# Real-time Grasp Pose Estimation for Novel Objects in Densely Cluttered Environment

Mohit Vohra, Ravi Prakash, and Laxmidhar Behera, *Senior Member, IEEE*

***Abstract*— Grasping of novel objects in pick and place applications is a fundamental and challenging problem in robotics, specifically for complex-shaped objects. It is observed that the well-known strategies like *i*) grasping from the centroid of object and *ii*) grasping along the major axis of the object often fails for complex-shaped objects. In this paper, a real-time grasp pose estimation strategy for novel objects in robotic pick and place applications is proposed. The proposed technique estimates the object contour in the point cloud and predicts the grasp pose along with the object skeleton in the image plane. The technique is tested for the objects like ball container, hand weight, tennis ball and even for complex shape objects like blower (non-convex shape). It is observed that the proposed strategy performs very well for complex shaped objects and predicts the valid grasp configurations in comparison with the above strategies. The experimental validation of the proposed grasping technique is tested in two scenarios, when the objects are placed distinctly and when the objects are placed in dense clutter. A grasp accuracy of 88.16% and 77.03% respectively are reported. All the experiments are performed with a real UR10 robot manipulator along with WSG-50 two-finger gripper for grasping of objects.**

## I. Introduction

With the rapid growth in the e-commerce world, order volumes are increasing tremendously. To fulfill customer requirements, many industries and researchers have shifted their interest in warehouse automation. One of the main components of warehouse automation is the autonomous grasping of objects. Grasping of the object is a two-step procedure: perception and planning. The goal of the perception module is to estimate the grasp configurations either by estimating the pose of the object or by segmenting the object region and using some post-processing steps. The goal of the planning module is to generate the required trajectory and motor torques to achieve the desired configuration. Traditional grasping techniques evaluates the grasp configuration according to some metric [1] [2] [3] on the assumption of simplified contact model, Coulomb friction and accurate 3D object model. But these methods often fail in the practical scenario due to inconsistencies in the simplified object and contact models.

On the other hand, data-driven method [4] [5] has shown some significant results on the real data but requires an extensive training data set. In [6], the author developed a data set including millions of point cloud data to train a grasp quality CNN (GQ-CNN) with analytic metric and used GQ-CNN to select the best grasp plan which achieves 93% success rate with eight types of known objects. [7] requires 50k grasping attempts to generate the training data set while [8] used 14 robots to randomly grasp over $800,000$ times and collected grasping data to train CNN that teach robots to grasp. The main drawback of the data-driven methods is the poor performance on the novel object *i.e.* objects which are not present in the data set, and in the warehouse industry because of the large variety of objects, it is difficult to include each one in the data set. Hence these methods are not reliable for grasping of novel objects.

All authors are with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Uttar Pradesh 208016, India. Email Ids: (mvohra, ravipr, lbehera)@iitk.ac.in.

Fig. 1: Grasping of the object

Various solutions have proposed in the literature for the grasping of novel objects. In [9], the author has formulated the grasping problem as a deep Reinforcement learning problem and has successfully demonstrated for unknown item with sampled grasp configurations. In [10] [11], author first samples the several thousand grasp candidates and then use CNN for selecting the best grasp configuration. Similarly, [12] segments the object region in the image with the assumption that the object surface is convex, and select the best grasp configuration among the sampled grasp configuration. One of the main drawbacks of the above method is that the final grasp configuration will always be selected from the set of sampled grasp configurations. Hence there is no surety that the final grasp configuration is optimal. So to get the near optimal grasp configuration, thousands of samples need to be evaluated, hence requires massive computation.

Various solutions have been proposed in the literature for directly predicting the grasp configurations (*i.e.* no sampling is required). In [13], authors develop a supervised learning framework with manually designed features as inputs, for predicting the affordances (push, pull or grasp) for each

object, and selects the best affordance and its associated action to execute. [14] trains a deep network on millions of synthetic images for predicting the grasp configuration. With the idea of domain randomization authors has demonstrated for a single and isolated object, that network trained on synthetic data can perform well on real data. In [15], the author develops an approach which can localize the handle-like grasp affordances in 3D point cloud data by satisfying the geometric constraints. [16] model the pose computation problem as a nonlinearly constrained optimization problem with the object and gripper both are modeled as superquadric functions.

In this paper, we consider the problem of grasping of novel objects. Our framework consists of estimating the boundary points in the point cloud data, segmenting the object region in the image plane, estimate the skeleton the object region, followed by predicting the grasp configuration at each skeleton location and selection of best grasp configuration. The main advantage of our approach is that this approach is simple and can easily be adapted for other grippers as well. The second advantage is that our framework does not require any grasp configuration sampling step, which makes it more stable. Here stable means that for the same input, the developed framework will predict the same grasp configuration while sampling based method will produce different grasp configuration for each trial.

The remainder of this paper is organized as follows. Gripper representation and problem formulation are presented in section II. A detailed explanation of our method is given in Section III. Experimental results for robotic grasping and quantitative results are mentioned in section IV. The paper is finally concluded in Section V.

## II. Gripper representation and Problem Statement

### A. Gripper representation

We represent two-finger gripper as a rectangle in the image plane, shown in Fig. 2. This representation of gripper is taken from [17] where length represents the maximum opening between the fingers, breadth represents the thickness of gripper fingers, the centroid of the rectangle is the palm of the gripper, and $\boldsymbol{\alpha}$ represents the orientation of gripper. Thus gripper configuration is represented by three parameters *i.e.* $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\alpha}$ where $\boldsymbol{x}$, $\boldsymbol{y}$ represents the centroid pixel location and $\boldsymbol{\alpha}$ represents the orientation.

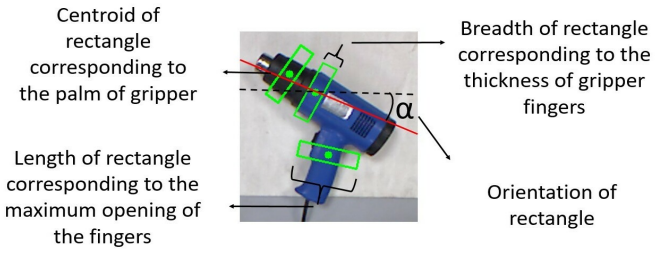



Fig. 2: Gripper representation

### B. Problem Statement

Given the current image $\mathcal{I}$ and Point cloud $\mathcal{P}$ of the scene or workspace, find the gripper configuration ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\alpha}$) to grasp the objects present in the workspace. Main steps in our approach are to estimate the skeleton of the object and at each skeleton location predict the grasp configuration using local object skeleton structure and for selecting the final grasp configuration, point cloud data corresponding to the rectangular region (*i.e.* gripper representation) will be used. A detailed explanation of each step is given in section III.

## III. Proposed Methodology

The proposed method for estimating the grasp configuration is a four-step procedure. In the first step, boundary points of the object are estimated using the point cloud data, and the corresponding object region is estimated in the image plane. In the second step, centroid and skeleton of the object are estimated. In the third step, grasp rectangles at each skeleton point are estimated, and in the final step, point cloud data corresponding to the grasp rectangle part and the centroid of the object is used to decide the final grasp rectangle or grasp configuration. Complete pipeline for estimating the final grasp configuration is shown in Fig. 3.

### A. Boundary points and object contour

*1) Boundary points in point cloud:* Let $\mathcal{P}$ be the raw point cloud data, $\boldsymbol{r_s}$ represents the user defined radius and $\boldsymbol{t_h}$ is user defined threshold value. Let $\boldsymbol{\mathcal{E}}$ is the point cloud data which contains the boundary points. To decide if a given query point $\mathbf{p_i} \in \mathcal{P}$ is a boundary point or not, we calculate $\boldsymbol{\mathcal{R}}(\mathbf{p_i})$, which is defined as set of all points inside a sphere, centered at point $\mathbf{p_i}$ with radius $\boldsymbol{r_s}$. In point cloud, this is achieved through a k-dimensional (K-d) tree. We call each point in set $\boldsymbol{\mathcal{R}}(\mathbf{p_i})$ as neighboring point of $\mathbf{p_i}$ and it can be represented as $\boldsymbol{\mathcal{R}}(\mathbf{p_i}) = \{\mathbf{n_1}, \mathbf{n_2}, \ldots, \mathbf{n_k}\}$. For each query point $\mathbf{p_i}$ and neighboring point $\mathbf{n_j}$ we calculate the directional vector as

$$\mathbf{d}(\mathbf{p_i}, \mathbf{n_j}) = \mathbf{n_j} - \mathbf{p_i} \tag{1}$$

$$\hat{\mathbf{d}}(\mathbf{p_i}, \mathbf{n_j}) = \frac{\mathbf{d}(\mathbf{p_i}, \mathbf{n_j})}{\|\mathbf{d}(\mathbf{p_i}, \mathbf{n_j})\|} \tag{2}$$

Then we calculate the resultant directional vector $\hat{\mathbf{R}}(\mathbf{p_i})$ as sum of all directional vector and normalize it

$$\mathbf{R}(\mathbf{p_i}) = \sum_{j=1}^{k} \hat{\mathbf{d}}(\mathbf{p_i}, \mathbf{n_j}), \hat{\mathbf{R}}(\mathbf{p_i}) = \frac{\mathbf{R}(\mathbf{p_i})}{\|\mathbf{R}(\mathbf{p_i})\|} \tag{3}$$

We assign a score $\boldsymbol{s}(\mathbf{p_i})$ to each query point $\mathbf{p_i}$ as average of dot product between $\hat{\mathbf{R}}(\mathbf{p_i})$ and $\hat{\mathbf{d}}(\mathbf{p_i}, \mathbf{n_j})$ for all neighboring points.

$$\boldsymbol{s}(\mathbf{p_i}) = \frac{\sum_{j=1}^{k} \hat{\mathbf{R}}(\mathbf{p_i}) \cdot \hat{\mathbf{d}}(\mathbf{p_i}, \mathbf{n_j})}{\boldsymbol{k}} \tag{4}$$

If $\boldsymbol{s}(\mathbf{p_i})$ is greater than some threshold $\boldsymbol{t_h}$ then $\mathbf{p_i}$ is considered as a boundary point else not. The boundary points in point cloud data is shown in Fig. 3, step 1 output.

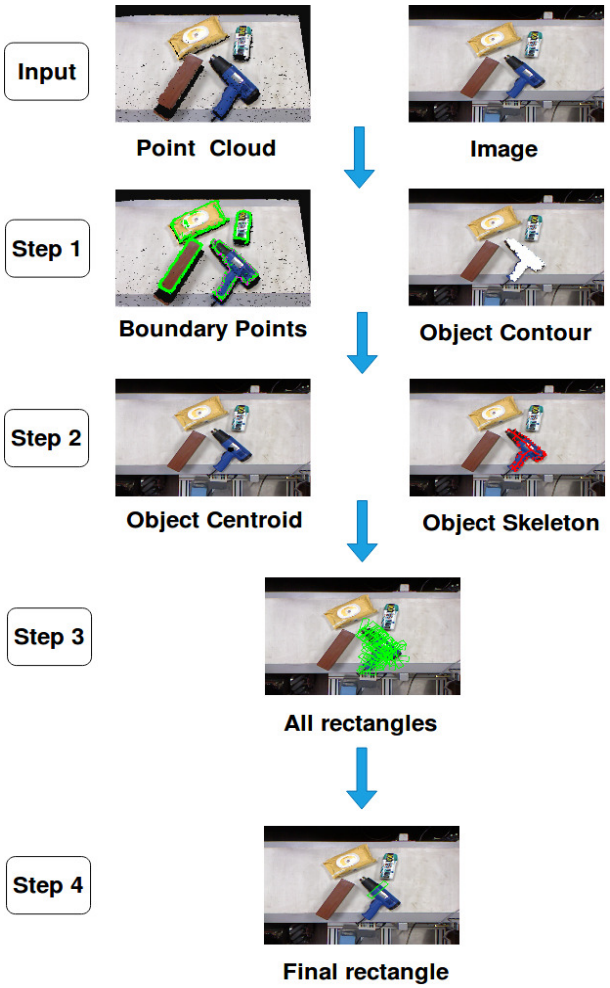



Fig. 3: Complete Pipeline

*2) Close contour in point cloud:* Once we got the boundary points, the next step is to select the boundary points, which forms a closed loop. This step ensures that unwanted boundary points because of sensor noise or because of any sharp features on the surface of the object can be ignored, thus avoiding the unnecessary computation. Following are the steps to find the closed loop points:

- Select the the starting point $\mathbf{s_i}$ with lowest $z$ value (closest to Kinect).
- Let the nearest neighbour of $\mathbf{s_i}$ be $\mathbf{n_j}$.
- Remove $\mathbf{s_i}$ from boundary point set and store $\mathbf{s_i}$ in an array $\mathbf{A}$.
- Find the nearest neighbour of $\mathbf{n_j}$, since we have removed $\mathbf{s_i}$, so we will get the nearest neighbor of $\mathbf{n_j}$ different from $\mathbf{s_i}$.
- Store $\mathbf{n_j}$ in $\mathbf{A}$ and repeat the step $\mathbf{3}$ and $\mathbf{4}$ with $\mathbf{n_j}$ till
  - Nearest neighbor is same as $\mathbf{s_i}$, hence all points in $\mathbf{A}$ forms a close loop.
  - If there is no neighbor, then remove last element from $\mathbf{A}$, find the neighbor for last element of $\mathbf{A}$ and repeat the step $\mathbf{3}$ and $\mathbf{4}$.

The boundary points which forms a closed loop are shown in pink color (Fig. 3, step 1 output)

*3) Object region in image plane:* Since Kinect gives the registered point cloud data, so we can find the pixel locations corresponding to the closed loop boundary points and using the standard flood fill algorithm we will find the object region in the image plane which is shown in Fig. 3, step 1 output.

### *B. Centroid and skeleton of object in image plane*

*1) Object Centroid:* Output of a flood fill algorithm is a binary mask with white pixels represents the object region and black pixels represents the non object region. We define centroid of the object as the average of the object pixel locations which is shown in Fig. 3, step 2 output.

*2) Skeleton of object:* In this work, we mainly focus on grasping the objects vertically using two-finger gripper. Hence we assume that grasping along the central axis of the object surface, has a high probability of successful grasping as compared to randomly selecting the grasp configurations. To find the central axis of the object surface, we find the skeleton of the object region in the image plane. Skeletonization reduces the foreground regions to a skeletal remnant that largely preserves the shape and connectivity of the original regions. Since the output of skeletonization is pixel locations which are along the central axis of the object region, hence this step provides several pixel locations for calculating the grasp configurations, instead of only the centroid pixel. Result of skeletonization is shown in Fig. 3, step 2 output.

### *C. Grasp rectangles at each skeleton point*

Since a gripper is represented by three parameters (center of the rectangle, $\boldsymbol{x}$, $\boldsymbol{y}$, and orientation of the rectangle $\boldsymbol{\alpha}$), here skeleton pixel represents the center and to find the orientation, we fit the straight line on the local skeleton structure around the skeleton pixel using orthogonal linear regression[1].

Let $\boldsymbol{q_j}$ represents the skeleton pixel location where we want to estimate the grasp configuration. For visualization, a white circle is drawn at a pixel location of $\boldsymbol{q_j}$, which is shown in Fig. 4. All skeleton pixels inside the circular region is local skeleton structure around $\boldsymbol{q_j}$. To find the slope of line let us assume that $\boldsymbol{p_i}$ represents the $\boldsymbol{i^{th}}$ skeleton pixel inside the circular region, $\boldsymbol{p_i(r)}$, $\boldsymbol{p_i(c)}$ represents the row and column number of $\boldsymbol{p_i}$ respectively and $\boldsymbol{n}$ be the total number of pixels inside the circular region.

$$\mu_r = \frac{\sum_{i=1}^{n} p_i(r)}{n} \tag{5}$$

$$\mu_c = \frac{\sum_{i=1}^{n} p_i(c)}{n} \tag{6}$$

$$\sigma_r = \sum_{i=1}^{n} (\mu_r - p_i(r))^2 \tag{7}$$

$$\sigma_c = \sum_{i=1}^{n} (\mu_c - p_i(c))^2 \tag{8}$$

[1]https://www.codeproject.com/Articles/576228/Line-Fitting-in-Images-Using-Orthogonal-Linear-Reg

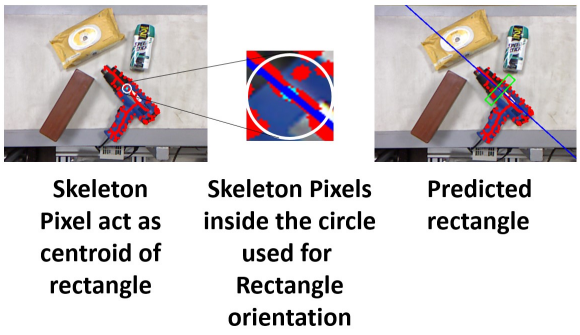



Fig. 4: Grasp Rectangle

$$\sigma_{rc} = \sum_{i=1}^{n} (\mu_r - p_i(r))(\mu_c - p_i(c)) \tag{9}$$

$$b = \frac{\sigma_r - \sigma_c + \sqrt{(\sigma_r - \sigma_c)^2 + 4\sigma_{rc}^2}}{2\sigma_{rc}} \tag{10}$$

$$slope = \tan^{-1}(b) \tag{11}$$

For visualization, we have shown the line with calculated slope passing through the $q_j$ and grasp rectangle at $q_j$ with an orientation of rectangle is same as the slope of the line which is shown in Fig. 4.

### D. Final grasp selection

Once we have the number of possible grasp configurations in the image plane, as shown in Fig. 3, step 3 output, next step is to select the final grasp configuration. For each grasp configuration, we partition the point cloud data into three regions *i.e.* object region (white) and two non object regions (red and blue) as shown in Fig. 5. We consider two conditions for configuration to be valid

- Object region must always exist in between the two non object regions.
- Difference between the $z$ values of each point in non object region and $z$ value at centroid should be greater than some threshold value *i.e.* $z(p_i) - z(centroid) > th$, where $p_i$ is any point in the non object region. This threshold mainly depends on the length of the gripper's finger.

Above conditions filter out the grasp configurations at the boundary of objects which has a high chance of gripper finger collisions with other objects. From the filtered configurations, we select the grasp configuration which is closest to the object centroid by calculating the distance between the object centroid and grasp rectangle centroid.

## IV. Experiments and Result

### A. Experimental Setup

Our robot platform setup shown in Fig. 6 consists of a UR10 robot manipulator with its controller box (internal computer) and a host PC (external computer). The UR10 robot manipulator is a 6 DOF robot arm designed to safely work alongside and in collaboration with a human. This arm

(a) Image (b) Point Cloud

Fig. 5: Grasp configurations

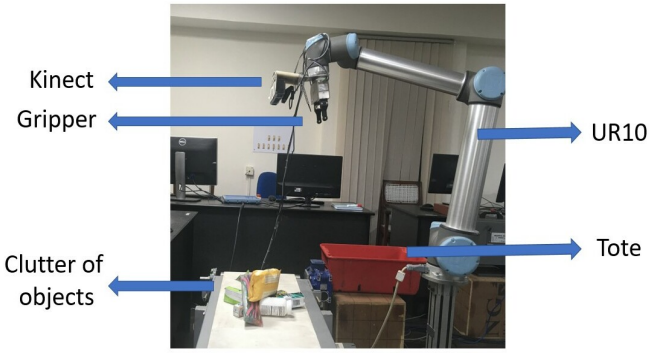



Fig. 6: Hardware Setup

can follow position commands like a traditional industrial robot, as well as take velocity commands to apply a given velocity in/around a specified axis. The low level robot controller is a program running on UR10's internal computer broadcasting robot arm data, receiving and interpreting the commands and controlling the arm accordingly. There are several options for communicating with the robot low level controller to control the robot including the teach pendent or opening a TCP socket (C++/Python) on a host computer. We used eye-in-hand approach i.e. the vision hardware consisting of RGB-D Microsoft Kinect sensor mounted on the wrist of the manipulator. The WSG 50 robotic parallel gripper made by SCHUNK, is used to grasp the objects. ROS drivers are used to communicate between the sensor, manipulator and the gripper.

### B. Motion Control Module

Order in which the the actions are executed to grasp the objects is shown in Fig. 7.

***i) Calling vision service:*** In first step vision service estimates the grasp configuration ($x$, $y$, $\alpha$). Here $x$, $y$, are the pixel coordinates, so from the registered point cloud data we can get the 3D location and since we always grasp the object vertically, hence $\alpha$ represents the angle of rotation around the gripper axis. Hence from $x$, $y$, $\alpha$, we calculate the final pose of the end effector.

***ii) Grasp Object:*** This task defines the trajectory from current pose to the final grasp pose estimated from vision service. Since final pose will always exist at the object surface, so to grasp the object vertically we define one intermediate trajectory point which is at some fixed height and use ROS modules to generate trajectory that starts at current pose, passes through intermediate point and ends at

(a) Calling vision service (b) Grasp Object (c) Lift object (d) Place object (e) View Pose

Fig. 7: Execution steps during robotic manipulation


Fig. 8: Objects used for experiments

final pose. After execution of the trajectory gripper close command is triggered to grasp the object.

***iii) Lift object:*** Once the object is grasped, we lift the object vertically up at some fixed height.

***iv) Place object:*** Next task is to place the objects in a tote. In our experiments we have fixed the tote and fixed joint angle motion is executed to reach the tote. After execution of the joint angle motion, gripper open command is triggered to place the object.

***v) View Pose:*** Final task is to reach at a desired joint space configuration so that the objects are in the proper view of the vision sensor. This motion is also the fixed joint angle motion and after the execution of the joint angle motion, vision service will be called and entire process will repeat till we clear the clutter. Video link for the real experiment is video link.

### C. Objects Used for grasping

We have used **15** objects for our experiments which is shown in Fig. 8. Most of the objects are very common. Objects are chosen in such a way that the thickness of object part should not be greater than the maximum opening of the gripper. We deliberately included the blower (blue gun like object) to test our algorithm for a complex shape object. For our experiments we randomly select the objects from the object set and place the objects either separately or in a clutter.

### D. Parameters setting

***i) Boundary points:*** Based on several trials we found that for extracting boundary points in point cloud data, $r_s$= 0.02m and $t_h$ = 0.35 are optimal values. It is because if we set $r_s$ at a minimal value, then the quality of the boundary points will be inferior and if we set $r_s$ at huge value than quality of boundary points will be very good but computation time will be very high. Similarly, if we set $t_h$ very low, then the quality of boundary points is inferior because almost every point in the point cloud data will be considered as a boundary point. If we set $t_h$ very high, then the only small number of point cloud data will be considered as boundary points, and hence there be less probability of finding the closed loop boundary points.

***ii) Object region in image plane:*** To find the object region or mask in the image plane, we use the flood fill algorithm with a starting point or seed point as the average of pixels corresponding to the closed loop boundary points. Because of the complex shape of objects, there is a high chance that seed point can exist outside the actual object region. So to get the proper object region, we count the number of pixels corresponding to the object region in the mask, and if the number of pixels is more than the half of the image size, then we invert the mask else remains the same.

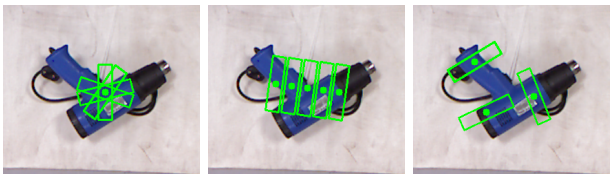
***iii) Object Skeleton:*** To find the object skeleton, we use the standard morphological operations defined in OpenCV library *i.e.* dilation and erosion with default parameter values.

***v) Grasp Rectangle:*** Dimension of the rectangle depends on the maximum opening of the gripper and the thickness of the gripper finger. For WSG-50 gripper, the maximum opening is of 10cm, and the thickness of the finger is 2.5cm. In our experiments, during vision service, Kinect is at the height of 0.8m from the flat surface where objects will be placed, so we draw a rectangle of 10cm by 2.5cm on the flat surface and find its projection in the image plane which gives a rough dimension of the grasp rectangle. In our experiments, we set length of the rectangle at **80** pixels and width at **20** pixels.

***v) Grasp Selection:*** We define a grasp configuration valid if difference between the $z$ values of each point in non object region and $z$ value at centroid is greater than some threshold value *i.e.* $z(p_i) - z(centroid) > th$, where $p_i$ is any point in the non object region and we set $th$ to be 3cm.

### E. Results and comparison

*1) Results:* To test our method, we perform 25 rounds of experiments with 5 rounds when objects are placed separately, and in 20 rounds, we place objects in clutter. In the case of isolated objects, our method performs very well for larger and thicker objects. We believe that this limitation is because of the Kinect sensor noise. So if we use Kinect V2 or Ensenso, which produces high-resolution point cloud data then our method can estimate the grasp regions for thinner objects as well. Overall we got a grasping



(a) Strategy 1 (b) Strategy 2 (c) Proposd strategy

Fig. 9: Grasp configurations for different strategies

accuracy of 88.16% when objects are placed separately, and in case of clutter, grasp accuracy drops to 77.03%. This is because most of the objects are occluded or central part of the object is covered with other objects. Hence our method always predicts the grasp configuration near the corners of the object and decreases the grasp success probability.

*2) Comparison:* We compared the proposed method with two other strategies. In the first strategy, we grasp the objects from the centroid of the object region. In this strategy, each possible grasp configurations has the same centroid but different orientation. In the second strategy, we compute two orthogonal axes for the object region using PCA and generate the grasp configurations along the major axis, and each possible grasp configurations has the same orientation but different centroid. We found that for simple shaped objects when placed separately, all three strategies provides equivalent results *i.e.* final grasp configuration is similar. But for complex shape objects, both strategy fails, while our strategy predicts some valid grasp configurations, as shown in Fig. 9. So we conclude that the grasping from centroid or grasping along the major axis are special cases of our method, and our method is more general and can be applied for various shaped objects.

## V. Conclusion

A novel grasp pose estimation technique was proposed and tested on various objects like a ball, hand weight, tennis ball container, and for complex shaped objects like blower (nonconvex shape). It is observed our method performs better than the well-known strategies *i*) grasp from the centroid of object and *ii*) grasp along the major axis of the object. Further, we tested the technique in two scenarios, when the objects are placed distinctly and when the objects are placed in a dense clutter, and overall we achieve a grasping accuracy of 88.16% and 77.03% respectively. All experiments are performed with UR10 robot manipulator along with WSG-50 two-finger gripper for grasping of objects.

We believe that if we could use a color-based segmentation or machine learning based semantic segmentation technique, then we can get the object contour and object skeleton with greater accuracy and can get better grasping.

## References


[1] C. Ferrari and J. F. Canny, "Planning optimal grasps.," in *ICRA*, vol. 3, pp. 2290–2295, 1992.

[2] V.-D. Nguyen, "Constructing force-closure grasps," *The International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988.

[3] F. T. Pokorny and D. Kragic, "Classical grasp quality evaluation: New algorithms and theory," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3493–3500, IEEE, 2013.

[4] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[5] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[6] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1957–1964, IEEE, 2016.

[7] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3406–3413, IEEE, 2016.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[9] M. Gualtieri, A. ten Pas, and R. Platt, "Pick and place without geometric object models," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7433–7440, IEEE, 2018.

[10] M. Gualtieri, A. Ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 598–605, IEEE, 2016.

[11] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[12] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[13] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz, "Perceiving, learning, and exploiting object affordances for autonomous pile manipulation," *Autonomous Robots*, vol. 37, no. 4, pp. 369–382, 2014.

[14] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, *et al.*, "Domain randomization and generative models for robotic grasping," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3482–3489, IEEE, 2018.

[15] A. Ten Pas and R. Platt, "Localizing handle-like grasp affordances in 3d point clouds," in *Experimental Robotics*, pp. 623–638, Springer, 2016.

[16] G. Vezzani, U. Pattacini, and L. Natale, "A grasping approach based on superquadric models," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1579–1586, IEEE, 2017.

[17] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3304–3311, IEEE, 2011.