robotics

MDPI

Article

Bin Picking for Ship-Building Logistics Using Perception and Grasping Systems

Artur Cordeiro [1,2], João Pedro Souza [1,3], Carlos M. Costa [1,3], Vítor Filipe [1,2], Luís F. Rocha [1] and Manuel F. Silva [1,4,*]

1 INESC TEC—INESC Technology and Science, 4200-465 Porto, Portugal
2 School of Science and Technology, University of Trás-os-Montes e Alto Douro (UTAD), 5000-801 Vila Real, Portugal
3 Faculty of Engineering, University of Porto (FEUP), 4200-465 Porto, Portugal
4 Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto, Rua Dr. António Bernardino de Almeida, 431, 4249-015 Porto, Portugal
* Correspondence: mss@isep.ipp.pt; Tel.: +351-22-834-05-00

Abstract: Bin picking is a challenging task involving many research domains within the perception and grasping fields, for which there are no perfect and reliable solutions available that are applicable to a wide range of unstructured and cluttered environments present in industrial factories and logistics centers. This paper contributes with research on the topic of object segmentation in cluttered scenarios, independent of previous object shape knowledge, for textured and textureless objects. In addition, it addresses the demand for extended datasets in deep learning tasks with realistic data. We propose a solution using a Mask R-CNN for 2D object segmentation, trained with real data acquired from a RGB-D sensor and synthetic data generated in Blender, combined with 3D point-cloud segmentation to extract a segmented point cloud belonging to a single object from the bin. Next, it is employed a re-configurable pipeline for 6-DoF object pose estimation, followed by a grasp planner to select a feasible grasp pose. The experimental results show that the object segmentation approach is efficient and accurate in cluttered scenarios with several occlusions. The neural network model was trained with both real and simulated data, enhancing the success rate from the previous classical segmentation, displaying an overall grasping success rate of 87.5%.

Keywords: deep learning; bin picking; grasping; segmentation

Citation: Cordeiro, A.; Souza, J.P.; Costa, C.M.; Filipe, V.; Rocha, L.F.; Silva, M.F. Bin Picking for Ship-Building Logistics Using Perception and Grasping Systems. *Robotics* **2023**, *12*, 15. https://doi.org/10.3390/robotics12010015

Academic Editor: Xinjun Liu

Received: 2 December 2022
Revised: 9 January 2023
Accepted: 12 January 2023
Published: 18 January 2023

1. Introduction

Over the past few years, smart logistics and smart warehouse technologies have been evolving at a rapid pace to provide reliable delivery of products to the supply chains and their customers. Automation and connectivity between machinery and equipment in smarter, more effective distribution centers fueled the development of new operational concepts while advancing the current manufacturing systems.

Large distribution centers currently use autonomous systems to improve their intralogistics efficiency and assist personnel with physically demanding and time-consuming tasks. Examples of these facilities are Amazon, UPS, and DHL facilities, employing systems detailed in [1]. Currently, these systems are mostly used in warehouses and manufacturing facilities, since they have a well-known environment, simplifying several tasks, such as robot navigation, boosting labor efficiency and decreasing costs and job execution times. Additionally, most of the applications are employed in autonomous guided vehicles (AGV), a concept deeply developed in the past few years.

Although shipbuilding has a significant impact on a number of vital transportation industries, it is currently under-explored. In order to achieve efficiency, safety, and affordable prices, it must evolve by implementing new technologies related to automatic data identification and autonomous vehicles, as developed in [2–6]. The shipbuilding process is

known for being challenging, since its environments can range from very small dark rooms to very large storage spaces, which may have hazardous chemicals and heavy machinery. As such, these unfavorable working environments may result in workplace accidents.

The primary goal of the proposed paper is to create an autonomous picking system employed on a mobile manipulator, focused on object segmentation, pose estimation, and grasping, that can move a variety of objects from their original containers to the appropriate places, freeing workers from risky material handling tasks. The current available industrial solutions are only capable of accomplishing this problem in well-structured and controlled environments, mostly resorting to cooperation between robots and operators, and they have difficulties with estimating a solution in highly cluttered scenarios composed of identical objects.

This paper presents a fully autonomous picking solution, composed of three configurable pipelines that were designed for object segmentation, pose estimation, and grasping. It is capable of:

1. Segmenting individual object instances in highly cluttered and complex scenarios with precision and efficiency;
2. Identifying and segmenting singular objects from environments composed of several layers of identical objects;
3. Estimating the 6-DoF pose of the segmented object;
4. Generating several grasp candidates and selecting a feasible grasp according to a set of heuristics;
5. Working in different environment conditions, for example, various levels of illumination and various positions and orientations of the mobile manipulator in relation to the bin.

Experimental results were acquired in a factory use-case scenario, evaluating the perception pipelines, from scanning of the bin to the placing of the object in the correct transportation container for executing intralogistics operations with an omnidirectional mobile platform equipped with a UR-10 robotic arm, a Robotiq 2 finger gripper, and a Photoneo PhoXi 3D scanner, which is exhibited in Figure 1.

Figure 1. Use-case scenario: (**Left**) omnidirection robot manipulator, (**Right**) bins.

The remainder of the paper is structured as follows: Section 2 presents related works, developed with similar objectives to the proposed system. Section 3 details the hardware we used and the proposed pipelines, followed by the results and a brief discussion in Section 4, and conclusions and future work in Section 5.

2. Related Work

Pick and place operations are mostly related to bin picking problems; however, they can be applied to several other situations—for example, vertical picking from shelves [7]. The work presented in this paper falls into that scenario. Currently, these operations constitute the majority of the industrial robotic applications and have been extensively studied by research institutes, schools, and industries due to the vast demand and the advantages that it delivers.

Image segmentation is a known computer vision research field that is crucial to perception systems; already, there are several developed techniques. However, a common obstacle is present in segmentation approaches—namely, applying the segmentation to objects in densely cluttered scenarios, due to the fact that these types of environments have multiple occlusions, objects in different depth planes, and distinct objects. A similar issue occurs in other perception algorithms, which struggle to recognize and select objects in cluttered environments, especially when the environment is made up of several identical objects, creating issues with the perception and recognition of a specific object.

2.1. Object Segmentation and Pose Estimation

In a common robot bin-picking task, scenes contain multiple instances of diverse object types, causing several occlusions; therefore, instance segmentation techniques are required before estimating the object pose. Deep-learning-based approaches have displayed remarkable performances in executing this task. Blank et al. [8] presented an object recognition pipeline combining convolutional neural networks (CNNs) and feature-matching methods to estimate the 6-degrees of freedom (DoF) poses of objects; it is capable of recognizing textureless objects. Wada et al. [7] proposed an object segmentation approach to pick objects in narrow spaces. It accumulates segmentation results with multiple camera angles, using a voxel grid map to generate 3D object segments. Le et al. [9] developed a segmentation-based deep learning approach for planar objects in a cluttered environment, while only requiring 2D data to estimate a 3-DoF pose, in order to pick an object with a vacuum gripper. Zhuang et al. [10] proposed a semantic image segmentation using the Part Mask R-CNN neural network combined with the point pair feature method to estimate 3D object poses. Kozák et al. [11] developed a pipeline to estimate poses from 2D images for textureless industrial metallic parts for semistructured bin-picking based on a CNN.

Furthermore, certain neural networks were developed to segment and estimate pose based only on 3D point-cloud data. Dong et al. [12] estimated 6-DoF poses in cluttered scenarios by employing deep learning-based instance segmentation of 3D point-cloud data with a point-wise pose regression network. Xu et al. [13] proposed a similar approach for instance segmentation based on a fast point-cloud clustering network and a fast clustering algorithm, extracting features of each point and inferring geometric center points of each instance simultaneously before clustering the remaining points to the closest geometric center.

2.2. Grasping Planner

Grasping planners are employed to plan how to pick an object in an efficient manner and without collisions. Buchholz et al. [14] proposed an optimal grasping-pose-estimation system with collision avoidance for bin-picking problems, allowing one to reduce system cycle number. He et al. [15], developed a novel approach to scooping based on a mobility analysis with a two-finger gripper. Ichnowski et al. [16] presented an algorithm that speeds up the execution of a bin-picking robot's operations based on a grasp-optimized motion planning. Leão et al. [17] suggested a bin-picking solution for entangled objects; the algorithm represents a curved tube as an ordered list of cylinders and joints using point-cloud-analysis algorithms.

In addiction, new solutions were developed, aggregating perception and grasping pipelines into a singular framework, based on deep learning techniques; these estimate grasps recurring to neural networks. Iriondo et al. [18] adapted a deep graph convolutional

network to predict affordances scores for suction and other gripper end-effectors in an industrial bin-picking environment instead of scene segmentation; this way, it does not require a pose-estimation method. Jiang et al. [19] proposed a system for textureless planar-faced objects using depth images based on a deep convolutional neural network to predicted grasp suction points and optimal grasp patterns for a hand with two vacuum cups. Several works developed a similar approach to estimate grasp points or affordance scores, such as [20–27]. However, these approaches are generally not applied in cluttered scenarios, as observed in the previous works.

Although there are several approaches to solving bin picking problems, the proposed solution provides a re-configurable pipeline using deep learning techniques that is capable of segmenting different object types in cluttered environments with challenging conditions. Some approaches presented in Section 2.2 can solve the problem of having several different types of objects; however, 2D grasp solutions generate several grasping difficulties, and the grasp solution is only selected based on the best estimated grasp candidate.

3. System Overview

To support the development of the proposed bin picking system, a use-case from the shipbuilding industry was considered. We focused on improving the image segmentation process, which currently is the main bin-picking bottleneck. It is composed of two object models, a 90° flow pipe (Model A) and a triangular wall support (Model B), illustrated by the CAD models depicted in Figure 2. These models are very different from one another, covering various features. Model A is made of reflective material, which can lead to a number of visual issues, especially when applied in an environment with a variable illumination system, as exhibited in this case. Model B is composed of a textureless material, which causes several difficulties in object detection, as detailed in [28], because the deep neural network has to analyze fewer features. In addition, when demonstrated in cluttered scenarios, there are some instances where the height difference between objects is close to zero, making it challenging to select the appropriate objects to grasp.

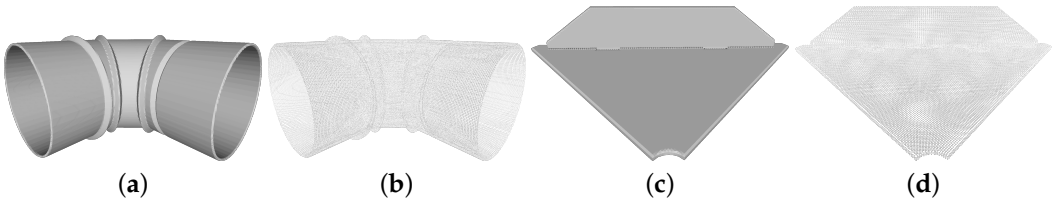

Figure 2. Use-case objects: (**a**) 90°-flow-pipe CAD model, (**b**) 90°-flow-pipe point cloud, (**c**) triangular-wall-support CAD model, (**d**) triangular-wall-support point cloud.

These object models were chosen because they are frequently used in ship building; however, the proposed framework can operate with any type of model, only requiring a deep learning (Mask R-CNN) training model and the object's CAD model.

The main objective of this work was to improve the perception approach adopted in a previous work [29] by developing a system capable of generating solutions in any type of environment, including dense cluttered ones, which is an important aspect to a bin-picking problem.

The implemented system is divided into three pipelines: segmentation, pose estimation, and grasping. The segmentation is the main objective of the presented paper and is detailed in Section 3.1, where we also briefly describe the pose estimation and grasping pipelines to understand the entire bin-picking system employed.

3.1. Proposed Segmentation Method

The proposed perception system focuses primarily on solving the segmentation problem, creating a solution with accurate perception and segmentation of objects in cluttered picking scenarios, as illustrated in Figure 3, which is consistent with observed industry

patterns. This enables the software to be used in various picking scenarios with either straightforward or complex perception issues.

Figure 3. Cluttered environment example for bin picking.

In this Section, we detail the full perception pipeline, from receiving the initial inputs to the segmented point cloud acquisition, describing the development of each process. The process of object segmentation is based on deep learning detection and point-cloud-segmentation methods and is responsible for detecting, selecting, and segmenting the best object to grasp, according to specific heuristics, such as the highest object, or the object segment with the most surface area, as briefly depicted in Figure 4 (adapted from [30]).

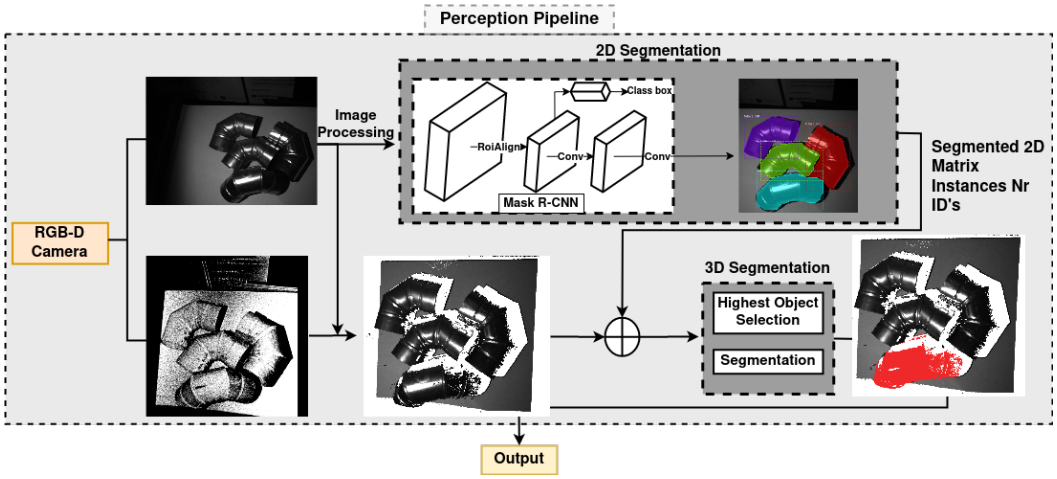

Figure 4. Perception pipeline for image segmentation (adapted from [30]).

The proposed approach is detailed in Figure 5, which depicts the system divided into two main algorithms, namely, 2D segmentation and 3D segmentation. The approach was implemented through a ROS action server, which receives goals from the client (mobile platform) and transmits results and feedback.

The system has three main stages. The first is preprocessing, which is executed when the robot manipulator is initially set up and corresponds to the loading of the system configuration. The second is loading, in which the requested object model is considered and the neural network model is loaded into memory. Lastly, the third stage is described as the process stage, which occurs after receiving the sensor data, performing the object segmentation.

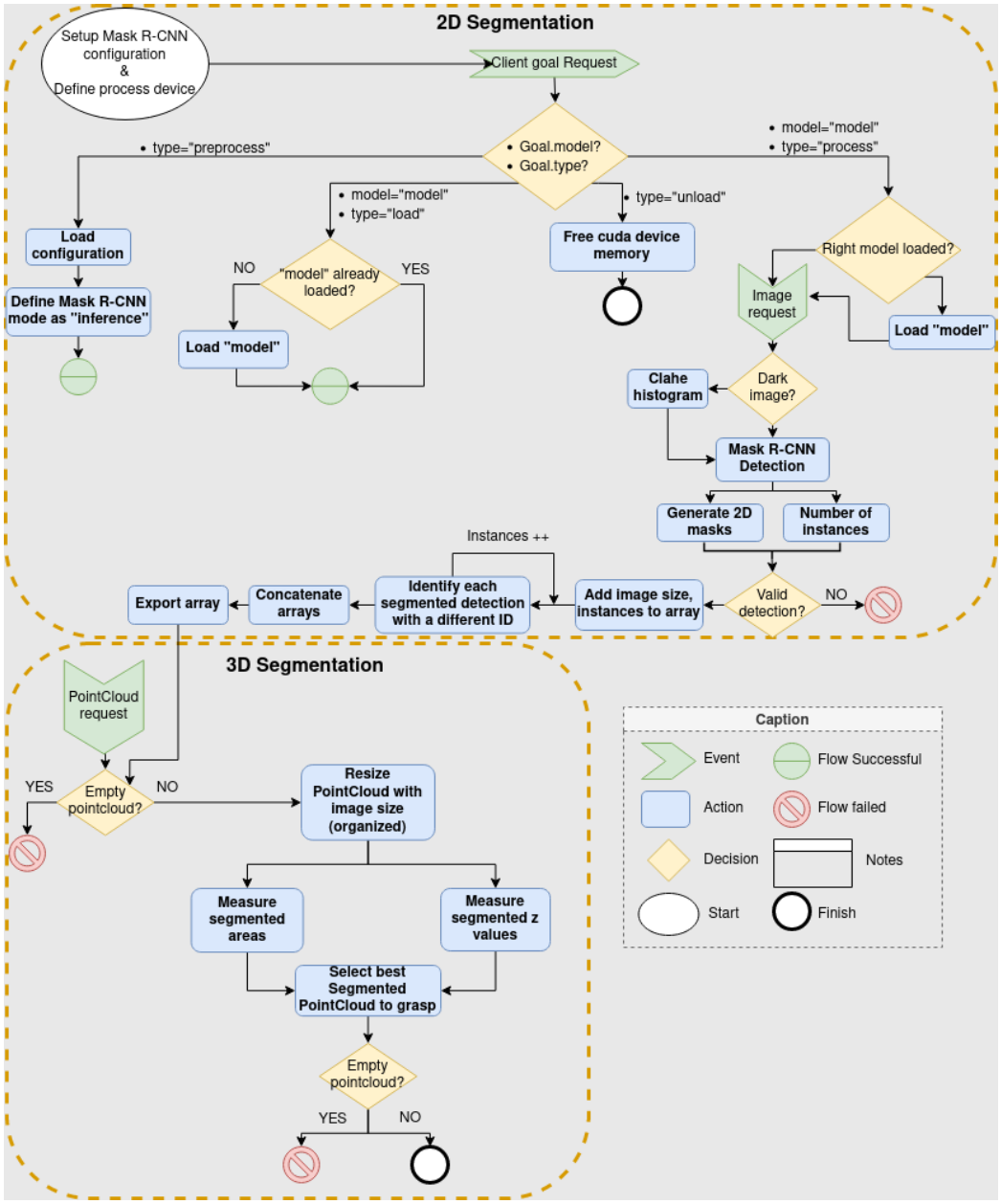

Figure 5. Perception pipeline overview.

In this study, we used a method based on image and point-cloud (generated through the 3D sensor with RGB and depth data) analysis to perform the 3D data segmentation. The Mask R-CNN was the deep neural network chosen, as detailed in Section 3.1.1. This framework was used for object-instance segmentation [31], which accurately and quickly identified the objects in the 2D image. Mask R-CNN was modified and trained using the parameters explained in Section 3.1.1 in order to obtain the findings shown in Section 4.

Mask R-CNN configurations are loaded into the system before choosing the device designed to execute the deep neural network inference. The appropriate training model and parameters are then loaded for the particular object that is to be detected. After receiving the captured 2D image, the system checks it to ensure it has valid data before feeding it to the detector, which will generate the detection masks. Each object segment mask is given a unique ID number in order to identify each object, and the data are later exported as a 2D array, which is sent to the 3D segmentation algorithm.

After all objects in the environment have been segmented in 2D, it is necessary to choose and segment the most fitting object to grasp using the 3D point-cloud data. The cur-

rent heuristic selects the object that is closest to the sensor and does not have any other objects on top of it.

The system begins by receiving the image array created during the first segmentation phase, ensuring that the point-cloud dimensions match the image array masks.

Later on, the point-cloud coordinate system is transformed from its camera optical frame to the sensor body frame, which is parallel to the bin. This is necessary because the Photoneo 3D scanner relies on structured light sensing, and as such, requires overlapping fields of view between the camera and the laser projector. For achieving this overlap, the scanner has the camera tilted 15° in relation to the sensor's body frame. This tilt must be taken into consideration for ensuring that the height measurements are properly made on frame M instead of frame C, as shown in Figure 6. As observed, the z axis of frame C has a 15° tilt, requiring a change in the coordinate system (represented in Equation (1)), which changes the original frame to the corrected frame, converting the z axis of frame C to the x axis of frame M and transforming point coordinates (X, Y, Z) to (X′, Y′, Z′).

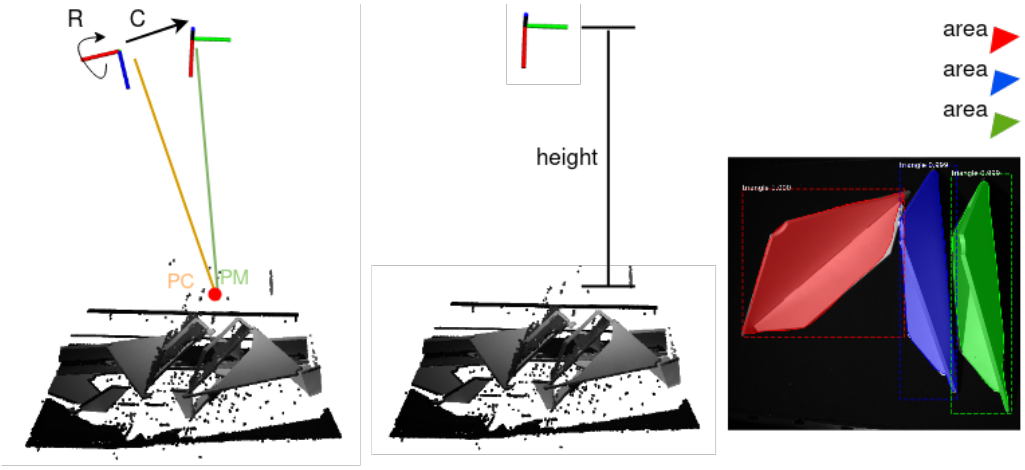

Figure 6. Height and area estimation illustration.

The measured height value (Equation (1)), acquired from the X′ value of the points is adjusted to 70% of the weight to select the best object, and the other 30% is estimated by measuring object areas, as described in Equation (2). The weight distribution was defined based on the measurement priority, in addition to evaluations realized to improve the object selection.

Finally, the segmented instance with a higher *object_selection_weight* (Equation (2)) was selected and extracted, concluding the 3D segmentation.

$$PM = R(PC - C) \Rightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \quad (1)$$

$$object\_selection\_weight = 0.7 \times height + 0.3 \times area \quad (2)$$

The method described allows the segmentation of objects in various environments, requiring a trained Mask R-CNN model with high precision and accuracy values. The pipeline can estimate results with repeatable average computation time, exhibiting a minimal standard deviation in different scenarios, involving one or more objects.

3.1.1. Mask R-CNN Training

As described, the 2D detection is realized with the support of a deep neural network referred to as Mask R-CNN [31]. Mask R-CNN was chosen from all the possibilities considering that in the first place, it was the newest stable version of R-CNN, a neural

network with good results that has made a lot of progress made in the segmentation field, as analyzed by works [32–34]. Secondly, in several published papers, Mask R-CNN obtained above-average results, surpassing several neural networks. Thirdly, as described in the main goal of this implementation, the objective was to find a specific known pose of the model to grasp the object, this way excluding most of the gripper-oriented neural networks designed to estimate the best grasp possibility. Lastly, it is an open-source neural network with good community support, and it has attained exceptional results concerning the detection and segmentation of different objects.

This neural network was adapted to train and inference in TensorFlow version 2 (2.5.1) and was configured with hyper-parameters to obtain better results, which are detailed in Table 1. In essence, it is a modified version of the framework defined in [35], composed of a similar architecture, including a ResNet-101 backbone based on a feature pyramid network (FPN). This backbone was employed considering the fact that, compared to all the other ones, such as ResNet-18, ResNet-34, ResNet-50, and ResNet-152, it has better results concerning average precision, as analyzed in [31,36]. However, it requires more time to converge than several other architectures.

Beyond the framework modification implemented, allowing us to use TensorFlow version 2, we added the ability to read both JSON and COCO labels, a transpose layer to enhance the mask resolution, and more augmentation techniques to augment the dataset, producing a better training process.

Figure 7 displays the result of adding a new transpose layer and modifying the mask shape of the neural network. As observed, the curling effect produced in the first segmentation, created by the predicted mask, is reduced in the second image, fixing the 4 mm peak value from the curling error presented in the edge of the mask.

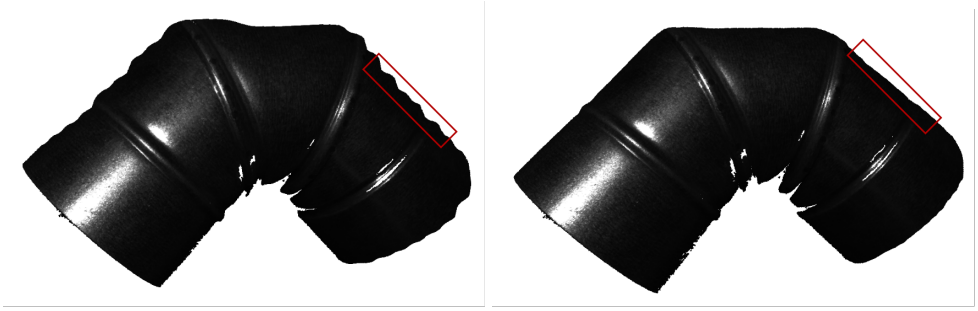

Figure 7. Transpose layer addition and mask-shape-modification results: (**Left**) without the proposed configuration; (**Right**) with the proposed configuration.

The system used to train the Mask-RCNN consisted of an RTX 3080 (12 GB) graphics card, 32 GB of DDR4 3600mhz RAM, and a Ryzen 5 5600x processor, with CUDA 11.4 and cuDNN 8.2.2 versions. In the initial version, real picking images were acquired with a RGB-D sensor and manually annotated by a human operator with the assistance of the VGG image annotator and the COCO annotator. After analyzing the procedure, a brand-new approach for creating datasets was suggested based on simulated data using Blender [37] and OpenCV [38] to minimize the time and work required for both data acquisition and label generation, as described in Section 3.1.2.

The set of parameters was defined empirically with the aim of enhancing the training process and the expected inference results. The most useful parameters to adjust are shown in Table 1; specific values were defined after analysis and testing various choices.

Mask shape, pixel mean, detection minimum confidence, and image size each have a significant impact on both detection precision and computation time, particularly for the inference stage. The most crucial phase of a deep neural network is training, which must be adjusted for the GPU memory available, dataset size, necessary average precision, and computational time.

The steps for epoch and batch size must also be adjusted in accordance with the dataset length to prevent the trained model from over-fitting or under-fitting. Mask size, learning rate, and Mask R-CNN losses boost the accuracy of the detection mask, but they can also extend the training required time, changing how the neural network learns particular patterns. Nearly all of the additional settings that can influence the training procedure are listed in the configuration file, such as maximum GT instances, which can speed up training.

Table 1. Mask R-CNN parameters.

Parameters	Values
Pixel mean	80.2
Maximum image dimension	512
Maximum GT instances	45
Number of classes	2
Scales of rpn anchors	(64, 128, 256, 512,1024)
Learning rate	0.001
Mask R-CNN mask loss	2.0
Gradient norm clipping	10.0
Backbone	Resnet-101
Batch size	2
Mask shape	[56, 56]

Training strategies are used to raise the detection average precision and intersection over union (IOU) outcomes. Several tensor-board metrics, including bounding box loss, class loss, epoch loss, and most crucially, mask loss, were examined to evaluate these results. The tensor-board results obtained in different training cases are displayed and analyzed in Section 4.2.

3.1.2. Real/Simulated Dataset Generation

We implemented two distinct approaches of providing input data to the neural network, namely, the RGB images and the respective annotation labels. The first approach was the classical dataset generation, requiring human interaction. The training data were created using images of actual picking scenarios acquired from a RGB-D sensor and labeled by a human operator using the VGG image annotator [39] and COCO annotator [40] software, requiring an average of 1 min to generate an image and 16 to 40 s to produce each instance label, depending on the type of object. To remove this systematic and tedious work, we developed an autonomous system employing Blender and OpenCV software, consisting of 3D simulated data and autonomous labeling, minimizing the average time consumed and required work for producing the final data.

The simulated training method using CAD models is an efficient approach for training deep neural networks, since it can generate a substantial amount of training data containing automatically labeled RGB images. The system can produce an average of 1600 images in 2 h and 36 min, with different features, such as view perspectives, object quantity, illumination, and object position and orientation. This system solves the problem of not having enough valuable data to train deep neural networks, while also reducing the time and work it requires.

The proposed simulated-data-generation system is divided into two stages: first, an offline phase (also referred to as configuration), executed before performing any actions; second, an online phase, executed while the algorithm is running. The offline stage is required to configure the types of objects, object quantity, view perspectives, and render configurations. For instance, the render engine used was Cycles, computed with an RTX 3080 graphics card with CUDA 11.4 installed. For each model, there was a texture configuration with similar configuration attributes, as illustrated in Figure 8, but with different parameter settings and input texture images. Beyond the realistic texture, different colors are defined based on the maximum number of items the user requires in the scene.

For instance, if the user demands a maximum of 10 objects, 10 distinct colors were defined, with fixed RGB values.

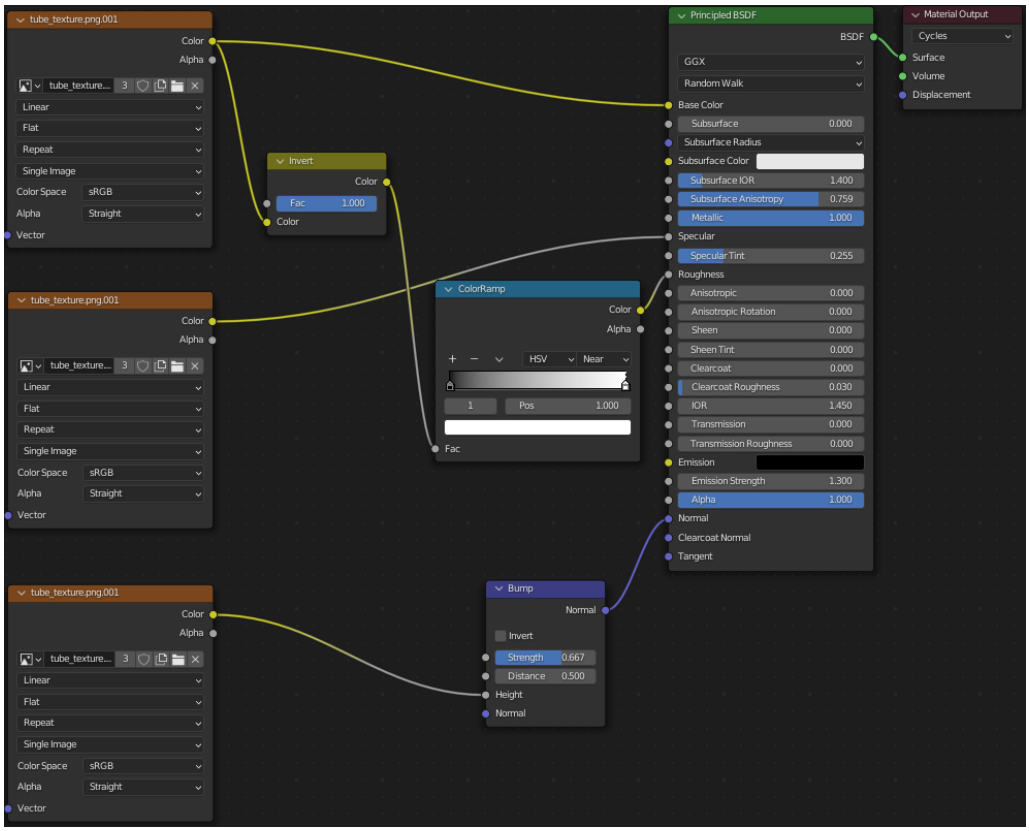

Figure 8. Object with a realistic texture.

The online stage begins by activating rigid-body physics properties, enabling gravitational and object collision forces. The generation simulation loop starts by associating each object with a predetermined color. Then, as shown in Figure 9, the objects are positioned on top of a funnel with a random pose, falling into the desired bin with various positions and orientations. Subsequently, two different scenes are rendered. The first one has the respective colors, thereby generating objects with constant color values regardless of position, orientation, and illumination system; and the second scene has the selected object's texture, with realistic settings (each loop supplies two images for each viewing perspective).

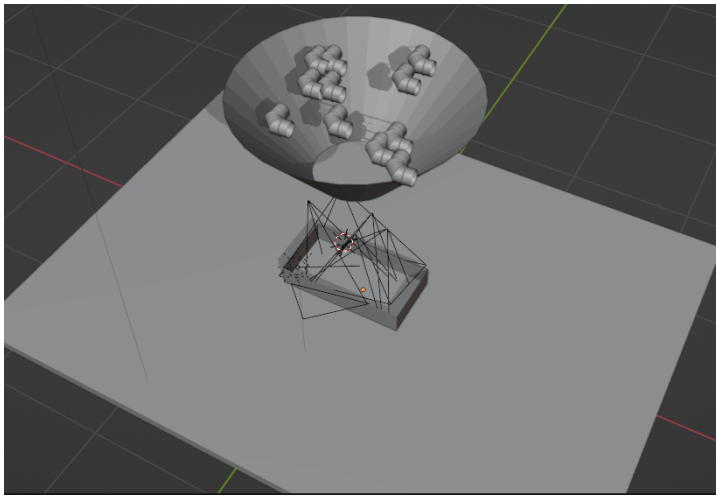

Figure 9. Blender environment.

Half of the images have colored textures, and the other half have realistic textures, as shown in Figure 10. The images created to be inputs for the labeling algorithms are on the left-hand side of the figure, and the original images created to train the deep neural network are on the right.

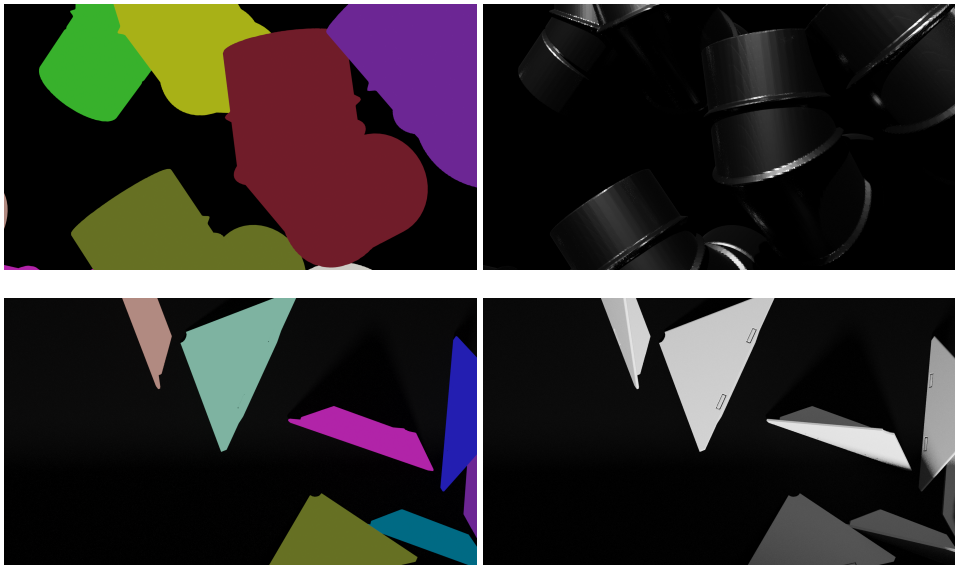

Figure 10. Synthetic dataset images: (**left**): labeling images, (**Right**) RGB images.

The labeling algorithm is split into two main stages, namely, mask generation and label generation, which are both explicitly shown in Figure 11. The objective is to generate one file in JSON format that can be easily converted to another format, such as COCO, containing instance segmentation annotations of the entire generated dataset. For instance, each image shown in Figure 10 has multiple instance segmentation annotations (in this case, five different ones for the top-row images and seven for the bottom-row images).

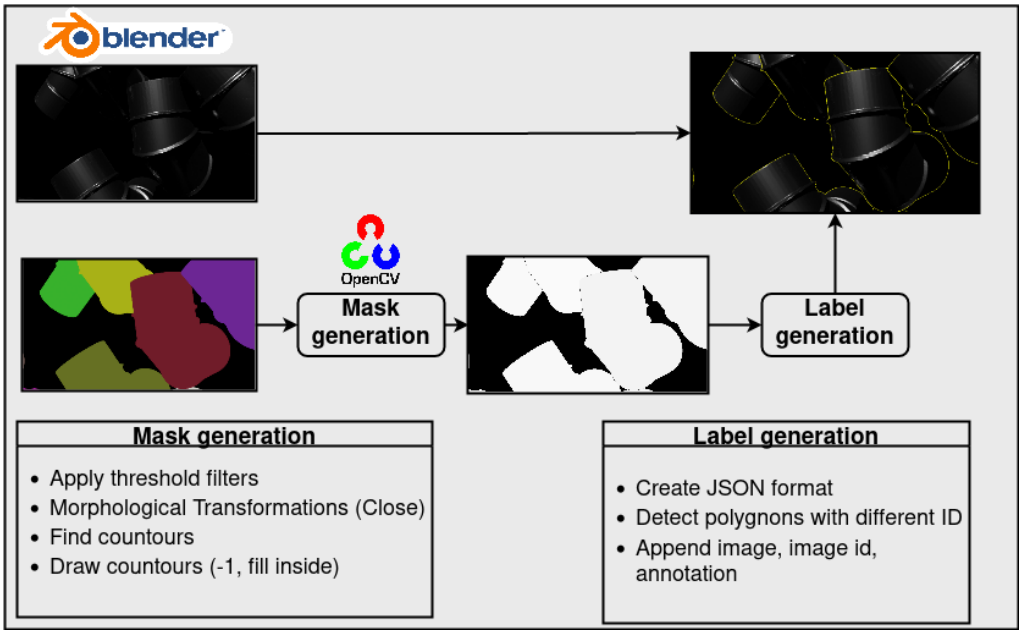

Figure 11. Simulated dataset generation overview.

The first stage, identified as mask generation, is implemented to generate a mask for each image, wherein segmented objects are identified with different IDs, hence the

requirement for images with differently colored objects. To simplify the segmentation (polygon detection), we applied specific OpenCV filters, according to the color defined in the previous Blender offline phase, thereby identifying each segment with a particular ID and obtaining the 2D pose of every object presented in the image.

After generating the masks of the dataset with success, the next stage occurs, namely, label generation, adapted from [41]. Currently, the system has several images with known segmentation data and needs to transform each segment into instance label data. To execute this task, categories are first created according to the JSON format instance segmentation labels, detailed in Table 2. Afterwards are defined the filename and the size of the original image, which will be used in the training stage. Afterwards, the image is interpreted, and we define different polygons and region attributes for each image, employing find_contours, polygon, and simplify functions to withdraw the exterior points of each segmentation.

Table 2. JSON segmentation format.

ID	Categories	Sub-Categories	Sub-Sub-Categories	Value (Examples)
image id	filename			"image_name.jpg"
	size on the disk			114,651
	regions	shape attributes	name	"polygon"
			array of "x" points	[112, 114, 117,. . .]
			array of "y" points	[559, 551, 532,. . .]
		region attributes	label	"label":"model"
	file attributes			"width":1980, "height":1080

An array of "x" points represents every point from each polygon of the *x* axis, and an array of "y" points represents every point from each polygon of the *y* axis.

Finally, the information of each image was collected into one file in JSON format, ending with the original simulated dataset generated in Blender and the annotation file. Each segment of the entire dataset was labelled. Each image has its own polygon segments labels, as displayed at the output of Figure 11.

3.2. Object Pose Estimation

The object pose estimation pipeline was developed by Costa et al. [42]. It is a modular robot localization solution that can be easily customized for object-pose-estimation applications. The configuration allows one to estimate object poses using a CAD reference model and a sensor point cloud.

To establish communication with this pipeline, the segmentation system publishes a segmented point cloud for a specific object, allowing the pose estimation system to match this segmentation with the CAD reference model, estimating the object 6-DoF position and orientation.

Figure 12 provides an overview of the main stages of the 6-DoF pose estimation system, namely:

- The first image shows the raw point cloud.
- The second image introduces the segmented object with a blue point cloud.
- The third image presents the green reference point cloud before alignment.
- The fourth image shows the initial alignment between the filtered sensor data and the green reference point cloud using fast point feature histogram (FPFH) descriptors along with random sample consensus (RANSAC) matching.
- The fifth image shows the green reference point cloud after alignment using the iterative closest point (ICP) algorithm.
- The sixth image shows the comparison of the filtered sensor data with the matched green reference point cloud. Each point in the filtered sensor point cloud was given a

color gradient between green and red based on the distance to the closest neighbor in the green reference point cloud. Green dots represent points with a very close neighbor and as such can be considered correctly matched, and red dots represent points with a far away neighbor and as such can be categorized as not having a matching point in the reference point cloud.

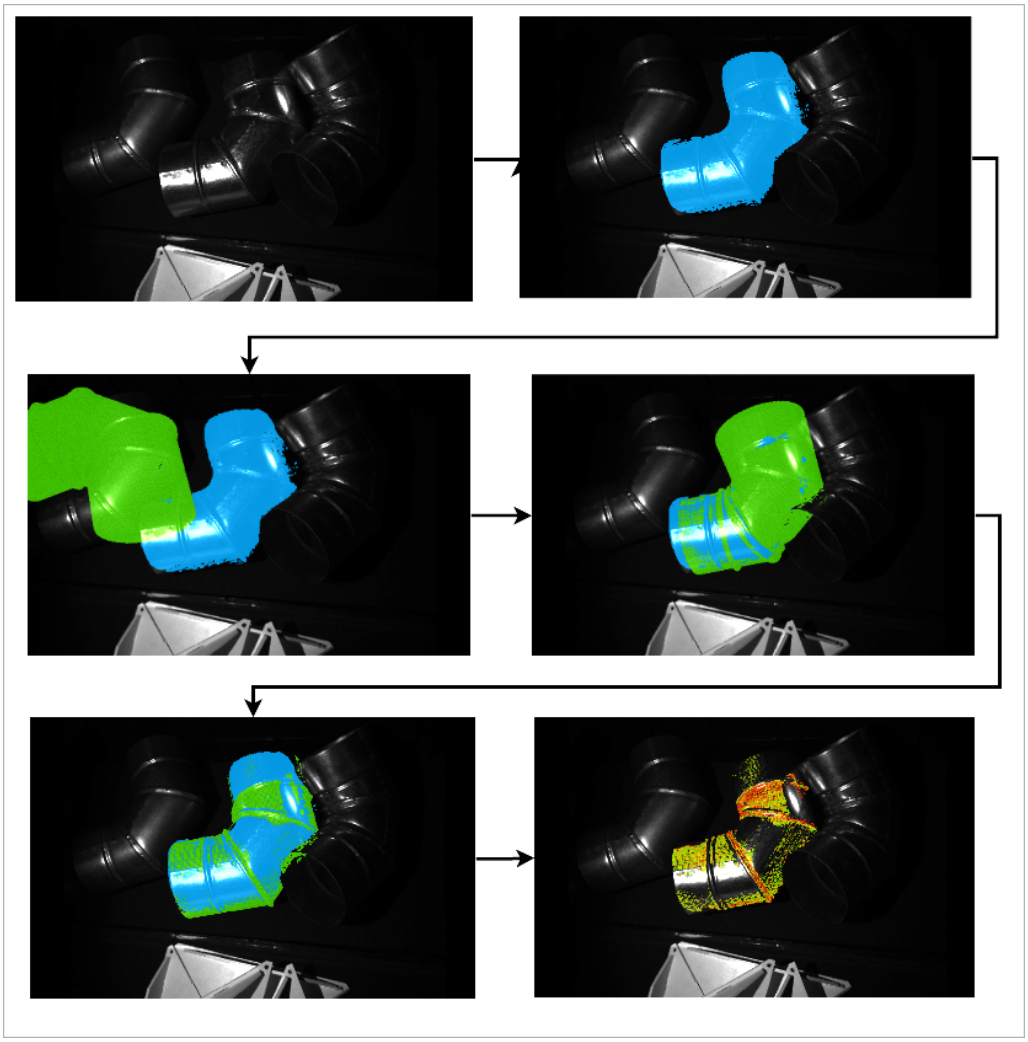

Figure 12. Alignment of the green reference point cloud with the blue segmented point cloud for 6-DoF pose estimation.

3.3. Grasping Planner

The grasping planning used in the current paper was developed by Souza et al. [29]. In this work, the authors proposed a re-configurable and modular grasping planning pipeline to be deployed by factory floor operators. This pipeline is divided into two stages of processing. The first, an offline stage, is in charge of automatically generating a set of grasping configurations oriented to the object 3D CAD model (more than 100 configurations), and the second determines which configuration is appropriate at run-time. A pipeline flow organizes user-defined decision metrics and heuristics. Figure 13 presents some examples of grasping configurations generated using grasping planning for the current use-case.

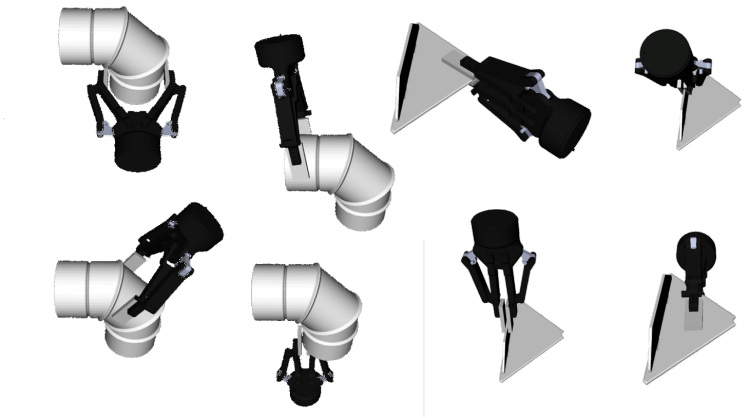

Figure 13. Some examples of generated grasping postures used in the current proposal.

The estimation of the selected grasp candidate was evaluated based on several metrics and heuristics, namely: (i) the required effort applied when moving the mobile manipulator to the estimated point, estimated by the roll, pitch, yaw, and Euclidean distance; (ii) the collision with the workspace or another object, excluding grasp candidates with collisions; and, lastly, (iii) the joint space, excluding candidates outside this space, due to the manipulator not being capable of reaching certain locations to execute the action. In other words, the grasp candidate inside the joint space, without any collisions, and presenting the lowest-effort trajectory, is going to be chosen as the best grasp pose.

If the end-effector accidentally holds another object, and the grasp occurs without any collision, the system completes the procedure and places the object in the bin of the mobile platform (this case only happens if the segmentation system chooses the wrong object). When grasping, if an object is holding or stuck to another object, the robot manipulator will try to execute the same action that was planned, because the algorithm does not have a solution for entangled objects.

4. Results and Discussion

This section discusses several bin-picking results obtained from several experiments, based on deep learning inference metrics, such as average precision, average recall, IOU, F1-score, deep learning training evaluation, data generation evaluation parameters, and real-life intralogistics experiments. This section analyses the entire bin-picking system proposed, focusing on the perception operation and closing with an analysis of realistic bin-picking experiments.

The suggested system was tested in several scenarios, with one or more objects, and at various levels of difficulty, allowing us to analyze its behaviors at various complexity levels.

4.1. Data Generation

The data were generated according to two different approaches described in Section 3.1.2. We analyze the efficiency of each approach (see Table 3) and its precision in Section 4.3.

Comparing the two approaches, it can be concluded that the acquisition based on real data and labeled by human operators requires more time than the synthetic-data approach to generate the sensor data and their labels. Considering a linear relation between elapsed time and data samples, and between elapsed time and number of annotations, the real data approach requires 900% longer to acquire images and around 500% to 1200% longer for labeling. Apart from this fact, the approach is not autonomous, requiring human operators with some experience in this field to execute the task. In light of these considerations, the dataset generation using synthetic data (which solely requires a computer, the CAD models, Blender, and OpenCV) is an effective autonomous method of producing training data for deep neural networks.

Table 3. Dataset generation time analysis.

Dataset	Nr of Images	Data Acquisition Total Time (h:m)	Nr of Annotations	Time per Annotation (s)	Labeling Total Time (h:m)
Real (Model A)	488	07:10	737	38	07:46
Real (Model B)	59	00:55	155	16	00:41
Simulation (Model A)	1600	02:36	10,593	3	08:32
Simulation (Model B)	1600	02:21	10,106	3	08:05

The label-generation process for synthetic data is not yet optimized, as it was the first iteration developed, so the time required to generate every label was not the algorithm's priority. It would be possible to reduce the labeling total time, enhancing the proposed approach's efficiency.

Figure 14 shows two different types of data generated with different approaches. In the left column are synthetic data generated with the CAD model of each object, and in the right column, real data obtained by a RGB-D sensor. As observed, the simulated data have colors, textures, forms, and sizes that are realistically close to those of the data acquired by a RGB-D sensor.

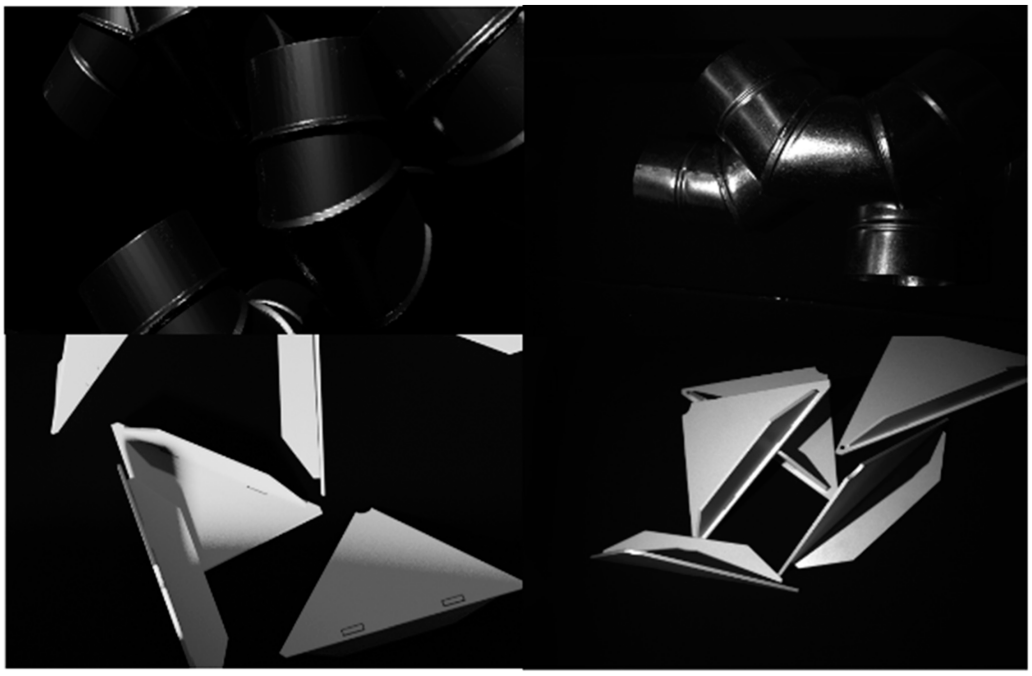

Figure 14. Difference between simulated and real data. (**Top-Left**) synthetic 90°-flow-pipe data; (**Top-Right**) real 90°-flow-pipe data; (**Bottom-Left**) synthetic triangular-wall-support data; (**Bottom-Right**) real triangular-wall-support data.

4.2. Mask R-CNN Training Results

As explained in Section 3.1.1, the Mask R-CNN was trained with specific parameters to enhance the precision and efficiency of the neural network. These were adapted according to the dataset size and complexity. For example, synthetic datasets generated with the new proposed system feature more training and validation images than real datasets, because generating these only requires time to produce more quantity instead of operator

time and effort. Due to this case, the training process with synthetic data had a lower learning rate and more steps per epoch.

The training process was slightly modified during the course of the work, including the image size, batch size, epochs, and Mask R-CNN layers, to analyze which parameter values produced the best results. The final parameters are defined in Table 1.

For simulated datasets with 1600 images, including both models, the total training time was on average 52 h. Regarding the training executed with datasets generated from real data, as it had fewer steps for epochs, it required an average of 6 h for the triangular-wall-support model and 9 h for the flow-pipe model. The applied training process was based on transfer learning with COCO weights, increasing the neural network's precision and enabling less training time.

The results were analyzed using the tensor board tool [43] to visualize metrics such as mask and bounding box loss, with a particular emphasis on the mask loss metric, which defines the neural network's 2D segmentation loss. The results were organized into categories based on the different models and dataset creation techniques and are shown in Figures 15 and 16.

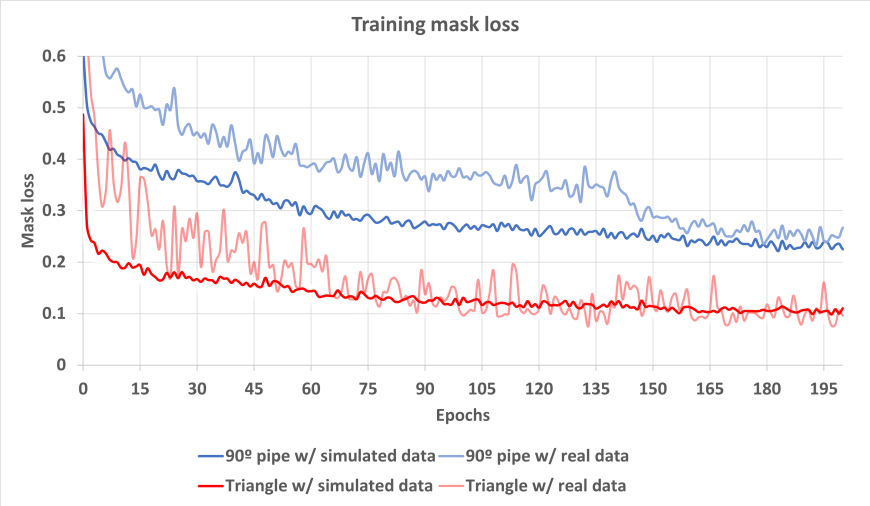

Figure 15. Training mask losses.

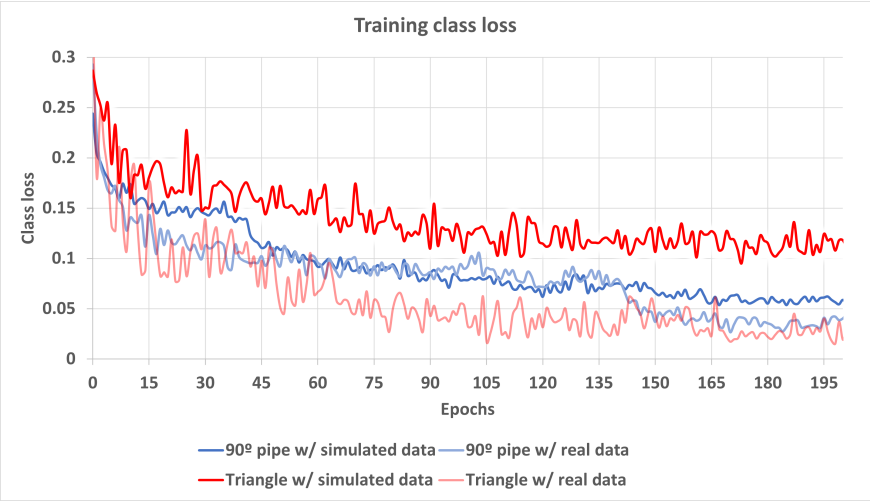

Figure 16. Training class losses.

Analyzing the displayed graphs, both dataset generation approaches have similar training statistics for both object models. On average, the mask and class loss stabilized after around 150 epochs, depending on the trained Mask R-CNN layers. In this case, three

types of layers were trained, namely, heads, 4+, and all. The mask loss varied between 10% and 28% and class loss between 5% and 12%.

The dataset created using real data for the triangular-wall-support model only contained 70 training and validation images, which led to more unstable training, as observed in Figure 15. Mask loss variation was between 5% and 15%; compared that to the 1% presented in the model trained by simulated data, illustrating how smaller datasets with fewer samples have worse training performance. In other words, the results produced by the simulated data are accurate, and if the dataset is appropriately constructed, with several viewpoints positions, and a sufficient sample size, it can outperform the dataset generation with real data.

Lastly, by analyzing all the results presented and comparing the two dataset generation approaches, it is possible to conclude that the proposed simulated data dataset generation method is an appropriate one for deep neural network training, presenting high efficiency and precision values, enabling the adaptation to various object models with minimal resource requirements. This conclusion is supported by the efficiency demonstrated in Section 4.1.

Furthermore, if we require the detection of a different type of object, the deep neural network will have to be re-trained using training from scratch or transfer learning methods. The employment of the second method requires a previous trained weight, allowing one to reduce the new training effort (time, GPU resources) required. The most common pre-trained weights used for instance segmentation are COCO and ImageNet; however, one can also use weights trained with specific datasets. Especially in transfer learning, the training has to be supervised, in order to not overfit or underfit in relation to the new dataset used. Nonetheless, when changing the object type, there is no need to change the segmentation pipeline due to its architecture, enabling the 2D and 3D segmentation of any type of object.

4.3. Proposed Perception Pipeline Results

To develop a thorough evaluation, specific evaluation datasets were created, with several images and various scenarios, from simple ones with only one item to complex ones with multiple objects and occlusions. Every trained model evaluation was assessed using the same datasets according to the type of object model (90° flow pipe and triangular wall support, as shown in Figure 17).

The datasets used for evaluation were acquired with real data from the RGB-D sensor and labeled by human operators. The 90°-flow-pipe dataset consists of 64 images,with different object quantities, view angles, and complexity; it has uncluttered scenarios with no or only a few occlusions, and cluttered scenarios with several occlusions. The triangular-wall-support dataset is similar, being composed of 59 images, only differing in the object model. Furthermore, instance segmentation labels were generated for both datasets, working as ground-truth data for the evaluation.

The evaluation system applied to the perception framework is based on dice coefficients, exhibited in Tables 4 and 5. Average precision (AP), established according to a specific IOU interval (for instance, we used AP_{50} (50% IOU), AP_{75} (75% IOU), AP_{90} (90% IOU), and mAP (from 50% to 95%)), was estimated by computing the division between true positives and the sum of true positives and false positives. Average recall (AR) is also based on IOU intervals and is estimated by dividing the true positives by the sum of true positives and false negatives. However, we only calculated the average recall between 50% and 95% IOU. Average IOU (mIOU) quantifies the overlap percentage between the ground truth mask and the prediction mask. Finally, the F1-score (mF1) is the weighted average of precision and recall, giving equal weight to precision and recall.

The results are divided into two tables, providing a deep analysis of the difference between training the Mask R-CNN deep neural network with real instance segmentation data and simulated instance segmentation data.

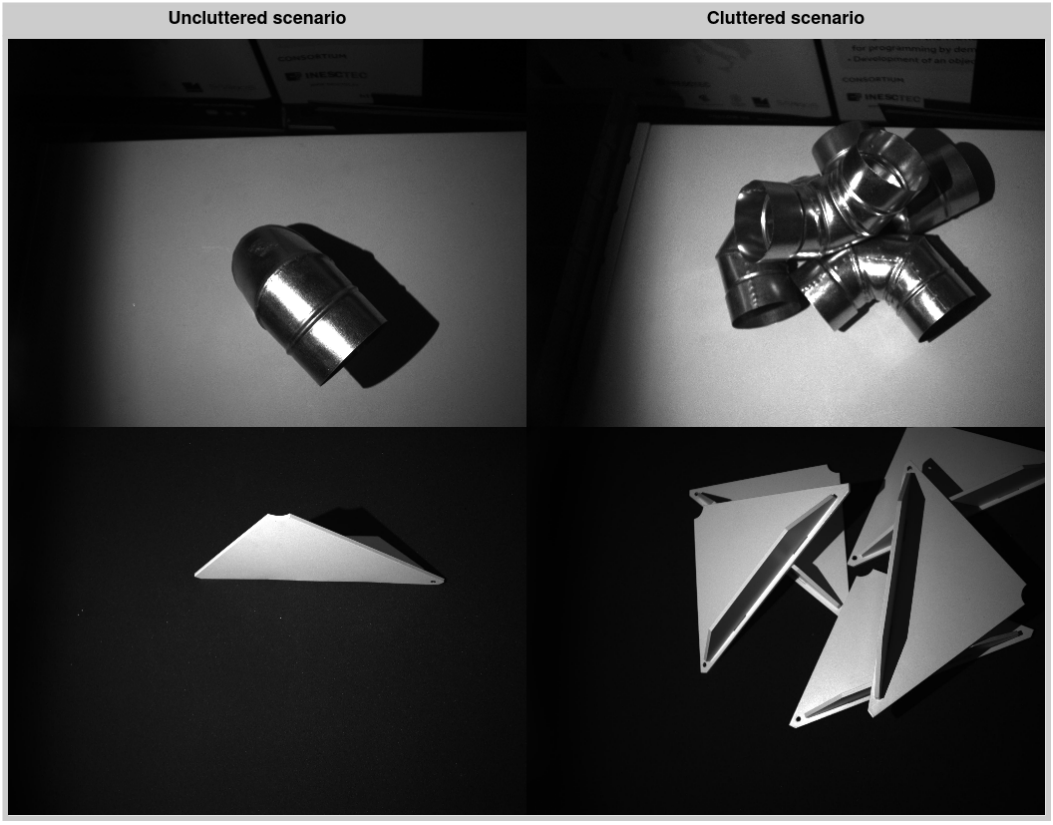

Figure 17. Perception evaluation dataset. (**Top**) 90° flow pipe; (**Bottom**) triangular wall support.

Table 4. Perception results (Mask R-CNN trained with real data).

Dataset Model	AP_{50}	AP_{75}	AP_{90}	mAP	mAR	mIOU	mF1
90° flow pipe	0.976	0.968	0.897	0.872	0.976	0.916	0.943
Triangular wall support	0.980	0.955	0.902	0.914	0.980	0.937	0.849

Table 5. Perception results (Mask R-CNN trained with simulated data).

Dataset Model	AP_{50}	AP_{75}	AP_{90}	mAP	mAR	mIOU	mF1
90° flow pipe	0.967	0.962	0.579	0.819	0.976	0.736	0.875
Triangular wall support	0.919	0.842	0.721	0.802	0.948	0.894	0.793

After analyzing Tables 4 and 5, it is possible to conclude that both dataset generation techniques for the two models exhibit similar results in almost every category. A difference in the average precision is noticeable though, especially for higher IOU values.

Overall, the object perception results based on this evaluation, according to the defined metrics, demonstrate that for every trained model, the pipeline is capable of estimating results with high precision, recall, and IOU in distinct environments, with several occlusions. Precise instance segments were defined relative to the ground truth previously defined.

Figure 18 displays several segmentation results acquired from bin-picking experiments with the two referred models (90° flow pipe and triangular wall support) containing from 1 to 6 objects. The segmented point cloud estimated by the object segmentation system is represented by the blue region. As observed, the segmentation system only selects the object most appropriate to be grasped by the robot manipulator. Considering these results, every segment was correctly estimated, though with a small mask error in some of the triangular-wall-support model segmentation.

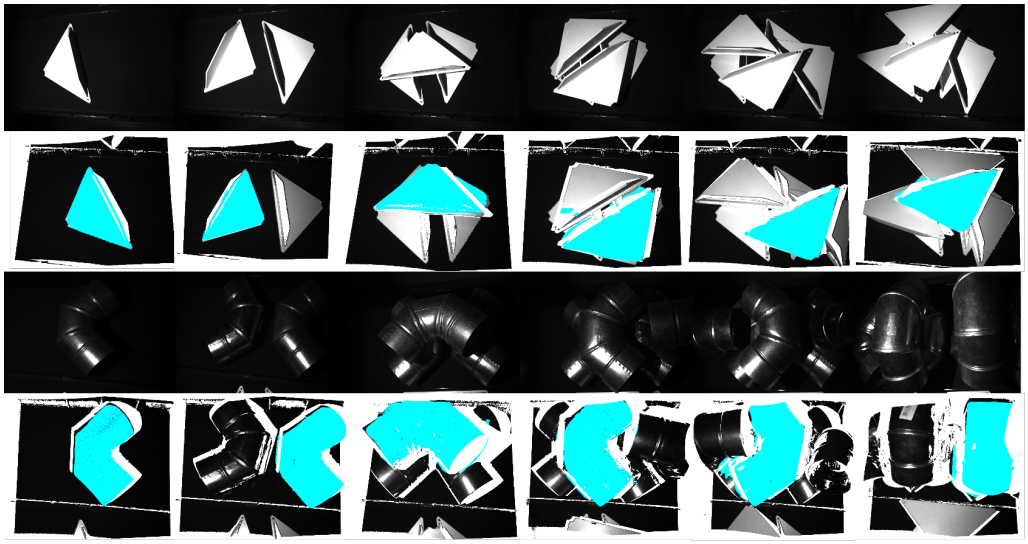

Figure 18. First scenario perception results. (**1st and 3rd row**) Original image; (**2nd and 4th row)** segmentation result.

Table 6 displays the mean and standard deviation of the time consumed in the proposed perception pipeline. The presented results were computed after 20 iterations for each model type. The inference was executed on a computer composed of an Intel i7 12700H processor, NVIDIA RTX 3060 graphics card, and 16 GB of RAM.

Table 6. Proposed object segmentation pipeline with elapsed time.

Dataset Model	Pre-Process (Offline) [s]		Loading (Offline) [s]		2D Segmentation (Online) [s]		3D Segmentation (Online) [s]		Overall Object Segmentation [s]
	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean
Knee tube	8.5	0.6	0.25	0.2	0.33	0.4	0.11	0.04	0.47
Triangular wall support	8.5	0.6	0.24	0.2	0.28	0.3	0.09	0.02	0.38

Mean represents the mean value and Std dev represents the standard deviation value.

The elapsed time was divided into several stages, and each was evaluated separately—namely, the pre-process, loading, and segmentation (main processing stage). The main stage is split into two steps, one for 2D image segmentation (estimated by the deep neural network) and one for 3D segmentation.

Pre-processing has little impact on the pipeline duration because it is only performed once when the robot is initialized. The 2D segmentation acquired as an output of the Mask R-CNN neural network, which takes an average of 0.3 s to compute, is the longest process, excluding the pre-process, according to an analysis of Table 6. This computational time can be decreased by changing certain parameters, such as the input image quality. Loading the trained model required, on average, 0.25 s; however, that is only when the type of object that is being detected changes.

At last, the overall object segmentation, shown in Table 6, is the most crucial result, exhibiting the average time consumed when the perception framework is called, composed of the time consumed by the Mask R-CNN detection, 3D segmentation, and communication for receiving and sensing data between the robot operating system (ROS) nodes.

4.4. Intralogistics Experiments

In order to evaluate the complete bin-picking system, covering the segmentation, pose estimation, and grasping system, we analyzed its behavior in scenarios with different difficulty levels—real laboratory scenarios that were created by randomly placing objects in the bin and only manipulating them to be at least partially inside the 3D sensor field of view. Each scenario contained only a singular type of object—for this use-case, a 90° flow-pipe model or a triangular-wall-support model. The evaluation of these experiments focused on the number of successes per iteration, to determine the system's consistency and robustness. Furthermore, the proposed system was compared with another implementation based on a plane and cluster segmentation, essentially only differentiating the segmentation pipeline.

The results were evaluated according to the metrics described by Souza et al. [44]. In summary, the evaluation parameters used, namely, grasp prediction (GPSR), grasping reaching (GRSR), grasping hold (GHSR) and handling grasping success rates (HGSR) allow the verification of different grasping procedure steps in a standard fashion. If a stage fails, the succeeding stages results are not affected, enabling one to estimate what stages contain more failures. The most significant results for this proposed work are demonstrated in the grasping prediction (GP) stage, associating the segmentation and pose estimation system.

The system was first tested according to the number of objects presented in the scene (from one to six objects), limiting the object quantity due to the dimensions of the bin. Based on this configuration, the results listed in Table 7 were obtained, and for each iteration, a different scenario with the same object quantity was constructed. In order to achieve an acceptable evaluation, the system was tested at least 20 times for each case (number of objects).

Table 7. Intralogistics experiments based on object number. Each category evaluated the grasping performance from grasp prediction (GPSR), grasping reaching (GRSR), grasping hold (GHSR), and handling grasping success rates (HGSR).

Parameters	1 Object	2 Objects	3 Objects	4 Objects	5 Objects	6 Objects
Iterations	20	20	20	20	20	20
GPSR	95%	85%	80%	95%	85%	85%
GRSR	100%	100%	100%	100%	100%	100%
GHSR	100%	100%	100%	95%	100%	100%
HGSR	100%	100%	100%	100%	100%	100%

Based on Table 7, the proposed system has similar results in scenarios composed of 1 or 6 objects, confirming its robustness. Failures mainly occurred at the GP stage, mainly due to matching miscalculations between the segmented point cloud and the CAD model. In general, the system achieved results with high grasping success rates, especially for cluttered scenarios with four or more objects.

The proposed system was once more evaluated, based on a realistic bin-picking problem. This new experiment, whose results are detailed in Tables 8 and 9, was created with the intention of emulating a standard bin-picking problem, starting with a defined number of objects in random positions and orientations, forming a cluttered scenario, as displayed in Figure 19. The objective was to grasp (remove) every object from the bin, evaluating the number of necessary iterations to empty each bin. If the system failed to pick and place an object, it would try again until emptying the bin, because the neural network's estimated results varied in each iteration, unlike most of the classical implementations.

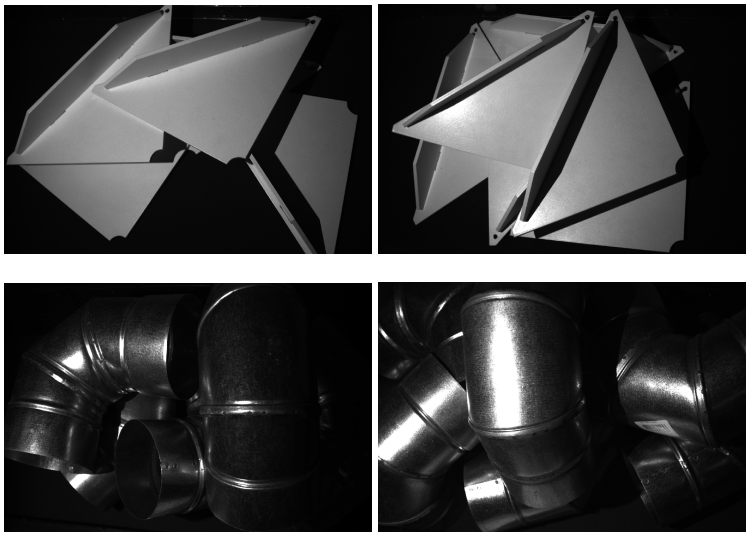

Figure 19. Bin-picking scenarios. **(Left)** First bin-picking scenario; **(Right)** second bin-picking scenario.

The first iteration of the first bin picking experiment is displayed in Figure 19 (left). The initial object quantity was five; this quantity decreased with each iteration realized, representing a realistic bin-picking problem. The second experiment was similar to the first, simply changing the initial quantity from 5 to 6 and the overall scenery.

Table 8. First bin-picking evaluation.

Parameters	90° Flow Pipe Model	Triangle Wall Support Model
Object quantity	5	5
GPSR	100%	80%
GRST	100%	100%
GHSR	100%	100%
HGSR	100%	100%

Table 9. Second bin-picking evaluation.

Parameters	90° Flow Pipe Model	Triangle Wall Support Model
Object quantity	6	6
GPSR	100%	83.3%
GRST	100%	100%
GHSR	100%	100%
HGSR	100%	100%

In order to analyze in depth the crucial phase referred as GP, Table 10 demonstrates the same experiments realized in the previous evaluations from Tables 7–9. However, now the evaluation is focused on the point-cloud segmentation, object selection, matching with the CAD model, and pose estimation stages—identifying which of them had the higher success rate. As each process is executed in order, the success rate is only dependent of each specific stage.

Table 10. GPSR evaluation.

Parameters	90° Flow Pipe Model	Triangle Wall Support Model
Iterations	71	71
Segmentation success	100%	99.98%
Selection success	99.98%	95.71%
Matching success	100%	82.00%
Pose estimation success	100%	100%

Taking into consideration Table 10, for every experiment realized in this paper, the majority of the failures were caused by the matching process, generally when applied to the triangular-wall-support model. Several of the matching failures originated because the smaller surface of the triangle was directed upwards, causing the RGB-D sensor to be perpendicular to this smaller surface, thereby only segmenting the top view, as exhibited by the straight line in blue in Figure 20 (Left). As a result, in some cases, the matching algorithm estimated the object in the opposite direction, as shown in Figure 20 (right), where the CAD model (red) is in the opposite direction of the segmented cloud (blue). The same happened with the first experiment, exhibited in Table 7, where most of the total failures (87.5%—portion of matching failures of total failures) were caused by this error. However, this error did not appear because of the segmentation, but rather because of the perspective of the robot manipulator. The solution for this problem is to acquire data from different perspectives, acquiring point clouds with more information about the object being scanned.

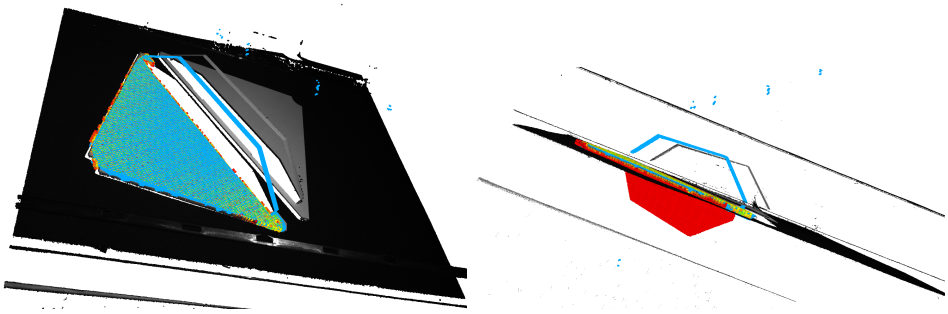

Figure 20. Proposed pipeline matching error. (**Left**) Point-cloud segmentation; (**Right**) Point-cloud matching

As observed, the system had an average success rate of 91.6% in bin-picking problems with complex and cluttered scenarios. Regardless of the object quantity, the proposed system achieved an average success rate of 87.5%, proving to be capable of segmenting and picking objects in densely cluttered environments, similar to those illustrated in Figure 19, composed of 5 and 6 objects. To conclude the previous discussion, the main factor influencing the system's success rate is the matching process between the segmentation pipeline and grasping pipeline, where the estimated point cloud segment is matched with the reference point cloud model to estimate the position and orientation of the object, presenting a higher grasping failure rate when executing the system with the triangular-wall-support model.

The segmentation of the point cloud estimated by the proposed pipeline is compared to a system that was previously developed based on a different perception technique, adopted in [29]. Several classical techniques, including plane segmentation, hue saturation value (HSV) segmentation, and cluster segmentation, were used in the segmentation pipeline of the prior methodology. As shown in Figure 21, the segmentation estimated from the previous method had issues when the objects were close to one another on the same plane field. This problem frequently occurred in cluttered scenarios, where the inability to separate the various objects resulted in a single cluster (represented by green dots) in the

same plane field, which provides misleading information about the object segment for the matching stage, making it impossible to match with the CAD model.

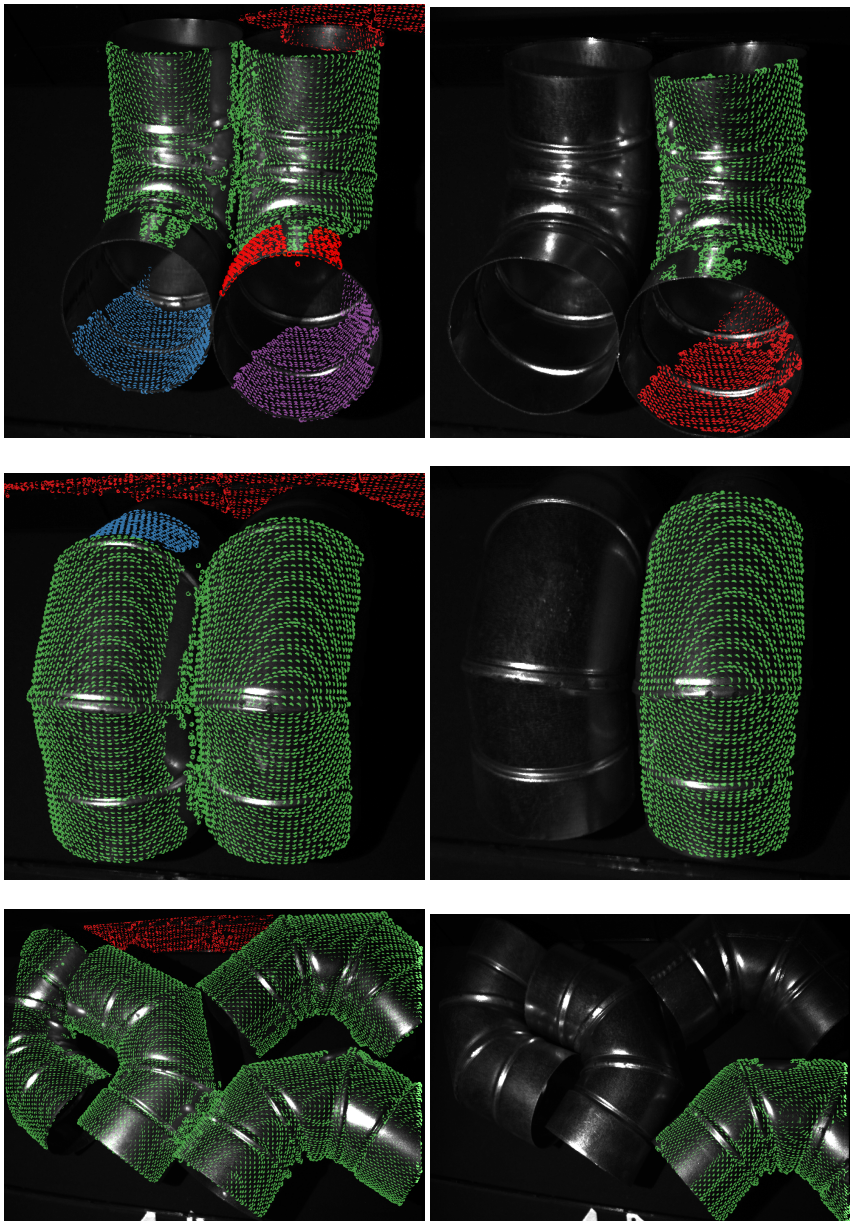

Figure 21. Results comparison between pipelines, for selecting a point cloud from a single object (shown with green dots). (**Left**) Previous pipeline [29]; (**Right**) proposed pipeline.

By comparing the results generated by the previous pipeline and the proposed pipeline in cluttered scenarios, displayed in Figure 21, it is possible to conclude that the proposed approach has more accuracy in these types of scenarios. As observed, the instance-segmentation technique produces similar results related to the mask. However, the new proposed pipeline can distinguish each singular object when they are close together, minimizing the error of estimating a segmentation with more than one object. In addition, the triangular-wall-support model has several segmentation difficulties because of its geometry, color, and material. Several experiments were realized with the previous pipeline, and it was noticed that with four objects in densely cluttered scenarios (composed of 90° flow-pipe model objects), several segmentation difficulties arose, and it achieved the worst results for the triangular-wall-support model, having difficulties with only one object in

the scene, due to the height difference between the background and the model being too small, showing it was merely accurately segmenting when there was a significant plane or cluster difference.

Concluding, the proposed method presents a higher grasping success rate than the previously developed method, based on classical techniques. However, it still presents a few limitations in random bin-picking scenarios. Its major limitations are related to the geometry and other features of the object to be picked—if the object does not present a significant height difference and is relatively uniform, this limits the matching process, and the corresponding success rate. As the defined threshold to accept or decline a match is applied to diverse objects, a few matches are estimated in the wrong orientation (most of the cases with a 180-degree offset) or slightly displaced from the original position. Nonetheless, these limitations do not cause a major problem, as concluded by the results obtained from the triangular-wall-support model—a textureless object with a hard geometric shape.

5. Conclusions and Future Work

A segmentation pipeline approach was proposed and implemented in a re-configurable bin-picking system based on object instance segmentation using deep learning techniques. This paper solves several pipeline difficulties in bin-picking problems. Our method segments/detects different types of objects (different shapes and sizes) in cluttered environments, showing the potential of deep learning techniques. In particular, we developed a method based on 2D and 3D segmentation that is capable of detecting the most appropriate object to pick up in a cluttered environment with several occlusions. We also proposed a new completely autonomous system to generate labeled data quickly for deep neural network training; we demonstrated that simulated data can be used to generate accurate deep learning models and avoid the time-consuming manual data acquisition and labeling by human operators. The experimental results show that the proposed segmentation pipeline can segment objects with an average precision of 89.3% when trained with real data and 81.0% when trained with simulated data, solving several difficulties presented in the previously developed system. Finally, the re-configurable pipeline displayed an average grasping success rate of 87.5%.

Furthermore, this work provided a quality comparison between deep learning training with real data acquired by a RGB-D sensor and labeled by human operators, and synthetic data labeled by an autonomous system using Blender and OpenCV. We also compared the new segmentation approach with a classical segmentation method.

Future work includes the improvement of the re-configurable pipeline with keypoints detection, allowing the segmentation pipeline to estimate the initial position and orientation of the object, reducing the iterative closest point (ICP) algorithm's number of iterations. It is also possible to avoid the usage of the pose estimation pipeline by employing a deep neural network capable of generating grasp solutions with 3D point clouds, having similar efficiency to deep learning frameworks used to plan 2D grasping solutions; however, we must work on its precision in cluttered environments with collisions and occlusions.

Author Contributions: Conceptualization, A.C., L.F.R. and M.F.S.; methodology, A.C., L.F.R. and M.F.S.; software, A.C., J.P.S. and C.M.C.; validation, A.C., J.P.S. and L.F.R.; formal analysis, A.C. and J.P.S.; investigation, A.C., J.P.S. and C.M.C.; resources, L.F.R.; data curation, A.C., J.P.S., C.M.C. and L.F.R.; writing—original draft preparation, A.C.; writing—review and editing, A.C. and J.P.S.; visualization, A.C. and V.F.; supervision, M.F.S., L.F.R. and V.F.; project administration, L.F.R.; funding acquisition, L.F.R. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 101006798.

Data Availability Statement: The datasets presented in this study can be obtained through email (A.C.).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Boysen, N.; de Koster, R.; Weidinger, F. Warehousing in the e-commerce era: A survey. *Eur. J. Oper. Res.* **2019**, *277*, 396–411. [CrossRef]
2. Ding, J.; Ni, C. Gird Based Line Segment Detector and Application: Vision System for Autonomous Ship Small Assembly Line. *J. Mar. Sci. Eng.* **2021**, *9*, 313. [CrossRef]
3. Ferreira, L.A.; Figueira, Y.L.; Iglesias, I.F.; Álvarez Souto, M. Offline CAD-based Robot Programming and Welding Parametrization of a Flexible and Adaptive Robotic Cell Using Enriched CAD/CAM System for Shipbuilding. *Procedia Manuf.* **2017**, *11*, 215–223. [CrossRef]
4. Liu, J.; Jiao, T.; Li, S.; Wu, Z.; Chen, Y.F. Automatic seam detection of welding robots using deep learning. *Autom. Constr.* **2022**, *143*, 104582. [CrossRef]
5. Kershaw, J.; Yu, R.; Zhang, Y.; Wang, P. Hybrid machine learning-enabled adaptive welding speed control. *J. Manuf. Process.* **2021**, *71*, 374–383. [CrossRef]
6. Gao, Y.; Ping, C.; Wang, L.; Wang, B. A Simplification Method for Point Cloud of T-Profile Steel Plate for Shipbuilding. *Algorithms* **2021**, *14*, 202. [CrossRef]
7. Wada, K.; Murooka, M.; Okada, K.; Inaba, M. 3D Object Segmentation for Shelf Bin Picking by Humanoid with Deep Learning and Occupancy Voxel Grid Map. In Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, 15–17 November 2016. . [CrossRef]
8. Blank, A.; Hiller, M.; Zhang, S.; Leser, A.; Metzner, M.; Lieret, M.; Thielecke, J.; Franke, J. 6DoF Pose-Estimation Pipeline for Texture-less Industrial Components in Bin Picking Applications. In Proceedings of the 2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic, 4–6 September 2019; pp. 1–7. [CrossRef]
9. Le, T.T.; Lin, C.Y. Bin-Picking for Planar Objects Based on a Deep Learning Network: A Case Study of USB Packs. *Sensors* **2019**, *19*, 3602. [CrossRef]
10. Zhuang, C.; Wang, Z.; Zhao, H.; Ding, H. Semantic part segmentation method based 3D object pose estimation with RGB-D images for bin-picking. *Robot. Comput.-Integr. Manuf.* **2021**, *68*, 102086. [CrossRef]
11. Höfer, T.; Shamsafar, F.; Benbarka, N.; Zell, A. Object detection and Autoencoder-based 6D pose estimation for highly cluttered Bin Picking. In Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 19–22 September 2021. [CrossRef]
12. Dong, Z.; Liu, S.; Zhou, T.; Cheng, H.; Zeng, L.; Yu, X.; Liu, H. PPR-Net:Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 1773–1780. [CrossRef]
13. Xu, Y.; Arai, S.; Liu, D.; Lin, F.; Kosuge, K. FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking. *Neurocomputing* **2022**, *494*, 255–268. [CrossRef]
14. Buchholz, D.; Futterlieb, M.; Winkelbach, S.; Wahl, F.M. Efficient bin-picking and grasp planning based on depth data. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 3245–3250. [CrossRef]
15. He, T.; Aslam, S.; Tong, Z.; Seo, J. Scooping Manipulation Via Motion Control With a Two-Fingered Gripper and Its Application to Bin Picking. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6394–6401. [CrossRef]
16. Ichnowski, J.; Avigal, Y.; Liu, Y.; Goldberg, K. GOMP-FIT: Grasp-Optimized Motion Planning for Fast Inertial Transport. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022. [CrossRef]
17. Leão, G.; Costa, C.M.; Sousa, A.; Veiga, G. Detecting and Solving Tube Entanglement in Bin Picking Operations. *Appl. Sci.* **2020**, *10*, 2264. [CrossRef]
18. Iriondo, A.; Lazkano, E.; Ansuategi, A. Affordance-Based Grasping Point Detection Using Graph Convolutional Networks for Industrial Bin-Picking Applications. *Sensors* **2021**, *21*, 816. [CrossRef]
19. Jiang, P.; Ishihara, Y.; Sugiyama, N.; Oaki, J.; Tokura, S.; Sugahara, A.; Ogawa, A. Depth Image–Based Deep Learning of Grasp Planning for Textureless Planar-Faced Objects in Vision-Guided Robotic Bin-Picking. *Sensors* **2020**, *20*, 706. [CrossRef]
20. Tang, B.; Corsaro, M.; Konidaris, G.D.; Nikolaidis, S.; Tellex, S. Learning Collaborative Pushing and Grasping Policies in Dense Clutter. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–June 2021; pp. 6177–6184.
21. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. *arXiv* **2017**. [CrossRef]
22. Kumra, S.; Joshi, S.; Sahin, F. Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2021. [CrossRef]
23. Breyer, M.; Chung, J.J.; Ott, L.; Siegwart, R.; Nieto, J. Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter. *arXiv* **2021**. [CrossRef]
24. Asif, U.; Tang, J.; Harrer, S. GraspNet: An Efficient Convolutional Neural Network for Real-time Grasp Detection for Low-powered Devices. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, Stockholm, Sweden, 13–19 July 2018; pp. 4875–4882. [CrossRef]

25. Pinto, L.; Gupta, A. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016. [CrossRef]
26. Shao, Q.; Hu, J.; Wang, W.; Fang, Y.; Liu, W.; Qi, J.; Ma, J. Suction Grasp Region Prediction using Self-supervised Learning for Object Picking in Dense Clutter. In Proceedings of the 2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR), Singapore, 3–5 May 2019. [CrossRef]
27. Jiang, P.; Oaki, J.; Ishihara, Y.; Ooga, J.; Han, H.; Sugahara, A.; Tokura, S.; Eto, H.; Komoda, K.; Ogawa, A. Learning suction graspability considering grasp quality and robot reachability for bin-picking. *Front. Neurorobot.* **2021**, *16*, 806898. [CrossRef]
28. Kozák, V.; Sushkov, R.; Kulich, M.; Přeučil, L. Data-Driven Object Pose Estimation in a Practical Bin-Picking Application. *Sensors* **2021**, *21*, 6093. [CrossRef]
29. Carvalho de Souza, J.P.; Costa, C.M.; Rocha, L.F.; Arrais, R.; Moreira, A.P.; Pires, E.S.; Boaventura-Cunha, J. Reconfigurable Grasp Planning Pipeline with Grasp Synthesis and Selection Applied to Picking Operations in Aerospace Factories. *Robot. Comput.-Integr. Manuf.* **2021**, *67*, 102032. [CrossRef]
30. Cordeiro, A.; Rocha, L.F.; Costa, C.; Silva, M.F. Object Segmentation for Bin Picking Using Deep Learning. In *Proceedings of the ROBOT2022: Fifth Iberian Robotics Conference*; Tardioli, D., Matellán, V., Heredia, G., Silva, M.F., Marques, L., Eds.; Springer International Publishing: Cham, Switzerland, 2023; pp. 53–66.
31. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017. [CrossRef]
32. Huang, Z.; Huang, L.; Gong, Y.; Huang, C.; Wang, X. Mask Scoring R-CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019. . [CrossRef]
33. Cordeiro, A.; Rocha, L.F.; Costa, C.; Costa, P.; Silva, M.F. Bin Picking Approaches Based on Deep Learning Techniques: A State-of-the-Art Survey. In Proceedings of the 2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Santa Maria da Feira, Portugal, 29–30 April 2022; pp. 110–117. [CrossRef]
34. Sunwoo, H.; Choi, W.; Na, S.; Kim, C.; Heo, S. Comparison of the Performance of Artificial Intelligence Models Depending on the Labelled Image by Different User Levels. *Appl. Sci.* **2022**, *12*, 3136. [CrossRef]
35. Abdulla, W. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. 2017. Available online: https://github.com/matterport/Mask_RCNN (accessed on 10 October 2022).
36. Ciaparrone, G.; Bardozzo, F.; Priscoli, M.D.; Londoño Kallewaard, J.; Zuluaga, M.R.; Tagliaferri, R. A comparative analysis of multi-backbone Mask R-CNN for surgical tools detection. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
37. Community, B.O. *Blender—A 3D Modelling and Rendering Package*; Blender Foundation, Stichting Blender Foundation: Amsterdam, The Netherlands, 2018.
38. Bradski, G. The OpenCV Library. *Dr. Dobb's J. Softw. Tools* **2000**, *11*, 120–123.
39. Dutta, A.; Gupta, A.; Zissermann, A. VGG Image Annotator (VIA). Version: 2. 2016. Available online: http://www.robots.ox.ac.uk/~vgg/software/via/ (accessed on 19 September 2022).
40. Brooks, J. COCO Annotator. 2019. Available online: https://github.com/jsbroks/coco-annotator/ (accessed on 13 October 2022).
41. Kelly, A. Create Coco Annotations From Scratch. 2020. Available online: https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch (accessed on 13 October 2022).
42. Costa, C.M.; Sobreira, H.M.; Sousa, A.J.; Veiga, G.M. Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms. *Robot. Auton. Syst.* **2016**, *76*, 113–140. [CrossRef]
43. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: tensorflow.org (accessed on 11 October 2022).
44. de Souza, J.P.C.; Rocha, L.F.; Oliveira, P.M.; Moreira, A.P.; Boaventura-Cunha, J. Robotic grasping: From wrench space heuristics to deep learning policies. *Robot. Comput.-Integr. Manuf.* **2021**, *71*, 102176. [CrossRef]