Robotica (2022), **40**, pp. 447–463
doi:10.1017/S0263574721000503

CAMBRIDGE
UNIVERSITY PRESS

RESEARCH ARTICLE

A novel method for finding grasping handles in a clutter using RGBD Gaussian mixture models

Olyvia Kundu, Samrat Dutta and Swagat Kumar*

TATA Consultancy Services, Bangalore 560066, India
*Corresponding author. Email: swagat.kumar@tcs.com

Received: 12 July 2019; **Revised:** 6 April 2021; **Accepted:** 7 April 2021; **First published online:** 16 June 2021

Keywords: grasp pose detection; graspable affordance; grasping; RGBD point cloud; Gaussian mixture model (GMM); surface normals; region growing algorithms; primitive shape identification

Abstract

The paper proposes a novel method to detect graspable handles for picking objects from a confined and cluttered space, such as the bins of a rack in a retail warehouse. The proposed method combines color and depth curvature information to create a Gaussian mixture model that can segment the target object from its background and imposes the geometrical constraints of a two-finger gripper to localize the graspable regions. This helps in overcoming the limitations of a poorly trained deep network object detector and provides a simple and efficient method for grasp pose detection that does not require a priori knowledge about object geometry and can be implemented online with near real-time performance. The efficacy of the proposed approach is demonstrated through simulation as well as real-world experiment.

1. Introduction

A robot capable of manipulating its environment is more useful than one that can only perceive it. Such robots will be able to carry out many of our daily chores, thereby relieving humans for more creative pursuits. For this to happen, the robots are required to have human-like manipulation and grasping abilities. While the manipulation abilities of robots have matured over the years, grasping still remains a difficult problem which has attracted a lot of attention in the recent past due to the increasing demand of such manipulative robots in almost all sphere of human activities. A number of methods exist in literature that attempt to solve this grasping problem [1]. Our focus is primarily restricted to vision-based methods that aim at finding or detecting graspable handles using visual sensor data as input. Some of these methods use visual features in 2D images to localize graspable regions, [2, 3] while others use range or depth data for this purpose [4, 5, 6, 7, 8], the latter becoming more popular owing to the availability of low-cost RGBD sensors. Recently, the deep learning-based methods are becoming increasingly popular for detecting graspable regions [9, 10, 11, 12, 13]. Most of the existing methods for vision-based grasping can be broadly classified into two categories: one that relies on the availability of accurate geometric information about the object (or a CAD model) [14, 15, 16] making them impractical in several real-world use cases, and the other that computes grasping affordances directly from a RGBD point cloud by harnessing local geometric features without knowing the object identity or its accurate 3D geometry [6, 17, 18, 19].

This paper looks into the problem of grasp pose detection (GPD) which concerns itself with finding suitable graspable affordances for a given object in a cluttered environment with application to warehouse robotics as specified by the Amazon Picking Challenge [8, 20, 21]. Specifically, it is expected that an articulated robotic system should be able to automatically pick objects from a rack to a tote and vice versa using an on-board vision system. The grasping problem is challenging in this case as

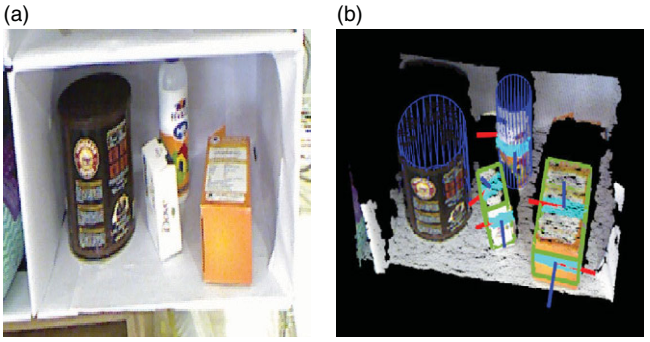

Figure 1. *Input and output of the proposed system. Input is a RGBD image obtained from a robot camera. The output is the detection of suitable graspable affordances with their corresponding grasp poses.*

the robot has to identify an object within a confined and cluttered space of a bin of the rack. This is different from other cases where people either consider isolated objects as in ref. [6] or table top cases as in refs. [22, 7]. The problem is challenging due to several factors like partial occlusion, poor illumination, change of shape and size of deformable objects, scaling as well as restricted field of view for the rack bins. Contrary to the existing methods that separate the grasping from object recognition problem [18, 6], an attempt is made here to combine these two aspects to improve the accuracy of GPD and find suitable graspable affordances necessary for picking objects by a robot manipulator. The object recognition is carried out by training a deep learning network like RCNN [23] which requires a large number of training examples generated through laborious manual effort. The objective of this work is to reduce the training requirement for the object detector while improving the accuracy of GPD through the inclusion of object identity in the process of computing graspable affordances. This is achieved using a Gaussian mixture model (GMM) [24] that combines color and curvature information to segment the surfaces of the target object from its background within the bounding box provided by the object recognizer. This allows us to use a weakly trained object recognizer for obtaining same level of GPD performance. In other words, the inaccuracy of object recognition module is compensated by using a GMM that segments the target from the background clutter within the bounding box. The GMM for the target can be trained using a few images from the training set used by the object recognition module and hence does not require any additional information or a priori knowledge of object shape or CAD model. The overall method of GPD involves the following four steps: (1) obtain a bounding box around the target object using a RCNN object detector, (2) create surface segments within the bounding box using a region growing algorithm [25] that exploits the continuity of surface normals [26, 27] and then remove the background clutter using a GMM model of the target, (3) apply a set of empirical rules on the remaining surface segments to identify primitive shapes of the object and (4) apply a modified version of curve-fitting algorithm [18, 19] to localize the grasping handles for a two-finger gripper. The working of the resulting GPD pipeline is shown in Fig. 1. The input to the proposed method is an RGBD image, shown in the left image, obtained from a kinect camera mounted on the robot wrist and the output is a set of graspable handles along with their corresponding grasp poses as shown in the right image.

There are several advantages of this approach. For instance, it does not require any computationally intensive training phase to compute GPD unlike other deep learning methods that attempt to process RGB or RGBD images directly [11, 12, 13, 28, 29] and in some cases, make use of simulators like Graspit! [14] to produce examples required for training. This allows us to carry out GPD on the fly in near real-time once the target item has been localized in the image view frame. Since most of the vision-based robotic applications for grasping use an object detector for locating the target object, including object identity can greatly simplify the process of GPD without any significant

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

increase in the computational requirement. Further simplification in GPD is achieved by identifying the closest primitive shape, rather than the actual shape of the object. This is adequate for grasping most objects using a two-finger gripper commonly used in industries [30]. By imposing the constraints of a two-finger gripper and approximating the exact shape by its closest primitive, the 6-dimensional grasp pose search problem is reduced to a 1D search problem. Finally, the proposed method can efficiently detect grasp poses for box-type objects which is considered to be a difficult problem compared to other shapes like spheres or cylinders [17]. Unlike the existing methods that primarily rely on curvature information for computing these affordances, the proposed work makes use of surfaces normals and their directions to create multiple surface segments which are later combined together using empirical rules to identify shape primitives. This allows us to grasp rectangular box-type objects more efficiently compared to the above approaches. This also allows us to take into account the presence of neighboring objects which could render the affordances unusable by the robot.

In short, the main contributions made in this paper are as follows:

- A novel method for GPD is proposed that incorporates object identity to simplify the process of finding grasping handles for objects in a clutter from RGBD point clouds. The proposed method is computationally efficient and can be implemented online with near real-time performance and does not require any a priori knowledge about the object geometry or shape.
- The proposed approach uses a novel GMM based on color and curvature to segment surfaces belonging to the target object from those in the background. This allows one to work with a weakly trained object recognizer needing less amount of training data.
- The proposed method identifies primitive shapes from the surface segments allowing us to effectively grasp box-type objects thereby overcoming the limitations of existing methods that primarily rely on curve-fitting algorithms to find grasp handles.
- The efficacy of the proposed method is established through rigorous experiments with a large number of household objects which are picked by a UR-10 robotic arm with an RG2 gripper from a vertical shelf.

The rest of this paper is organized as follows. The proposed method is explained in Section 2. The analysis of various simulation and experimental results are provided in Section 3 followed by conclusion in Section 4.

2. Proposed method

As described earlier, the objective of this work is to detect graspable handles for an object in a clutter by directly processing the RGBD point cloud without making use any a priori knowledge about the object geometry. The flow diagram depicting the steps involved is shown in Fig. 2. The proposed method consists of the following four modules. The first module is the *object recognition module* that uses a deep learning-based algorithm (Faster R-CNN) [23] to detect the target object in the workspace. A bounding box with maximum likelihood (score) is returned by this module for a given query object. A bounding box is a rectangular region on the image frame defined by a tuple (x, y, w, h), where (x, y) is the pixel coordinate of the left-bottom corner of the box and (w, h) are its width and height, respectively. The object detection module returns this bounding box for the target object, thereby specifying its location within the image frame. Depending on the accuracy of the detection algorithm, this bounding box could be bigger including other objects in the vicinity of the target object. The second module, called *surface extraction module*, processes the RGBD point cloud data within this bounding box to extract surfaces by applying region growing algorithm to the surface normal attribute of each pixel. The region growing algorithm may not always provide perfect boundary of the target object due to over or under segmentation. This problem is remedied using a GMM to segment the target object from its background. The third module, called *shape-fitting module*, recognizes the primitive shape for the segmented target using principal component analysis (PCA) and empirical geometric rules. The fourth module, called *grasp*

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

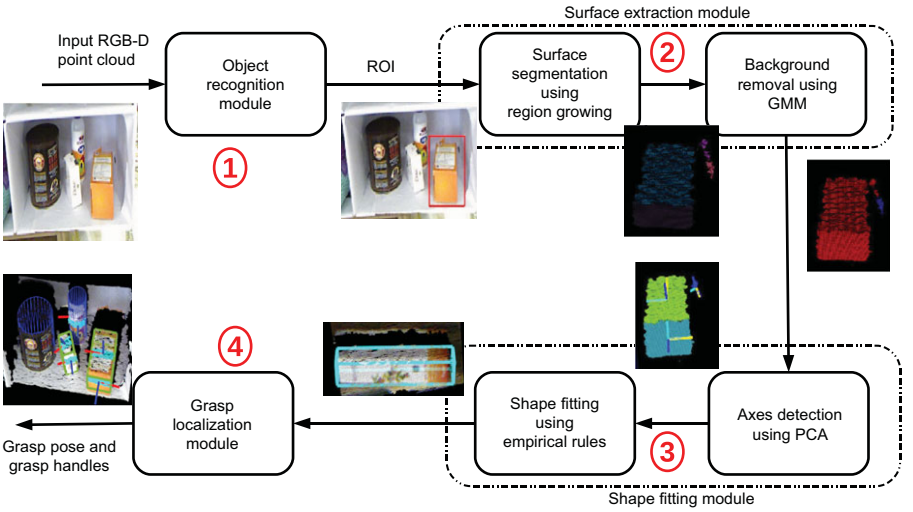

Figure 2. *Flow diagram of our proposed method. The pipeline consists of four major modules: (1) RCNN-based object recognition module, (2) surface segmentation module, (3) shape fitting module and (4) grasp localization module.*

localization module, detects the grasping handles by taking into account the geometry of two-finger grippers and the symmetry of the primitive shape identified. The details of each individual module are described next in this section.

2.1. Object recognition module

The advent of deep learning [31] has transformed the field of object recognition and detection in computer vision. Particularly, the deep convolutional neural networks (CNNs) were shown to provide double digit improvement over traditional methods in the ImageNet challenge [32]. Since then a number of deep networks have been developed to recognize and detect objects. Some of these models are RCNN, Faster-RCNN [23], YOLO [33], R-FCN [34]. Many researchers have explored scene segmentation network such as PSPNet [35] for object segmentation. [36, 37]. Researchers have also applied deep learning for recognizing objects directly in RGBD point clouds.[38, 39, 40]. In general, deep learning-based methods require large amount of data for training the detection or recognition models. Many of these datasets are generated through laborious manual effort requiring thousands of man-hours. Additionally, processing RGBD datasets are computationally more expensive compared to processing only RGB images. Considering all these factors, a middle path is taken that combines a simple deep learning-based object recognition module with a traditional feature-based post-processing module to obtain the grasping handles for objects in a clutter. A lightly trained Faster-RCNN [23] is used to recognize objects which returns multiple bounding boxes for a given object with varying likelihood score. The bounding box with a maximum score is usually taken for the further processing.

2.2. Surface extraction module

It is the second module in our pipeline that takes the bounding box obtained from the object recognition module as the input and processes the RGBD point cloud within this bounding box to extract surfaces which will be used for identifying primitive shapes in the next step. It consists of two steps – first, applying region growing algorithm to create surface segments and then use GMM-based models to remove background clutter. These are explained in the following two sub-sections.

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

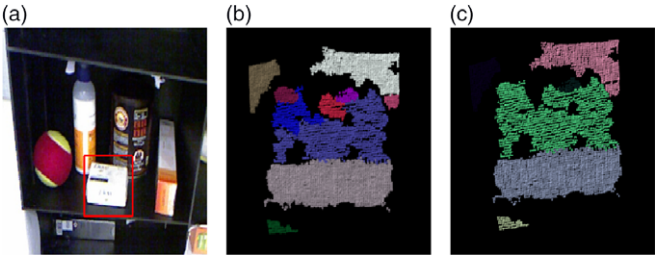

Figure 3. *Creating surface segments using region growing algorithm: (a) the RCNN bounding box (marked with red color) obtained from the object recognition module, (b) the patches of surface segments obtained after applying the first iteration of region growing algorithm and (c) the surface segments obtained after merging neighboring patches. Each color represents a different surface.*

2.2.1. Creating surface segments using region growing algorithm

Inside each window given by faster RCNN, the corresponding depth data are cropped and considered for further processing. The main idea is to group the points which belong to same surface. For instance, it is desirable to have different surface segments, one for each face of a box. Similarly, there should be two surface segments for a cylinder – one curved surface on the side and one planar top surface (depending on the view). This goal is achieved by applying region growing algorithm [27] that uses the angle between surface normals as the smoothness criterion to decide whether the neighboring pixels belong to the same surface or not. The algorithm first considers a seed point, and then the surface normal between the point and neighborhood points is, compared. If the angle between them is less than a threshold, then it is considered that the point belongs to the current region. Otherwise, it is considered as a new seed point. This process is repeated until all points are checked. The implementation of region growing algorithm available with PCL [41] is used for this work. The above process may result in multiple patches for a given surface. These patches are further merged with the neighboring patches if they have very similar average surface normals. The similarity is checked by taking the dot product between the average surface normals of two neighboring patches. If the dot product is above a certain threshold, the neighboring patches are considered to be the parts of the same surface. The output of the region growing algorithm is shown in Fig. 3. The first figure shows the bounding box around the target object obtained from the object recognition module. The surface segments created using region growing algorithm based on smooth of surface normals are shown in the second image. The effect of second stage of smoothing is shown in the last figure which shows more contiguous segments for each side of the box.

2.2.2. GMM-based segmentation using color and curvature

The bounding box returned by the object detection module may not exactly enclose the object. It may also include other background objects inside it. The surface segments for the target object created using region growing may include points from other objects within the target bounding box. This happens because the surface normal information is itself quite noisy and may not be adequate to detect subtle changes in direction at object boundaries. This limitation is shown in Fig. 4(b) where the surface segment belonging to the target object is shown in red color and the background points are shown in blue color. The background objects lying close to the target object become a part of the target surface when region growing is applied to the surface normals. To solve this problem, the foreground target object is modeled using a GMM that uses color and depth curvature attribute of the RGBD point cloud. The hue and saturation component of the HSV color space is utilized instead of RGB color information. The V channel is discarded to remove the illumination effect. The depth curvature is obtained using the Taubin Fitting method [18] described in the next section. The GMM model is trained using the templates for each target object. Since the identity of the object is known from the object recognition module, the corresponding GMM model could be used to localize it in the RGBD point cloud within the bounding

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

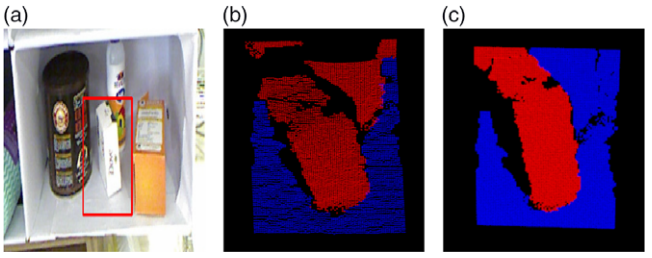

Figure 4. *Gaussian mixture model is used to segment the target object from its background. (a) The bounding box obtained using RCNN object recognition module; (b) the surface segments obtained using region growing module. The surface segment with red color includes points from background object: (c) the target surface after applying GMM model. It clearly segments the target object from its background within the bounding box. Red color denotes the points that belong to the target and the blue points correspond to other background objects.*

box. The proposed architecture uses two mixture models: one is trained to estimate the probability of the foreground pixels and the other learns the background pixels from the given image. The predictions are then compared to decide the category of a pixel. The GMMs are designed as follows:

Given the dataset $\mathscr{D}_v = \{[h, s, d, c_v]_{j=1}^{N}\}_{i=1}^{D}$, where $v = \{target, background\}$[1]; h, s, d represent the hue, saturation and depth curvature and c_v is the class of the corresponding pixel, N is the total number of pixels in the patch and D is the total number of training images, the joint probability distribution of the dataset is given by:

$$P_v(h, s, d, c_v|\boldsymbol{\theta}) = \sum_{k=1}^{K} P_v(k) P_v(h, s, d, c_v|k), \tag{1}$$

where K is the number of Gaussians in the mixture model, $P_v(k)$ is termed as the prior and $P_v(h, s, d, c_v|k)$ is the conditional probability density function which is given by

$$\begin{aligned} P_v(h, s, d, c_v|k) &= G_v^k\left(h, s, d, c_v; \mu_v^k, \Sigma_v^k\right) \\ &= \frac{\exp^{-\frac{1}{2}\left(\left([h;s;d;c_v]-\mu_v^k\right)^T\left(\Sigma_v^k\right)^{-1}\left([h;s;d;c_v]-\mu_v^k\right)\right)}}{\sqrt{2\pi^3\left|\Sigma_v^k\right|}}, \end{aligned} \tag{2}$$

where μ_v^k and Σ_v^k are the mean and covariance matrix, respectively, of the k-th Gaussian. The posterior $P_v(c_v|h, s, d)$ gives the probability of the class v. Therefore, the probability of the target pixel belonging to class v ($= target$) is

$$P_{target} = \frac{P_v(c_{v=target}|h, s, d)}{P_v(c_{v=target}|h, s, d) + P_v(c_{v=background}|h, s, d)} \tag{3}$$

A pixel is considered to be a part of the target object if this probability $P_{target} > 0.5$, otherwise it is considered as a part of the background. All the pixels inside the bounding box are subjected to this test. Figure 4(c) shows the performance of the proposed GMM-based segmentation. It can be observed that the proposed technique is able to successfully identify the true background pixels and segregate the pixels from the target object (shown in red).

The performance of segmentation depends on the quality of training. It is also important to select a right number of clusters which can be determined empirically through several trials. An incremental clustering method is used to arrive at the optimal number of clusters required for the GMM model. The pseudocode of the algorithm is provided in Algorithm 1 below. It takes the RGBD point cloud as input and returns the number of clusters K and their cluster centers $\{\mu_k\}$, $k = 1, 2, \ldots, K$ as output.

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

Algorithm 1 Incremental Clustering for Gaussian Mixture Model

Require: Input data points $p_i(h, s, d),\ i = 1, .., N$; {$h$ = Hue, s = Saturation, d = Depth Curvature}

Initialize the number of clusters: $K \leftarrow 0$
for $i = 0$ to $i = N$ **do**
  if $K = 0$ **then**
    Add a cluster $\mu_K \leftarrow p_i$
    $K \leftarrow K + 1$, Cluster Size $S_K \leftarrow 1$
  else
    Compute the distance of the point p_i from all clusters $\mu_k,\ k = 1, 2, \ldots, K$: $d_{ik} = \|p_i - \mu_k\|$
    Compute the cluster label *cmin* with minimum distance $d_{\min}$: $cmin = \arg\min_k d_{ik}$ and $d_{\min} = \min_k d_{ik}$
    if $d_{\min} < \theta$ {θ is a user-defined threshold} **then**
      Assign the point p_i to the cluster *cmin*
      Update its cluster center: $\mu_{c\,\min} \leftarrow \frac{S_{c\,\min}\mu_{c\,\min}+p_i}{S_{c\,\min}+1}$
      Increment its cluster size $S_{c\,\min} \leftarrow S_{c\,\min} + 1$
    else
      Create a new cluster: $K \leftarrow K + 1$; $\mu_K \leftarrow p_i$; $S_K \leftarrow 1$
    end if
  end if
end for
return $K, \mu_k, k = 1, 2, \ldots, K$

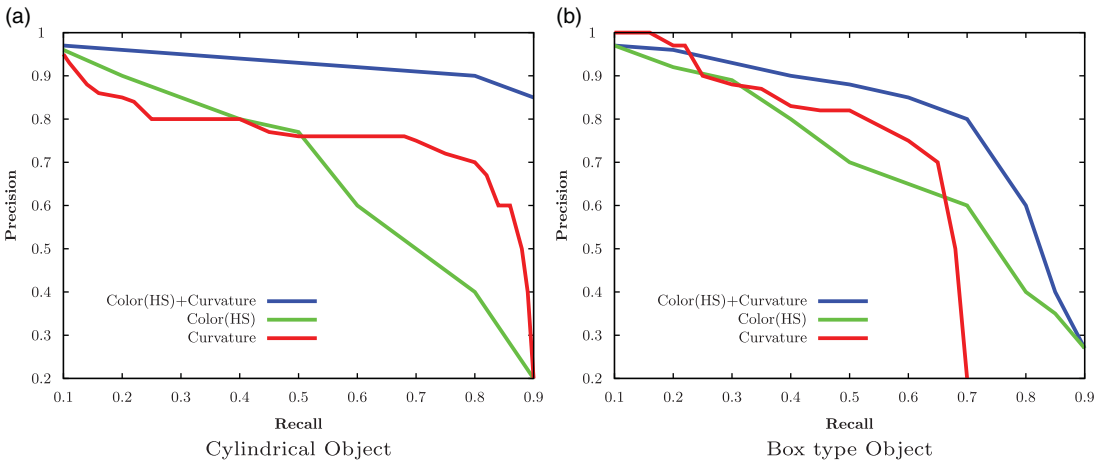

Figure 5. *Precision-recall curve for detecting two types of objects – (a) cylinder and (b) box, using Gaussian mixture model based on color, curvature and combination of both. It shows that combining color with depth curvature information provides better discrimination compared to using either color or curvature alone.*

As mentioned before, each point is represented by three attributes of hue (h), saturation (s) and depth curvature (d).

The positive effect of combining color and depth curvature in GMM can be further corroborated by analyzing Fig. 5 which shows the precision–recall curve for detecting two different kinds of objects – cylindrical and box types using GMM models. The red line shows the performance of GMM models in detecting objects using only depth curvature attribute. The green line shows the performance of GMM models that uses only hue and saturation color components to detect objects. The blue line shows

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

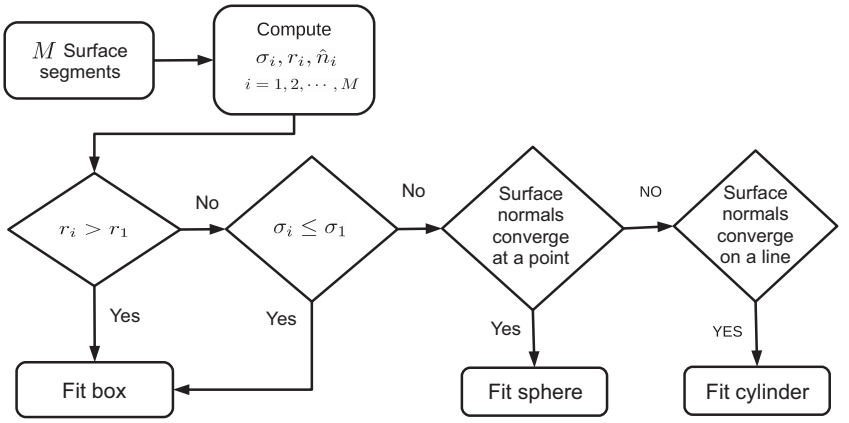

Figure 6. *Flow diagram of method for fitting three shape primitives – box, sphere and cylinder. Primitive shapes are identified by applying empirical rules to geometrical attributes such as, radius of curvature (r_i), variance of surface normals (σ_i) and mean surface normal ($\hat{n}_i$).*

the performance of GMM models that use both color (h,s) and depth curvature (d). The figure clearly shows that the combination and color and curvature improves performance of detecting targets against its background. By comparing the plot presented in Fig. 5(a) and (b), it can be concluded that it is easier to detect curved objects compared to flat/box-type objects. The former produces a flatter P–R curve with high precision compared to the latter where it bends quickly with higher recall.

2.3. Primitive shape fitting module

This is the third module of our proposed pipeline. It takes the surface segments obtained from the previous module as input and gives out the shape information about the object. While it is not absolutely mandatory to detect shapes to find grasping handles, the knowledge of shape can be used for simplifying the computation of graspable affordances for a given object. The use of a two-finger gripper further simplifies the grasping problem by obviating the need for computing the exact shape information. In such cases, the object can be approximated by its closest primitive shapes or a combination of such shapes. The closest primitive shape for a given object is identified by applying empirical geometrical rules to the surface segments obtained from the previous module. Without any loss of generality, the method for identifying three primitive shapes, namely cuboidal (box-type), spherical and cylindrical shapes, is demonstrated in this section. It is possible to derive similar rules to identify other primitive shapes as well. The flow diagram of the method for identifying the above three primitive shapes is shown in Fig. 6. It is assumed that M surface segments are obtained for a given target object from the previous module (surface extraction module). The following three parameters are computed for each of these surface segments: (1) radius of curvature r_i, (2) variance of surface normals σ_i and (3) mean surface normal $\hat{n}_i$. The segments for which the radius of curvature is more than a certain threshold r_1 and variance of surface normals is below a threshold σ_1 must represent flat surfaces belonging to a box-type cuboidal object. The segments not satisfying this condition must represent curved surfaces belonging to either a cylindrical or a spherical object. Additional conditions need to be applied to distinguish a spherical object from a cylindrical one. The surface normals for a spherical surfaces will meet at a point when extended in the direction opposite to surface normals as shown in Fig. 7(b). On the other hand, the surface normals for a cylindrical object converge to a line as shown in Fig. 7(a). Similarly, additional constraints could be applied to confirm box-type shapes. For instance, the surface normals for two adjacent surfaces will be orthogonal to each other. These are shown as blue arrows in Fig. 7(c). One can also observe that one of the other two major axes obtained using PCA for these two adjacent surfaces will be parallel to each

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

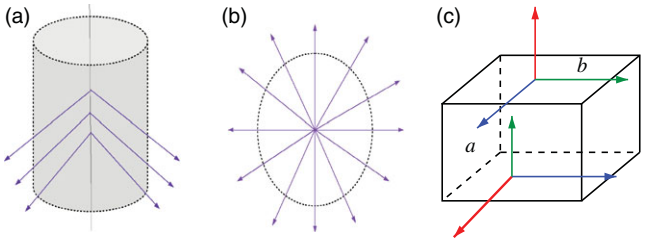

Figure 7. *Additional conditions to be applied to fit a shape primitive for three kinds of objects: (a) cylinder, (b) sphere and (c) box. Surface normals for a sphere converge to a point. Surface normals for a cylindrical object converge to a line. The surface normals of two adjacent surfaces of a box-type object are orthogonal to each other.*

other. The parallel axes are shown in red in this figure. Length along the parallel axis gives the length of the box, while length of the segment along the other axis gives the height of the box. Through these three examples, it is demonstrated that many of these rules are very simple to derive from observation and can be utilized to resolve the ambiguities arising from the inaccuracies of previous modules. The user-defined thresholds are also decided empirically through repeated experiments. The values used in this paper are $r_1 = 0.1$ m and $\sigma_1 = 0.1$. The method for computing the curvature and variance attribute of each surface is described next in this section.

2.3.1. Curvature estimation

The surface curvature is a reliable feature in determining the shape of a given 3D object. The Taubin quadric fitting method provided in ref. [18] is used to find curvature for a given segment. The method involves approximating a set of points in a Cartesian coordinates by fitting a quadratic surface to it in a least square sense. Once this parametric surface is known, maximum curvature is computed for a set of randomly sampled points from the local surface. The median of these maximum curvature is taken as the final curvature of the segment under consideration.

2.3.2. Variance of surface normals

Sometimes the curvature alone is not adequate for deciding the shape of a given object. The variance of the surface normals can be used to decide if a given surface is flat or curved. The variance of surface normals will be less compared to a curved surface where the variance could be high. The variance of surface normals for a given segment i is given by:

$$\sigma_i^2 = E(\hat{n}_i^2) - [E(\hat{n}_i)]^2; \quad i = 1, 2, \ldots, M \tag{4}$$

where $E(.)$ is the expected value computed over all points in a given surface i. The primitive shapes for few objects computed using the above method are shown in Fig. 8. It also shows the three main axes for a given surface. The blue axis represents the surface normal. The other two (major and minor) axes are obtained using PCA and are shown in yellow and cyan colors. The knowledge of the object shape will be utilized for computing a suitable grasping handle as explained in the next section.

2.4. Grasp localization module

This is the last module in our pipeline that computes the graspable handles for a given target object. The knowledge of object shape is useful in finding a suitable grasp handle required for picking up the object. The grasp handles are computed only for a two-finger parallel jaw gripper which is the most common type of gripper used for most pick and place tasks in industries. Finding 6 degrees of freedom (DOF) grasp pose for a general gripper is a difficult problem. We simplify this requirement by making several assumptions about how the gripper will pick an object. This can be understood by studying Fig. 9

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

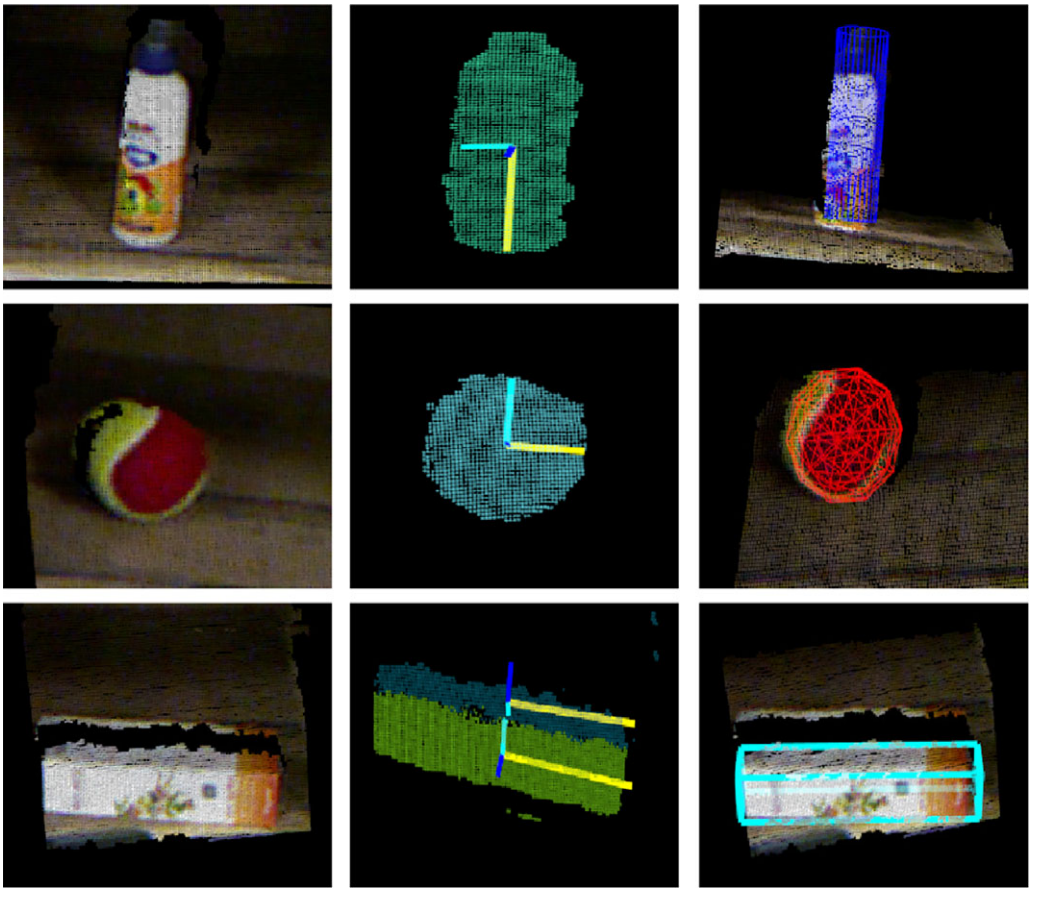

Figure 8. *Primitive shape fitting for few 3D objects – cylinder, sphere and a box.*

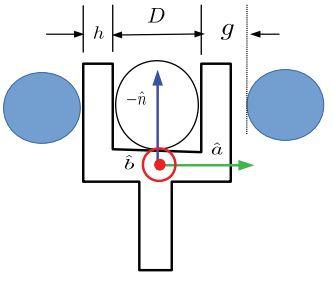

Figure 9. *Gripper parameters. Empty circle represents the target object to be picked by the gripper. The blue circles are the obstacles. The condition $g > h$ must be satisfied to avoid collision with other objects.*

which shows a top view of how a two-finger gripper will approach an object in a clutter. The maximum clearance between the fingers, the maximum hand aperture, is denoted by D. The width of finger in horizontal plane is denoted by h. It is assumed that the gripper will approach the object in a direction opposite to the surface normal of the object in the view. If multiple surface normals are available, each one will be evaluated for its suitable and validity as will be explained shortly. This approach direction is shown by the blue arrow and carries a notation $-\hat{n}$. The green arrow denoted by $\hat{a}$ is the minor axes of

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

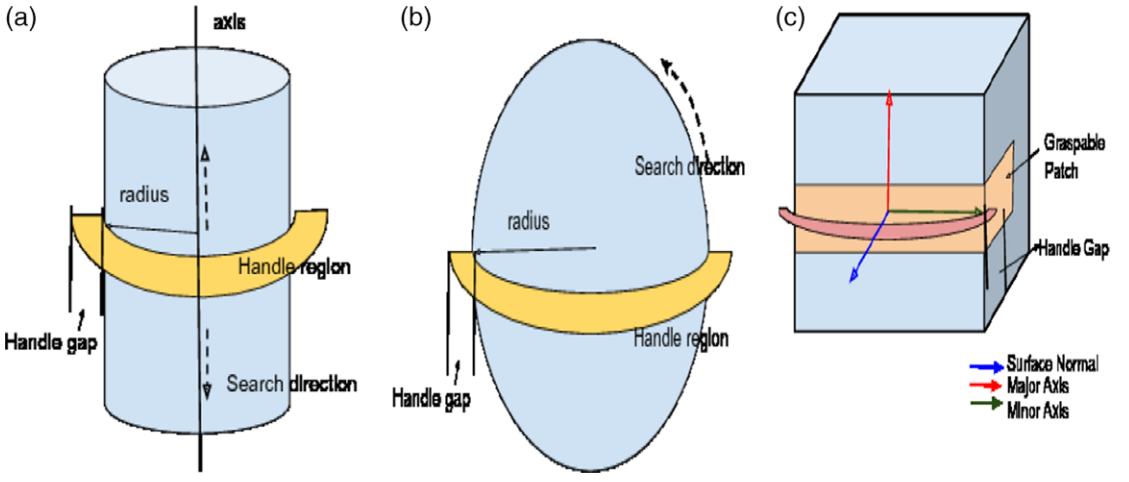

Figure 10. *Search for a suitable grasp handle for different types of object: (a) cylindrical object, (b) spherical object and (c) box-type object.*

the surface segment and the red dot (vector pointing out of the paper plane) denoted by $\hat{b}$ is the major axes of the surface segment. The major and minor axes for a given surface are obtained using PCA as explained earlier. If the object width is less than the maximum hand aperture D, the gripper can be used to grasp the object. Additionally, it is also necessary to check if the gap between two objects g is greater than the width of fingers h. This will avoid collision during grasping. Since the object shape is known, the search for grasping handle is started at the geometric center of the surface segment as shown in Fig. 10. Essentially, a surface patch of dimension $D \times w$, where w is the depth of the finger, is searched in the vertical plane (along the major axis $\hat{b}$). If a patch is found that satisfies the gripper constraints mentioned above, the centroid and the gripper pose are returned to the robot for execution. If the gripper constraints are not satisfied, the search for another patch is carried out along the direction shown in Fig. 10. For cylindrical and box-type objects, the search is continued along the major axis while for spherical objects, the search is continued along the circumference by rotating the gripper handle about the surface normal.

The output of the proposed pipeline for grasp handle detection in a clutter is shown in Fig. 11. The primitive shapes for box, cylinder and spherical type of objects are shown in green, blue and red colors, respectively. Only one grasping handle is shown for cylindrical and spherical objects, and one handle per visible segment is shown in case of box-type objects. The blue line shows the direction of surface normal. The grasp handles are shown in cyan color. The red line shows the minor axis here which aligns with the gripper plane. It only shows the valid grasp handles which can be executed by a robot without any collision with any other objects in the neighborhood. This is validated through a real-world experiment as will be described in the next section.

3. Simulation and experimental results

This section provides details of several experiments carried out to evaluate the performance of the proposed method for detecting grasping handles of objects in a clutter. This is described next in the following subsections.

3.1. Datasets

The performance of the proposed pipeline for computing grasping handles is evaluated on two different datasets comprising of various household objects. One of them is created by us and consists of 482

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

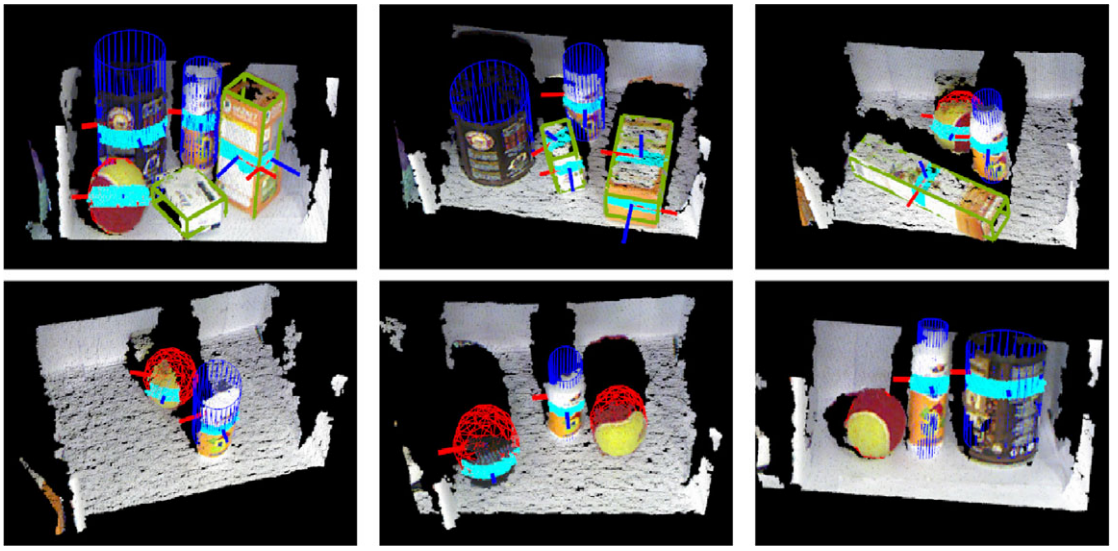

Figure 11. *Detecting grasping handles in a clutter. The three primitive shapes – boxes, spheres and cylinders – are represented by green, red and blue colors, respectively. The handles are shown by cyan color. For each handle, the dark blue line is the normal direction.*

frames of RGBD images collected using a Kinect camera. This includes nearly 382 frames for 9 individual objects in multiple poses and about 100 frames having multiple objects in a clutter. This dataset is being made available publicly for the convenience of readers [42]. The second dataset is a subset of the BigBIRD dataset [43] comprising of nearly 4000 frames having about 26 different type of objects. This dataset contains RGBD frames of only individual objects. The ground truth data for primitive shapes is identified through visual inspection.

3.2. Software and hardware setup

The proposed pipeline is implemented in C/C++ using the point cloud library [41] on a GNU/Linux system (Ubuntu 16.04) running on an i7 processor with 16 GB of RAM. This performance is further corroborated through a real-world experiment using a 6 DOF UR10 robot manipulator with a RG2 two-finger gripper [44]. The experimental setup is shown in Fig. 12(a). It uses a Kinect camera [45] mounted on its end-effector to detect objects and finding grasp handles for each object. The intermediate steps of the grasping pipeline are shown in Fig. 12(b)–(f). The video demonstrations [46], source codes [47] and datasets [42] have been made available online for the convenience of readers. The object recognition module makes use of a deep network (RCNN) module which returns a bounding box containing the target object in the RGB image frame. This module is implemented using Caffe deep learning framework [48]. It is to be noted that the proposed approach can also be used with any other deep network model that returns bounding box for the target object in the image frame.

3.3. Analysis of simulation results

The output for shape fitting algorithm for the above two datasets is shown in Tables I and II, respectively. GMMs for each target object are trained using 150 frames per object. The average accuracy of our algorithm in detecting shapes is 92.06% for the first dataset and 93.4% for the second dataset. The performance of our algorithm in detecting grasping handles is shown in Table III. If one valid handle can be detected for a given object, it is considered as a success. This table shows the performance

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

Table I. *Primitive shape detection results for the first dataset (our own) comprising of 382 RGBD frames of individual objects [42]. The average accuracy is 92.06% over 9 different objects of three shape types. The ground truth shape is decided through visual inspection.*

S. No.	Object	Ground truth	#frame	#pose	#cylin	#sph	#box	%Correct
1	toothpaste	Box	40	10	1	0	37	92.5
2	doveSoap	Box	50	10	1	1	46	92
3	tissuePaper	Box	40	10	2	0	34	85
4	garnetBulb	Box	50	10	3	0	45	90
5	careMate	Cylinder	60	5	57	0	3	95
6	deviCoffee	Cylinder	50	5	46	0	4	92
7	fevicol	Cylinder	32	5	48	0	0	96
8	blackBall	Sphere	30	1	3	44	0	88
9	redGreenBall	Sphere	30	1	1	49	0	98
	Total # frames		382					
	Average accuracy							92.06

of detecting primitive shapes and handles for both individual objects and the objects in a clutter. As one can see in this table, the performance of detecting handles is slightly lower than detecting shapes. It is expected as a number of gripper-specific constraints are applied to detect valid grasp handles to avoid collision with neighboring objects. Also, one can see that detecting handles for box-type objects is comparatively difficult and hence results in lower accuracy compared to the cylindrical or spherical shapes. Fig. 11 shows the output of the proposed algorithm for objects in a clutter. The primitive shape identified is shown as a mesh superimposed on the actual object. The normal to each surface is shown as a blue line. The detected grasping handle for each object is shown as a cyan color band. As one can observe in this figure, the grasping handles (cyan band) are not shown for some of the objects even when the object shapes are identified. This happens because of the constraints imposed by the gripper to avoid collision with neighboring objects. We also compare the grasping performance of the proposed method with that of an existing state-of-the-art approach that harnesses the geometrical attributes of point cloud to compute grasping handles [5, 18]. This is shown in Table IV. As one can see, the baseline approach performs reasonable well with the BigBird dataset which contains only individual objects. However, its performance deteriorates significantly for our dataset that includes several rectangular type box objects and several frames having multiple objects in clutter (see Fig. 11). In contrast, the proposed approach is able to overcome these limitations providing superior performance.

4. Conclusion

This paper attempts to solve the vision-based grasping problem for a warehouse pick and place robot where the robot has to pick and stow items from and to a bin of a rack. The problem is difficult as the robot has to find a grasp pose as well as a suitable graspable affordance to pick the target object. The problem becomes even more challenging when it has to pick the object from a clutter within the confined space of a bin. The image bounding box obtained from the recognition module acts as the input to the grasping module. The first step involves creating several surface segments in the RGBD point cloud using a region growing algorithm that uses a similarity criterion based on surface normals to decide when to create a new surface. These surface segments are then processed to segment the target object from the background clutter using a GMM that exploits the color and curvature attributes to model the target object. This allows one to use a poorly trained object detection module returning larger bounding boxes for the target object. Once the target object is segmented in the RGBD point cloud, a set of empirical rules based on surface attributes is applied to identify basic primitive shapes such as a box, a cylinder or a sphere.

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

Table II. *Primitive shape detection results for BigBIRD dataset. Only a subset of the original dataset comprising of 26 items is used for our evaluation. The average accuracy of shape detection for this dataset is 93.4%. The ground truth shape is decided through visual inspection.*

S. No.	Object	GT	#Frames	#cylin	#sph	#box	%Correct
1	3mHighTackSprayAdhesive	Cylinder	313	310	1	2	99
2	auntJemimaOriginalSyrup	Cylinder	46	42	4	0	91
3	bai5SumatraDragonfruit	Cylinder	278	278	0	0	100
4	bandAidClearStrips	Box	117	8	0	109	93
5	bandAidSheerStrips	Box	134	5	0	129	96
6	blueCloverBabyToy	Box	30	13	5	12	43
7	campbellsChickenNoodleSoup	Cylinder	228	214	12	2	94
8	campbellsSoupAtHandCreamy	Cylinder	170	166	0	4	97
9	canonAckWE0Box	Box	199	1	0	198	100
10	cheezItWhiteCheddar	Box	166	0	0	166	100
11	chewyDippsChocolateChip	Box	158	3	0	155	98
12	cholulaChipotleHotSauce	Cylinder	127	110	17	0	87
13	cinnamonToastCrunch	Box	136	3	0	133	98
14	clifCrunchChocolateChip	Box	123	2	0	121	98
15	niceHoneyRoastedAlmonds	Cylinder	248	240	0	8	97
16	clifCrunchWhiteChocolate	Box	122	3	0	119	98
17	coffeeMateFrenchVanilla	Cylinder	288	260	4	24	90
18	colgateCoolMint	Cylinder	108	104	2	2	96
19	natureValleyCrunchyOatsNHoney	Box	191	3	0	188	98
20	cupNoodlesShrimpPicante	Box	84	21	0	63	75
21	eatingRightForHealthyLivingblue	Box	170	1	0	169	100
22	haagenDazsButterPecan	Cylinder	235	235	0	0	100
23	haagenDazsCookieDough	Cylinder	233	232	1	0	100
24	hersheysCocoa	Box	53	6	3	44	83
25	huntsPaste	Cylinder	204	203	0	1	100
26	nutrigrainAppleCinnamon	Box	142	3	0	139	98
	Total # frames		4303				
	Average accuracy						93.42

Table III. *Average accuracy (%) of detecting shape primitives (P) and grasping handles (H). The performance of detecting graspable handles is slightly lower compared to detecting primitive shapes as the former is a more difficult problem and a number of gripper specific constraints are applied to select valid handles.*

	Box		Cylinder		Sphere	
Scene	P	H	P	H	P	H
Single_Object	90	89.3	95	94.4	94	93
cluttered	84	83.6	92	91	91	90.2

The final step involves computation of a suitable grasping handle for the object using an improved curve fitting method. These modifications allow us to grasp rectangular or box more successfully compared to the previous work. It also allows us to account for neighboring objects that can obstruct grasping, while computing a suitable affordance for the target object. The efficacy of the approach is demonstrated

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

Table IV. *Grasping performance comparison with a baseline approach. Compared to the state-of-the-art baseline approach, our method provides much superior performance in detecting graspable handles.*

S.No	Dataset	Proposed method (%)	Baseline approach (%) [5]
1	BigBird [43]	92	85
2	Our own Dataset [42]	89	48

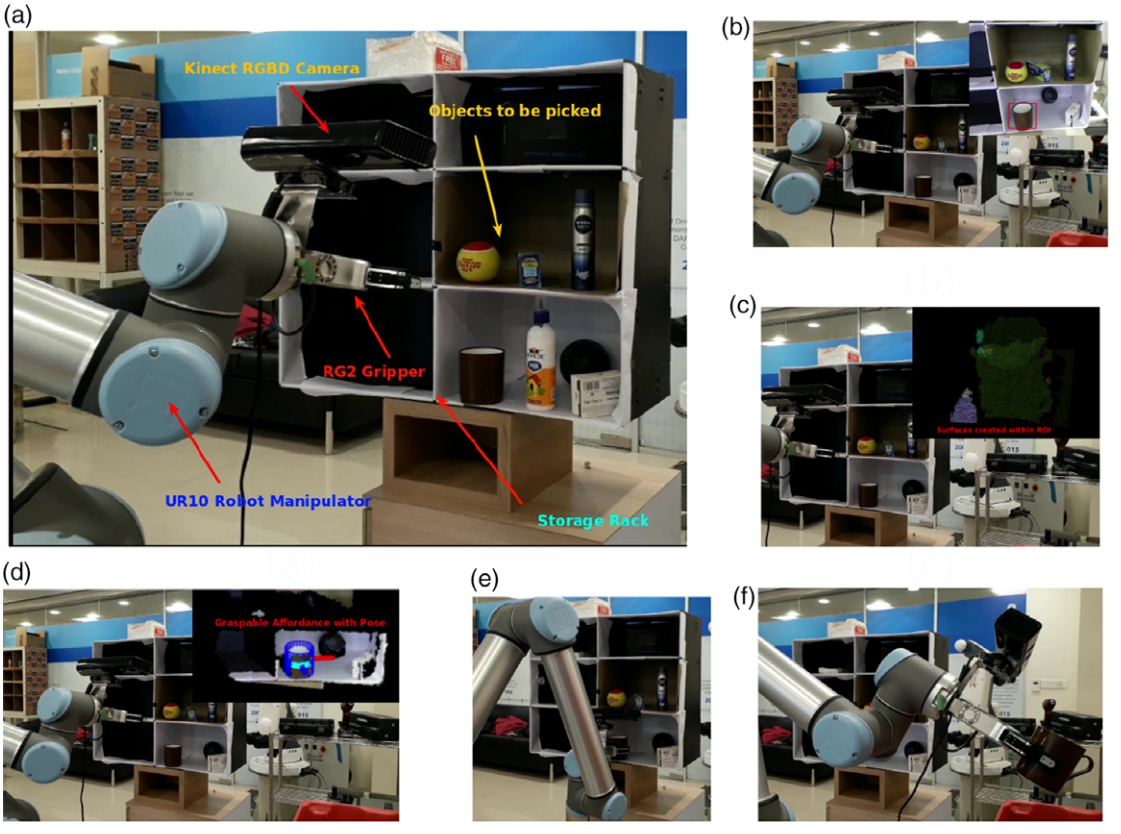

Figure 12. *Experimental setup for testing the proposed grasping algorithm. It uses a hand-mounted Kinect camera for perception and a RG2 two-finger gripper for grasping. (a) Experimental setup, (b) object detection using RCNN, (c) surface segmentation within the ROI, (d) primitive shape identification and pose detection, (e) robot motion for grasping and (f) dropping of object into a tote.*

through rigorous simulation and real-world experiments. The proposed approach is shown to provide significant improvement over an existing baseline approach on two different datasets. The datasets and program source codes are being made available publicly for the convenience of readers.

There are a few limitations which form the scope for future investigation. Reducing the number of examples required for training the GMM model will be one such direction. Secondly, an attempt will be made to directly compute the grasping handles while bypassing the intermediate step to identify the primitive shape of the object. This will make the pipeline simpler and faster.

Supplementary material. To view supplementary material for this article, please visit https://doi.org/10.1017/S0263574721000503

https://doi.org/10.1017/S0263574721000503 Published online by Cambridge University Press

References

[1] Q. M. Marwan, S. C. Chua and L. C. Kwek, "Comprehensive review on reaching and grasping of objects in robotics," *Robotica*, 1–34 (2021). doi: 10.1017/S0263574721000023

[2] A. Saxena, J. Driemeyer and A. Y. Ng, "Robotic grasping of novel objects using vision," *Int. J. Rob. Res.* **27**(2), 157–173 (2008).

[3] L. Montesano and M. Lopes, "Learning Grasping Affordances from Local Visual Descriptors," *IEEE 8th International Conference on Development and Learning, 2009. ICDL 2009* (IEEE, 2009) pp. 1–6.

[4] Y. Jiang, S. Moseson and A. Saxena, "Efficient Grasping from RGBD Images: Learning Using a New Rectangle Representation," *2011 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2011) pp. 3304–3311.

[5] M. Gualtieri, A. ten Pas, K. Saenko and R. Platt, "High precision grasp pose detection in dense clutter," CoRR, vol. abs/1603.01564 (2016).

[6] S. Jain and B. Argall, "Grasp Detection for Assistive Robotic Manipulation," *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016) pp. 2015–2021.

[7] D. Fischinger and M. Vincze, "Shape based learning for grasping novel objects in cluttered scenes," *IFAC Proc, Vol.* **45**(22), 787–792 (2012).

[8] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo and N. Fazeli, "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching," *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 3750–3757.

[9] I. Lenz, H. Lee and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Rob. Res.* **34**(4-5), 705–724 (2015).

[10] J. Redmon and A. Angelova, "Real-Time Grasp Detection Using Convolutional Neural Networks," *2015 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2015) pp. 1316–1322.

[11] D. Kim, A. Li and J. Lee, "Stable robotic grasping of multiple objects using deep neural networks," *Robotica* **39**(4), 735–748 (2021). doi: 10.1017/S0263574720000703

[12] M. Dong, S. Wei, X. Yu and J. Yin, "Mask-gd segmentation based robotic grasp detection," arXiv preprint arXiv:2101.08183 (2021).

[13] S. N. Aslan, A. Uçar and C. Güzeliş, "Semantic Segmentation for Object Detection and Grasping with Humanoid Robots," *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* (IEEE, 2020) pp. 1–6.

[14] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Rob. Autom. Mag.* **11**(4), 110–122 (2004).

[15] U. Klank, D. Pangercic, R. B. Rusu and M. Beetz, "Real-Time CAD Model Matching for Mobile Manipulation and Grasping," *9th IEEE-RAS International Conference on Humanoid Robots, 2009. Humanoids 2009* (IEEE, 2009) pp. 290–296.

[16] W. Yan, Z. Deng, J. Chen, H. Nie and J. Zhang, "Precision grasp planning for multi-finger hand to grasp unknown objects," *Robotica* **37**(8), 1415–1437 (2019).

[17] A. T. Pas and R. Platt, "Using geometry to detect grasps in 3d point clouds," (2015). arXiv preprint arXiv:1501.03100.

[18] A. Ten Pas and R. Platt, "Localizing Handle-like Grasp Affordances in 3d Point Clouds," **In:** *Experimental Robotics* (Springer, 2016) pp. 623–638.

[19] A. ten Pas and R. Platt, "Localizing grasp affordances in 3d points clouds using taubin quadric fitting," CoRR, vol. abs/1311.3192 (2013).

[20] P. R. Wurman and J. M. Romano, "The amazonpicking challenge 2015," *IEEE Rob. Autom. Mag.* **22**(3), 10–12 (2015).

[21] C. Eppner, S. Höfer, R. Jonschkowski, R. Martın-Martın, A. Sieverling, V. Wall and O. Brock, "Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems," *Proceedings of Robotics: Science and Systems, AnnArbor, Michigan* (2016).

[22] M. Richtsfeld and M. Vincze., "Grasping of Unknown Objects from a Table Top," *Workshop on Vision in Action* (2008).

[23] S. Ren, K. He, R. B. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," CoRR, vol. abs/1506.01497 (2015).

[24] Z. Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction," *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, vol. 2 (IEEE, 2004) pp. 28–31.

[25] C. Revol and M. Jourlin, "A new minimum variance region growing algorithm for image segmentation," *Pattern Recogn. Lett.* **18**(3), 249–258 (1997).

[26] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison and A. Fitzgibbon, "Kinectfusion: Real-Time 3d Reconstruction and Interaction Using a Moving Depth Camera," *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (ACM, 2011) pp. 559–568.

[27] T. Rabbani, F. Van Den Heuvel and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **36**(5), 248–253 (2006).

[28] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," arXiv preprint arXiv:1703.09312 (2017).

[29] S. Kumra and C. Kanan, "Robotic Grasp Detection Using Deep Convolutional Neural Networks," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017) pp. 769–776.

[30] L. Birglen and T. Schlicht, "A statistical review of industrial robotic grippers," *Rob. Comput. Integr. Manuf.* **49**, 88–97 (2018).
[31] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature* **521**(7553), 436 (2015).
[32] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," **In:** *Advances in Neural Information Processing Systems* (2012) pp. 1097–1105.
[33] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 779–788.
[34] J. Dai, Y. Li, K. He and J. Sun, "R-FCN: Object Detection via Region-Based Fully Convolutional Networks," **In:** *Advances in Neural Information Processing Systems* (2016) pp. 379–387.
[35] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 2881–2890.
[36] K. Mahajan, A. Majumder, H. Nanduri and S. Kumar, "A Deep Framework for Automatic Annotation with Application to Retail Warehouses," *BMVC* (2018).
[37] C. K. Singh, A. Majumder, S. Kumar and L. Behera, "Deep Network Based Automatic Annotation for Warehouse Automation," *2018 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018) pp. 1–8.
[38] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller and W. Burgard, "Multimodal Deep Learning for Robust RGB-D Object Recognition," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015) pp. 681–687.
[39] L. Bo, X. Ren and D. Fox, "Unsupervised Feature Learning for RGB-D Based Object Recognition," **In:** *Experimental Robotics* (Springer, 2013) pp. 387–402.
[40] D. Maturana and S. Scherer, "Voxnet: A 3d Convolutional Neural Network for Real-Time Object Recognition," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015) pp. 922–928.
[41] PCL, "The point cloud library." http://pointclouds.org/.
[42] O. Kundu, "TCS grasping dataset." https://sites.google.com/view/grasping-tcs-research/home.
[43] A. Singh, J. Sha, K. S. Narayan, T. Achim and P. Abbeel, "Bigbird: A Large-Scale 3d Database of Object Instances," *2014 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2014) pp. 509–516.
[44] OnRobot, "RG2 Industrial Gripper." http://onrobot.dk/Product.html.
[45] J. Smisek, M. Jancosek and T. Pajdla, "3d with Kinect," **In:** *Consumer Depth Cameras for Computer Vision* (Springer, 2013) pp. 3–25.
[46] O. Kundu, "Demonstration video for the proposed grasping algorithm." https://www.youtube.com/watch?v=P9BIPXtnQrw (2017).
[47] O. Kundu, "Program source code on gitlab." https://gitlab.com/olyvia/primitives_fit.git.
[48] University of Berkeley, "Caffe deep learning framework." https://caffe.berkeleyvision.org/ (2020).

Cite this article: O. Kundu, S. Dutta and S. Kumar (2022). "A novel method for finding grasping handles in a clutter using RGBD Gaussian mixture models", Robotica **40**, 447–463. https://doi.org/10.1017/S0263574721000503