Localizing Handle-Like Grasp Affordances in 3D Point Clouds

Andreas ten Pas and Robert Platt

College of Computer and Information Science, Northeastern University
Boston, Massachusetts, USA

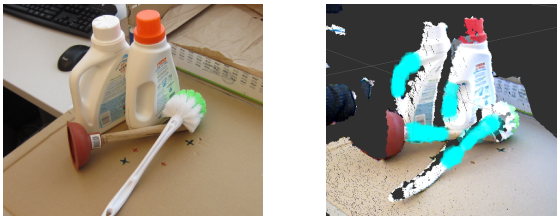
Abstract. We propose a new approach to localizing handle-like grasp affordances in 3-D point clouds. The main idea is to identify a set of sufficient geometric conditions for the existence of a grasp affordance and to search the point cloud for neighborhoods that satisfy these conditions. Our goal is not to find *all* possible grasp affordances, but instead to develop a method of localizing important types of grasp affordances quickly and reliably. The strength of this method relative to other current approaches is that it is very practical: it can have good precision/recall for the types of affordances under consideration, it runs in real-time, and it is easy to adapt to different robots and operating scenarios. We validate with a set of experiments where the approach is used to enable the Rethink Baxter robot to localize and grasp unmodelled objects.

Keywords: Grasping; 3-D point clouds; Grasp affordances; Handle grasping

1 Introduction

Robust robot grasping in novel and unstructured environments is an important research problem that has many practical applications. A key sub-problem is localization of the objects or object parts to be grasped. Localization is challenging because it can be difficult to localize graspable surfaces on unmodelled objects. Moreover, even small localization errors can cause a grasp failure. In this paper, we develop an approach to localization-for-grasping based on localizing parts of objects rather than localizing the entire object. We refer to these graspable object parts as *grasp affordance geometries*: object geometries that can be grasped in a particular way by a particular robot hand. Although the idea of a grasp affordance has existed in the literature for a long time [5], the idea has new promise now because the availability of accurate range sensing information (*i.e.* the Microsoft Kinect) may make grasp affordance localization easier. In this paper, we develop an approach to searching a 3-D point cloud for grasp affordance geometries.

The main idea is to identify a set of sufficient geometric conditions for the existence of a grasp affordance and to search the point cloud for neighborhoods that satisfy these conditions. Here, we concern ourselves with "handle-like" grasp affordance geometries. Our goal is not to find *all* possible grasp affordances, but



(a) (b)

Fig. 1. (a) An RGB image of a typical scene. (b) Handle-like grasp affordances localized using our algorithm highlighted in cyan.

instead to develop a method of localizing important types of grasp affordances quickly and reliably. Developing an efficient search is a key challenge. A complete handle configuration is determined by seven parameters and a brute force search of the point cloud would be infeasible in real time. We structure the search in two ways. First, we constrain the robot hand to grasp in a plane orthogonal to the minor principal curvature axis of the local object surface at the point where the grasp occurs. This constraint makes sense intuitively and ultimately enables us to reduce the search space down to three (spatial) dimensions. Second, we require a cylindrical gap to be present around an object surface to accommodate the grasping robot hand. This constraint enables us to eliminate many grasp candidates quickly. Figure 1 illustrates typical results of the overall process. The strength of this method relative to other current approaches is that it is very practical: it has good precision/recall for the types of affordances under consideration, it runs in real-time, and it is easy to adapt to different robots and operating scenarios. In addition, we have created an easy-to-use ROS package [15] that implements the algorithm and allows it to be used in most robotic manipulation operational scenarios.

2 Related Work

The problem of localizing graspable geometries has been the subject of extensive recent research. An important class of methods work by searching a height map or a range image for graspable regions. For example, Klingbeil *et al.* search for geometries in a range image that can be grasped by a parallel-jaw gripper [10]. A three-dimensional search (x, y, θ) is performed over the range image. The gripper is constrained to approach the object from a single direction. The work of Jiang *et al.* is related [8]. They search a registered RGBD image for regions that score high on a linear-in-the-features grasp score function, where feature weights were learned off-line. Closely related to the work of Jiang *et al.* is that of Fischinger and Vincze [4]. Rather than searching an RGBD image, they perform a 3-DOF search of a height map (calculated from a point cloud). They key element of this work is the introduction of a new type of feature used to develop

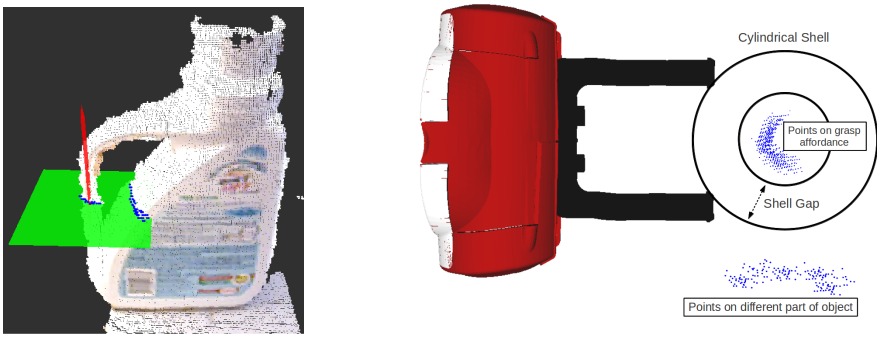
a graspability score function. Our current work is distinguished from the above chiefly because we do not use a depth image or height map to structure our search for grasp affordances, but we operate directly on the point cloud instead. This brings several advantages including the ability to structure the search in different ways, and a looser coupling between how the affordance was perceived and the approach direction of the arm. Overall, our grasp success rates are at least as good as those of any of the work mentioned above. However, it is important to remember that this success rate assumes that objects can always be grasped by a handle that is within reach of the robot.

Other work loosely related to the above includes that of Herzog *et al.*, who learn graspable height map "templates" based on user demonstrations of good and bad grasps [7]. Katz *et al.* develop a method that depends on physical interaction with the objects to be grasped [9]. The robot pushes the object under consideration and uses the resulting motion to perform segmentation accurately. The resulting system is very robust, but can require significant pushing interactions prior to grasping. Another line of current research approaches the problem of localization-for-grasping by searching for known modeled objects in a scene. Here, it is common to use feature-matching approaches. Appropriate 3-D features for use with point clouds include Fast Point Feature Histograms (FPFH) [11] and the SHOT feature [16]. It is typical to use RANSAC or Hough voting [17, 13] to align features found on an object model with features found in a scene. However, Glover and Popovic recently proposed a new method (loosely related to ICP [1]) that has demonstrated robustness advantages [6]. Often, the system may be ignorant of which object is present in a scene. Brook, Ciocarlie, and Hsiao develop a database-driven method that segments the point cloud into clusters and compare these clusters against 3D models in a database [2]. A Bayesian framework is used that incorporates uncertainty in object shape, object pose, and robot motion error.

3 Localizing Grasp Affordances

An enveloping grasp affordance is a handle-like object geometry that can be grasped by encircling it with the thumb and fingers of the robot hand. We locate these geometries in a 3D point cloud by searching for cylindrical shells that satisfy certain criteria with respect to local neighborhoods of the point cloud. A cylindrical shell is a pair of co-linear cylinders with different radii. We require the following conditions on the local point neighborhood to be satisfied:

1. Points near the center of the neighborhood must lie on a curved object surface (with respect to a parametrized threshold on curvature).
2. The axis of the cylindrical shell must be parallel to the secondary axis of curvature of the local object surface.
3. The gap between the inner and outer cylinders must contain zero points and be wide enough to contain the robot fingers.
4. The radius of the innermost cylinder must be no larger than the maximum hand aperture.



(a) (b)

Fig. 2. Illustration of the affordance search. (a) points in a local neighborhood are projected onto a plane orthogonal to the minor principal curvature axis of the object surface. (b) a shell is found that contains points within the inner circle but has a gap between the inner and outer circle.

If the above conditions are satisfied, we say that an enveloping grasp affordance exists in the corresponding configuration. These are sufficient conditions for an enveloping grasp in the sense that if we assume they are satisfied and if we assume that points lie densely on all object surfaces in the neighborhood and if we assume the neighborhood can be reached by the robot hand, then we know that an object can be grasped using an enveloping grasp. This is illustrated in Figure 2. In Figure 2(a), a locally curved surface has been found (at the root of the red arrow), and a plane has been drawn orthogonal to the secondary axis of curvature. Figure 2(b) shows the points after they have been projected onto the plane and a circular shell (a projection of the cylindrical shell) that satisfies the enveloping grasp affordance conditions.

Our overall algorithm has the following steps (see Algorithm 1). First, we randomly sample spherical point neighborhoods approximately two or three cm in radius. This is accomplished by sampling points uniformly at random from the cloud and then taking a point neighborhood about each sample (Step 3). Second, we fit an implicit quadratic function (in three variables) to each of these point neighborhoods using a least squares algebraic fit with Taubin normalization [14] (Step 4). As a result of fitting, we obtain an accurate measurement of the magnitudes and axes of principal surface curvature in the point neighborhood (Step 5). We eliminate from consideration all neighborhoods with an associated surface curvature below some parametrized threshold (Step 6), and project the point neighborhood onto the plane orthogonal to the axis of minor principal curvature (Step 7). Next, we fit a circle to the projected points (Step 8). We then fix the center of the shell to the center of the fitted circle and perform a 1-D search for cylindrical shells satisfying the enveloping grasp affordance conditions (Step 9). Last, given the found enveloping grasp affordances, we search for sets of affor-

Algorithm 1 Handle Localization

1: $\mathcal{A} = \emptyset$
2: **for** $i = 1$ to I **do**
3: Sample x uniformly from cloud; calculate point neighborhood about x.
4: Fit a quadratic surface S to point neighborhood.
5: Estimate the median curvature $\hat{\kappa}$ of S.
6: **if** $\hat{\kappa} > K$ **then**
7: Project point neighborhood onto orthogonal plane
8: Fit a circle to points in plane; calculate cicle center, c.
9: Search for cylindrical shell, a, centered at c.
10: **if** a is found **then**
11: $\mathcal{A} = \mathcal{A} \cup a$
12: **end if**
13: **end if**
14: **end for**
15: $\mathcal{H} \leftarrow$ findHandles($\mathcal{A}$).

dances that are roughly aligned and that exceed a minimum length (Step 15). Key elements of the algorithm are detailed in the subsections below.

3.1 Estimating Object Surface Curvature by Fitting an Implicit Quadratic Surface

In order to find high-curvature regions of the point cloud and to estimate the axes of curvature accurately, we fit an implicit quadratic surface in three variables to points in the local neighborhood. A quadratic can be described by $f(\mathbf{c}, \mathbf{x}) = 0$, where

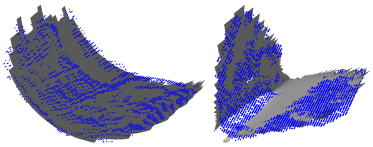
$$\begin{aligned} f(\mathbf{c}, \mathbf{x}) =& c_1 x_1^2 + c_2 x_2^2 + c_3 x_3^2 + c_4 x_1 x_2 + c_5 x_2 x_3 + \\ & c_6 x_1 x_3 + c_7 x_1 + c_8 x_2 + c_9 x_3 + c_{10}, \end{aligned} \tag{1}$$

and $\mathbf{c} \in \mathbb{R}^{10}$ denotes the parameters of the quadratic and $\mathbf{x} \in \mathbb{R}^3$ denotes the Cartesian coordinates of a point on the surface.

It turns out that there is no known fast (convex or closed form or *etc.*) method for finding the implicit quadratic surface that minimizes least squares geometric distances to a set of points (called the *geometric fit*). However, there do exist fast methods for solving for an *algebraic fit*, that is, a surface that solves the following optimization problem:

$$\min_c \sum_{i=1}^{n} f(\mathbf{c}, \mathbf{x}^i)^2 = \mathbf{c}^T M \mathbf{c}, \tag{2}$$

where $M = \sum_{i=1}^{n} l(\mathbf{x}^i) l(\mathbf{x}^i)^T$, $\mathbf{x}^1, \ldots, \mathbf{x}^n \in R^3$ are the points to which the curve is fitted,



(a) (b)

Fig. 3. Two examples of implicit quadratic surfaces fit using Taubin normalization.

and

$$l(\mathbf{x}) = (x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3, x_1, x_2, x_3, 1)^T.$$

To avoid the trivial solution $\mathbf{c} = 0$, it is necessary to impose constraints on this problem. Different constraints produce different results. One that seems to produce fits that are intuitively close to the geometric fit is known as Taubin's method [14]. Taubin's method sets the constraint $\|\nabla_{\mathbf{x}} f(\mathbf{c}, \mathbf{x}^i)\|^2 = 1$. Equation 2 is reformulated as the generalized Eigen decomposition, $(M - \lambda N)\,\mathbf{c} = 0$, where

$$N = \sum_{i=0}^{n} l_x(\mathbf{x}^i) l_x(\mathbf{x}^i)^T + l_y(\mathbf{x}^i) l_y(\mathbf{x}^i)^T + l_z(\mathbf{x}^i) l_z(\mathbf{x}^i)^T.$$

Here, $l_x(\mathbf{x})$ denotes the derivative of $l(\mathbf{x})$ taken with respect to x_1 and the other derivatives are defined similarly. The eigenvector corresponding to the smallest eigenvalue provides the best-fit parameter vector.

To fix the axis of the cylindrical shell to lie along the axis of minor principal curvature, we need to estimate the magnitude and direction of the curvature of the quadratic surface. The curvature at a particular point can be calculated by evaluating the shape operator[1] on the plane tangent to the point of interest. The eigenvectors of the shape operator describe the principal directions of the surface and its eigenvalues describe the curvature in those directions. This can be calculated for a point, $\mathbf{x}$, on the surface by taking the Eigenvalues and Eigenvectors of:

$$\left(I - N(\mathbf{x})N(\mathbf{x})^T\right) \nabla N(\mathbf{x}),$$

where $N(\mathbf{x})$ denotes the surface normals of the quadratic surface. It is calculated by differentiating and normalizing the implicit surface:

$$N(\mathbf{x}) = \frac{\nabla f(\mathbf{c}, \mathbf{x})}{\|\nabla f(\mathbf{c}, \mathbf{x})\|},$$

where

$$\nabla f(\mathbf{c}, \mathbf{x}) = \begin{pmatrix} 2c_1x_1 + c_4x_2 + c_6x_3 + c_7 \\ 2c_2x_2 + c_4x_1 + c_5x_3 + c_8 \\ 2c_3x_3 + c_5x_2 + c_6x_1 + c_9 \end{pmatrix}.$$

Once a quadratic is fit to a point neighborhood, we evaluate the median curvature of the quadratic surface in the point neighborhood. This is accomplished by randomly sampling several points from the local quadratic surface and calculating the maximum curvature (maximum of the two principal curvatures) magnitude at each of them. Then, we take the median of these maximum curvature values and accept as grasp affordance candidates all quadrics where the median curvature is larger than that implied by the hand capture radius. On the assumption that all enveloping grasp affordances will be located in a high-curvature neighborhood, we eliminate from consideration all neighborhoods with an associated surface curvature below some parametrized threshold.

[1] In general, the shape operator, S, can be calculated using the first and second fundamental forms of differential geometry: $S = \mathbf{I}^{-1}\mathbf{II}$.

It is important to note that rather than fitting a quadratic surface in order to calculate local curvature magnitudes and axes, an alternative is to estimate curvature from surface normals associated with each point in the neighborhood. This works as follows. Each point is associated with a surface normal, $\mathbf{n}_i \in S^3$. Then, an Eigen decomposition is performed for the following matrix: $\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T$, $i \in [1, n]$. The major principal curvature axis is determined to lie in the direction of the Eigenvector associated with the minimum Eigenvalue. The curvature magnitudes are approximated by taking ratios between the eigenvectors. Although this type of approach is somewhat common in point cloud processing [12], our experience informally indicates that the method we present here is better: it seems to be more accurate, it is less noisy, and it can be computed faster than estimating surface normals for a (potentially large) set of points.

3.2 Cylindrical Shell Search

Once the directions and magnitudes of the axes of principal curvature are estimated and low-curvature regions are eliminated, we search for cylindrical shells in three steps. First, we project the points in the local neighborhood onto the plane orthogonal to the minor principal curvature axis (see Figure 2(a)). Second, we calculate the center of the shell by fitting a circle to the points near the center of the neighborhood (*i.e.* points near the sampled point, x, in Step 3 of Algorithm 1). This is accomplished by minimizing algebraic distance as follows. Let x^i and y^i denote the two coordinates of the i^{th} point in the plane. Let h_x, h_y, and r denote the coordinates of the center and radius of the circle. We calculate:

$$\mathbf{w} = -\left(\sum_{i=1}^{n} \mathbf{l}_i \mathbf{l}_i^T\right)^{-1} \sum_{i=1}^{n} \lambda_i \mathbf{l}_i, \tag{3}$$

where $\lambda_i = (x^i)^2 + (y^i)^2$ and $\mathbf{l}_i = (-x^i, -y^i, 1)^T$. Then calculate the center and radius using: $h_x = -0.5a$, $h_y = -0.5b$, and $r = \pm\sqrt{h_x^2 + h_y^2 - c}$.

Once the best-fit circle is calculated, the third step is to fix the center of the shell to the center of the circle and search (brute-force 1-D search) over different radii for a shell such that the gap contains no points and the radius of the inner cylinder is less than the diameter of the robot hand (conditions 3 and 4 for the existence of an enveloping grasp affordance).

3.3 Handle Search

The presence of an enveloping grasp affordance guarantees that a grasp is possible in that configuration as long as all object surfaces in the local area are densely covered with points. Unfortunately, this is not always the case. The assumption is particularly problematic for objects that are hard for the range sensor to perceive. For example, the PrimeSense device does very poorly measuring distances to highly reflective surfaces such as the body of the pot shown in Figure 4(a).

(a) (b) (c)

Fig. 4. Illustration of handle search. (b) shows all grasp affordances found in the point cloud. (c) shows the handles found that satisfy alignment and minimum length constraints. The affordance search finds false positives on the surface of the pot and brush caused by measurement errors (the PrimeSense device fails to find accurate depths on reflective surfaces). However, they are eliminated in the handle search.

One way to mitigate this problem is to search for sets of enveloping grasp affordances that form "handles", *i.e.* sets of affordances that are roughly aligned and that cover some minimum length. This helps reduce the number of false positives. True enveloping grasp affordances are typically found aligned along object handles. False positives (caused by sensor error) are typically found in arbitrary configurations. Figure 4(b) and (c) shows an example where the handle search eliminates all false positives.

We search for handles using brute-force search over all pairs of enveloping grasp affordances. For each pair of grasp affordances, i and j, with centroids h_i and h_j, major principal axes v_i and v_j, and radii r_i and r_j, we compute the following three distances: $d_o = \|(I - v_i v_i^T)v_j\|$, $d_c = \|(I - v_i v_i^T)(h_i - h_j)\|$, and $d_r = |r_i - r_j|$. An enveloping grasp affordance i is considered to be aligned with affordance j if d_o, d_c, and d_r are below parametrized thresholds. If an enveloping grasp affordance i is aligned with at least a minimum number of other grasp affordances, then it is considered to define a handle affordance. The handles found using this method constitute the output of our algorithm (Step 14, Algorithm 1).

3.4 Sampling Strategy

Sampling plays a key role in our algorithm. As shown in Algorithm 1 (Step 3), the basic approach is uniform random sampling. We sample a point uniformly randomly from the point cloud and operate on the neighborhood of points around that sample. Our experience indicates that in the manipulation scenarios outlined in Section 4, 20,000 samples are sufficient to localize all handles in a scene. With 20,000 samples, Algorithm 1 takes approximately 1.7 seconds to execute (see Section 4.3). A natural way to speed things up is to use a more effective sampling strategy. Here, we explore a sequential importance sampling method that can be viewed as an implementation of the Cross Entropy Method [3]. The

method samples a fixed number of point neighborhoods in a series of rounds. In the first round, neighborhoods are chosen uniformly at random from the point cloud. After the first round, samples are drawn from a proposal distribution parametrized by the positions of the enveloping grasp affordances found in all prior rounds.

The form of the proposal distribution is a key choice that affects the performance of sampling. Here, we explore two variations on the Gaussian kernel density proposal distribution: a distribution expressed as a sum of Gaussians and a distribution expressed as a maximum over Gaussians. Let $x_i \in \mathbf{R}^3$, $i \in [1, n]$ denote the centroids of the n enveloping grasp affordances found in all prior rounds. The sum of Gaussians proposal distribution is:

$$g_{sum}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x|x_i, \Sigma),$$

where Σ is a constant parameter. The maximum of Gaussians proposal distribution is:

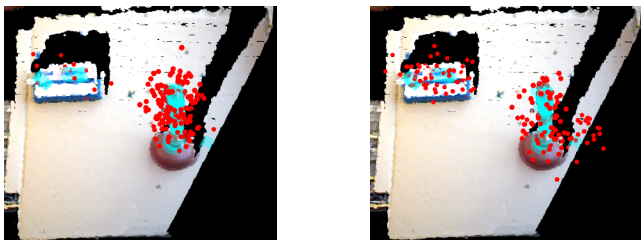
$$g_{max}(x) = \eta \max_{i \in [1,n]} \mathcal{N}(x|x_i, \Sigma),$$

where η is the normalization constant. It is relatively easy to sample from either of these proposal distributions. In order to draw k samples from g_{sum}, initialize $\mathcal{X} = \emptyset$ and do the following k times: choose an enveloping grasp affordance index, $j \in [1, n]$, uniformly randomly; 2) draw one sample from $\mathcal{N}(x : x_i, \Sigma)$ and add it to $\mathcal{X}$. Sampling from g_{max} is slightly more complicated. A method based on rejection sampling is shown in Algorithm 2.

Algorithm 2 Sampling from a distribution expressed as a maximum over Gaussians

1: $\mathcal{X} = \emptyset$
2: **for** $i = 1$ to n **do**
3: Choose i uniformly from $[1, n]$
4: Sample $\hat{x} \sim \mathcal{N}(x|x_i, \Sigma)$.
5: $m \leftarrow \max\{\mathcal{N}(\hat{x}|x_1, \Sigma), \mathcal{N}(\hat{x}|x_2, \Sigma), \ldots, \mathcal{N}(\hat{x}|x_n, \Sigma)\}$.
6: **if** $\mathcal{N}(\hat{x}|x_i, \Sigma) \geq m$ **then**
7: $\mathcal{X} \leftarrow \mathcal{X} \cup \hat{x}$.
8: **end if**
9: **end for**

These two distributions, g_{sum} and g_{max}, differ in the way that they "allocate" samples to particular regions of space (*i.e.* to regions about potential handle locations). g_{sum} allocates samples to a region in direct proportion to the number of grasp affordances that have been found in that region. This can be a problem if there are multiple handles present in a scene, but one handle is more likely to be populated by enveloping grasp affordances than the others (perhaps it is larger, longer, or is more densely covered with points in the cloud). In this case,



(a) (b)

Fig. 5. Illustration of difference in sampling strategy. (a) shows samples drawn from g_{sum}. (b) shows samples drawn from g_{max}. Notice that the distribution in (b) covers the two handles more evenly.

the handle where grasp affordances are more likely to be found is sampled even more densely on the next round. The result is that g_{sum} has a tendency to oversample some handles in the scene and ignore others. g_{max} corrects for this effect somewhat by sampling from all handle regions with a more even probability.

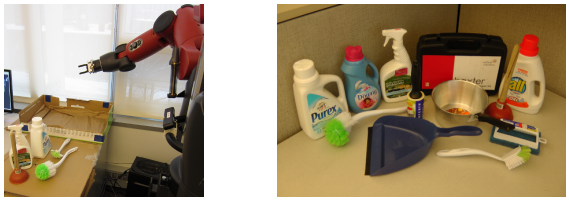
This difference is illustrated in Figure 5. Suppose that on a particular round of sampling, the algorithm has found all of the enveloping grasp affordances shown in cyan. Figure 5(a) shows a set of 100 samples drawn from g_{sum} and Figure 5(b) shows the same number of samples drawn from g_{max}. Notice that the distribution drawn from Figure 5(a) samples the object on the right more densely than the object on the left. This is because the object on the right was more densely covered with enveloping grasp affordances on prior rounds. Figure 5(b) shows that samples drawn from g_{max} cover both objects more evenly.

4 Experiments

4.1 Experimental Setup

We performed grasping experiments using the Rethink Robotics robot, Baxter. An Asus XTion Pro range sensor, mounted near the bottom of the robot's "chest", was used to sense a 3D point cloud containing the objects in front of the robot (see Figure 6(a)). A typical grasp was performed as follows. First, the Asus range sensor captured a range image of the target objects that was immediately converted to a 3D point cloud. Second, our algorithm was run for this point cloud and handle affordances were localized. Third, the right arm reached toward the handle closest to the base of the right arm. The arm was moved to a position such that a point between the two gripper fingers was placed at the handle centroid. The target orientation was such that the gripper was perpendicular to the handle axis and an axis pointing outward from the gripper was co-linear with a line between the handle and the base of the right arm. After reaching the target pose, the gripper was closed, the object lifted into the air, and transported to a large box where the object was deposited. If a grasp failed

on the first attempt, the robot continued to try to grasp by repeating this process. During each motion, the arm followed a straight line through configuration space.



(a) (b)

Fig. 6. (a) Typical grasping scenario. (b) The 12 objects used in our experiments. Notice that all objects have handles.

(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Fig. 7. Illustration of a typical clear-the-table experiment.

We tested our localization and grasping algorithms in two ways. First, we performed a series of 12 single-object grasp trials for each of the 12 objects (shown in Figure 6(b)) where each object was presented by itself. On each trial, the robot repeatedly attempted to grasp the presented object until either the object was grasped or it was pushed out of range. A grasp trial was run for each object in four different orientations at three different positions. Objects were placed such that a significant number of points on the handle were visible to the Asus range sensor and such that the handle was within the workspace of the robot's right arm. Second, we performed a series of 10 clear-the-table trials where we evaluated the capability for our approach to grasp a series of objects

in the presence of clutter. On each clear-the-table trial, the robot attempted to clear five objects (selected from the set shown in Figure 6(b)). Figure 7 shows a typical run of a clear-the-table experiment.

4.2 Localization Results

The single-object experiments indicate that our approach is capable of robustly grasping objects with handles. Table 1 shows the results. Out of the 12 grasp trials for each object, the table shows the number of successful grasps performed on the first try (column 2), by the second try (column 3), and by the third try (column 4). Notice that our method successfully grasped each object on the first try approximately 85% of the time. By the third try, it had nearly perfect grasp success. The only exception was for the Carrying Case where the object was pushed out of the workspace during a failed grasp attempt (collision between gripper and target object). Table 2 shows the results of ten clear-the-table experiments. The results show that our method sometimes failed to grasp one of the five presented objects. They also show that it sometimes took up to eight grasp attempts before all five objects were grasped.

Table 1. Results for the single-object experiments

Object	Grasped on 1st attempt	Grasped on 2nd attempt	Grasped on 3rd attempt
Blue Bottle	10/12	10/12	12/12
White Purex Bottle	11/12	12/12	12/12
White All Bottle	9/12	12/12	12/12
Carrying Case	11/12	11/12	11/12
Brush 1	10/12	11/12	12/12
Pot	11/12	12/12	12/12
Plunger	11/12	12/12	12/12
Sprayer	11/12	12/12	12/12
Dust Pan	11/12	12/12	12/12
Brush 2	8/12	12/12	12/12
Sponge	8/12	12/12	12/12
Lint Roller	11/12	12/12	12/12

Table 2. Results for the clear-the-table experiments

Trial num:	1	2	3	4	5	6	7	8	9	10
Number of objects grasped out of total objects:	5/5	4/5	5/5	4/5	5/5	4/5	5/5	5/5	4/5	5/5
Total grasp attempts:	5	5	5	7	7	6	5	5	5	8

4.3 Algorithm Runtime

This number is a conservative estimate of the maximum number of neighborhoods needed to localize all handles in our application scenarios. The algorithm was implemented in C++ on an Intel i7 3.5GHz system (four physical CPU cores) with 8GB of system memory. Runtime was averaged over 10 runs. The results are shown in Figure 8. As they show, total runtime is a little more than

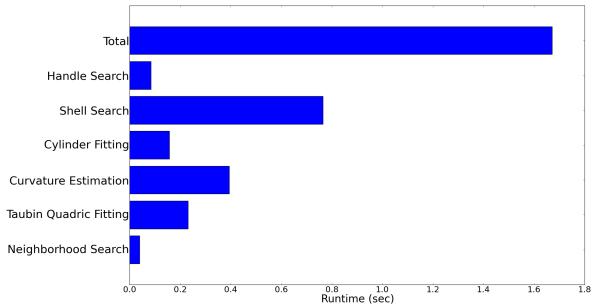


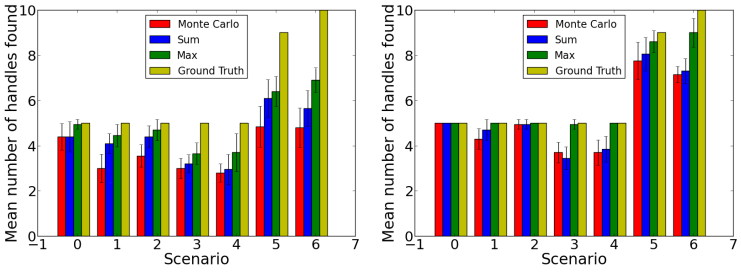
Fig. 8. Runtime of the localization algorithm for 20,000 samples averaged over 10 runs.

0.5Hz with the majority of the time taken by the brute-force 1-D shell search. We suspect that a closed-form approximation to the brute-force search exists that would reduce this time. Nevertheless, we expect this runtime to be fast enough for most application scenarios.

4.4 Comparison of Different Sampling Strategies

We also performed experiments to evaluate the number of handles in a scene missed by the algorithm as a function of the number of neighborhoods (I in Algorithm 1) and as a function of the sample strategy used. We tested with point clouds from seven scenes. The first five scenes contained exactly five different handles each. The last two scenes contained nine and ten handles, respectively. On each of these seven scenes, we tested the performance of our algorithm using three different sample strategies: uniform random Monte Carlo (MC), sequential importance sampling with g_{sum}, and sequential importance sampling with g_{max}. For each sample strategy, we performed experiments with 2000 and 5000 sampled neighborhoods. For uniform random MC we just sampled 2000 or 5000 samples in one batch. For sequential importance sampling with 2000 samples, we sampled 1000 neighborhoods in the first round and then 100 more neighborhoods in each of ten successive rounds. For sequential importance sampling with 5000 samples, we sampled 2000 samples in the first round and then 300 samples in each of 10 successive rounds.

Figure 9 show the results. Each bar shows the mean and standard deviation of 20 runs in the corresponding test scenario. The ground truth bar (yellow) shows the actual number of handles present in each scene. These results indicate the following. First, our method can be expected to find two or three handles in any scene with as few as 2000 samples using any sampling method. This is sufficient for some tasks (such as table clearing), where it is only necessary to grasp one object at a time. However, even 5000 sampled neighborhoods might not be enough to find all handles in a complex scene, especially if uniform random Monte Carlo is used. We found that it was necessary to use as many as 20000 sampled neighborhoods in order to localize all handles using this method. The



(a) 2000 sampled neighborhoods (b) 5000 sampled neighborhoods

Fig. 9. Performance comparison between the three sampling strategies for 2000 (a) and 5000 (b) samples, averaged over 20 runs. The error bars show the standard deviation.

results also indicate that it is generally better to use a sequential sampling method. Moreover, the results show that sequential importance sampling using the g_{max} proposal distribution has the best performance. This strategy finds nearly all handles with 5000 sampled neighborhoods.

5 Conclusions

The paper proposes a new approach to localizing handle-like grasp affordances in 3-D point clouds. The core of the idea is to identify sufficient geometric conditions for the existence of a class of grasp affordances and to search the point cloud for point neighborhoods where these conditions are satisfied. Our work makes use of an approach to implicit quadratic curve fitting that (to our knowledge) has not been used in the robotics literature. Our reported results show high grasp success rates similar to those reported in Klingbeil *et. al.* [10] and Fischinger *et. al.* [4]. Moreover, our method has important advantages relative to other approaches including fast run time, the ability to operate on 3D point clouds rather than range images or height maps, and the ability to localize handles. In our single-object experiments, nearly all grasp failures were caused by attempting to grasp false positives found because of depth measurement errors or because of insufficient point density on object surfaces in the neighborhood of the false positive. For example, the grasp failures of Brush 2 were caused mainly by the algorithm localizing the brush part of the object because of significant measurements errors in that area. Our clear-the-table experiments also suffered from localization failures. However, there, the effects of localization errors were more serious because of the clutter. A failed attempt to grasp one object sometimes pushed other objects out of the workspace such that a complete clearing of the table became impossible. In general, we found the grasping process to be very robust as long as multiple re-grasp attempts were allowed. Overall, the results in Tables 1 and 2 indicate that our approach is practical for many real robot application scenarios. We have incorporated our work into a ROS package [15].

Acknowledgements. This work was supported in part by NASA under Grant No. NNX13AQ85G and ONR under Grant No. N000141410047.

References

1. P. Besl and N. McKay. A method for registration of 3d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239256, 1992.
2. P. Brook, M. Ciocarlie, and K. Hsiao. Collaborative grasp planning with multiple object representations. In *IEEE Int'l Conf. on Robots and Automation*, 2011.
3. Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.
4. D. Fischinger and M. Vincze. Empty the basket - a shape based learning approach for grasping piles of unknown objects. In *IEEE Int'l Conf. on Intelligent Robot Systems*, 2012.
5. J. Gibson. *The Ecological Approach To Visual Perception*. Psychology Press, 1979.
6. J. Glover and S. Popovic. Bingham procrustean alignment for object detection in clutter. In *IEEE Int'l Conf. on Intelligent Robot Systems*, 2013.
7. A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. Template-based learning of grasp selection. In *IEEE Int'l Conf. on Robotics and Automation*, 2012.
8. Y. Jiang, S. Moseson, and A. Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *IEEE Int'l Conference on Robotics and Automation*, 2011.
9. D. Katz, M. Kazemi, D. Bagnell, and A. Stentz. Clearing a pile of unknown objects using interactive perception. In *IEEE Int'l Conf. on Robotics and Automation*, 2013.
10. E. Klingbeil, D. Rao, B. Carpenter, B. Ganapathi, A. Ng, and O. Khatib. Grasping with application to an autonomous checkout robot. In *IEEE Int'l Conf. on Robotics and Automation*, 2011.
11. R. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE Int'l Conf. on Robots and Automation*, 2009.
12. R. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Intl. Conference on Robotics and Automation*, 2011.
13. M. Sun, B. Xu, G. Bradski, and S. Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *European Conference on Computer Vision*, 2010.
14. G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Trans. PAMI*, 13:1115–1138, November 1991.
15. A. ten Pas and R. Platt. Handle detector ROS package. *http://wiki.ros.org/handle_detector*.
16. F. Tombari, S. Salti, and L. Stefano. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision*, 2010.
17. F. Tombari and L. Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *Pacific-Rim Symposium on Image and Video Technology*, 2010.