DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

SCHRIFTENREIHE DER FAKULTÄT FÜR TECHNIK
DER DUALEN HOCHSCHULE BADEN-WÜRTTEMBERG
RAVENSBURG

2022/02

An efficient bin picking using deep learning with a minimal dataset

Vishnuprasad Prachandabhanu M.Eng., B. Eng Madan Bakthavatchalam,
Dr. Kris Dalm MBA

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

SCHRIFTENREIHE DER FAKULTÄT FÜR TECHNIK DER DUALEN HOCHSCHULE BADEN-WÜRTTEMBERG RAVENSBURG

2022/02

An efficient bin picking using deep learning with a minimal dataset

Vishnuprasad Prachandabhanu M.Eng., B. Eng. Madan Bakthavatchalam, Dr. Kris Dalm MBA

IMPRESSUM

Schriftenreihe der Fakultät für Technik
der Dualen Hochschule Baden-Württemberg Ravensburg

Herausgeber

Prof. Dr. Heinz-Leo Dudek
Prorektor und Dekan der Fakultät für Technik

Duale Hochschule Baden-Württemberg Ravensburg
Baden-Wuerttemberg Cooperative State University
Marienplatz 2
88212 Ravensburg
Deutschland

http://www.ravensburg.dhbw.de

2022/02, März 2023

ISBN 978-3-945557-12-9
ISSN 2199-238X
DOI 10.12903/DHBW_RV_FN_02_2022_PRACHANDABHANU_BAKTHAVATCHALAM_DALM

Gestaltung
DHBW Ravensburg
Marienplatz 2, 88212 Ravensburg

An efficient bin picking using deep learning with a minimal dataset.

Vishnuprasad Prachandabhanu M.Eng.[1], B. Eng. Madan Bakthavatchalam, Dr. Kris Dalm MBA[2].

Keywords:

Deep Neural Network, 3D Object detection, Bin Picking

[1] IWT Wirtschaft und Technik GmbH, Software Developer

[2] IWT Wirtschaft und Technik GmbH, Area Manager; Lecturer DHBW Ravensburg

1 INTRODUCTION AND AIM

Today in Industrial automation, engineers strive to make the robotic system smarter and more flexible to increase productivity in manufacturing. A common task for the robot in a production line is repetitive picking and placing of a randomly placed object with an effective automated manipulation system. Grasping a random object in its real-world environment is a challenging task for the robot, which involves perception, planning and execution. This process is also called bin picking [1], which can recognize and localize the random object to effectively perform robot pick and place.

Deep learning-based methods have achieved excellent results in robotic grasping detection in recent years. The deep convolution network can extract the object's features by utilizing the two-dimension image and projecting it into a three-dimension to generate a robot grasp [2]. Many works, such as [3–5], trained with huge datasets to predict the grasp pose of an object. This work utilized 70+ images for training a deep convolution network model to segment the desired objects in the robot's environment and successfully perform a 2D-planar grasp.

2 RELATED WORK

In the past few years, bin picking solution has been increasing in demand, and various research has proposed different techniques to tackle this problem. The most common answer is STL or CAD matching with Point Cloud from the camera. The research by [6] uses the Interactive Closest Point algorithm(ICP) on Point Cloud from the camera to determine the best fitting points with respect to the reference CAD or STL object model. For the better points to reference model for ICP, [7] improves density-based spatial clustering of applications with noise (DBSCAN) algorithm for better point cloud segmentation. The study by [8] proposed a cost-effective solution to fuse RGB camera and Time of Flight (ToF) depth camera to improve object segmentation accuracy. However, these approaches are limited by 3D sensor accuracy.

In order to overcome the limitations of the 3D sensor, a 2D planar robotic grasp has been researched using a Convolutional Neural Network (CNN). In [3] proposed region base grasp network trained with Cornell Dataset [9] to extract features and achieve real-time performance by removing the process of searching potential grasps. A multi-modal fusion method is introduced in [10], combining RGB and depth data. The method improves the grasp detection accuracy on the Cornell grasping dataset by fusing RGB and depth

features. In the research [11], fully CNN based on Gaussian kernel to encode the central point of grasping is trained and compared with Cornell and Jacquard datasets. A Grasp Quality Convolutional Neural Network (GQ-CNN) was presented [12] to classify the potential grasps, which is trained with Dex-Net 2.0. However, these approaches are trained with large datasets but still fail to detect the grasp in some cases. In this paper, we utilized the U-Net [13] model trained with 70+ images to segment the object and projected it into 3D Point Cloud to determine the grasp location on the object's surface reliably.

3 DATASET

The dataset preparation starts with the image acquisition from the Azure Kinect camera. In total, 74 images are captured, and the model is trained based on these 74 images. Then, the images are cropped down to the Region of Interest (RoI). i.e., to the location of the bin. Next, the cropped images are labelled using online tools to obtain the annotation, and the dataset is prepared. These images and masks are then augmented with various methods, for example, shuffling the colour channels to enhance the learning process during training. The process flows as shown the Figure 1.

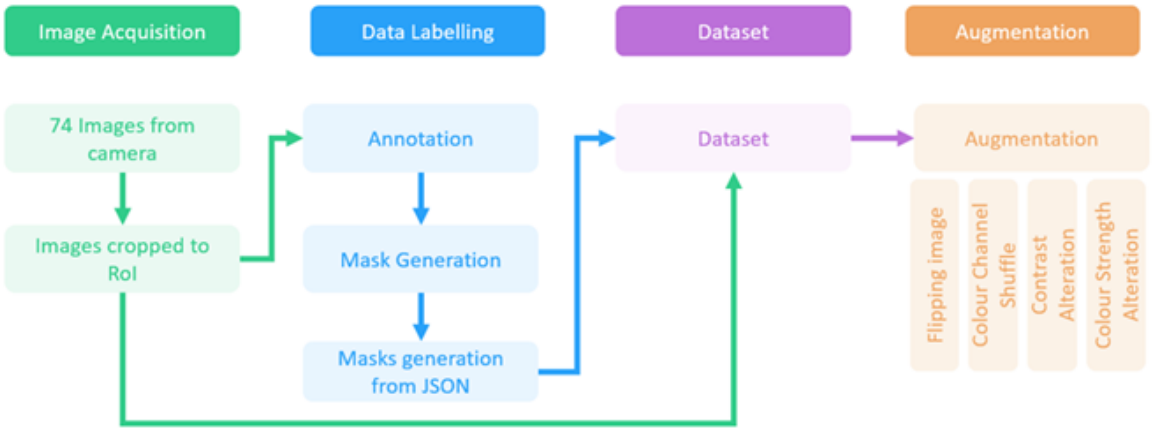

Figure 1: Process of Data Preparation.

3.1 IMAGE ACQUISITION AND LABELLING

Image Acquisition is the initial process in data preparation. This study uses the Azure Kinect camera positioned above the robot's environment, as shown in Figure 2 (a), to capture the RGB images and 3D Point Clouds. A total of 74 images are captured for the data preparation.

One of the work motivations is to achieve the goal with minimum labelling effort. Since the bin is placed in a fixed location with respect to the camera, the RoI is easily determined, and images are cropped to RoI with a dimension of 450 pixels x 650 pixels (H x W) as in Figure 2 (b). Finally, the cropped images are labelled with the help of open-source online tools to generate annotation files.

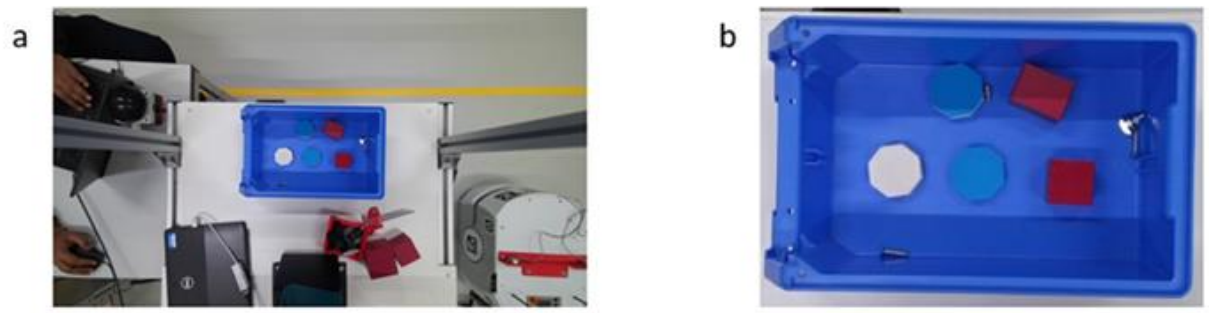

Figure 2: (a) Actual image from Azure Kinect (b) Cropped image to RoI.

3.2 MASK GENERATION

The annotation file obtained from the labelling process is used to generate the masks using the OpenCV library [14]. The dataset consists of 5 classes: background, the surface of the cube or cuboid, the surface of the octagonal prism, the border of the cube or cuboid and the border of the octagonal prism. The example is shown in Figure 3.

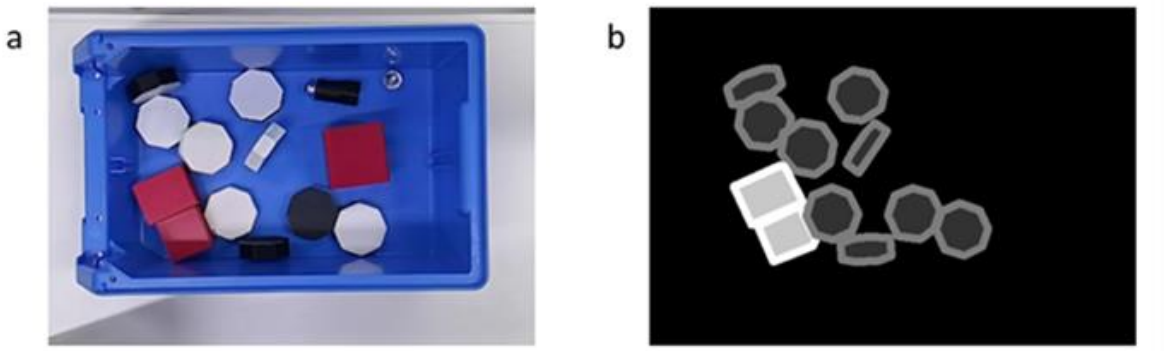

Figure 3: Dataset with Mask with Border (a) Actual Image (b) its corresponding Mask.

3.3 DATA AUGMENTATION

One of the advantages of using U-Net is its efficiency in learning the segmentation task from a smaller dataset. However, data augmentation is used to increase the training more efficiently and avoid over-fitting.

In this work, a few augmentation methods are used, such as flipping the image and mask horizontally or vertically, shuffling the color channels of the images, and altering the contrast of the images to make the model robust in changing lighting conditions and changing the strength of the object colors.

Figure 4: Image augmentation of input images A.

4 MODEL IMPLEMENTATION

This work trains U-Net with the dataset prepared in the Section 3. Mean Intersection-Over-Union (IOU) is chosen as the metric to monitor the model. The model weights with the best validation mean IOU score is saved. The dataset is divided in the ratio of 90: 5: 5, i.e., 90% of the data are taken for training the model, 5% of the data for validating the model and the remaining 5% for testing the model's performance.

4.1 MODEL PARAMETER

The U-Net architecture is taken from [13] as the Keras model and trained to recognize five classes. Furthermore, U-Net accepts images with height and width in the multiples of 32. The images are padded with zeros. Therefore, the input layer is 656 x 656 x 3 and the output layer of the model is 656 x 656 x 5. Since it is a multi-class classification problem, categorical cross-entropy [15] is used for calculating the loss. With batch size set to 4, the model is trained for 120 Epochs. All other model parameters and configurations are mentioned in **Error! Reference source not found.**.

Parameters	Configurations
1. Input Image Size	656 x 656 x 3 (H x W x C)
2. Mask Size	656 x 656 x 5(H x W X C)
3. No of Classes	5
4. Batch Size	4
5. Model Input Layer Size	656 x 656 x 3
6. Model Output Layer Size	656 x 656 x 5
7. Trainable Parameters	1,941,173
8. No of Epochs	120
9. No of Iterations	13
10. Loss Function	Categorical Crossentropy
11. Evaluation Metric	Mean IOU
12. Optimiser	Adam Optimiser
13. Learning Rate	0.001

Table 1: U-Net Model Training Configuration.

4.2 INFERENCE

In this section, the results of the training process, such as training statistics, Model prediction, are analyzed.

4.2.1 TRAINING STATISTICS

The U-Net model is trained for 120 Epochs, and the best weight of the model is saved by monitoring the validation set mean IOU score. The Figure 5 shows the Mean IOU score observed during the training process and validation of the model. The maximum mean IOU achieved for the validation set is 83.3%.

Figure 5: Training Statistics for the model

4.2.2 MODEL OUTPUT

In this section, the model outputs are compared Figure 6 shows the model's predictions. The Figure 6 has two rows and three columns. Row 1 is the input image to the model, and row 2 is the predictions of the respective input images. Columns represent three example images taken for comparison.

In example 1, the input image has a few objects such as a screw, a 3D printed part with a screw and a bigger cube. These objects are considered background. The bigger cube is used in the training process to check whether the model can also learn the objects' size. The model has learned the object's size and provided better results for the example images. The borders in masks also helped the model learn the difference between the background and the objects. This helps detect object instances.

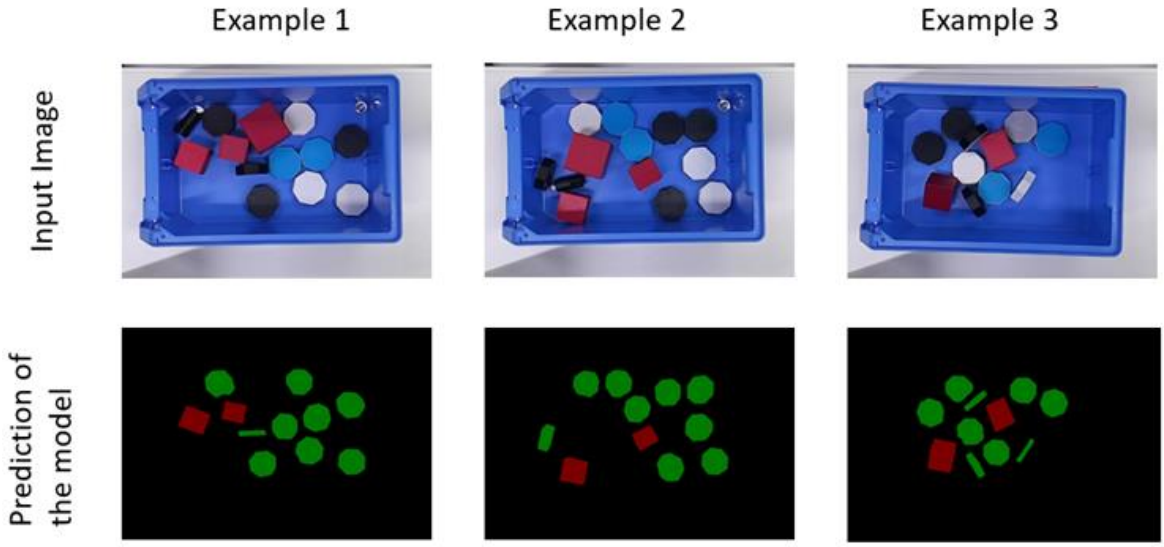

Figure 6: Model predictions

5 POST PROCESSING

U-Net provides the segmented output to the input image. To perform pick and place operations, it is necessary to have the object instances. Therefore, post-processing is developed to find the object instances. The model's output containing the cube or cuboid surface area and octagonal prisms are only taken for computation.

5.1 IMPLEMENTATION

The post-process consists of four steps. These steps are formulated to remove the noise or false predictions in the model's output, detect the object instances and extract the contours of the respective objects.

In Figure 7, the instance for the octagonal prisms is extracted using the post-process. The input to the post-process is the segmented output of the model. The instances of the objects are numbered in the order of performing the pick and place operation.

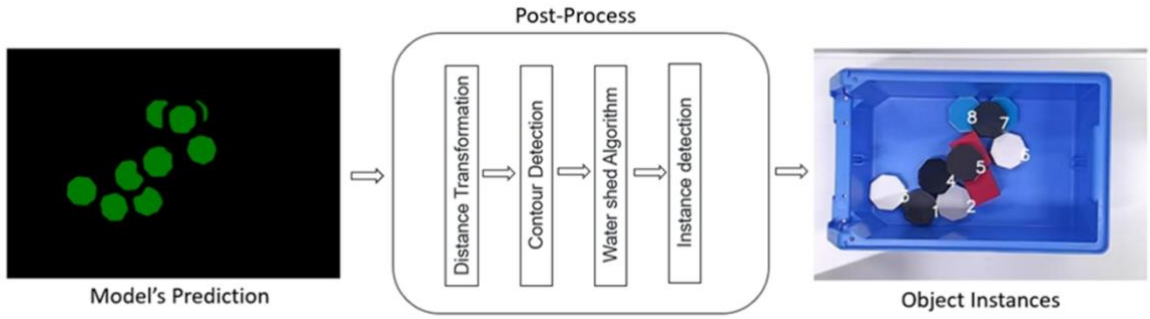

Figure 7: Steps in Post-process.

5.1.1 DISTANCE TRANSFORMATION

The distance transformation provides the greyscale image in which the intensity indicates the closest boundary between object and background. The region of pixels separating the object and background has a lower distance, and the peak distance values are found at the object's center. Furthermore, the distance map is normalized between 0 and 1. It is then filtered for the values ranging between 0.4 and 1.0 to eliminate any unwanted false predictions in the model's output. [16] Distance transformation is done using Euclidean distance in the OpenCV library.

5.1.2 CONTOUR DETECTION

After the distance transformation, contour detection is used to determine the contours of the objects. The contours of the objects are collected to determine the object instances. The Figure 8 shows the result after applying contour detection.

5.1.3 WATERSHED ALGORITHM

The resulting contours from the distance transformation are smaller because of the threshold used to filter the noise. To restore the contours detected to their original sizes, Watershed Algorithm is used.

5.1.4 INSTANCE DETECTION

In this final step, the instances of the object are detected, and the object's center is determined. In Figure 8, it can be noted that all the objects are marked with "X" and numbered in (f), which is the result of instance detection.

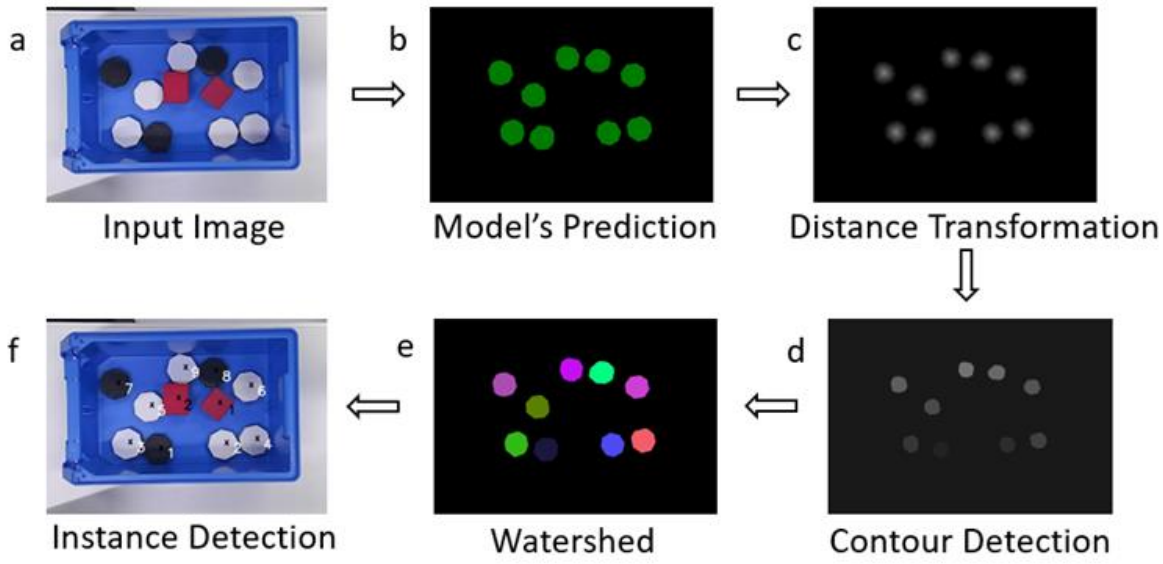

Figure 8: (a) is the input image. (b) is the model prediction. (c) is the output of distance transformation.

5.2 RESULT COMPARISON

To decide whether the model can be used for the actual detection process, it is essential to compare the outputs of the model combined with the post-processing algorithm. The aspects such as the number of objects detected, the model's ability to learn the size of the object, number of false detectors are considered.

The Figure 9 shows the comparison of the outputs of the model combined with the post-processing algorithm with the real images. The figure consists of three rows and two columns. Columns two and four represent the outputs of the model combined with the post-processing algorithm. Columns one and three represent the input images. Rows represent the different images taken for the process of selection. To measure the performance of the algorithm, F1 score is used.

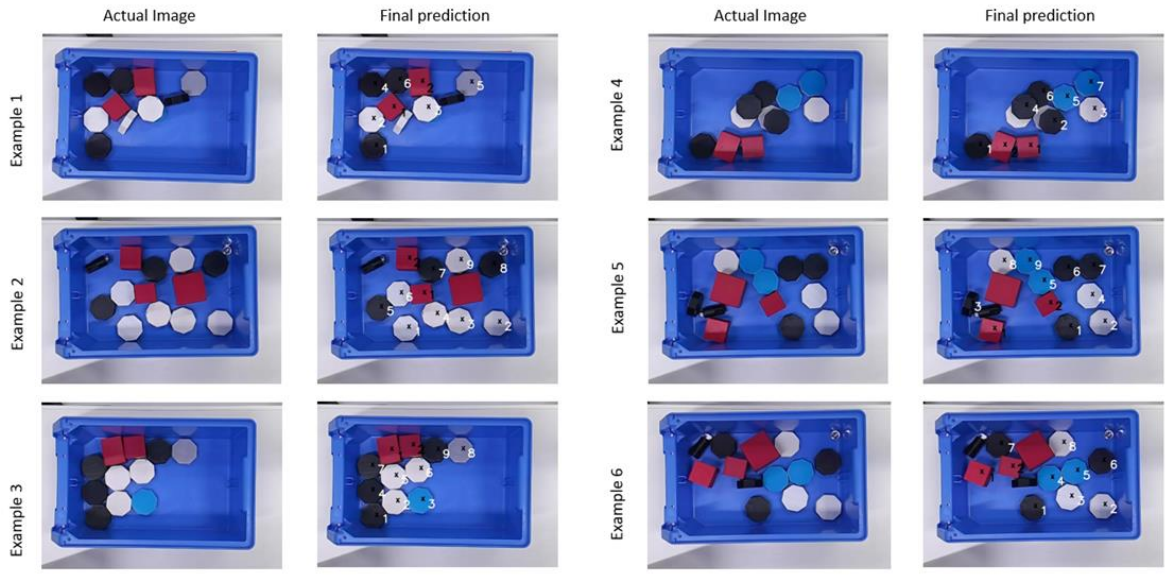

Figure 9: Comparison of Input Image and final prediction.

F1 score comparison: F1 score is the harmonic mean of precision and recall [17]. It considers the True Positives (TP), False Positives (FP) and False Negatives (FN) as shown in equation (1).

$$F1 = \frac{2TP}{2TP + FP + FN} \quad (1)$$

From Figure 9, the following table **Error! Reference source not found.** is derived for the algorithm. All the entries in the table are derived from the figures. The mean F1 score of the algorithm is 95%. Therefore, the algorithm is suited for this work.

Examples	TP	FP	FN	F1 Score
1	8	0	2	0.88
2	11	0	0	1.0
3	10	0	0	1.0
4	9	0	2	0.9
5	11	0	0	1.0
6	10	0	1	0.95
Mean F1 Score				0.95

Table 2: F1 score computation

6 POINT CLOUD

Point clouds are the 3D data form used to obtain detailed information about objects and environments produced by the 3D camera [18]. In this study, the Azure Kinect camera is used to convert the 2D depth map from the depth camera into a 3D point cloud of the camera coordinate system. This provides the RGB image with the pixel's location with respect to the origin of the camera coordinate system.

6.1 DETERMINATION OF PICKING POINT'S LOCATION

From the post-process (Section 5), the picking point of the object is determined, and the object's location with respect to the camera is selected. To find the location of this pixel with respect to the camera coordinate system in the XYZ image is the array containing the location of each pixel with respect to the camera in the RGB image.

Figure 10: Steps to extract pick point from XYZ image.

The Figure 11 has 2 images. (a) is the result provided by the algorithm (b) is the pick point (green point at the center of the green circle) location in the 3D point cloud which belongs to the square 2 in (a).

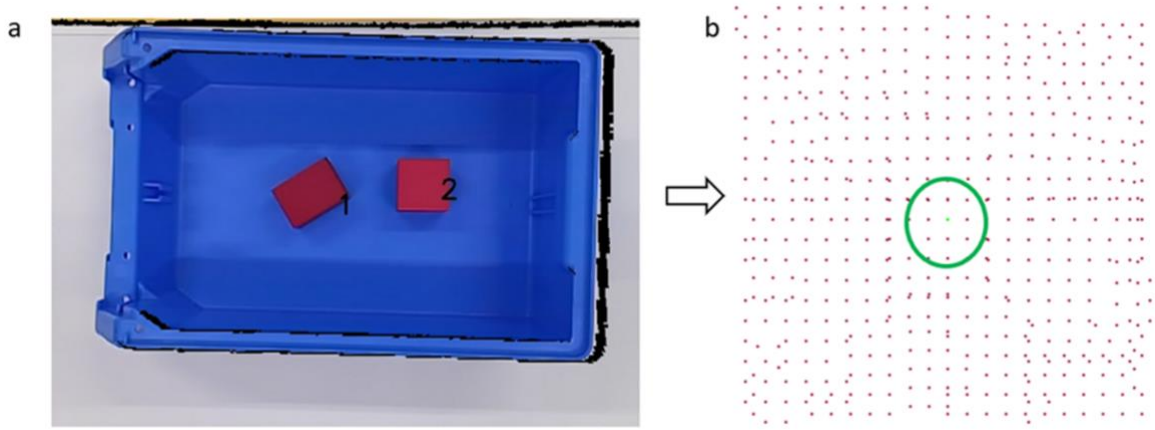

Figure 11: Pick point extraction from XYZ image.

7 ROBOT ARM MANIPULATION

Robot arms have found a wide application in many sectors such as production, medicine, food service and many more. Depending upon the application, the structure and other aspects of the Robot are determined. When the application involves a difficult position to be reached, the Robot structure also changes.

7.1 HARDWARE

In this work, Horst600, an industrial robot arm used in this work. It is manufactured by fruitcore robotics GmbH. It uses a vacuum gripper from Schmalz, which is ideal for gripping light-weighted objects with smooth flat surfaces. The Robot has six joints, and all the joints are rotational joints. Rotation of the joints is restricted to a certain extent.

7.2 COORDINATE SYSTEM

To localize an object precisely, a 6D pose needs to be estimated with respect to the camera. 6D represents the object's location in x, y and z and the orientation of the object roll, pitch, and yaw along the axes. The coordinate relationship between the robot and camera is well defined to achieve object pick action.

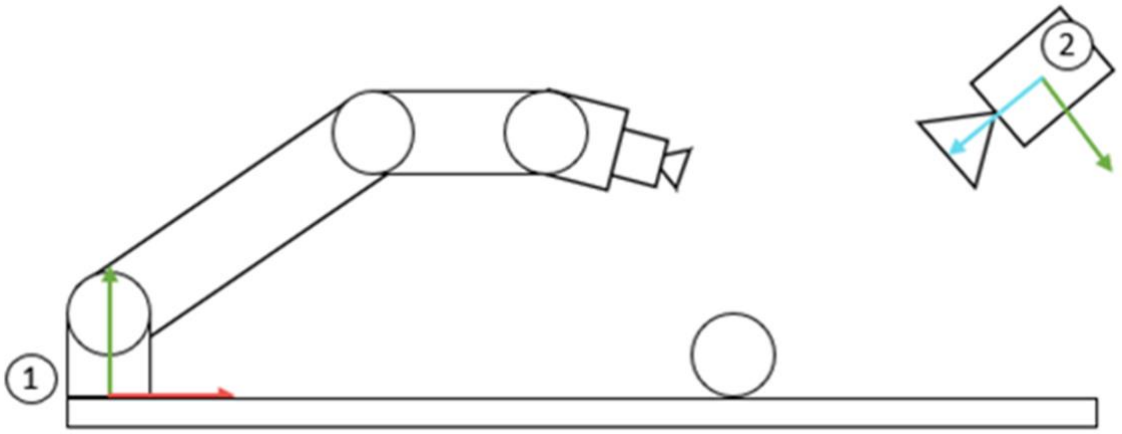

Figure 12: Simplified representation of coordinate systems. 1 is the robot coordinate system and 2 is the camera coordinate system.

7.2.1 ROBOT COORDINATE SYSTEM

The Robot Coordinate system represents moving parts of the robot in a single coordinate system. Usually, the coordinate system of the robot's base is chosen as the origin, as shown in Figure 12. The end-effector (i.e. gripper) pose is calculated with respect to the robot's origin coordinate system.

7.2.2 CAMERA COORDINATE SYSTEM

Camera Coordinate is like the robot coordinate system. When the object is localized using point clouds, the object is located with respect to the camera coordinate system.

7.2.3 RELATIONSHIP BETWEEN THE COORDINATE SYSTEM

The relationship between the camera and robot coordinate system needs to be determined by estimating the position and orientation of the camera with respect to the robot origin coordinate system. By performing rigid body transformation [19], the object's position and orientation with respect robot are computed.

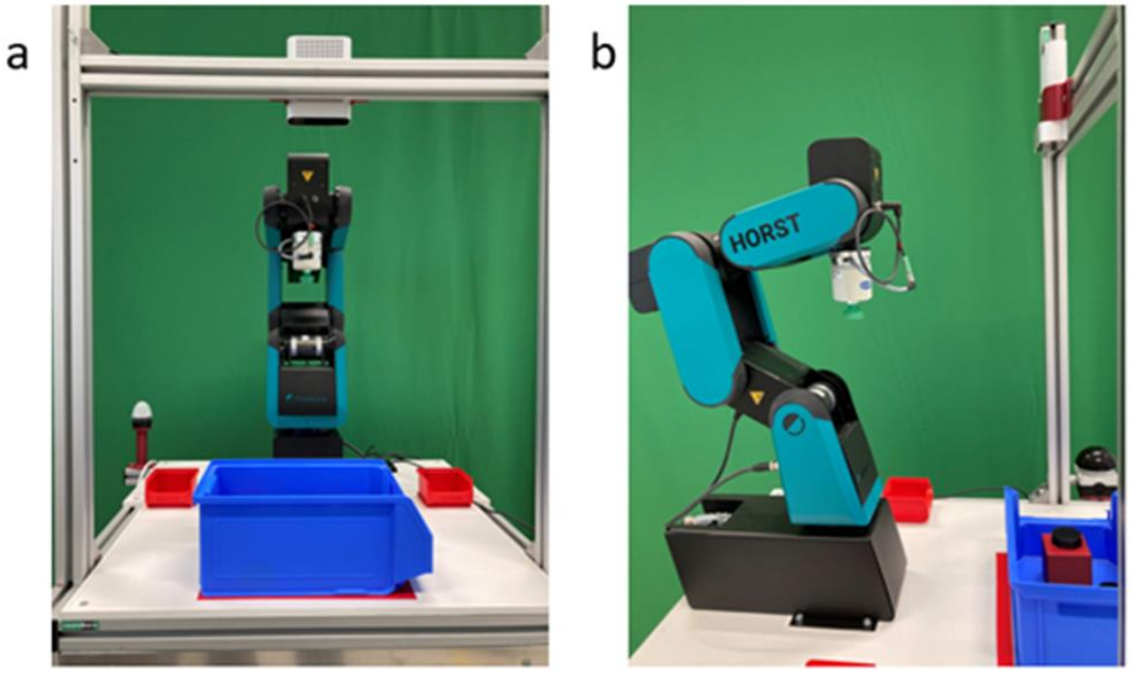

Figure 13: Robot setup in working environment.

The Figure 13 shows the front (a) and side (b) view of the working robot and camera setup, where the camera is fixed above the bin.

$$R = \begin{pmatrix} \cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma & -\cos\alpha\cos\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta \\ \sin\alpha\cos\beta\cos\gamma + \cos\alpha\sin\gamma & -\sin\alpha\cos\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta \\ -\sin\beta\cos\gamma & \sin\beta\sin\gamma & \cos\beta \end{pmatrix}$$

(2)

$$T = \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix}$$

(3)

Equation (2) represents rotation matrix R calculation where α, β and γ are roll, pitch and yaw, respectively between frames. Using equation (2), the transformation matrix T of size 4x4 is calculated as shown in equation (3), where p represents the position x, y and z, the distance between two frames. Finally, object pose with respect to the robot T_{rp} is generated by multiplying the transformation matrix between robot and camera T_{rc} and the transformation matrix between the camera and recognized object T_{cp}.

$$T_{rp} = T_{rc} \times T_{cp}$$

(4)

In the study, we are performing a 2D-planar grasp; therefore, the object's orientation is not considered during the pick and place robot action.

8 CONCLUSION AND FUTURE SCOPE

The work aims to develop a Bin picking solution using Deep Learning. One of the primary purposes of this work is to train a model with fewer samples. Usually, for training Deep Learning models, one may require hundreds of samples for training. To achieve these goals, U-Net is selected, and the dataset size is limited to 74. U-Net gives only segmented output. Therefore, a post-processing algorithm is developed to obtain the instances and pick points of objects detected.

F1 scores for the algorithm's output are checked to find if the model is suitable for the task. The mean F1 score of the model was 95%. Therefore, the model is selected for implementation for bin picking.

The relation between the Robot and camera coordinate systems was determined. Sequences of trajectory paths were formed and used to pick and place the objects according to their class. However, when the objects are placed at complex orientations,

the computer vision algorithm cannot determine the orientations of the objects required for grasping.

In future, this bin picking solution can be improved further by determining the 6D pose of the objects. This will enable the algorithm to grasp the object placed at complex orientations.

REFERENCES

[1] H. Tachikake and W. Watanabe, “A Learning-based Robotic Bin-picking with Flexibly Customizable Grasping Conditions,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2020, pp. 9040–9047.

[2] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *The International Journal of Robotics Research*, vol. 34, 4-5, pp. 705–724, 2015, doi: 10.1177/0278364914549607.

[3] . U. Asif, J. Tang, and S. Harrer, “Densely Supervised Grasp Detector (DSGD),” Jan. 2018. [Online]. Available: https://arxiv.org/pdf/1810.03962

[4] J. Redmon and A. Angelova, “Real-Time Grasp Detection Using Convolutional Neural Networks,” Sep. 2014. [Online]. Available: https://arxiv.org/pdf/1412.3128

[5] G. Wu, W. Chen, H. Cheng, W. Zuo, D. Zhang, and J. You, “Multi-Object Grasping Detection With Hierarchical Feature Fusion,” *IEEE Access*, vol. 7, pp. 43884–43894, 2019, doi: 10.1109/ACCESS.2019.2908281.

[6] Y. Chen and G. Medioni, “Object modeling by registration of multiple range images,” in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, USA, 1991, pp. 2724–2729.

[7] J. Guo, L. Fu, M. Jia, K. Wang, and S. Liu, “Fast and Robust Bin-picking System for Densely Piled Industrial Objects,” Dec. 2020. [Online]. Available: http://arxiv.org/pdf/2012.00316v2

[8] A. Gupta, A. Jadhav, and P. V. N. Korupolu, “Low Cost Bin Picking Solution for E-Commerce Warehouse Fulfillment Centers,” doi: 2019.

[9] I. Lenz, H. Lee, and A. Saxena, “Deep Learning for Detecting Robotic Grasps,” Jan. 2013. [Online]. Available: https://arxiv.org/pdf/1301.3592

[10] Z. Wang, Z. Li, B. Wang, and H. Liu, “Robot grasp detection using multimodal deep convolutional neural networks,” *Advances in Mechanical Engineering*, vol. 8, no. 9, 168781401666807, 2016, doi: 10.1177/1687814016668077.

[11] H. Cao, G. Chen, Z. Li, J. Lin, and A. Knoll, “Lightweight Convolutional Neural Network with Gaussian-based Grasping Representation for Robotic Grasping Detection,” Jan. 2021. [Online]. Available: http://arxiv.org/pdf/2101.10226v1

[12] J. Mahler *et al.,* “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics,” Mar. 2017. [Online]. Available: https://arxiv.org/pdf/1703.09312

[13] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” May. 2015. [Online]. Available: https://arxiv.org/pdf/1505.04597

[14] G. Bradski, “The OpenCV Library,” 2000.

[15] S. Mannor, D. Peleg, and R. Rubinstein, “The cross entropy method for classification,” in *Proceedings of the 22nd international conference on Machine learning - ICML '05*, Bonn, Germany, 2005, pp. 561–568.

[16] T. Strutz, “The Distance Transform and its Computation,” *TECHP.* [Online]. Available: https://arxiv.org/pdf/2106.03503

[17] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, “Thresholding Classifiers to Maximize F1 Score,” Aug. 2014. [Online]. Available: https://arxiv.org/pdf/1402.1892

[18] S. May, K. Pervoelz, and H. Surm, “3D Cameras: 3D Computer Vision of Wide Scope,” in *Vision Systems: Applications*, G. Obinata and A. Dutt, Eds.: I-Tech Education and Publishing, 2007.

[19] P. R. Evans, “Rotations and rotation matrices,” *Acta crystallographica. Section D, Biological crystallography*, vol. 57, Pt 10, pp. 1355–1359, 2001, doi: 10.1107/S0907444901012410.

Herausgeber

Prof. Dr. Heinz-Leo Dudek
Prorektor und Dekan der Fakultät für Technik

Duale Hochschule Baden-Württemberg Ravensburg
Baden-Wuerttemberg Cooperative State University
Marienplatz 2
88212 Ravensburg

ISBN 978-3-945557-12-9

ISSN 2199-238X

DOI 10.12903/DHBW_RV_FN_02_2022_PRACHANDABHANU_BAKTHAVATCHALAM_DALM