Object segmentation for bin picking using deep learning

Artur Cordeiro[1,2], Luís F. Rocha[2], Carlos Costa[2,3], and Manuel F. Silva[1,2]

[1] ISEP/IPP - School of Engineering, Polytechnic Institute of Porto, Porto, Portugal,
[2] INESC TEC - INESC Technology and Science, Porto, Portugal,
[3] FEUP - Faculty of Engineering, University of Porto, Porto, Portugal,
artur.j.cordeiro,luis.f.rocha,carlos.m.costa,manuel.s.silva}@inesctec.pt

Abstract. Bin picking based on deep learning techniques is a promising approach that can solve several analytical methods problems. These systems can provide accurate solutions to bin picking in cluttered environments, where the scenario is always changing. This article proposes a robust and accurate system for segmenting bin picking objects, employing an easy configuration to adjust the framework according to a specific object. The framework is implemented in Robot Operating System (ROS) and is divided into a detection and segmentation system. The detection system employs Mask R-CNN instance neural network to identify several objects from two dimensions (2D) grayscale images. The segmentation system relies on the point cloud library (PCL), manipulating 3D point cloud data according to the detection results to select particular points of the original point cloud, generating a partial point cloud result. Furthermore, to complete the bin picking system is employed a pose estimation approach based on matching algorithms, such as Iterative Closest Point (ICP).
The system was evaluated for two types of objects, knee tube and triangular wall support, ion cluttered environments. It displayed an average precision of 79% for both models, an average recall of 92% and an average IOU of 89%. As exhibited throughout the article, this system demonstrates high accuracy in cluttered environments with several occlusions for different types of objects

Keywords: Bin picking, Deep learning, Mask R-CNN, Grasping, Neural networks, Artificial Intelligence, ROS, RGB-D multimodal data.

1 Introduction

Robotic bin-picking based on deep learning techniques is a technology that applies a relatively recent concept, utilizing neural networks, intending to enhance the standard bin picking approaches. These approaches are increasingly being used in bin picking areas due to the potential of deep learning policies to further increase the intelligence and learning capabilities of robots.

This article provides an accurate solution to robotic bin-picking picking problems in highly cluttered environments for any type of object, similar to an industrial scenario, where the robot's objective is to pick several objects from different bins, each of them composed of only one type of object. To accomplish this task the framework has different models that accurately detect different types of objects, alternating them according to the determined action, achieving a robust process that detects different types of objects in highly cluttered environments by essentially modifying only the particular Mask R-CNN trained model.

Given these ideas, and after this brief introduction, the second section details several related works to this article. In the third section is described the deep learning training applied to Mask R-CNN neural network with TensorFlow 2, and details how the dataset was generated. The fourth section provides a brief overview of the bin picking algorithm and specifies the main stages of the implemented framework. The succeeding section displays several results acquired with evaluation datasets. To conclude, the paper briefly resumes the implementation and presents possible future developments.

2 Related work

Over the years, several methods have been developed to solve bin picking problems, such as the one proposed by Doumanoglou *et al.* [1] in which an unsupervised feature learning from depth-invariant patches for highly cluttered environments. Pochyly*et al.* [2] implemented a revolving vision system with a specific gripper for organized industrial environments. Choi*et al.* [3] developed a voting-based pose estimation algorithm for highly clutter environments with several occlusions. Yan*et al.* [4] implemented a pipeline for different types of objects in cluttered scenarios using an adaptive threshold segment to accelerate the pose estimation. Leão *et al.* [5] proposed a framework for entanglements objects.

This article focus on solutions based on deep learning techniques, resorting to distinct approaches, namely neural networks, and picking orientations. Lenz *et al.* [6] implemented one of the first Red, Green, Blue - Depth (RGB-D) multimodal parallel gripper-oriented methods, evaluating grasp candidates for random objects (Cornell grasping dataset objects) in a scenario without occlusions, to select the final grasps pose were applied two deep networks with two inputs RGB and depth data. Similar to Lenz *et al.*, Mahler *et al.* [7] developed a research project for generating synthetic datasets, parallel grasps and metrics of grasps based on physics, called Dex-Net. Dex-net 2.0 [8] presented a parallel gripper-oriented method to rapidly predict the probability of success of grasps from depth images.

Diverging from gripper-oriented approaches and focusing more on object-oriented segmentation methods, Zeng *et al.* [9] developed an RGB-D multi-view object-oriented approach for Amazon Picking Challenge (APC). This approach estimates the 6D pose by segmenting and labeling multiple views of the scenario with a fully convolutional neural network and then fitting pre-scanned 3D object models. Le *et al.* [10] developed a work very similar to the proposed implementation in this article, using the same neural network (Mask R-CNN), however, the solution proposed by these authors was for planar objects in cluttered environments using a vacuum gripper.

3 Deep learning training

Deep neural networks are mainly known and selected from their architectures, presenting different features, such as bounding-box, and classifications, among many others. As for this work, it was intended to detect and segment instances of objects in a cluttered environment. Mask R-CNN was chosen from all the possibilities considering that in the first place was the newest stable version of R-CNN, a neural network with good results and a lot of progress made in the segmentation field. Secondly, in several published papers, Mask R-CNN obtained results above average, surpassing a lot of other neural networks. Thirdly, as previously described, the main objective of this implementation was to find a specific known place of the model to grab the object, this way excluding most of the gripper-oriented neural networks. Lastly, it is an open-source neural network with good community support, and with exceptional results concerning the detection and segmentation of different objects with several occlusions [10], [11], [12].

Mask R-CNN consists of two stages. The first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage, which is in essence Fast R-CNN [13], extracts features using region of interest pooling (RoIPool) from each candidate box and performs classification, bounding-box regression, and outputs a parallel binary mask for each RoI [14].

Mask R-CNN was trained in a system composed of a Ryzen 5 5600x processor, GTX 1060 6 GB graphic card and 32GB DDR4 memory. Unlike the original neural network [15], this framework was processed with TensorFlow version 2 (2.4.1), ubuntu 20.04, CUDA toolkit 11.0, Nvidia driver 450.191.01, cuDNN 8.0, and Keras 2.4.0. This adjustment included a few changes to the neural network, assuring the compatibility between the used Mask R-CNN framework, functions and libraries [15], [16]. Nearly all the adjustments required were due to the incompatibility from TensorFlow functions, such as log_graph, set_intersection, among others, that changed between TensorFlow version 1 and TensorFlow version 2 [17]. Furthermore, the dataset class in the main python training script was adapted to load different annotations formats (JSON and COCO) from the custom dataset generated, different neural network parameters were adjusted, and lastly, Mask R-CNN neural network layers (2+,3+, heads,+4,...) training procedure was adapted according to the learning results.

3.1 Dataset generation

The dataset generation was executed with real data, acquired by an RGB-D sensor, and manually labeled by an human operator with third-party annotators support. The RGB-D sensor, identified as Photoneo PhoXi 3D Scanner, is capable of acquiring depth and grayscale data, allowing different types of deep learning training inputs. The scanner has up to 3.2 million 3D points of resolution with a throughput of 16 million points per second, and from 384 to 520 mm of scanning range. The resolution of the 2D images taken is up to 2060x1544, including grayscale and depth images. It can be controlled by software, called

PhoXi Control, via ethernet, which includes a Graphical User Interface (GUI) and an Application Programming Interface (API) [18]. In this implementation, the datasets used for both training and evaluation were divided into two models: 90º elbow tube (Model A) and triangular wall support (Model B).

The training was executed with 2D grayscale data, although in later stages of the process was necessary point cloud data, in order to implement pose estimation and segmentation methods. The 3D point cloud data is automatically generated by the sensor through multi-modal approaches from the initial data acquisition (RGB or grayscale and depth map).

The generated dataset for model A was divided into 4 batches. The first batch was introduced with different 3D views of the scenario, including only one tube or model. Similar to the first, the other batches, from second to fourth, presented a similar scenario with different views, differing in the number of objects presented, in this case, 2 to 4 objects, respectively. The higher the number of batches, the more complex was the environment.

The second resorted approach was very similar to model A: the dataset was divided into three different batches, differing in the number of objects presented. To add robustness to the model it was implemented an augmentation process, with added noise, rotating, cropping, flipping, saturating, brightness, among other processes, enhancing the training process.

Table 1 depicts information about the dataset generation of the two models. Model A dataset is larger than model B, and with more geometrically complex objects to segment, requiring on average 21 seconds to annotate a single object. Taking into account that some images have 4 objects, it requires on average 84 seconds to complete an image annotation. Model B dataset was originally much smaller, including 59 images with 155 annotations; however, with the application of augmentation steps, were generated 179 images from the original 59.

Table 1. Dataset information

Dataset	**Nr of images**	**Nr of annotations**	**Time per annotation (s)**
Model A	488	737	21
Model B	59	155	11.49
Model A + Augmentation	732	-	-
Model B + Augmentation	238	-	-

4 Proposed bin picking system

The basis of the system developed to detect objects in a cluttered environment is depicted in Figure 1. As perceived in this image, there are three main areas, that will be described, namely 2D detection, 3D Segmentation and Pose Estimation.

The idea of the proposed system was to enhance the performance of the recognition framework, and part of the segmentation approach with techniques

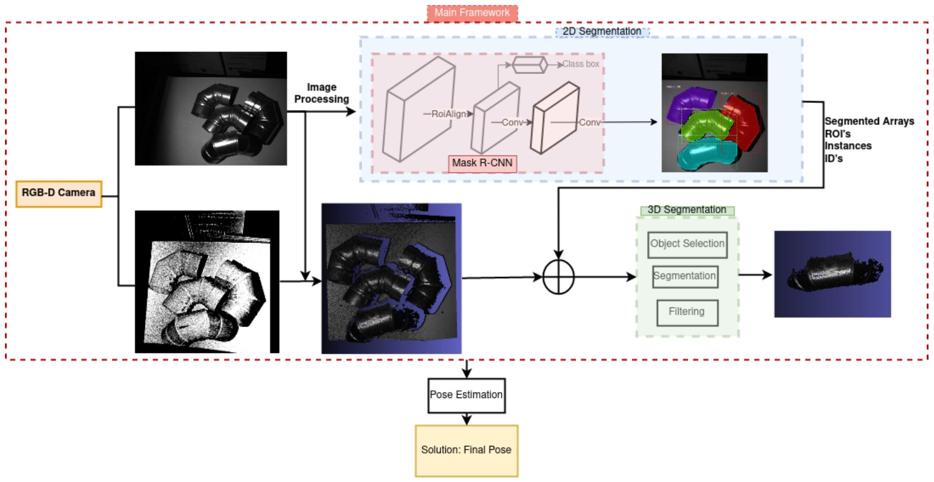

Fig. 1. Object detection pipeline overview

based on deep learning methods, whilst adopting 3D point cloud based heuristics for final the segmentation of the objects and their respective pose estimation. The primary reasons for this proposed adjustment is that deep learning models, when well trained, can provide a high level of robustness to the system, meaning that, when encountering novel and difficult environments (highly cluttered environments), this system can accurately segment the target objects. Another reason is the easy system modification to detect a novel object, only requiring a new neural network model with the knowledge to identify this new object, therefore not involving several complex steps.

4.1 2D object detection

The detection is generated in a 2D space, resorting to an object instance segmentation framework (Mask R-CNN), which can identify several segments, with a mask and bounding box, of the same types of objects.

The detection algorithm starts by loading the specified model. After the loading is completed and is called the process command from the client, the algorithm waits for an image message publication to read its data (lines 3 and 4 from Algorithm 1).

After reading the input image, image processing techniques are performed to enhance the object's visualization. First is implemented a denoise method, with the fastNlMeansDenoising OpenCV function, followed by a Contrast Limited Adaptive Histogram Equalization (CLAHE). The functions were applied in this order to first remove noise from the image, because CLAHE causes noise in near-constant regions as the background, and to improve the contrast in the image, improving several detections with darker images.

Mask R-CNN detection, explicit in Algorithm 1 line 9, extracts the number of objects, mask (segmentation) of each object and bounding boxes. Afterwards is generated a 2D array with all the masks discriminated by a different id for each instance, the number of objects, the center coordinates (*x*,*y*) of each bounding box, and the image resolution, and is published as an 8-bit image (2D array) topic to be received by the 3D point cloud segmentation algorithm. In the end is published a result to the client: successful if is collected an acceptable point cloud from the 3d point cloud sensor, and unsuccessful if the point cloud is not acceptable or is not received.

Algorithm 1 $mask\_img \leftarrow Object\_Detection(img, m)$

```
    Input: $img = image, m = trained\_model$
 1: $Device$ = "cpu"
 2: Subscribe to img topic
 3: wait for img msg
 4: $img = fastNlMeansDenoising(img)$
 5: $limit = exponential\_function(image\_meanvalue)$
 6: $CLAHE = createCLAHE(cliplimit = limit, tileGridSize = (3, 3))$
 7: $img = CLAHE.apply(img)$
 8: $detection = m.detect(img)[0]$
 9: $info\_array = instances, img.size$
10: for i in range(0, instances) do
11:     $roi\_x = (rois[i, 1] + rois[i, 3])/2$
12:     $roi\_y = (rois[i, 0] + rois[i, 2])/2$
13:     $info\_array = info\_array, roi\_x, roi\_y$
14:     $binary\_mask = masks[:, :, i].astype(np.uint8) * (i + 1)$
15:     $id\_mask = (binary\_mask == (i + 1))$
16:     $final\_mask[id\_mask] = binary\_mask[id\_mask]$
17: end for
18: $mask\_img = append(final\_mask, info\_array)$
19: Publish $mask\_img$
```

4.2 3D point cloud segmentation

The 3D point cloud segmentation algorithm expects to receive an image array containing all the detection information provided from Object_Detection, and the original point cloud acquired at the same time as the grayscale image. When the point cloud is received (lines 1-2 Algorithm 2), the point cloud message is transformed from a ROS message `sensor_msgs::PointCloud2` to PCL `pcl::PointCloud< XYZRGBA >`.

As described in the previous sub-section, the information exported through the image array is organized in specific variables, as presented in Algorithm 2 Lines 9-14. Succeeding the data management, the original point cloud is resized, allowing to treat this point cloud as a 2D array with the same shape as the original grayscale image acquired, and is computed the mean 8x8 center region z value of each segment (instance), shown in lines 16-26 (Algorithm 2), to predict the object closest to the sensor in line 27(Algorithm 2).

After estimating the highest point cloud segment, the segment is extracted from the original point cloud throughout its ID (line 28 to 34 of Algorithm 2) and are applied several filters, such as voxelgrid, pass-through and statistical outlier removal (line 35 of Algorithm 2), that essentially were used to remove noise and unnecessary information. This algorithm structure improves the total time consumed because, instead of selecting and filtering every point cloud segment detected, it only selects and filters the highest segment points, which is the essential instance to pick up.

Algorithm 2 $PointCloud\_Segmentation(mask\_im)$

```
    Input: $cloud = pointcloud$
 1: Subscriber to cloud_msg
 2: Subscriber to mask_im
 3: while !$cloud\_received\_flag$ do
 4:     $ros :: Duration(0.01).sleep();$
 5:     $ros :: spinOnce();$
 6: end while
 7: for int i=0; i¡Instances; i++ do
 8:     $cloud\_center\_x[i] = info\_array.roi\_x$
 9:     $cloud\_center\_y[i] = info\_array.roi\_y$
10: end for
11: Resize $cloud\_received(height * width)$
12: for $l = 0; l < Instance, l++$ do
13:     for $i = cloud\_center\_x[l] - 8; i \leq cloud\_center\_x[l] + 8, l++$ do
14:         for $j = cloud\_center\_y[l] - 8; j \leq cloud\_center\_y[l] + 8, l++$ do
15:             if $cloud\_received \to at(i, j).z > 0$ then
16:                 cnt[l]++
17:                 $h[l] = h[l] + cloud\_received \to at(i, j).z$
18:             end if
19:         end for
20:     end for
21:     h[i]= h[i] / cnt[i]
22: end for
23: $higher\_cloud = std :: distance(h, std :: min\_element(h, h + Instances))$
24: for int i=0; i¡width; i++ do
25:     for int j=0; j¡height; j++ do
26:         if id_cloud(i,j)==higher_cloud+1 then
27:             $partial\_cloud = push\_back(cloud\_received- > at(i, j))$
28:         end if
29:     end for
30: end for
31: $filtered\_cloud = Filters(partial\_cloud)$
32: Publish $filtered\_cloud$
```

The following images, displayed in Figure 2, are results from the detection and segmentation system from one random sample in each dataset of model A and B. As observed, the system receives the original image (left column of Figure 2) and detects the distinct instances in the scene, identified by colour segmentation masks for easy visualization (Middle-Left column). Analysing the results, even in novel and arduous environments, the system can detect and segment with high accuracy each object, as observed in the last columnrow 2 of Figure 2.

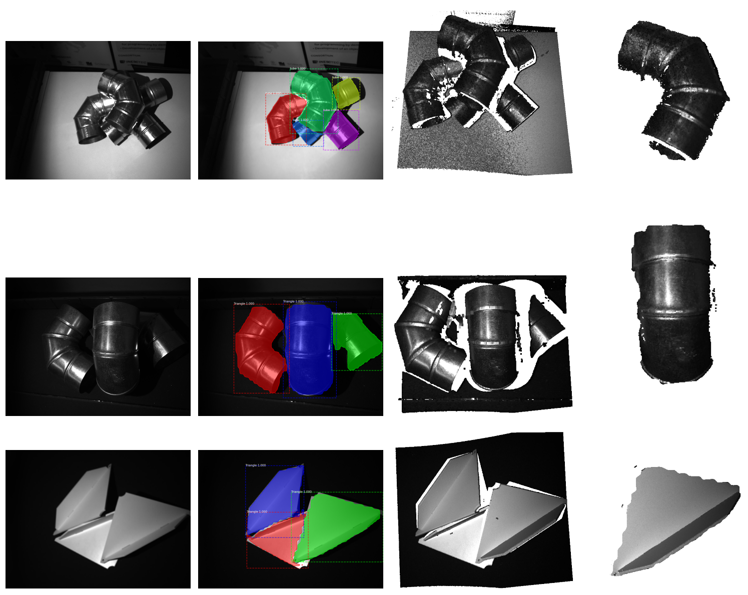

Fig. 2. Evaluation results. (Left) original grayscale image (Middle-Left) Detection mask (Middle-Right) Original point cloud (Right) Partial point cloud segmentation

4.3 Pose estimation and grasping

The pose estimation algorithm is implemented after the point cloud segmentation, and is based on cloud matching, between the reference point cloud (displayed in Figure 3) and the acquired point cloud. Essentially, the segmented point cloud is matched with a reference point cloud provided through the CAD model of the specific part that is intended to be picked; subsequently is generated a grasp solution from the grasp evaluation system, detecting the best possible grasp (from all the pre-computed candidates) to pick the object in its current position [19].

The algorithm begins loading the pipeline configurations and the reference point cloud, acquired either from the 90º elbow tube CAD model or the trian-

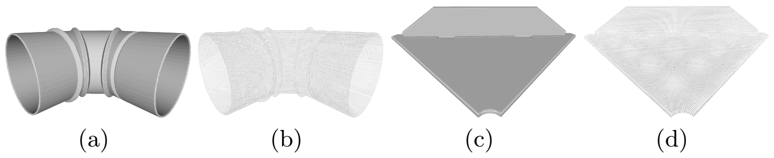

Fig. 3. Reference point cloud. (a,c) CAD model (b,d) Generated point clouds

gular wall support CAD model, displayed in Figure 3. Subsequently, the point cloud is filtered and are estimated the surface normals and curvatures. Concluding the first step, the reference point cloud is stored, and are applied and stored the keypoints and their descriptors.

The initial pose is estimated through RAndom SAmple Consensus (RANSAC), and an initial alignment of the two point clouds is refined, using ICP algorithm, as detailed in Figure 4, applying a transformation from the reference point cloud, displayed at middle right image of Figure 4 to the detected point cloud in the middle left image of Figure 4.

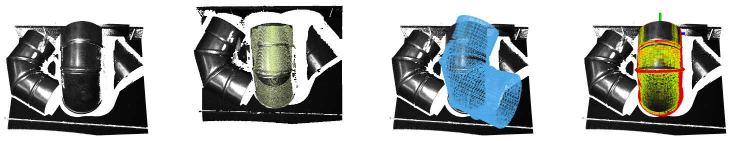

Fig. 4. Pose estimation overview

Posterior to the pose estimation, are checked possible object collisions to evaluate the best grasp candidate among all the previously taught grasp poses exhibited in Figure 5. This decision is formed through scores that are given based on heuristics, such as euclidean distance, depth distance; roll, pitch and yaw distance, among others. In the end, the grasp candidate with the lower cost is chosen [20].

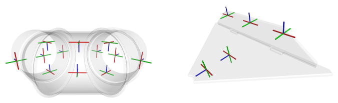

Fig. 5. Grasping candidates

After the full procedure (detection, segmentation, pose estimation and grasp estimation) the robot manipulator starts the bin picking action, displayed in Figure 6, moving from the initial static pose (Figure 6 (a)) to the estimated grasping pose ((b) and (d)), and concluding with a trajectory to place the object in the bin ((c) and (e)). The same process can be repeated for any other object, modifying only some features of the main object detection framework.

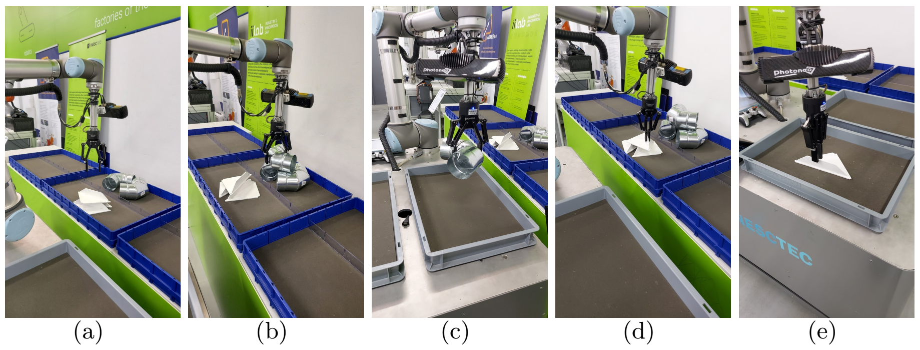

Fig. 6. Bin picking action sequence

5 System tests and evaluation

The object detection and segmentation framework was tested on eleven types of scenarios (datasets), detailed in Table 2, divided into six types of datasets for model A and five types of datasets for Model B. The computer utilized for this evaluation is described in Section 3, it was only used the computer processor for the inference stage, only using the GPU for neural network training. The evaluation datasets (depicted on Figure 7, are divided according to the number of objects presented in the bin and the type of environment, varying between a known environment (images that trained the Neurak Network (NN)) and a novel environment, with less illumination, new positions, viewpoints and backgrounds, creating an arduous scenario to apply and test the developed framework.

Table 2. Evaluation dataset

Evaluation dataset	**Number of images**	**Number of annotations**
tubes_eval	118	269
triangles_eval	54	102

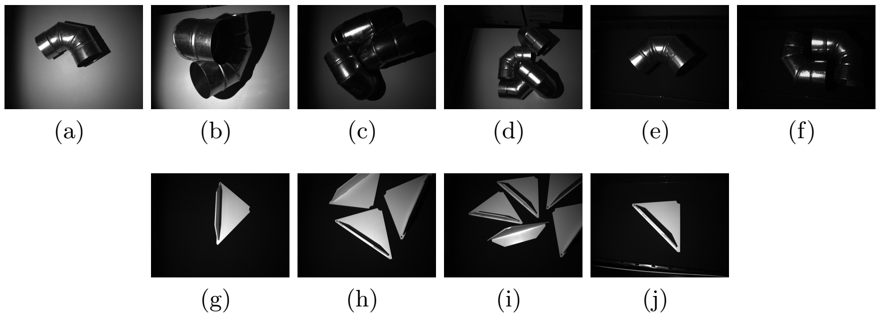

Fig. 7. Evaluation Datasets. Top- Model A (knee tube), Bottom- Model B (triangle)

5.1 Object detection and segmentation efficiency and performance

The object segmentation framework average results are illustrated in Table 3, depicting all the standard deep learning metrics, such as Intersection over Union mask (IoU_mask, represented as mIOU), Average precision (Ap), Average Recall (Ar), and F1 score.

Table 3. Metrics precision

Overall	Ap	Ap_50	Ap_75	mIOU	Ar_50	F1_50
Model A	0.829	0.971	0.937	0.905	0.955	0.904
Model B	0.751	0.893	0.819	0.873	0.890	0.885

Compared to model A, model B results display a lower average score on every metric, however still exhibiting high values, around 85% on every metric apart from Ap with 75%. These results are expected because model B had a lower quality training compared to model A, with less training time, fewer images, just one camera viewpoint and fewer batches with a distinct number of objects.

Comparing these results to the results of the original Mask R-CNN [14], it is observed that the results acquired with this trained models are better. The improvement is understandable due to the type of objects displayed in an image (classes), the simpler background due to the application in an industrial environment and the training phase of the neural network.

The execution time of the object detection and segmentation framework, illustrated in Table 4, is divided into the detection stage, segmentation and segmentation with filtering. The values presented are an average measurement from all datasets samples.

The detection stage, on average, requires around 0.49 seconds to generate the necessary masks of the input image. The segmentation without filters requires an average of 0.04 seconds, generating the segmented point cloud of the input mask closest to the sensor, however exhibiting all the noise and outliers produced by

Table 4. Execution time (CPU)

Overall	**Instances number**	**Det (s)**	**Seg (s)**	**Seg + Filters (s)**
Model A	1	0.475	0.048	0.452
	2	0.481	0.070	0.431
	3	0.519	0.045	0.417
	4	0.518	0.038	0.483
Model B	1	0.464	0.033	0.299
	3	0.508	0.037	0.278
	5	0.504	0.028	0.193

the depth sensor in the original point cloud. The segmentation with filtering requires around 0.36 seconds, applying pass-through, voxelgrid and remove outlier filters.

The average system total time is 0.53 seconds without filtering and 0.85 seconds with filtering. Therefore, the application of filters requires on average 0.32 seconds, increasing or decreasing the time required according to the number of points given to the filters.

Analysing these results, one big advantage is that the time elapsed is not dependent on the instance number, due to the segmentation algorithm structure. As displayed in Table 4, the only noticeable difference is in the segmentation and filtering stage between model A and model B, diverging from 0.44 seconds on average to 0.25 seconds, on average, respectively. This difference is due to the original point cloud generation; as shown later in the results, the model B point cloud has much less noise and outliers than the model A point cloud.

These average execution time values can fluctuate according to the model resolution, namelly the values presented in this article are for 1024x1024 image resolution, which was the resolution that the images were resized when training the model. The time can be reduced by decreasing this resolution.

6 Discussion and Future work

This paper addressed bin picking problems in clutter environments using a standard robot manipulator with a parallel gripper, implementing a system based on deep learning techniques.

After analysing different frameworks, it was proposed an approach to detect and segment instances based on instance segmentation techniques and point cloud procedures, using grayscale image and point cloud data generated with an RGB-D sensor. This implementation estimates the different instance segments in a grayscale image by applying image processing techniques and Mask R-CNN neural network. Subsequently, it is received point cloud data to estimate and extract the object (partial point cloud) closest to the sensor, to be sent to the pose and grasp estimation systems, and finally, the robot manipulator grasps the object through the desired position. The proposed system is capable of segmenting several types of objects in cluttered environments. However, in this case, it was only trained to detect two models, a 90º elbow tube model (model A) and a triangular wall support model (model B).

The accuracy and execution time achieved by the proposed system allows to detect and segment instances with several occlusions in hard environments, being able to estimate the closest object to the sensor, which will probably be the easiest to pick up, with acceptable execution times for an industrial solution.

The current implemented system can be further improved by estimating an initial pose of the segmented objectoutput cloud, given the fact that the partial point cloud has the same (x, y) position as the original one, improving the efficiency of the matching algorithm in the pose estimation system, and decreasing the iterations required to match the two point clouds. In addition to initial pose estimation, the system can be implemented with multiple channels of neural network inputs, modifying Mask R-CNN to predict the output based on, for example, two types of data (such as grayscale and depth images), instead of segmenting instances based only on a 2D image. On the other hand,The last future work for this system is to enhance system performance with Mask R-CNN models with less resolution and similar precision values.

Considering that the neural network training stage is very time-consuming and exhausting to create instance segment labels for all the original images, this stage can be improved with an automated image generation and labelling system, for example, using 3D creation software, such as Blender.

7 Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101006798.

References

1. A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016. [Online]. Available: https://doi.org/10.1109/cvpr.2016.390
2. A. Pochyly, T. Kubela, V. Singule, and P. Cihak, "Robotic bin-picking system based on a revolving vision system," in *2017 19th International Conference on Electrical Drives and Power Electronics (EDPE)*. IEEE, Oct. 2017. [Online]. Available: https://doi.org/10.1109/edpe.2017.8123228
3. C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam, "Voting-based pose estimation for robotic assembly using a 3d sensor," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, May 2012. [Online]. Available: https://doi.org/10.1109/icra.2012.6225371
4. W. Yan, Z. Xu, X. Zhou, Q. Su, S. Li, and H. Wu, "Fast object pose estimation using adaptive threshold for bin-picking," *IEEE Access*, vol. 8, pp. 63 055–63 064, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.2983173

5. G. Leão, C. M. Costa, A. Sousa, and G. Veiga, "Detecting and solving tube entanglement in bin picking operations," *Applied Sciences*, vol. 10, no. 7, p. 2264, Mar. 2020. [Online]. Available: https://doi.org/10.3390/app10072264
6. I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," 2013. [Online]. Available: https://arxiv.org/abs/1301.3592
7. J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroger, 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)*. IEEE, May 2016. [Online]. Available: https://doi.org/10.1109/icra.2016.7487342
8. J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017. [Online]. Available: https://arxiv.org/abs/1703.09312
9. A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," 2016. [Online]. Available: https://arxiv.org/abs/1609.09475
10. T.-T. Le and C.-Y. Lin, "Bin-picking for planar objects based on a deep learning network: A case study of USB packs," *Sensors*, vol. 19, no. 16, p. 3602, Aug. 2019. [Online]. Available: https://doi.org/10.3390/s19163602
11. T. Höfer, F. Shamsafar, N. Benbarka, and A. Zell, "Object detection and autoencoder-based 6d pose estimation for highly cluttered bin picking," 2021. [Online]. Available: https://arxiv.org/abs/2106.08045
12. M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data," 2018. [Online]. Available: https://arxiv.org/abs/1809.05825
13. R. Girshick, "Fast r-cnn," 2015. [Online]. Available: https://arxiv.org/abs/1504.08083
14. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017. [Online]. Available: https://arxiv.org/abs/1703.06870
15. W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.
16. "Tested build configurations," https://www.tensorflow.org/install/source\#gpu, accessed: 2022-03-03.
17. "Mask r-cnn for object detection and segmentation using tensorflow 2.0," https://github.com/ahmedfgad/Mask-RCNN-TF2, accessed: 2022-03-13.
18. "Industrial 3d scanner: Phoxi®." [Online]. Available: https://www.photoneo.com/phoxi-3d-scanner/?gclid=EAIaIQobChMI_4y38LLp-AIVCcPVCh2E0A8eEAAYASAAEgJMwfD_BwE
19. C. Costa, H. Sobreira, A. Sousa, and G. Veiga, "Robust 3/6 dof self-localization system with selective map update for mobile robot platforms," *Robotics and Autonomous Systems*, vol. 76, pp. 113–140, 02 2016.
20. J. Carvalho de Souza, C. Costa, L. Rocha, R. Arrais, A. Moreira, E. Pires, and J. Cunha, "Reconfigurable grasp planning pipeline with grasp synthesis and selection applied to picking operations in aerospace factories," *Robotics and Computer-Integrated Manufacturing*, vol. 67, 07 2020.