \approx 3$ mm, which was able to project one v-cloud point to each corresponding pixel of the grasp image. It is to note that we used the same resolution for the LeNet grasp image (60×60 pixels) as applied in the reference method [11].

Using the procedure explained above, the circle and sphere v-cloud patterns are generated as shown in Figure 6(b) and (c), respectively. The v-cloud is generated by selecting a suitable pattern shape depending on the type of cloud points representing objects in a scene. The last two columns of Table 1 describe the relationship between the object-point types and the corresponding v-cloud patterns.

2.5. Type identification of the cloud points

To determine whether a cloud point on an object represents a surface, an edge, or a corner, in a given scene, we consider the number of neighboring points whose depth difference from the sensor compared to the point under consideration is larger than a threshold d. The 'depth threshold' (d) is kept slightly bigger than the thickness of the v-cloud (l) to avoid overlapping of objects. There can be four directions on a depth-image (Figure 8): left(X-), right(X+), up(Y-), and down(Y+). This leads to four types of neighboring points: left, right, up, and down. The categorization of the four neighbor points is given in the first four columns of Table 1. If the depth of a point under consideration is found larger (smaller) than d, the corresponding cell in Table 1 for that neighboring point contains 'o' ('–'). In certain special cases (*e.g.* for long and highly cylindrical objects), the objects start appearing just as a line in the depth image. To cater those special cases, we use 'horizontal-spine' and 'vertical-spine' edges as shown in Figure 8.

Table 1. Types of the edges of an object, cloud points and the corresponding v-cloud patterns.

Neighbor points					
Left	Right	Up	Down	Point types	v-cloud patterns
–	–	–	–	surface	short line
o	–	–	–	left edge	left half circle
–	o	–	–	right edge	right half circle
–	–	o	–	up edge	up half circle
–	–	–	o	down edge	down half circle
o	o	–	–	vertical-spine edge	horizontal circle
–	–	o	o	horizontal-spine edge	vertical circle
o	–	o	–	left-up corner	left-up quarter sphere
o	–	–	o	left-down corner	left-down quarter sphere
–	o	o	–	right-up corner	right-up quarter sphere
–	o	–	o	right-down corner	right-down quarter sphere
o	o	o	–	up tip	up half sphere
o	o	–	o	down tip	down half sphere
o	–	o	o	left tip	left half sphere
–	o	o	o	right tip	right half sphere
o	o	o	o	noise	–

3. Filtering out the unsafe grasps

This section explains how the mapped v-cloud is used to remove the incorrect (i.e. unsafe) grasp candidates. As mentioned earlier, a good grasp configuration is defined as the one in which the physical gripper of the robot remains completely immersed inside the v-cloud during a desired grasping action. The proposed approach to determine a successful grasp simulates the grasping action for a given grasp candidate. The percentage of the gripper volume that remains outside (uncovered) of the mapped v-cloud determines whether to discard or accept the given grasp candidate. This can be time consuming if applied to a large number of grasp candidates directly. Therefore, to counter this issue we introduce the concept of gripper key-points to remove the risky grasp candidates in a faster way. It is done by filtering out the grasp candidates for which the key points on the gripper are not fully covered in the v-cloud (see Section 3.1). Then, for each of the remaining grasp candidates, we generate two grasp images: (i) one describing the area that the gripper-fingers need to cover when the gripper is closed to grasp the object, and (ii) the second having the area of the gripper covered by the v-cloud for that particular grasp configuration. The purpose of these two grasp images is to determine the percentage of gripper area that remains visible (i.e. inside the v-cloud) when performing the grasp action using that particular grasp configuration. We define it as the 'grasp visibility' (V_g), which is the ratio of the intersection of grasp image (i) and (ii), and the area covered by the gripper-fingers (from grasp image

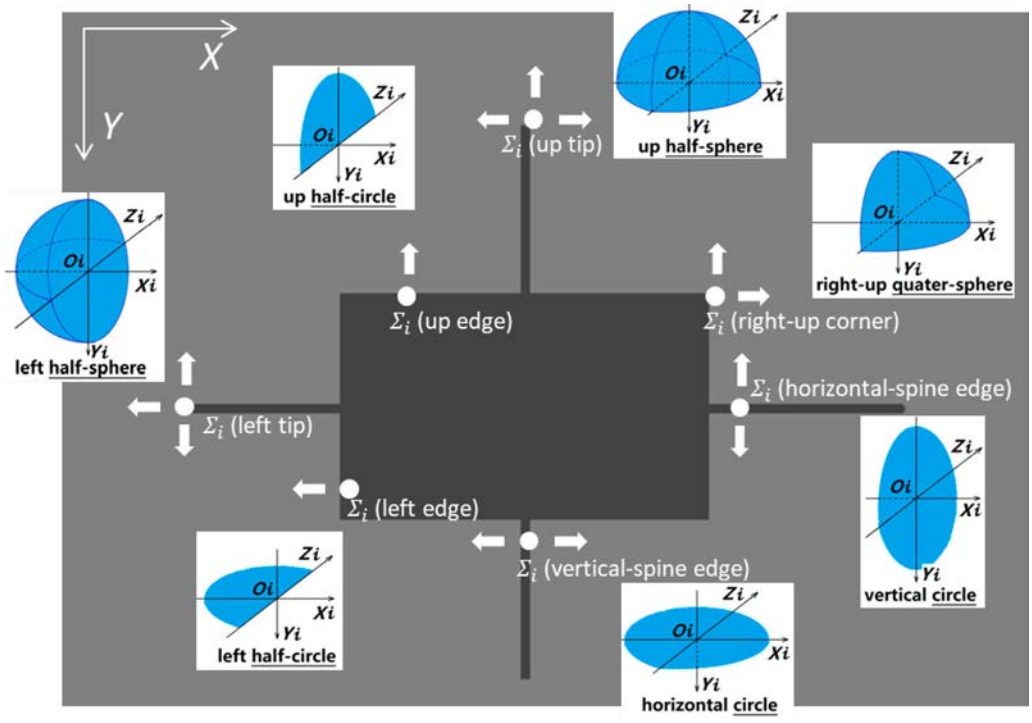


Figure 8. The types of various edge-points in a depth image and their relationship with the corresponding shape-patterns of the v-cloud are illustrated. The region in gray color represents the distance of the cloud points from the sensor; light-gray represents the background, while the dark-gray color represents a hypothetical object. The white circles represent several types of pixels (i.e. cloud points). Let us consider a reference point (white point) on the object. It has four neighbor points (pixels), left, right, up, and down. If the difference between depth distance of the reference point and that of a neighbor point is larger than a threshold d ($d = 0.07$m), a white arrow is depicted to the neighbor point direction from the reference point. The number and directions of white arrows indicate the number of neighboring pixels whose depths are greater than that of the reference point and their location(s) (e.g. left, right, up, down). In this figure, seven types of the reference object point are shown. For example, 'up-edge' means out of four neighboring pixels only one on the Y -direction has depth difference greater than the threshold (d). Similarly, the types of other edge-points can be identified. There are two special cases (horizontal- and vertical-spine) where the object is too thin and the edge-point appears to fall on a line. The goal of each shape-pattern is to make sure that the object is surrounded by the v-cloud from every unoccupied direction.

(i)). The grasp candidates with low V_g score are deemed unsafe and, therefore, are removed. Further details are given in Section 3.2.

3.1. Removing the incorrect grasps by using the gripper key-points

Figure 9 shows a 2-D visualization of three (one blue and two red) key points on the robot gripper. The blue key point is placed at the palm of the gripper. In order to avoid an unwanted contact between the gripper palm and the object, the blue key point shall be at least at a distance $2r$ (r being the resolution of the v-cloud) from the surface of the object. The blue key point serves as the first condition to be met for a successful grasp candidate. In other words, if the blue key point is covered by the v-cloud and has a distance $\geq 2r$, the corresponding grasp candidate is considered further. Otherwise, it is discarded. Similarly, one red point is placed on the mid-length of each finger of the gripper. As there are two fingers – with one joint on each – in our case, we have used two red key points. The gray area represents the v-cloud and yellow curves indicate the point cloud representing the object(s). Note that Figure 9 shows 2-D top views of the gripper. In real practice, the mapped v-cloud and position of the key points are placed in 3-D.

The proposed method takes into consideration only those grasp candidates for which a base (blue) key point is covered in the mapped v-cloud. This means all the grasp candidates where the blue key point is not covered by the mapped v-cloud are discarded, *e.g.* Figure 9(d). There can be two possible situations where this happens. First, when the grasp candidate is located out of the sensing area and second when the direct-sight of the camera is blocked by the presence of an object in between. Both situations are regarded as unsafe to carry on with the grasping action. To determine whether a key point is covered by the mapped v-cloud or not, we directly use the Octo-tree. The process of generating the mapped v-cloud using the Octo-tree is already described in Section 2. It

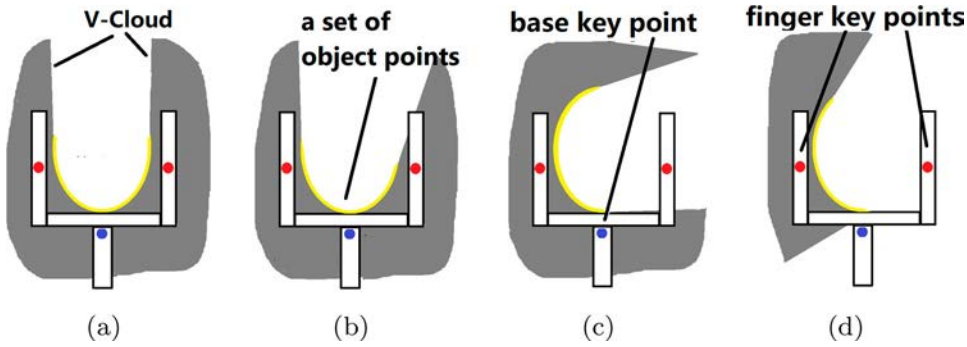


Figure 9. Three key-points on the gripper are used to determine whether it is approaching the object from a suitable direction, i.e. is covered by the v-cloud. (a) and (b) illustrate a correct gripper approach because it is immersed in the v-cloud, while (c) and (d) represent the gripper approach where a portion of the gripper is not covered – means it may be colliding with another object – by the v-cloud, and, therefore, is considered unsafe.

is worth noting that about 90% of the grasp candidates are removed using this technique, without applying any additional filters for a single depth-camera.

After removing the unsafe grasp candidates, we rank the remaining candidates by looking whether all the finger (red) key points are covered in the mapped v-cloud or not. This helps differentiate between different grasp situations, for example Figure 9(a), (b), and (c). As the robot gripper used in this research has two fingers, we only consider the grasp candidates that have both red key points covered by the mapped v-cloud. All the remaining grasp candidates are discarded.

3.2. Removing the incorrect grasps by visibility

Using the gripper key-points (as explained above) is fast but not enough to ensure that the gripper always remains in the visible region of interest for a given grasp configuration. For example, there is a possibility that the gripper may pass through some occluded (unsafe) regions even for the class-1 grasp candidates (all gripper key-points are covered in the mapped v-cloud) due to the limited numbers of key points. One such situation is shown in Figure 9(b). Also, as the key points represent a static position of the gripper, we cannot ensure that the gripper will not pass through any occluded region when closing its fingers. This is important to avoid any untoward contact with objects present in the occluded region.

To avoid this, we generate two grasp images. The first grasp image describes the area swept by the gripper-fingers when simulating the grasp action in 3-D and projects to the considered plane, shown with green color in Figure 10(b). This grasp image is generated using Figure 10(a), which is the grasp image fed to the LeNet [19] neural network for classification of grasp configurations. The second grasp image describes the v-cloud covering that grasp handle, indicated by the gray area in Figure 10(c). Using these two grasp images, we then determine the *grasp visibility* (V_g), which is a measure to indicate the grasp suitability of each individual grasp configuration. To determine V_g, we need to know the area swept by the gripper-fingers that remain covered by the v-cloud for a given grasp configuration. In the proposed method, this area is determined by taking the intersection of the gray area (representing pixels covered by the v-cloud) in Figure 10(c) and the green area (pixels representing the area swept by the gripper-fingers when grasping) shown in Figure 10(b). Using the above information, V_g is then calculated as expressed in Equation (1). Here, $N_{\text{intersection}}$ denotes the number of pixels that are present in both the green (Figure 10(c)) and as well as in the gray area (Figure 10(b)). N_{gripper} represents the pixels that are green (in Figure 10(b)). If the 'grasp visibility' of a grasp candidate is smaller than a threshold (90% in our experiments), that grasp candidate is regarded as unsuitable and, hence, is discarded.

$$V_g = \frac{N_{\text{intersection}}}{N_{\text{gripper}}} \tag{1}$$

3.3. Determining the best grasp

After removing the grasp candidates that have low scores from the 'grasp visibility' metrics and the classifier, the best grasp candidate is determined from the remaining grasp candidates using the cost function below,

$$E = a_1 S_{\text{visibility}} + a_2 S_{\text{classifier}} + a_3 S_{\text{manipulability}} + a_4 S_{\text{arm\_config}}. \tag{2}$$

where $a_1, \ldots, a_4$ are weight coefficients that express the importance of each component. $S_{\text{visibility}}$ represents the min–max normalized value of grasp visibility V_g and has a range of [0,1]. $S_{\text{classifier}}$ is the min–max normalized score from the neural network classifier, which also

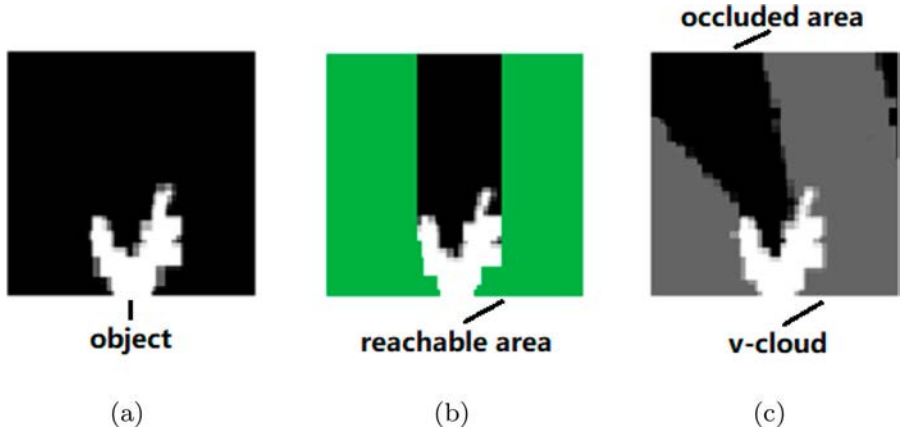


Figure 10. Example grasp images: (a) The image used as input to the neural network classifier. White area represents the object. (b) The green color shows the area where the gripper can reach and perform a grasping action. (c) The gray area indicates the v-cloud.

has a range of [0,1]. $S_{manipulability}$ is the min-max normalized manipulability value [20] of the robotic arm in the approaching direction, which has a range of [0,1]. Higher manipulability for a particular grasp location means the robot arm can move quickly (with high speed) and easily in that direction to/from the current configuration. $S_{\text{arm\_config}} = 1 - \sum_{1}^{M} \frac{|u_m - x_m|}{\pi}$ denotes the similarity between the current robot state and desired robot state required for performing the grasping action [21], which has a range of [0,1]. x_m represents the joint angles of the current robot state, u_m is the joint angles of the desired robot state and M denotes the number of joints of the robotic arm. A bigger value of S_{arm_config} means the current robot state is closer to the desired robot state required to perform the grasping action, and, therefore, the robot can easily approach the grasp-location. The value of the coefficients $a_1, \ldots, a_4$ in the experiments were 0.4, 0.4, 0.1, and 0.1, respectively.

4. Experimental evaluation and results

We evaluate the effectiveness of the proposed method by conducting real-world experiments using a physical robotic arm (Kinova Mico2) equipped with a two-finger (Kinova KG-2) gripper. The goal is to determine the effectiveness of the proposed method with incomplete point cloud information and complex backgrounds. We also compare the performance of the proposed method with another state-of-the-art grasp detection method [11].

4.1. Experimental setup

The experimental setup is shown in Figure 11. The robot arm (Kinova Mico2), equipped with a two-finger gripper (Kinova KG-2), is parked nearby the experiment field to lift the objects and move them to the nearby recycling box. Since the gripper (Kinova KG-2) has a grasping and lifting force of 25 N and 500 g, respectively, it is capable to lift light-weight objects (up to 50 g) without slippage.

The experiment field was setup with three different background settings to evaluate the performance of the proposed method for heterogeneous backgrounds, first using a single camera and then using two cameras to obtain a more complete point cloud information. In these experiments, we set the number of grasp candidates (N) equal to 2, 000 for step IV in the proposed algorithm (Section 2.1). The first experimental field, shown in Figure 12(a), is built using wooden cubes on a table top. The objects are placed on the wooden steps in random order. The background heterogeneity is intentionally kept high to test the robustness of the proposed method. The second background type, shown in Figure 12(b), also has wooden cubes on a table top but in a vertical orientation. The third background type, shown in Figure 12(c), is a table top with no wooden cubes. This is the simplest background is used to test whether the proposed method has any disadvantages in the absence of any serious occlusion.

The details of the camera locations and the robot position on the experiment field are shown in Figure 13. The objects with three different backgrounds were placed in the center position of the field. The robotic arm was parked close (on the X_f axis and kept at a distance of 0.5 m from the center) to the field so that it could easily access the objects from different angles. In the experiments, first a single (Camera 1) and then two Intel® RealSense™ D435 depth-cameras (Camera 1 and Camera 2) were used to obtain the point cloud. The two cameras were placed facing toward the center of the field, each having a distance of 0.6 m from the center. The cameras had +60° and -60° angles with respect to axis

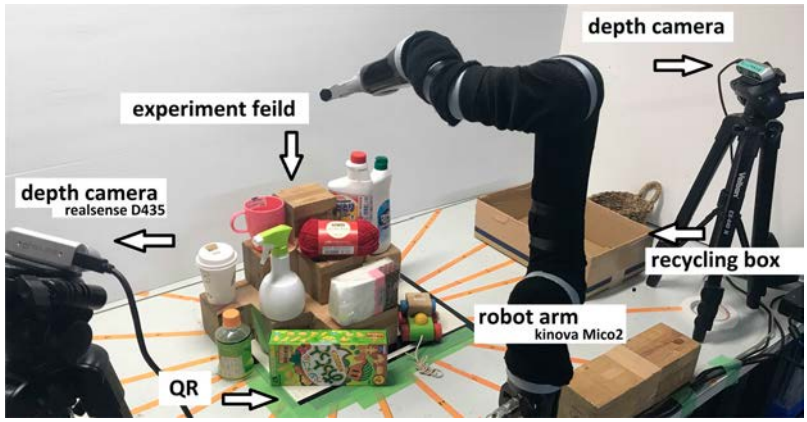


Figure 11. Experimental setup: Ten objects are randomly chosen from a pool and placed in the field. Two depth cameras (Intel RealSense D435) are used to obtain the point cloud and a robot arm (Kinova Mico2), equipped with a two-finger gripper (Kinova KG-2), is used to perform the grasping task. The QR code on the floor is used to coordinate the position of camera(s) and robot gripper.

Figure 12. Three types of experimental fields: (a) Objects are placed on random wooden steps to describe a relatively heterogeneous background. (b) The wooden cubes are placed in a vertical orientation like pillars. (c) The objects are placed on a flat table top to describe a simple background.

X_f, respectively. The height of each camera was kept as 0.6 m above the field/table surface. The communication between the robot arm and the RealSense™ Camera was handled using the Robot Operating System (ROS) on a ThinkPad™ P51 laptop with 3.8 GHz Intel® Core i7-7700HQ CPU (four physical cores) and 32 GB of system memory. We used the CPU-Only version (Caffe [22]) of the LeNet neural network for classification which was pre-trained, as used in the reference method [11]. The *manipulability* of the robotic arm is obtained using inverse kinematic (IK) solutions through software package *MoveIt!* [21], which uses IKFast in OpenRAVE [23].

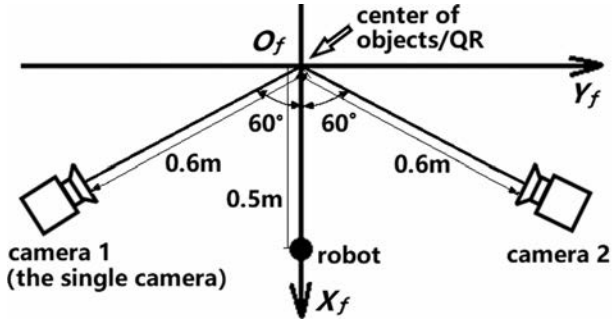


Figure 13. The experimental setup: The objects were placed on the center position O_f. Two Realsense™ D435 cameras (camera 1 and camera 2) located at +60° and -60° angles to X_f axis, respectively, and 0.6 m from the center O_f were used. The robot arm was placed on the X_f axis and 0.5 m away from the center O_f. The pose information of the cameras and the robot arm was obtained using a QR code pasted at the center O_f. When the experiments of a single camera, the camera 1 is used. When the experiments of two cameras, both camera 1 and camera 2 are used.

4.2. Experimental procedure

Six experiments, one for each background setting, first with (i) a single camera, and then with (ii) two cameras, were conducted and each experiment consisted of two stages. In the first stage, grasp detection was performed using the proposed method. The evaluation function (Equation (2)) determined the best grasp candidate and the robotic arm was moved to grasp the object. In the second stage, the reference method [11] was applied to the same experimental setup. Each stage of the experiment

consisted of ten rounds. At the beginning of each round, 10 objects were randomly selected from a pool of 34 objects and placed in the experimental field by a operator. The robotic system was used to perform the perception task and remove the objects one by one from the field in a random order. This procedure was repeated for ten times (trials), which counted as one round. In each stage of an experiment, a total of 100 trials (10 rounds) were performed. During each trial, if a 'best grasp-pose' was found, the robot was commanded to grasp and remove the object from the scene. If the object was removed successfully, i.e. the object is grasped, lifted, transported, and dropped in to a box, a success was counted. Otherwise, it was recorded as a failure. In both cases (success or failure), an attempt was counted. However, if the system could not find any suitable grasp-pose, the robot did not move and the next trial was started. In other words, the environment remained unchanged and the perception was carried out again to obtain new N grasp candidates. Such a situation was counted as a 'trial', but not as an 'attempt'. The average grasp detection time was recorded and the pick-up action was executed to count the number of objects removed successfully. Then, in the second stage, the same procedure was applied using the reference method and average grasp time and objects removed were calculated.

As mentioned earlier, each stage of the six experiments was divided into ten rounds, where each round had ten trials. We recorded the number of grasp attempts, grasp success, and the number of remaining objects in the scene after ten trials. If all objects were not removed by the system, the operator would remove them manually at the end of each round. After ten rounds, the experiment ended.

4.3. Benchmarking

To benchmark the performance, we set up a standard task for the proposed method and as well as the reference method [11]. In one round, a operator picked up ten objects randomly from a pool of 34 objects and placed them in the experiment field in a random order. The system detected the best grasp-pose using the proposed method and started taking out the objects with the help of the robotic arm. If the object was lifted successfully and transported to the goal position (dropped into a box), one success was counted. If the object was not transported to the goal position successfully, this attempt was counted as a failure. After the failed attempt, the next trial was carried out without changing the environment. In case of a planning failure (i.e. the planning software (*MoveIt!* [21]) could not find a solution, neither was an 'attempt' counted, nor a 'trial'.

4.4. The degree of point cloud (in)completeness

In the real-world applications, it is difficult to obtain a complete point cloud information. Therefore, one major objective of the proposed method is to work well even with incomplete point clouds. However, there is a lack of standardization when it comes to describing the 'degree of completeness' (or incompleteness) of a given point cloud. In this research, we describe the 'degree of completeness' in terms of percentage by generating a visible region generated by ray-traversal algorithm [17], as shown in Figure 14. In Equation (3), N_{scene} denotes the number of total points in a visible region when there are no objects but only background, i.e. the field is empty. N_{object} denotes the number of total points in a visible region when the objects are placed in the scene. The degree of completeness of the point cloud $P_{completeness}$ is calculated before each round of the experiment,

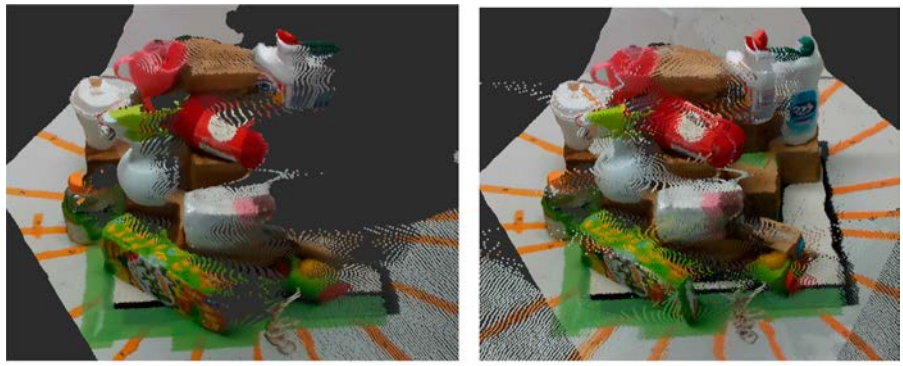
$$P_{completeness} = \frac{N_{\text{object}}}{N_{\text{scene}}}. \tag{3}$$

Note that $P_{\text{completeness}}$ depends on type of the field and the number of point-cloud cameras used. However, it is completely independent of the method of perception or any grasp-pose optimization techniques.

4.5. Experimental results

The experimental results are summarized in Table 2. We recorded the following six measures: The average completeness of the point cloud, the number of attempts, the number of removed objects, the success rate (ratio of the number of objects removed to the number of total attempts), the percentage of objects removed (ratio of the number of objects removed to the total number of objects), and the average calculation time (second).

Single camera means a significantly incomplete point cloud, with average completeness from 62.4% to 67.8%, was available. The proposed method has shown a success rate of 82.4% for the background with wooden blocks lying horizontally ((a) steps), 83.0% with wooden blocks placed vertical ((b) pillars), and 83.9% for background a simple background over the (c) table top. On the other hand, the success rate of the reference method remained 62.0%, 77.8%, and 79.8% for backgrounds (a), (b), and (c), respectively. For a more complete point cloud using two cameras (with average completeness from 80.1% to 83.0%), the proposed method demonstrated a success rate of 85.3%, 82.5%, and 85.4% for each background, respectively. On the other hand, the success rate of the reference method was recorded as 80.8%, 80.6%, and 84.5%.



(a) (b)

Figure 14. Degree of the point-cloud completeness: (a) shows a 59.2% complete point cloud which is obtained using a single camera. (b) shows a 83.8% complete (a more complete) point cloud obtained using two cameras.

Table 2. Performance of the proposed method vs. the reference method [11].

Field type	Camera number	Avg. completeness of point cloud	Method	Attempts	Removed	Success rate %	Objects removed %	Avg. calculation time (s)
(a) Steps	Single camera	62.4%	Proposed	91	75	82.4%	75.0%	2.31
			Reference	100	62	62.0%	62.0%	2.78
	Two cameras	80.1%	Proposed	95	81	85.3%	81.0%	3.01
			Reference	99	80	80.8%	80.0%	3.37
(b) Pillars	Single camera	67.8%	Proposed	94	78	83.0%	78.0%	2.23
			Reference	99	77	77.8%	77.0%	2.59
	Two cameras	82.0%	Proposed	97	80	82.5%	80.0%	2.75
			Reference	98	79	80.6%	79.0%	2.91
(c) Table top	Single camera	66.4%	Proposed	93	78	83.9%	78.0%	1.98
			Reference	99	79	79.8%	79.0%	2.03
	Two cameras	83.0%	Proposed	96	82	85.4%	82.0%	2.55
			Reference	97	82	84.5%	82.0%	2.48

Next, we counted the number of objects removed successfully from the scene during each round. With a single camera, the proposed method removed 75.0%, 78.0%, and 78.0% for the scene (a), (b), and (c), respectively. On the other hand, the reference method removed 62.0%, 77.0%, and 79.0% of objects for each scene. With two cameras, the proposed method removed of 81.0%, 80.0%, and 82.0% for each scene. The reference method removed as 80.0%, 79.0%, and 82.0% for each scene.

In addition, we recorded the computational time to grasp an object for both methods. When using a single camera, the average calculation time of the proposed method was 2.31 s for the scene (a), 2.23 s for the scene (b), and 1.98 s for the scene (c). For the reference method, it was 2.78 s, 2.59 s, and 2.03 s. This means the proposed method consumed 16.9%, 13.8% and 2.5% less calculation time as compared to the reference method with a single depth-camera for each scene, respectively. With two cameras, the average calculation time of the proposed method was 3.01 s, 2.75 s, and 2.55 s. For the reference method, it was 3.37 s, 2.91 s, and 2.48 s. This means the proposed method required 10.1% and 5.5% less calculation time for the scene (b) and (c) respectively, but it took 2.8% more calculation time for the scene (a) table top.

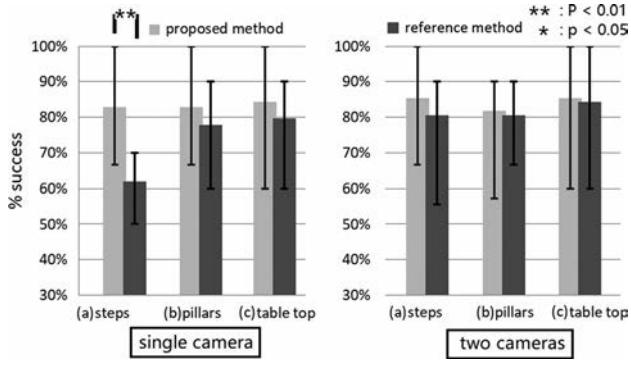


Figure 15. The success rate (%) of the proposed method and the reference method [11] for each background type with a single (one left), and then two cameras (on right).

The results of all six experiments are graphically shown in Figures 15–17.

4.6. Discussion

A comparison of the success rate for the proposed method and the reference method is shown in Figure 15. The results demonstrate that the proposed method has a higher success rate than the reference method, especially

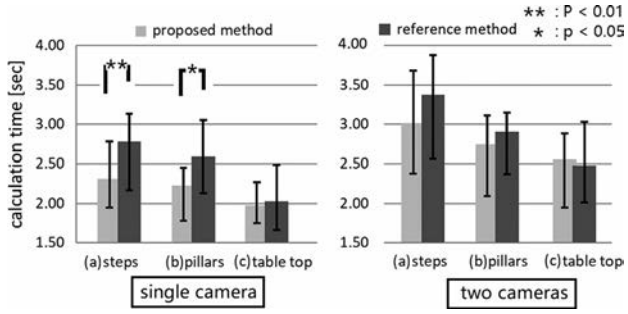


Figure 16. The calculation time (s) of the proposed method and the reference method [11] for each background type with a single (one left), and then two cameras (on right).

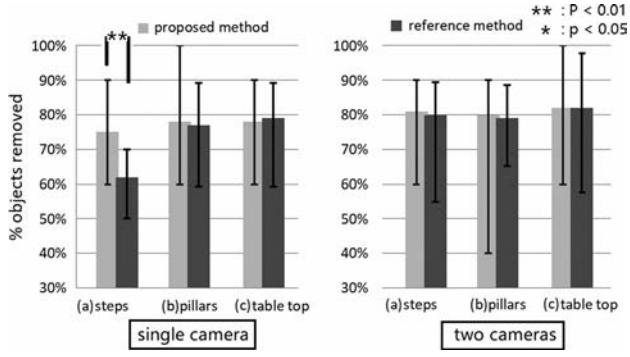


Figure 17. Percentage of the objects removed using the proposed grasp detection method and the reference method [11] for each background type with a single (one left), and then two cameras (on right).

when the point cloud information is significantly incomplete (single-camera case) and the experimental field has a relatively complex background (i.e. steps). In addition, the success rate of the proposed method is not affected much due to changes in the 'degree of completeness' of the available point cloud. In both cases (with single and double cameras), the success rate stays above 80%, which indicates the robustness of the proposed method as compared to the reference method. On the other hand, the difference between the success rates and the percentage of object removal decreases if the background becomes simpler, i.e. less heterogeneous. This means the proposed method gives a real advantage when the point cloud is highly incomplete and the background is relatively complex.

The average calculation time (second) for each method is plotted as shown in Figure 16. The proposed method outperforms the reference method for complex background situations in both cases when the point-cloud information is incomplete (single-camera). The percentage of successfully removed objects for the proposed method and reference method are plotted in Figure 17. The proposed method removes more objects, especially when the point cloud information is significantly incomplete and the experimental field has a complex background. Moreover, the proposed method shows a clear advantage over the reference method when the point cloud is significantly incomplete. The proposed method can be useful for applications that have tight space-constraints and mounting several sensors onto a robot's body is not feasible.

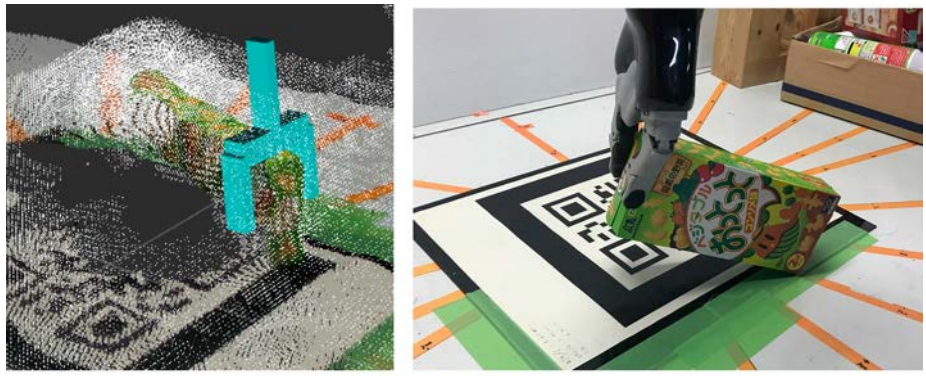
The novelty of the proposed method for being faster lies in the fact that it removes most of the incorrect grasp candidates quickly at a very early stage. This results in time-saving and reduces the classification burden for the neural network classifier. When using a single camera, the proposed method on average of (a)steps feeds only 6.6% of the total grasp candidates to the neural network, which is 127.5 grasps out of each trial 2000 grasp candidates. On the other hand, the reference method feeds 19.2% of the total candidates, which accounts for 384.4 grasps out of the total of 2000 grasps. When two depth-cameras are used, the proposed method on average of (a)steps feeds only 7.4% of the total grasp candidates to the neural network, which is 149.1 grasps out of the total of 2000 grasps. With two cameras, the reference method feeds 19.7% of the total candidates, which is 393.3 grasps out of the total 2000 grasps.

4.7. The failure cases

The failure cases observed during our real-world experiments can be classified into three types:

(a) failures due to sensing/positioning errors,
(b) failures due to classification errors,
(c) failures due to high-scored but incorrect grasp handles.

Type (a) and (b) failures that comprise the majority of failure cases (about 90% of the total failures) in our experiments are not caused by the proposed method. For example, type (a) failures are caused by the sensing/positioning errors of cameras and the robotic arm. Since the objects are light-weight, they can easily be tumble or pushed forward when the gripper-finger touches them with a small positioning error. Similarly, type (b) failure happens when the image classifier (LeNet) cannot recognize the object correctly. Since the proposed method and the reference method [11] both use the same LeNet classifier, these problems do not affect our comparison. Type (c) failures are caused mainly by the proposed method when a high-scored grasp candidate fails. However, this occurrence is very rare (about one in a hundred trails). When analyzed closely, we observed that this rare



(a) (b)

Figure 18. A failure case using the v-cloud: (a) the detected grasp handle is fully covered in the v-cloud but located too close to the edge of a box-shaped object, (b) when the robot arm tries to grasp the object, the object slips between the fingers due to lack of grasp stability.(a) Illustration of the grasp. (b) Execution in real-world.

failure is caused by the lack of grasp stability, as shown in Figure 18. One way to avoid this problem is to filter out all the grasp candidates that are located very close to the edges of the objects. However, this can make the proposed highly conservative. In addition, as this is a very rare failure case so the success rate is not expected to improve significantly by solving this problem.

We believe a combination of the *grasp visibility*, proposed in this research, and the *grasp stability* may help remove this rare failure case as well. Therefore, it will be an interesting future work to consider grasp stability as a part of the cost function and analyze its impact on the computational time and success rate.

5. Conclusion

This paper introduced a new method of autonomous grasp detection using a novel approach that divides the scene into visible (detectable) and occluded (undetectable) regions using the camera pose (position and orientation) and point cloud information. A *visible point cloud* (*v-cloud*) is generated to quickly exclude the potentially incorrect (unsafe) grasp candidates. The proposed method has been evaluated for object grasping in three different backgrounds with varying heterogeneity and levels of point cloud completeness. The experimental results show that the proposed method offers a significantly higher success rate compared to another state-of-the-art method, especially with incomplete point cloud (obtained using a single camera). In the case of a more complete point cloud (obtained using two cameras), the proposed method has a slightly higher success rate than the reference method. However, the proposed method requires a smaller calculation time for experimental situations with complex backgrounds. The results demonstrate that the proposed method is robust enough to achieve a high success rate for autonomous grasping tasks even with limited point cloud information and complex background. In the future, we plan to reduce the processing time further by leveraging the advantages offered by the v-cloud.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Xixun Wang is a doctor course student at Kyoto University, Japan. He received his M.S.at Waseda University, Japan. He is working on an autonomous rescue robot. His research interests are trajectory planning, force/position control, human interface, and perception of robotic grasping.

Sajid Nisar received the MS (M.Eng.) and PhD (Dr. Eng.) degrees in Mechanical Engineering and Science, with majors in Robotics, from Kyoto University, Japan, in 2016 and 2019, respectively. He is currently a Junior Associate Professor with the Department of Mechanical and Electrical System Engineering, Kyoto University of Advanced Science (KUAS), Kyoto, Japan. He is the founder and Principal Investigator of Novel Intelligent Systems and Advanced Robotics (NISAR) Laboratory at KUAS, where his research focuses on the design and development of robotic systems for applications in robot-assisted surgery, haptics, mobility, and other human-centered areas. Previously, he was a Visiting PhD Scholar with the Department of Mechanical Engineering and Center for Design Research, Stanford University, USA. Dr. Nisar holds several awards and distinctions; including the Young Award from the IEEE Robotics and Automation Society (RAS), Japan, for best paper at the International Conference on Robotics and Automation (ICRA'19), Engineering Dean's Award from Kyoto University, Best Innovation Award from the IEEE World Haptics 2019, and a Fellowship from the Japan Society for Promotion of Sciences (JSPS). He is a member of ASME, JSME,

New York Academy of Science, and several other professional organizations.

Fumitoshi Matsuno received the Dr. Eng. degree from Osaka University, Japan, in 1986. In 1986, he joined the Department of Control Engineering, Osaka University. Since 2009, he has been a Professor with the Department of Mechanical Engineering and Science, Kyoto University, Japan. He is also the Vice-President of the Robotics Society of Japan (RSJ) and the NPO International Rescue System Institute, Japan, and served as the President of the Institute of Systems, Control and Information Engineers. His current research interests include robotics, swarm intelligence, the control of distributed parameter systems and nonlinear systems, and rescue support systems in disaster. Prof. Matsuno received many awards, including the Outstanding Paper Award in 2001, 2006 and 2017, the Takeda Memorial Prize in 2001 and the Tomoda Memorial Prize in 2017 from the Society of Instrument and Control Engineers (SICE), the Prize for Academic Achievement from Japan Society of Mechanical Engineers (JSME) in 2009, the Best Paper Award in 2013 from the Information Processing Society of Japan, and the Best Paper Award in 2018 from the RSJ. He is a General Chair of International Symposium of Distributed Autonomous Robotic Systems (DARS) 2021 and Asian Control Conference (ASCC) 2022. He also served as a General Chair of the IEEE SSRR2011 and the IEEE/SICE SII2011, SWARM2015, SWARM2017 etc. He is a Fellow Member of the SICE, the JSME, and the RSJ.

References

[1] Kragic D, Miller AT, Allen PK. Real time tracking meets online grasp planning. Proceedings of International Conference on Robotics and Automation; Seoul, South Korea: 2001 July. p. 2460–2465.

[2] Wu C, Jiang S, Song K. CAD-based pose estimation design for random bin picking using A RGB-D camera. J Intell Robot Syst. 2017;87:455–470.

[3] Kehl W, Manhardt F, Tombari F, et al. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. Proceedings of International Conference on Computer Vision; Venice, Italy: 2017 Dec. p. 1521–1529.

[4] Krull A, Brachmann E, Michel F, et al. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. Proceedings of International Conference on Computer Vision; Santiago, Chile: 2015 Feb. p. 953–962.

[5] Guo D, Sun F, Kong T, et al. Deep vision networks for real-time robotic grasp detection. Int J Adv Robot Syst. 2017;14:1–8.

[6] Kumra S, Kanan C, Robotic grasp detection using deep convolutional neural networks. Proceedings of International Conference on Intelligent Robots and Systems; Vancouver, Canada: 2017 Dec. p. 768–776.

[7] Fang K, Bai Y, Hinterstoisser S, et al. Multi-task domain adaptation for deep learning of instance grasping from simulation. Proceedings of International Conference on Robotics and Automation; Brisbane, Australia: 2018 May. p. 3516–3523.

[8] Morrison D, Corke P, Leitner J. Learning robust, real-time, reactive robotic grasping. Int J Robot Res. 2020;39:183–201.

[9] Wu B, Akinola I, Allen P, Pixel-attentive policy gradient for multi-fingered grasping in cluttered scenes. Proceedings of International Conference on Robotics and Automation; Macau, China: 2019 Nov. p. 1789–1996.

[10] Kanoulas D, Lee J, Caldwell D, et al. Visual grasp affordance localization in point clouds using curved contact patches. Int J Human Robot. 2017;14(1):1–21.

[11] Pas AT, Gualtieri M, Saenko K, et al. Pose detection in point clouds. Int J Robot Res. 2017;36:1455–1473.

[12] Patten T, Park K, Vincze M. DGCM-Net: dense geometrical correspondence matching network for incremental experience-based robotic grasping. 2020.

[13] Choi C, Schwarting W, DelPreto J, et al. Learning Object grasping for soft robot hands. IEEE Robot Autom Lett. 2018;3(3):2370–2377.

[14] Mousavian A, Eppner C, Fox D. 6-DOF GraspNet: variational grasp generation for object manipulation. Proceedings of International Conference on Computer Vision; Seoul, Korea: 2019 Oct. p. 2901–2910.

[15] Liang H, Ma X, Li S. PointNetGPD: detecting grasp configurations from point sets. Proceedings of International Conference on Robotics and Automation; Montreal, Canada: 2019 May. p. 3629–3635.

[16] Amanatides J, Woo A. A fast voxel traversal algorithm for ray tracing. Proceedings of Eurographics; Amsterdam, Netherlands: 1987.

[17] Hornung A, Wurm KM, Bennewitz M, et al. An efficient probabilistic 3D mapping framework based on octrees. Auton Robots. 2013;34:189–206.

[18] Hameiri E, Shimshoni I. Estimating the principal curvatures and the darboux frame from real 3-D range data. IEEE Trans Syst Man Cybern. 2003;33:626–637.

[19] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86:2278–2324.

[20] Yoshikawa T. Manipulability of robotic mechanisms. Int J Robot Res. 1985;4:3–9.

[21] Chitta S, Sucan I, Cousins S. Moveit![ROS topics]. Proceedings of International Conference on Robotics and Automation; 2012 April. p. 18–19.

[22] Jia Y, Shelhamer E, Donahue J, et al. Convolutional architecture for fast feature embedding. Proceedings of the 22nd ACM international conference on Multimedia; 2014 Jun. p. 675–678.

[23] Diankov R. Automated construction of robotic manipulation programs [Ph.D. thesis]. Robotics Institute, Carnegie Mellon University; 2010.