A Low Cost Bin Picking Solution for E-Commerce Warehouse Fulfillment Centers

Avnish Gupta, Akash Jadhav and Pradyot VN Korupolu
Addverb Technologies Pvt. Ltd., India
{avnish.gupta, akash.jhadav, pradyot.korupolu}@addverb.in

Abstract
In recent years, the throughput requirements of e-commerce fulfillment warehouses have seen a steep increase. This has resulted in various automation solutions being developed for item picking and movement. In this paper, we address the problem of manipulators picking heterogeneous items placed randomly in a bin. Traditional solutions require that the items be picked to be placed in an orderly manner in the bin and that the exact dimensions of the items be known beforehand. Such solutions do not perform well in the real world since the items in a bin are seldom placed in an orderly manner and new products are added almost every day by e-commerce suppliers. We propose a cost-effective solution that handles both the aforementioned challenges. Our solution comprises of a dual sensor system comprising of a regular RGB camera and a 3D ToF depth sensor. We propose a novel algorithm that fuses data from both these sensors to improve object segmentation while maintaining the accuracy of pose estimation, especially in occluded environments and tightly packed bins. We experimentally verify the performance of our system by picking boxes using an ABB IRB 1200 robot. We also show that our system maintains a high level of accuracy in pose estimation that is independent of the dimensions of the box, texture, occlusion or orientation. We further show that our system is computationally less expensive and maintains a consistent detection time of 1 second. We also discuss how this approach can be easily extended to objects of all shapes.

1 Introduction
Industrial robots used for tasks like assembly operations or pick and place operations can work only on objects whose location and orientation are predefined. These robots without vision system hence are used only for repetitive tasks. This design of a blind robot system increases the cost of automation considering the requirement of expensive object feeding conveyor systems.

An efficient solution to this situation can be provided by considering a manual assembly environment that has several objects placed randomly in bins around the work area from which required objects can be picked for assembly. Automation of this process requires developing an intelligent robot capable of recognizing and handling objects in bins. This problem, called bin picking, thus consists of picking an object from a group of similar (homogeneous) or different (heterogeneous) objects. The objects in the bin are generally in random orientation and occlude each other; this poses a great challenge to the vision sensors in segmenting the objects to be picked.

A great deal of previous research on vision-based bin picking has been limited to recognition of objects and STL matching where the object is predefined due to the complexity involved in segmenting the bin images. In recent years, much work has been proposed on the recognition of objects for the bin picking problem. Some techniques are limited to detection of the objects while others concentrate on picking an object and analyzing the object for pose determination and recognition using the object database [Takenouchi et al., 1998], [Ohba and Ikeuchi, 1996], [Yanagihara and Kita, 1991].

Image segmentation is another well-researched area that involves the division of an image into meaningful sub-regions. If all the objects are segmented individually, then identifying them is relatively easy; however, when the objects are partially occluded more than one object will be considered as a single object and thus making it difficult to identify each one of them. In contrast to the approaches mentioned above, our algorithm attempts to provide a solution for segmentation and localization of boxes of different sizes in a partially occluded bin using a dual vision system based approach. In the dual vision system, the RGB image is fused with a 3D point cloud
solutions have a satisfactory performance in the case of using CAD file of an object [Kuo et al., 2017]. Another research by [Yang, C. and Medioni, G, 1992] uses the Interactive Closest Point algorithm (ICP) on 3D range data for finding the best fitting cloud point with respect to the reference CAD file. These types of instance recognition solutions are limited to organized environments and are dependent on sensor accuracy. They also pose a major issue of maintaining the object CAD database for all the objects which we need to pick.

Research has also focused on matching extracted 2D features with geometries of known 3D objects using chamfer matching technique [Liu et al., 2012]. One approach discusses Oriented Chamfer Matching (OCM) on an object database containing 360 images of one object each one representing an orientation at equal intervals created using CAD file of an object [Kuo et al., 2017]. These solutions have a satisfactory performance in the case of unoccluded and uncluttered objects in the bin and are not quite robust [Kuo et al., 2017].

Another approach to tackle the problem of cluttered object heap is the use of Convolutional Neural Networks (CNNs) trained on a large dataset of images to recognize the object [Bousmalis et al., 2014]. With the development of 3D RGB-D sensors, there was an increase in the point cloud data. This motivated researchers to further use CNNs for the segmentation of object classes directly from the point cloud of the scene. [Gualtieri et al., 2016] and [Viereck et al., 2017] trained CNNs on 3D dense point clouds from synthetic datasets to predict antipodal grasps. These methods have a very good performance if the data is collected from various viewpoints and is large enough so that the algorithm can generalize well. The drawbacks of these methods are the requirements of a powerful GPU system and several months taken in the collection of datasets.

The major issue with all of the above-mentioned approaches is scalability for a large number of classes of objects. [Mahler and Goldberg, 2017] have discussed an approach using Reinforcement Learning to solve the problem of scalability. Our approach addresses the similar problem of scalability but is rather a computationally less intensive and cost-effective solution.

The objective of this research is to develop a robust vision system for a manipulator robot to perform automated bin-picking operations generalized for boxes of any sizes and textured surfaces in cluttered environments. The task of bin picking is split into two subtasks, namely box segmentation, and box localization. The two subtasks and the experimental setup is detailed in Section 3, which describes the proposed segmentation algorithm using robust edge detection and elaborates on the box localization process to obtain 6DOF of the box detected. Section 4 discusses the experimental results obtained from the proposed algorithm and real-time bin picking experiments conducted on an ABB IRB 1200 robot. Finally, section 5 and section 6 presents the conclusion and future work. This paper presents a 3D vision based bin-picking for cuboidal objects but this can be further extended for randomly shaped objects.

3 Methodology

The proposed dual vision system consists of a monocular RGB camera with a 3D depth sensor that provides an organized point cloud. A calibration technique using perspective transform [16] is implemented to fuse the RGB image from the monocular camera with the organized point cloud of the 3D depth sensor such that for every point in the point cloud we have a specific RGB value for it.

The dual sensor setup shown in Figure 2 consists of a monocular camera and the 3D depth sensor placed par-
parallel to each other on a rigid mount. The segmentation of the box is done using the RGB information from the monocular camera. A binary mask is generated for the segmented boxes, where the detected boxes have a value of one and rest as zero. Further for the pixels in the mask, corresponding points in the point cloud are considered for fitting a plane to obtain its centroid and orientation. Finally, for sending obtained 6DOF information of the box detected by the dual vision system to the robot, a calibration procedure [OpenCV] is carried out between the ABB IRB 1200 robot and the 3D depth sensor to transform the coordinates obtained with respect to the 3D depth sensor frame to the robot base frame.

### 3.1 Box Segmentation

To segment out boxes, we have incorporated a robust edge detection algorithm, which detects all possible edges in the image and filters out the unnecessary edges, while preserving the structural properties of the detected boxes to be used for further processing. The proposed algorithm also performs efficiently for differentiating two objects kept very close to each other. The algorithm further consists of a parent-child hierarchy, such that if a small box is placed on top of a large box, the identity of the smaller box should be preserved within the larger box. The smaller box is termed as a child and the larger box is termed as a parent by the algorithm. The child box is given a priority to be picked first by the algorithm.

The algorithm runs in 6 separate steps:

**Set Bin ROI:** In the image taken from the camera, the region of interest (ROI) of the bin is extracted and used in the further processing of the algorithm. Bin ROI is shown in Figure 4a.

**Smoothing:** Images taken from a camera might contain some amount of noise. To prevent that noise from being mistaken for edges, a gaussian filter with kernel 3x3 is applied to smoothen the input image as shown in Figure 4b.

**Finding Gradients:** Popular Canny edge algorithm [Canny, 1986] is known for finding edges whose intensity gradients fall in a range of predefined threshold values. We have incorporated an auto canny algorithm [Auto-Canny] that adaptively decides the threshold based on the region of interest of the bin in the image. The edges are determined by applying gradient at each pixel in the smoothened image using Sobel operator as shown in Figure 4c. To approximate the gradient in the x- and y-direction respectively the kernels shown in Equation 1.
\[
KGY = \begin{bmatrix}
  -1 & 0 & 1 \\
  -2 & 0 & 2 \\
  -1 & 0 & 1 \\
\end{bmatrix}
\] (1)
\[
KGX = \begin{bmatrix}
  1 & 2 & 1 \\
  0 & 0 & 0 \\
  -1 & -2 & -1 \\
\end{bmatrix}
\] (2)

**Refining Gradients:** Over the edges obtained in the image from the auto canny edge detection, contours are detected. These are further refined over the contour area of 2500 pixel-sq in a 2K pixel density image to remove unnecessary edges. Contour detection aims at quantifying the presence of a closed boundary at a given image location through local measurements. This step results in the identification of surfaces of boxes as shown in Figure 4d.

**Parent-Child hierarchy:** A contour hierarchy is implemented to solve the stacking of smaller boxes over larger boxes. Contour detection is used to detect objects in the image, but in some cases, objects are present on top of another object, just like nested figures. In this case, we call the outer one as a parent and the inner one as a child. This way, contours in an image constitute a relationship with each other. Representation of this relationship is called the Hierarchy. An individual object is determined as a parent contour as shown in Figure 4d.

**Mask Generation:** A binary mask is generated for the child and parent boxes individually, where the full surface of the detected box has a value of one and rest part of the bin as zero. While the child contour is given a priority, the parent contour has a zero value. The mask is generated for parent contours only after all the child contours have been picked by the ABB 1200 robot as shown in Figure 4e. This binary mask is further used on the point cloud to detect planes and generate the 6DOF of the plane.

### Algorithm 1: Plane Segmentation and Extraction

**Input:** Masked Point Cloud Clusters (clusters)
**Output:** Planes, Model Coefficients, Plane Normals

```python
for cluster in clusters:
    plane = plane, plane_coeff, normal_vecs = []
    while cluster.size() >= min_cluster_size:
        points, coeff = ransac_seg_plane(cluster)
        if prev_coeff == coeff:
            break
        if points.size() >= min_object_size:
            plane.append(points)
            plane_coeff.append(coeff)
            cluster = rm_seg_plane(cluster, points)
        plane_num = planes.size()
        if plane_num == 0:
            continue
        else:
            dp_mat = dot_product(normal_vecs)
            group, visited = [],
            for m in range(plane_num):
                if m not in visited:
                    visited.append(m)
                    g = []
                    g.append(m)
                    for n in size(plane_num)
                        if n not in visited & dp_mat[m][n] ~ 1.0:
                            if check_overlapping(planes[m], planes[n]):
                                g.append(n)
                        end
                    end
                    group.append(g)
                end
            end
            for g in group:
                uni_plane.append(comp_grp_plane(g))
                uni_plane_coeff.append(comp_grp_plane_coeff(g))
                normal_vecs.append(comp_grp_plane_normal(g))
            end
        end
    end
return uni_plane, uni_plane_coeff, normal_vecs
```

**Filtering**

Since the data consists of thousands of points, it will take a huge amount of time to directly compute planes on this data. Moreover, the data is prone to noise points because of sensors’ measurement capability and repeatability. There are various methods available for filtering the point cloud \cite{Xian-Feng et al., 2017, Moreno and Ming L., 2006}. We have selected two filters based on their efficiency and time complexity to reduce the num-
ber of points and to minimize noise, after which an output shown in Figure 6b is obtained.

Firstly, we apply the Voxel Grid filter to downsample the cloud. This method assumes a 3D voxel grid over the cloud and then in for each voxel, a centroid is assumed to approximate all the points in that voxel. This method robustly reduces the number of points based on the leaf size which defines the size of voxel and it represents the underlying surface more accurately. We can optimally set this parameter so that geometric features don’t get lost and the number of points also gets significantly reduced.

Secondly, we use Statistical Outlier Removal (SOR) filter [HarisBalta et al., 2018]. Generally, point clouds registered from depth sensors have some sparse outliers which can corrupt the estimation of characteristics like surface normals. SOR computes a distribution of the mean distance of a point to all its k nearest neighbors specified by a parameter. The resulting distribution is assumed to be Gaussian with a mean and a standard deviation. All the points whose mean distances are above the threshold defined by the global distances mean and standard deviation parameters are classified as outliers and removed from the cloud.

Resampling and Difference of Normals
Filtering, discussed in the above section, is incapable of removing some erroneous points caused by measurement errors. Thus we perform resampling using Moving Least Squares (MLS) surface reconstruction method [Fleishman et al., 2005]. This resampling technique also helps in accounting irregularities in the cloud created due to glossy surfaces and occlusions. The algorithm estimates missing points in the surfaces by using higher-order polynomial interpolations between the neighboring data points.

Following this, we perform a Difference of Normals based segmentation [Ioannou et al., 2012] to remove the edges between the planes of the cuboidal objects. The normals calculated on the plane point cloud are shown in Figure 6c. This method enhances the performance of plane segmentation algorithm discussed later by separating the faces of boxes. We first calculate two unit point normals $\hat{n}(p,r_s), \hat{n}(p,r_l)$ for every point $p$ such that $r_s > r_l$ as shown in Figure 5. Succeeding this we calculated a normalized Difference of Normals vector field using equation 3 where $r_s, r_l \in \mathbb{R}, r_s < r_l$.

$$\Delta \hat{n}(p,r_s, r_l) = \frac{\hat{n}(p,r_s) - \hat{n}(p,r_l)}{2} \quad (3)$$

Taking the norm of this vector field yields values in the range (0,1) for all the points. We select the points for which the norm has value less than a user-defined threshold parameter.
classified as inliers or outliers. Then this model is again re-estimated by using all classified inliers. This process is repeated until we get a segmented plane with its coefficients from the cluster. The coefficients of this segmented plane are saved and all the points belonging to this plane are removed from the cluster. The updated cluster is again processed for plane segmentation. This process keeps on repeating until the number of points in the cluster becomes less than the user-defined parameter for minimum points in the cluster. A similar kind of processing is done for all the clusters. The parameters of the RANSAC algorithm should be chosen wisely such that the points which are not on the surface to be grasped don’t become part of the segmented plane.

Following the segmentation of all the planes from all the clusters, we need to extract unique planes. Generally, due to measurement noise in sensors, the same surface can be detected as 2 different planes. In order to address this problem, we group all segmented planes which are parallel to each other. The two parallel planes found from different clusters (Figure 6d) after RANSAC but are part of the same surface are combined into a single plane as shown in Figure 6e. This is done by combining all those planes which are overlapping using the following checks:

- Distance between the centroid is less than a certain threshold. This threshold is defined based on the minimum object size to be picked.
- Perpendicular distance between planes is less than the measurement noise of the depth sensor. Correct selection of this parameter ensures that planes which are part of same surface gets combined.

The pseudo code shown in [1] contains the functions `check_overlapping` to check for overlapping planes and `comp_grp_plane` to group overlapping planes.

### 6DOF Pose Estimation

After finding all the unique planes from the cloud, we need to generate a 6DOF pose over each plane for the robot to grasp the object as shown in Figure 6f. Since in our implementation we used a vacuum gripper, we need to calculate the centroid of the plane and Euler angles to represent the orientation of the plane.

The centroid of the plane is calculated using Mean Shift clustering [Comaniciu and Meer, 1981] method. The advantage of using this algorithm over others is that it is a centroid based algorithm. This algorithm almost accurately calculates the center of the region even in the case of planes having sparse points. The centroid is calculated with reference to the depth sensor frame.

We estimated Euler angles by attaching a reference frame at the previously computed centroid for each plane. For each plane, we have a normal vector from estimated plane coefficients from which we approximate an axis along the plane by finding a unit vector along the plane. Finally, we compute the cross product between the unit normal vector and unit vector along the plane to find the third axis of the reference frame. Following this, a rotation matrix is computed for defining the rotation of this object frame with respect to the depth sensor frame. Further, this rotation matrix is converted into Z-Y-X Euler angles with respect to the depth sensor frame.

\[
Z_1Y_2X_3 = \begin{bmatrix}
 c_{1}c_{2} & c_{1}s_{2}s_{3} - c_{3}s_{1} & s_{1}s_{3} + c_{1}c_{3}s_{2} \\
 c_{2}s_{1} & c_{1}c_{3} + s_{1}s_{2}s_{3} & c_{3}s_{1}s_{2} - c_{1}s_{3} \\
 -s_{2} & c_{2}s_{3} & c_{2}c_{3}
\end{bmatrix}
\] (4)

, where c represents cos(), s represents sin(), 1 and 3 are Euler angles about X, Y and Z axes respectively.

### 4 Experiment Test-bench

We performed our experiments with ABB IRB 1200 robot and dual vision system. The dual vision system consisted of a Monocular RGB camera with 2K resolution and a Time of Flight based 3D depth sensor which gives an organized point cloud, of resolution 224x172. This visual system was mounted at a height of 1.2m above the bin having dimensions 400mm x 600mm. All the experiments were performed on a laptop having Ubuntu 16.04 with a 2.30GHz Intel Core i5-8300H CPU and 4GB DDR3 RAM.
5 Results

This section is divided into three subsections:

5.1 Verification

To evaluate the accuracy of the 3D pose estimated by our system, a plate with an Aruco marker [Ho Chuen Kam et al., 2018] is attached to the last link of the robot with a known static transform as shown in Figure 7.

![Figure 7: Experimental setup for performing verification procedure](image)

The robot is moved by the teach pendant such that the plate gets all possible orientations of a box in the bin. The Aruco maker is detected in the RGB image of the camera and a mask is generated for it [Ho Chuen Kam et al., 2018]. Further, a plane is fitted in the point cloud for this mask and its 6DOF pose is estimated by our system which is compared with the ground truth values of the teach pendant. In our experiment, the performance was tested on a total of 9 orientation cases of the plate. For each test case, we have recorded 20 samples and taken the average of the error in pose calculated by the algorithm. All the Euler angles are measured with respect to the robot base frame and the z-axis of the robot is pointing upward with respect to the ground. The results obtained for the 9 test cases of the plate are shown in Table 1. Additionally, the system demonstrated stable performance with an overall mean translation error of less than 3.3 mm in all directions and a mean rotation error of less than 3.26 deg on all axes.

![Table 1: Computation of system error for 6DoF pose](image)

5.2 Box Test Case

The surface details of the boxes used in experimentation are presented in Table 2. We have used boxes with a minimum dimension of 50mmX75mmX20mm and various boxes having a surface with no texture, minimal texture, maximum texture, and gloss. Further, some special test cases are taken into consideration where two objects of the same dimensions are kept adjacent to each other and a smaller object is kept on top of a larger object. For planar surfaces, the deviation along the x and y axes determines the accuracy of the system. This accuracy further determines if the system is good enough to perform random boxes picking tasks, whereas the deviation and orientation on the z-axis with respect to the gripping tool are less important as the single suction cup type gripper can easily search for the box within a certain range in the z-direction. After the box being picked by

![Figure 8: Segmentation results on boxes with different surface types as mentioned in test cases](image)
the robot, it is put in the barcode scanning zone where it is identified by the system. The maximum average percentage error in the result of the translation estimate is found to be 3.12% along the x and y axes of the box. This error is acceptable as the box is not to be picked from its barcode which is generally present at the edges of the box. The error computed ensured that the proposed method worked well on the boxes in the cluttered environment for random bin-picking tasks.

5.3 Complete System Performance

The performance of our system for the box specifications mentioned in Table 2 is shown in Figure 8. The boxes are placed individually in the bin and the image from the RGB camera is first processed using the box segmentation module and then fused with the point cloud data of the depth sensor. This is given as input to the box localization module to extract the 6DOF pose data of the box.

Figure 8a shows the performance for the box with no texture, Figure 8b shows performance for the box with minimal texture, Figure 8c shows performance for the box with high texture and Figure 8d shows performance for the box with a glossy surface. The pose estimation error is slightly higher in the case of glossy surfaces because of sensor limitations caused due to reflectivity.

The performance of our system in the case of two adjacent boxes of the same dimensions is shown in Figure 9a and the case of smaller box placed on top of a larger box is shown in Figure 9b. Where the 1st row shows the input RGB image ROI of the bin, 2nd row shows the smoothened image, 3rd row shows the Edges found, 4th row shows the refined edges found, 5th row shows the mask of child contour and the 6th row shows the mask of parent contour, after the child was picked by the robot.

Figure 10 explains the parent-child hierarchy in both cases. When two adjacent boxes are taken into consideration as shown in Figure 10a, they are not identified as two individual boxes. Instead, their contours are detected as one contour (child) is inside another contour (parent) and the child is given a priority to be picked up. Once the child box is picked up, in the next detection the parent box is picked up. When the case of a smaller box is kept on top of a larger box is taken into consideration as shown in Figure 10b, the smaller box is detected as child contour and the larger box is detected as parent contour. The vision system first segments the child contour boxes and computes their midpoint [x, y, and z coordinates] and orientation [Euler angles] and further goes for the parent contour boxes when it has picked all the child contour boxes. Figure 11 shows the performance of our system in the cluttered bin. Our proposed algorithm was able to detect smaller boxes on the top of larger boxes due to the parent-child hierarchy used in generating masks. These masks were further used to extract an accurate plane from the point cloud and estimate the 6DoF pose.

Table 3 shows the total processing time for both box segmentation and box localization modules. The total processing time comes out to be 0.9 seconds. These
results are comparable to the results demonstrated in [Mahler and Goldberg, 2017]. As opposed to Mahler and Goldberg, 2017, which requires very high computation resources to generate pick points, our approach works very well on a regular laptop. This makes our approach cost-effective. In addition, our approach is easily scalable to multiple boxes since it does not require any prior knowledge of the boxes or any form of training.

6 Conclusion and Future Work

We presented a dual sensor system for box segmentation and box location computation in bin picking applications. Two algorithms for box mask segmentation and box localization are proposed. The box mask segmentation algorithm is found to be robust in case of no texture, minimal and high texture boxes. However, the performance for the glossy surface is limited by 3D depth sensor measurement. The unified vision algorithm is capable of computing accurate poses in all three axes with an average error of +/- 3.3mm; however, the picking operations were unaffected due to the error in z-axis because of the tool with single suction type vacuum gripper which has a tolerance of up to +/- 5 mm. The real-time experimental demonstrates a consistent detection time of approximately 1 second on the specified hardware and test cases. We have empirically proved the feasibility of the proposed algorithms for bin-picking application and it can be adopted for many real-world box picking operations in warehouses and e-commerce industry. This solution has the potential to be extended for other objects such as cylinders using shape decomposition algorithms.

References

[Takenouchi et al., 1998] Ayako Takenouchi, Naoyoshi Kanamaru and Makoto Mizukawa. Hough-space-based Object Recognition Tightly Coupled with Path Planning for Robust and Fast Bin-picking. Proceedings of International Conference on Intelligent Robots and Systems, pages 1222 – 1229, Canada, October 1998.

[Ohba and Ikeuchi, 1996] Kohtaro Ohba and Katsushi Ikeuchi. Recognition of the Multi Specularity Objects for Bin-picking Task. Proceedings of IROS, IEEE, pages 1440 – 1447, October 1996.

[Yanagihara and Kita, 1991] Yoshimasa Yanagihara and Toshiro Kita. Parts Picking in Disordered Environment. IEEE International Workshop on Intelligent Robots and Systems, pages 517-522, Osaka, Japan, 1991.

[Martin et al., 2000] B.Martin, B. Bachler and S. Stefan. Vision Guided Bin Picking and Mounting in a Flexible Assembly Cell. Thirteenth International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems, USA, 2000.

[Ioannou et al., 2012] Yani Ioannou, Babak Taati, Robin Harrap and Michael Greenspan Difference of Normals as a Multi-Scale Operator in Unorganized Point Clouds. Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT).

[AutoCanny] https://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/.

[Yang, C. and Medioni, G, 1992] Yang, C. and Medioni, G. Object modelling by registration of multiple range images. Image and Vision Computing 10(3), pages 145–155, 1992.

[Liu et al., 2012] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa Fast object localization and pose estimation in heavy clutter for robotic bin picking. The International Journal of Robotics Research, 31(8):pages 951–973, 2012.

[Kuo et al., 2017] Kuo, H. Y., Su, H. R. and Lai, S. H. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. IEEE International Conference on Robotics and Automation(ICRA), 2017.

[Bousmalis et al., 2014] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor and K. Konolige. 3D Object Detection and Pose Estimation from Depth Image
for Robotic Bin Picking. IEEE International Conference on Automation Science and Engineering (CASE), pages 1264-1269, 2014.

Gualtieri et al., 2016] M. Gualtieri, A. t. Pas, K. Saenko, and R. Platt. High precision grasp pose detection in dense clutter. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016.

Viereck et al., 2017] U. Viereck, A. t. Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using easily simulated depth images. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2017.

Mahler and Goldberg, 2017] Jeffrey Mahler and Ken Goldberg. Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences. CoRL 2017.

[OpenCV] https://docs.opencv.org/trunk/da/d6e/tutorial_py_geometric_transformations.html

Strobl and Hirzinger, 2006] K. H. Strobl and G. Hirzinger. Optimal hand-eye calibration. Intelligent Robots and Systems, 2006.

Canny, 1986] J. Canny. A computational approach to edge detection. PAMI, 1986.

Xian-Feng et al., 2017] Xian-Feng Han*, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao and Liping Xiao. A review of algorithms for filtering the 3D point cloud. Signal Processing: Image Communication, pages 103-112, September 2017.

Moreno and Ming L., 2006] Carlos Moreno and Ming L. A Comparative Study of Filtering Methods for Point Clouds in Real-Time Video Streaming. Proceedings of the World Congress on Engineering and Computer Science 2016 Vol I WCECS 2016, San Francisco, USA, October 19-21, 2016.

HarisBalta et al., 2018] HarisBalta, JasminVelagic, WalterBoschchaerts, GeertDeCubber and BrunoSiciliano. Fast Statistical Outlier Removal Based Method for Large 3D Point Clouds of Outdoor Environments. IFAC-PapersOnLine, Volume 51, pages 348-353, Issue 22, 2018.

Fleishman et al., 2005] Shachar Fleishman, Daniel Cohen-Or, T. Silva. Robust Moving Least-squares Fitting with Sharp Features. Proceeding of SIGGRAPH ’05 ACM SIGGRAPH, Pages 544-552, Los Angeles, California, 2005.

Campello, Moulavi and Sander, 2013] Campello R.J.G.B., Moulavi D. and Sander J. Density-Based Clustering Based on Hierarchical Density Estimates. In: Pei J., Tseng V.S., Cao L., Motoda H., Xu G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science, vol 7819. Springer, Berlin, Heidelberg, 2013.

Oehler et al., 2011] Oehler B., Stueckler J., Welle J., Schulz D. and Behnke S. Efficient Multi-resolution Plane Segmentation of 3D Point Clouds. In: Jeschke S., Liu H., Schilberg D. (eds) Intelligent Robotics and Applications. ICIRA 2011. Lecture Notes in Computer Science, vol 7102. Springer, Berlin, Heidelberg, 2011.

Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Magazine Communications of the ACM C ACM Homepage archive Volume 24 Issue 6, pages 381-395, ACM New York, NY, USA, June 1981.

Comaniciu and Meer, 1981] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Machine Intell., pages 603–619, 2002.

Ho Chuen Kam et al., 2018] Ho Chuen Kam, Ying Kin Yu and Kin Hong Wong. An Improvement on ArUco Marker for Pose Tracking Using Kalman Filter. 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2018.