Occlusion-aware part recognition for robot manipulation

Ziyad Tareq Nouri

Master’s thesis

University of Eastern Finland
School of Computing
Computer Science
September 2020
This thesis studies occlusion-aware part recognition for robot manipulation. The task of this thesis is to simulate robotic picking from a pile with heavy occlusion. The picking task requires at minimum three subsystems together, one for detecting and identifying objects, one for attaching to objects, and one for manipulating objects. In this thesis, we focus on identifying the objects i.e. object recognition, and manipulating the objects, i.e. motion planning. The object recognition has been studied widely. Still, in computer vision, occlusion is a common and difficult task. The applications of occlusion-aware recognition models are various: it could be used, for example, in autonomous cars, as well as in the warehouses. Because of that, it has economical potential and it has lately been studied a lot.

Our occlusion handling solution is inspired by the human way of solving the occlusion problem by using his/her previous knowledge of the full shape of the occluded object. In this work, the occlusion is solved by the state-of-the-art instance segmentation model Mask R-CNN. We train the Mask R-CNN by full shape annotation dataset of the objects in the scene to simulate the humans’ previous knowledge for the object shape. Usually, the Mask R-CNN is trained on the visible part annotation dataset. We developed a Data Generation Tool to generate synthesized images and do the full shape annotation dataset, and an Accuracy Tool to test the Mask R-CNN, and a Dataset Statistical Tool to analyze the quality and the features of the dataset. We developed a software application, the decision-maker, that gives to the robot the orders, what to do handle the occlusion. Besides grasping the objects, the robot can also sweep the scene, in case that the object is not seen on the surface of the pile. We create a tool to generate a depth map dataset for testing the system and to simulate a depth camera. We also create a tool to check the accuracy of our system.

The advantage of training the Mask R-CNN with the full shape annotation dataset is that in this way, it can predict also the occluded parts of the object. When we know the occluded parts of the object, we can know which objects are occluded or occluding other objects by just finding an intersection between the masks. By implementing the depth map to this scenario, we can easily recognize which one from the objects is on the top of the pile and can be picked up safely.

We show that the full shape annotation dataset can be used to train the Mask R-CNN to predict the full shape mask of an object. Our segmentation accuracy for the full shape masks reached 88.1 % which is only slightly less than the visible mask segmentation accuracy 89.4 % that we got with the same dataset. That means that training with the full shape annotation dataset does not really worsen the result of Mask R-CNN.

We show that the decision-maker can make a grasping order to reach the target in case that it is occluded by other objects. The decision-maker application uses the results of Mask R-CNN in making decisions and its result depends on the Mask R-CNN result. The decision-maker finds the target in 66.2 % of the cases using Mask R-CNN. But if the target is not found by Mask R-CNN, the decision-maker can choose to sweep the scene and make a new prediction. The decision-maker finds a correct object to pick up in 68.3 % of the cases. Better segmentation results would improve its performance.

Keywords: Occlusion, Part Recognition, Robotic Picking, Object Manipulation, Image Synthesis
Foreword

This thesis was done at the School of Computing, University of Eastern Finland and the Department of Computer Science and Engineering, Toyohashi University of Technology, during the study year 2019-2020.

I wish to thank my supervisor, professor Jun Miura from the Toyohashi University of Technology, who was always ready to help in anything. It was easy to communicate with him, he truly stood by my side. It really helped me to succeed in this thesis. I also thank my supervisor from the University of Eastern Finland, doctor Ville Hautamäki, for the guidance, and professor Xiao-Zhi Gao for his support.

I want to extend my gratitude to Honda R&D for the trainee period, and the useful discussion throughout my work. My special thanks go to doctor Yasuhiro Taniguchi, together with Keiichi Sawada and Atsuki Osanai.

I also thank my parents for always pushing me towards science and my friends who helped me to develop my writing skills.
**List of abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>Amazon Robotics Challenge</td>
</tr>
<tr>
<td>ILSVRC</td>
<td>ImageNet Large Scale Visual Recognition Challenge</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>STD</td>
<td>Short-term decision</td>
</tr>
<tr>
<td>LTD</td>
<td>Long-term decision</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Introduction 1

2 Related work 6
   2.1 Bin picking systems 6
   2.2 Deep learning-based object detection 7
   2.3 Deep learning-based image segmentation 10
   2.4 Preprocessing and synthesizing datasets 14
   2.5 Occlusion-aware object recognition 18

3 Occlusion-aware object recognition 22
   3.1 Overview of the method 22
   3.2 Full shape segmentation for solving the occlusion 24
   3.3 Data generation 26
   3.4 Experimental evaluation 30
      3.4.1 The Dataset Statistical Tool 30
      3.4.2 The Classifier Accuracy Tool 32
      3.4.3 Evaluation results 33

4 Occlusion-aware object manipulation system 36
   4.1 Overview of the system 36
   4.2 Deciding the robot action 37
      4.2.1 Decision-maker 37
      4.2.2 Grasping order 38
   4.3 Hand motion strategies 41
      4.3.1 Sweep estimation 41
      4.3.2 Grasp estimation 43
   4.4 Experiments and evaluation 44
      4.4.1 The Dataset Generation Tool for Decision-Maker 44
      4.4.2 The Decision-maker Accuracy Tool 46
      4.4.3 Evaluation results for Decision-maker 47

5 Conclusion 51

References 53
1 Introduction

This thesis studies occlusion-aware part recognition for robot manipulation. The topic has been studied around, for example, robotic bin-picking systems, and this thesis focuses on robotic bin-picking as the main task as well. The applications of robotic bin picking are various: it could be used in human assisting robotics as well as in the warehouses. Because of that, it has economical potential and it has lately been studied a lot.

In order to perform tasks of humans, a robot needs to have the expertise that humans have about interacting with the environment without harming or breaking anything while performing the task in the required way. Recognizing the object and being able to move it is not enough in order to interact with the environment in the best way - not even for a human. For example, a child in his first years can recognize his toys and he knows how to use his hand to grasp them. Still, if his favorite storybook is half occluded under other books, his grasping attempt might end up falling the pile of books on himself. The child doesn't have the expertise to understand the consequences that moving an occluded object causes. Because of that, also a robot interacting in the human environment needs to understand if an object is occluded and by what it is occluded. With this knowledge, it can avoid the consequences of picking up an occluded object. Occlusion handling aims to generate enough information about occlusion which allows safe picking for all the objects. Occlusion-aware recognition solutions are needed in many other applications too, like self-driving cars or robot-assisted surgery.

Recently a lot has been done to improve the development of object recognition solutions. For example, Amazon has organized an Amazon Robotics Challenge (ARC [32]) in the field of bin picking and there have been several ImageNet challenges (ILSVRC [33]) in the field of object detection. These challenges among other research have introduced deep learning solutions, especially convolutional neural network (CNN) based models, for different object recognition tasks like object detection and semantic segmentation. The solutions have been developed rapidly. State-of-the-art methods in object recognition include methods like YOLO [11] and Faster R-CNN [9] while state-of-the-art methods in segmentation include methods like Mask R-CNN [10] and YOLACT [12]. One of the remaining challenges in the field is to handle occlusion.
Occlusion is a common situation in real life. Most of the time humans have just one viewpoint to a scene. Sometimes the human can move a little in order to get a better viewpoint with less occlusion to the scene. Still, if possible, humans first try to understand the scene with occlusion imagining the occluded parts. For example, if a pencil is partly under a pencil case, humans can easily understand how much of the pencil is occluded and if a good grasping point is visible or must be gained by picking first the pencil case from above it. When part of an object is occluded, the robot also has to estimate the occluded part to understand the structure of all the objects as a whole in that environment, like in figure 2 is shown. Only with this knowledge, it can deal safely with the object depending on the task. The tasks can be various, picking an object is just one of them. For example, a self-driving car must be able to estimate the real size of a partly occluded car to decide if it’s safe to pass it or not. The task of this thesis is similar to that of the Amazon bin-picking challenge. A robot must understand which object is covered by which other objects, and how to pick it up, for example, does it need to pick other objects from above to reach it.
Figure 2. The chip can occludes the bowl and the bowl occludes the chip can. The visible segmenting (in the middle) separates the bowl and chip can while the full shape segmentation (on the right side) shows the real shape and area of both objects.

In this thesis, we develop a method to simulate a human way of picking a target from a pile of objects with occlusion. That includes the human’s ability to recognize the target object with occlusion, analyze its position in the hierarchy of the pile, decide on picking order, and finally pick up the target. It also includes logical thinking. If the target is not visible at all in the beginning, the robot should estimate the most probable place for the target to be occluded and sweep there to see if the object is under there. The process is explained also in figure 3.

Figure 3. A high-level system diagram. The system will first take images from the scene. These images are analyzed by a fine-tuned deep learning model and the analyzing result is then used in deciding on robot action, after which comes either the picking or sweeping action.
Hence the robot should be able to

1. recognize a target object from any viewpoint in a pile of multiple objects,
2. if the target is not found, sweep the surface and try again to find it,
3. if the target is found, predict its occluded parts,
4. understand the positions of the objects in the hierarchy of the pile with help of depth camera,
5. make a decision of picking order (how to clear all the objects from above the target object).

To make the robot recognize the objects from any viewpoint even with heavy occlusion, we use a deep learning model for instance segmentation. We fine-tune it to our occlusion handling task by training it with a special training dataset that we call full shape annotation dataset. Normally the datasets are annotated just on the visible part of the objects but our dataset is annotated on the full shape of the object, i.e. both visible and occluded parts of the object. Because of that, we compare our full shape recognition results with visible part recognition results that we get from the same dataset which is just annotated on visible parts of the object only. To evaluate our dataset, we develop a dataset statistical tool. The tool aims to give us a better understanding of the features of our dataset which is important in analyzing the learning results of the model. We also create an accuracy tool to measure the learning results. Finally, we develop an algorithm on how the robot could use the knowledge from our model in deciding the picking order.

The main contributions in this thesis are as follows.

1. We create a dataset of synthesized images that we annotate according to the full shape (the visible and occluded parts) of the objects. We compare the results of our recognition model after training on our dataset with results of the same model after training on an identical dataset that is annotated according to the visible parts of the objects.
2. We create a couple of tools to understand the quality and features of the created dataset and to measure the accuracy of the model after training.

3. We create an algorithm to decide on how to reach and pick up the target object if it’s in the scene or how to sweep if the target is not found.

4. We simulate a depth camera by synthesizing depth maps for the images in our dataset. The depth maps are used in the decision-maker algorithm.

5. We create a tool to measure the accuracy of the decision-maker.

In this thesis, we will first present some related work from the field of bin picking and object recognition in chapter 2. The system overview of object recognition and detailed explanation will be given in chapter 3. The decision-making algorithm will be discussed in chapter 4. Finally, the conclusion will be given in chapter 5.
2 Related work

2.1 Bin picking systems

Multiple kinds of research have been conducted around bin-picking systems, especially around the Amazon Robotic Challenge (ARC) task and tasks similar to it. One of the tasks of ARC in 2017 was to find target objects and pick them up from a bin, where there were other objects too, see for example [2, 7]. The task is similar to this thesis, but in our case, we also pick up objects from a pile with heavy occlusion. In ARC 2017 one big challenge in the task was that the system had to be able to recognize not only well-known objects but also novel objects in short (30 min) training time. This task has been solved in many ways but, for example for [3], occlusion caused failed grasplings. In this thesis, there are only well-known objects and no time limits for training, but unlike the ARC task solutions [3, 7], we handle occlusion too.

The bin-picking task requires at minimum three subsystems together, one for detecting and identifying objects, one for attaching to objects, and one for manipulating objects, i.e. robot vision, a gripping system, and motion planning [1, 2]. The detection and identifying will be discussed more in detail in chapter 2.2-2.5. In this chapter we focus more on robot hardware, attaching to objects and manipulating them.

A lot in recognition depends on the images that the robot will get. The robot usually has multiple cameras, from which one or more can be depth cameras, see for example [2]. The camera can be located in the hand of the robot [27] or anywhere else, even outside the robot, like in [1, 7]. Like Zeng et al. [7] mention, the pose of the camera's effects on the viewpoints the images will be taken from, so it is important to think about it. Usually, RGB or RGB-D cameras are used. RGBD cameras also include a depth map of the objects in the image [24: p. 23-24].

The picking can happen either after the object has been recognized from the pile [1, 2], or before it is recognized [7]. In [26] the objects can look similar from some surface and because of that the recognition sometimes requires picking so that the object can be seen from all its
sides. Picking here verifies that the robot has found the right object. Other hand motions can also be included in the object searching process, for example, sweeping. The sweeping motion was used in the research of [26] to find the object when it wasn’t found directly on the surface of the pile. In the pile of objects, it is possible that the object is mostly or even fully occluded by other objects and in that case, it is useful to sweep the surface to see the objects in the bottom as well. Another way to find an object that is not seen in the surface is given by [2]. They check first if the object was recognized in earlier searches of other objects and if it was, they used information from that recognition to approximate in which location the object might be occluded. They then decide to remove objects from that area to find the target object. If the object was not found before, they pick up the biggest and highest objects supposing that the biggest chance to find the target object is under the biggest objects.

The picking can be done, for example, through a sucking system or a grasping system, depending on the object that needs to be picked up. In this thesis, we will focus on grasping systems as in this task the robot doesn’t have a sucking system. Most of the Amazon Robot challenge approaches use the sucking system though as it is more efficient. According to [1] 99 % of the challenge instances could be sucked.

Before the robot can grasp, a grasping point must be predicted. With help of machine learning even if the object is not known or recognized, a good grasping point can be predicted [7]. Also, other parameters (except the grasping point) must be taken into account when making a good grasp, for example, the weight of the object [7], physical constraints of the grasping tool, object pose, material properties, and surrounding [2].

2.2 Deep learning-based object detection

A robot sees an image as an array or grid of numbers. How can it recognize an object in it and know which object it is? This is the object recognition problem that needs to be solved for all the robot tasks with robot vision, also bin-picking systems. Nowadays in the field of machine vision, many deep learning models are created to solve this problem. Deep learning is used to
detect objects by localizing them with bounding boxes (object localization) and by categorizing the pixels with the identity of the object (object classification). Deep learning approaches are also used in object segmentation and in image synthesis. [25: p. 448.]

Deep learning is about artificial neural networks. A neural network is a mathematical way to simulate the human brain’s neurons in action. Feedforward networks often form the basis of neural network applications, it’s the basis of the convolutional neural network that is used in this thesis as well. The feedforward network consists of three parts: an input layer, hidden layers, and an output layer. The hidden layers can be many and the more layers there are in a network, the deeper it is. The layers are vector-valued functions connected to each other. From the layers, just the output layer is specified by the training data. It is the value to which the earlier layers (functions) must adapt to. Because the way to adapt is not specified by the training data, the layers are called hidden. [25: p. 164-165]

The object recognition task becomes much easier if the object can be recognized from its features, otherwise, you must compare the found object with every image in your dataset. Only if there is a corresponding or near enough image in your dataset, the object can be recognized, see for example [1]. Convolutional neural network (CNN) is one kind of deep learning approach for feature extracting and it is used often when working on images. It is an advanced way to use a neural network. Convolution is a linear operation that is used for extracting the most important feature of input that will be then fed to the feedforward neural network. CNN is really popular at the moment and rapidly developing machine vision. Indeed, CNN is used in many machine vision models at the moment, like ConvNet, YOLO, R-CNN, Faster R-CNN, and Mask R-CNN, and many more.

In short, CNN consists of the same input layer, hidden layers, and output layer as any other neural network. At least one layer of CNN must use convolution to be called a CNN [25: p. 326]. In convolution, the network’s input values are added to kernel values thus creating an output called feature map. In the case when CNN works on images both input and kernel are 2D grids of numbers. [25: p. 328-330.] Pooling operations are often used in between the layers of convolutional neural networks [25: p. 326]. The pooling is a way to decrease the number of parameters (i.e. the size of the feature map) and to make the network invariant to
small location translations. This is achieved by comparing a neighborhood of pixels together, for example by selecting from three pixel values the biggest number (max pooling) or counting an average from three pixel values (average pooling) and making a new layer of values from these values. Because of the pooling layer, there is less variety in the information of the second layer and thus the amount of parameters is reduced. The good in pooling is that it helps to keep the number of inputs smaller and thus makes all the CNN run faster and use less memory while it also makes the network invariant to small translations of the images. But on the other side, in the pooling, some information is always lost. Because of that, the number of pooling layers can vary depending on the task. [25: p. 336-339] For example CNN called VGG-19 uses 7 times pooling when ResNet-34 uses it just twice [8].

One important thing to understand when using neural networks is that the training set errors and validation set errors behave differently. The algorithm can overfit, which means that after a certain amount of training epochs the network is having fewer and fewer errors in the training set but more errors in the validation set. [25: p. 241-243.] The network has learned too well the training dataset and performs poorly with new data [24: p. 35]. By following these error values and saving all the weights, it is possible to restore the best parameters at the end of the training. This is called the Early Stopping method and it is one of the regularization strategies in machine learning. [25: p. 241-243.]

Next, we will introduce two famous and different CNN architectures for the object recognition task: YOLO and R-CNN based networks.

YOLO [11] is a CNN based object detection method. It is really fast in detection, to the level of real-time. While R-CNN based models have bounding boxes and classification components trained separately, YOLO is unifying the bounding box prediction and classification and doing them simultaneously in all the pictures. Unlike R-CNN based models, YOLO doesn’t use region-based algorithms to predict bounding boxes, instead, it studies all the input image by dividing it to a grid and letting the grid cells then predict the bounding boxes and their confidence scores. The confidence score tells if there is an object in the bounding box and how accurate the bounding box is. YOLO’s pipeline is simple, it has a single convolutional network predicting multiple bounding boxes and class probabilities for those boxes
simultaneously. Because of the grid system, the bounding boxes might not hit the object's location exactly and YOLO makes more localization errors, compared to R-CNN. It might also not find all the objects because every grid cell predicts only two bounding boxes even though the objects around the grid cell might be more. Still, it predicts less false positives in the background.

Region-based convolutional neural networks (R-CNNs) differ from YOLO, as mentioned above, in the way they predict bounding boxes. The Faster R-CNN’s [9] method for bounding box predicting is another CNN, the Region Proposal Network (RPN). It is a network that is trained just for finding the best bounding boxes that are then proposed with objectness scores for object detection and classification. The RPN is almost cost-free because it shares parameters with the detection network. Still, Faster R-CNN can handle just 5-17 frames per second which is much less than the speed of YOLO, 155 frames per second. R-CNN based models use a fully connected structure in their architecture and that makes them slower than YOLO. YOLO doesn’t use it. Another difference that makes Faster R-CNN slower from YOLO is that it trains the classification and bounding boxes separately. The accuracy of Faster R-CNN is better though, as the bounding box predictions are already more accurate. According to an experiment of [11] the mAP (mean average precision) of YOLO was 63.4% and of Fast R-CNN (71.8%).

2.3 Deep learning-based image segmentation

When localizing an object by a bounding box, we get as a result a rectangular area, where the object is located. We don’t have its exact location in pixel-level and we don’t know exactly how big an area it reserves from the image. If the goal is that a robot picks the object, the bounding box doesn’t give enough information in order to predict the grasping or sucking point. In order to know that we must use a method called segmentation. Segmentation algorithms segment areas of an image. The semantic segmentation algorithm studies the image in pixel level and labels every pixel according to the given categories. For example, it can label the pixels to belong to a category from a category set consisting of background, cat,
and dog. Semantic segmentation models label the pixels by category without separating different objects inside that category. The way to segment objects from the same object category separately is called instance segmentation. [13].

Figure 4. The differences between Object Detection, Semantic Segmentation, and Instance Segmentation. The image is taken from [13].

Semantic segmentation can be done in many ways. Nowadays common to most of them is that they are using convolutional neural networks. Even though semantic segmentation has been studied for a few decades, the recent success in semantic segmentation is due to convolutional networks. A method called Fully Convolutional Network (FCN) has specially set the development of semantic segmentation into rapid progress.

In this thesis, we focus on segmentation methods that have been inspired in object segmentation based models, namely Mask R-CNN [10] and YOLACT [12]. Mask R-CNN belongs to the R-CNN family, where R-CNN has been developed into Fast R-CNN, Faster R-CNN, and finally to Mask R-CNN. R-CNN, Fast R-CNN, and Faster R-CNN are models for object detection and classifications. These models study every object in the image separately.
Figure 5. Mask R-CNN classifies and segments every region of interest (ROI) of the image in separate branches in parallel. [10]

Mask R-CNN is developed for instance segmentation. Object detection aims to classify each object and localize them in the image by a bounding box. Semantic segmentation aims to classify each pixel to a category. Mask R-CNN combines these two tasks. Its goal is to classify each object and to segment them separately from each other even if they belong to the same object class, i.e. to do instance segmentation. It is the current state-of-art method in instance segmentation. The classification and segmentation are done in two separate branches simultaneously and independently. It is fast and accurate, its speed is 5 fps. Mask R-CNN is easy to use and fine-tune and it has been developed already for example to SD Mask R-CNN [6] and Mask R-CNN with relook architecture [16].

Figure 6. Mask R-CNN with relook architecture by [16].
YOLACT [12] is developed in the same way and with the same goal as mask R-CNN, they just use YOLO as the base model. YOLACT does segmentation better than mask R-CNN but still, it has less mAP because of the poor object detection caused by the problems of YOLO. However, YOLACT can process the images with speed above 30 fps, so it is a real-time approach to instance segmenting and 6 times faster than Mask R-CNN.

Training a deep CNN like mask R-CNN or YOLACT from scratch requires time, expertise, and a lot of labeled data but it is not the only way to use CNNs. It is a common practice to use these pre-trained CNN models for new tasks. By transfer learning, we can take useful knowledge that is learned in one setting and use it in another setting. In this way, the representation of the input data that we have can benefit from both training sets of the two settings. Transfer learning is especially useful when the two settings/tasks are similar to each other. [25: p. 536.] Still, even quite different tasks can be seen as similar: Tajbakhsh et al. [14] showed that even the domain can be changed from everyday images to medical images. They divide the transfer learning into two groups, one of them being fine-tuning i.e. adapting the CNN model to the task at hand, and the second one of them being the use of it as a feature generator which output can then be used for something else. Tajbakhsh et al. [14] proved that fine-tuned CNNs can outperform CNNs that have been trained from scratch and that fine-tuned CNNs are more robust to the size of training data.

Fine-tuning the pre-trained models is indeed really popular and it can be done in many ways: Kentaro Wada et al (2018) [16] fine-tuned Mask R-CNN for their method to handle occlusion. They changed the last layer of the Mask R-CNN from sigmoid to softmax-layer and added new layers to the architecture. Danielczuk et al. [6] fine-tuned Mask R-CNN to adapt to depth images. The model’s name is synthetic depth (SD) mask R-CNN. Their results revealed that SD mask R-CNN outperforms point cloud clustering and the results were near to the mask R-CNN that is trained with a massive dataset of hand-labeled RGB images and fine-tuned on real images. Like Causo et al. [1] mentions, in challenges like the Amazon robot challenge, there is not enough time nor data to create and train a CNN from scratch so transfer learning can help there. They solve the time and data limitation problem using partly non-learning solutions and partly a fine-tuned CNN, Inception v3, from which the last layer is retrained.
Another ARC competitor, the winner of the 2017 ARC [3], fine-tuned for both classification and segmentation tasks a CNN model called RefineNet. The network was trained with a 3D dataset of synthesized images from Point Cloud and fine-tuned later with the task-specific images. Dwibedi et al. [4] fine-tune for their task the pre-trained Faster R-CNN.

Different metrics can be used to evaluate the success of object recognition and segmentation. IOU (intersection over union) and F1 are common ones but there is also a metric called F0.5. The different metrics take different things better into consideration. The F0.5 penalizes more false positives than false negatives. (False positive means that the model detects an object that is not in the image, the false negative means that the model doesn’t find all the objects.) Morrison et al. [2] argue that it is more applicable in robotic bin picking because the false positives result in more failed grasps than false negatives. IOU and F1 penalize both negatives equally. Mean average precision (mAP) is a metric often used in instance segmentation but it can only be used for the visible regions. Wada et al. [16] prefer panoptic quality PQ measurement for the instance segmentation with occlusion handling. PQ evaluates both detection accuracy and segmentation accuracy. Like F1 and IOU, PQ doesn’t take into consideration the different importance of false positives and false negatives.

2.4 Preprocessing and synthesizing datasets

Robot vision is not the same as human vision. Computer vision works on an image which is actually a 2D array of numbers [24: p. 36]. Still, robots are not limited to see the same or see in the same ways as humans do. It is important to know which goal we have for robot vision. [24: p. 22.] Is it useful to know the depth? Is it useful to know the colors? How wide an area does the robot need to see? How far the robot needs to see? In the case of this study, the robot needs to understand the depth and colors but does not need to see an especially wide area.

Image processing and computer-vision algorithms are used to make the robot understand its visual environment [24: p. 24]. Nowadays that is often done by using deep learning based methods. In deep learning, a dataset is an essential part of the learning process because the
features are determined from its data (in our example, the images). The quality (and relevance) of the dataset is thus closely related to the success of all the deep learning applications. According to [23], dataset quality depends on its relative density and diversity i.e. the number of images that differ from each other.

The aim of deep learning based methods is that after training on a training dataset the deep learning model will generalize its learning results. Generalization in the case of object classification means that the network is able to handle an object even though it has not been trained with the exact image of it. [24: p. 35.] In other words, after seeing 1000 images of cats, the network should be able to recognize all kinds of cats to be cats. The more training data, usually the better generalization result the network can give [25: p. 236]. Because of that a dataset really can have hundreds of thousands or even millions of images [21, 22, 23, 6]. The training data must also be annotated [24: p. 24], in case that the network is supposed to learn how to predict segmentation.

Getting a dense and diverse dataset with hundreds of thousands or millions of images and that suits your specific task is important but it can also be quite difficult. Collecting a new one can be really time-consuming but on the other hand, finding an existing dataset for the task might not be possible, see for example [17, 21, 22, 23]. When collecting a new dataset from scratch, finding images might be easy, but being sure about the quality of the dataset is harder. Zhou et al. [23] collected their dataset from a few internet search engines. The search words were used as labels of the images. After automatic processing of the data (deleting duplicates and too small pictures), they used human annotation by Amazon Mechanical Turk two times. Manual annotation is time-consuming and expensive. The human annotation was needed to correct all the labeling that was taken from the search words, i.e. to delete images that didn’t actually represent the label. After cleaning the raw data in this way for better quality, the amount of the images in the Places dataset of Zhou et al [23] dropped from 40 million to 7 million. We can understand that the annotation process was indeed needed even though it surely was really expensive and time-consuming.

There is another way to get a dataset, too. It is popular nowadays to generate synthesized image datasets for training machine vision models, see for example [4, 5, 6, 17]. Goodfellow
et al [25: p. 236] call this method as generating fake data. The benefit of the synthesized image dataset is that it is automatically annotated correctly, it requires little effort and a neural network trained on synthesized images can generalize even to real images, see for example [4, 5, 6]. Another difference in generating images for a dataset, instead of collecting, is that it requires much less real input images. In [16], the amount of 2D input images per object is 4-6 images. All the images are taken from the same object but from different viewpoints. This way is, of course, useful only when the task handles just certain objects very well, i.e. the object classes are quite specific. This is the case, for example, in the Amazon robotic challenge, where the amount of objects to handle is limited and the environment settings are stable.

The synthesized image datasets can serve different goals. For example, Dwibedi et al. [4] and Hodaň et al. [5] generated a synthesized image dataset for instance detection, while Danielczuk et al. [6] generated it for segmentation. Generating synthesized images is quite simple. In [4 and 16] the objects are cut from images and pasted randomly on backgrounds. A common goal for all synthesized training datasets is to generalize the network also to real-life images. In order to reach that, Hodaň et al. [5] were careful about positioning the objects in the background. They manually annotated stages where the objects could be put in real life. Then they calculated the possible poses of the objects at any stage. They also used a lot of calculation power to arrange the objects physically plausibly, according to gravity. Both Dwibedi et al. [4] and Hodaň et al. [5] use real scene images as a background. Dwibedi et al. [4] used FCN (Fully Convolutional Network) to achieve the masks for every object. The mask is then used to cut the objects from the original images. This way of cutting objects is similar to the way of this paper too, even though the way to get the masks is different. Hodaň et al. [5] and Danielczuk et al. [6] take images of every object from the 3D object model and from it they conclude the mask. In [4, 5, 6], the input images also have a corresponding depth image, which makes them have 3D images as input images.

Usually, the computer will learn faster from a preprocessed training dataset. The dataset can be processed, for example, by using different algorithms as feature extractors. [25: p. 34.] The synthesized images also can be preprocessed to gain better generalization results and better overall results. Cutting and pasting often cause pixel artifacts between the pasted object and
the background image. To decrease the negative effect of pixel artifacts on generalization, Dwibedi et al. [4] fine-tune the detector so that it will ignore the artifacts in training. In other words that means that they use different blending techniques for the training data. The best results come when the model is trained on the original images and images from two used blending techniques, Gaussian blurring, and Poisson. That means that the same image is three times presented in the training data with small differences that are generated by blending techniques.

In order to generalize the neural network to different poses and locations of the objects, Milan et al. [3], Dwibedi et al. [4] and Wada et al. [16] use geometric data augmentation. Geometric augmentation can be seen as a way to preprocess the training data to better generalize to the above mentioned real-life conditions. It reduces the generalization error. [25: p. 448.] As we said already, the more diverse data in training, the better generalization. Data augmentation allows many different ways to synthesize varying fake images from a few real input images. So instead of taking more real images, different real-life occasions can be simulated through geometric augmentation [16, 3, 4]. Data augmentation is especially popular among different classification tasks, because, according to Goodfellow et al. [25: p. 452-453], the class is invariant to a lot of geometric transformation.

There are different geometric augmentation methods, like flipping, rotation, transformation, and translation. These augmentation methods can solve the problem of positioning biases that are obvious when having just a few input images. The disadvantage of them is that they might require additional memory, have computing costs, or increase the training time. [20.] Many augmentation algorithms like translation, rotation, and scaling have been proven to be efficient in improving the generalization [25: 236-237.]. Dwibedi et al. [4] use 2D and 3D rotation. Wada et al. [16] use both color augmentation and geometric augmentation. From geometric augmentation methods, they use affine transformation, translation, rotation, and shear.
2.5 Occlusion-aware object recognition

One difficulty that worsens the object recognition results is occlusion. For example, in [2] 10.3 % of failed grasping attempts were caused by occlusion, and Zeng et al. [7] chose to pick the objects up before recognition is done to avoid occlusion handling. When an object is in the robot’s hand, it can’t be occluded by other objects so it’s faster to train the network for the new objects, it’s also making the recognition easier. Semantic segmentation or instance segmentation doesn’t handle occlusion anyhow.

In real life, there is occlusion everywhere. Because of that in recent years, there has been increasing interest in occlusion handling and researchers have proposed different solutions to handle it in different situations. So far many proposals have been quite task-specific and the solutions cannot always be generalized to other kinds of tasks. Because occlusion is really common in real life, it needs to be studied more.

One occlusion topic which many studies have been carried on is recognizing an occluded human [19, 28, 29, 30]. Liu et al. [19] made an early approach to occlusion detection and it has a unique solution to it. The research studies what is occluding a human by Deformable Part Model with images and verbal phrases. Deformable Part Model is not a CNN. Instead of showing the occlusion in the image, Liu et al. [19] create a phrase that explains what is occluding a human, i.e. “horse occludes human”. The research was published in 2015 and its results are quite weak compared to the results of the year 2019 on the same field of occluded humans.

Cheng et al. [28] and Li et al. [29] can handle occlusion and make estimations of the human’s occluded areas. Cheng et al. [28] create an occlusion-aware deep learning framework to estimate occluded 2D and 3D key-points (i.e. joints) of a human. They outperformed current models in the cases when the studied human occluded himself/herself but if there were another human occluding the studied human, their framework couldn’t handle it. Another neural network-based model to understand and expect people’s actions in dark conditions or occluded persons was proposed by [29]. Their model could handle many people over time.
The way they handle occlusion is radio frequency which is quite a unique solution. The result is still comparable to the visible action recognition systems and widens the area of seeing to occluded areas. Discriminative feature transformation which improves pedestrian feature separation and assists non-pedestrian models to deal with occlusion was performed by [30]. By pushing non-pedestrian examples toward the centroid of the easily-distinguished non-pedestrian examples, the missing parts of the occluded objects are obtainable. As a result, the performance of pedestrian detection is significantly enhanced. Their approach depends on employing one transformation network in the fast R-CNN framework. Commonly-used datasets for pedestrian detection Caltech and City Persons were used to validate the proposed approach. Their results showed a very promising potential for both occluded and none-occluded pedestrian detection.

An early approach towards the topic of this thesis, object segmentation with occlusion handling is carried by [18] in 2015. They use the Pascal2012 dataset on a convolutional neural network called SDS CNN. This approach is similar to Mask R-CNN in a way that it simultaneously detects the objects and segments them. It is made for visible objects though so to make the system occlusion-aware, a really complicated algorithm is applied to it. The algorithm studies if two object proposals are overlapping. This algorithm contains likelihood maps of every image, shape prediction with examples, and occlusion dataset and in the end, they are all formulated to an energy minimization framework to handle the occlusion and make the final output segmentation. Despite being so complicated, the model manages to get AP (average precision) with IoU 0.5 to be 38.4 with images with occlusion, which was better than the previous state-of-art model of that time. The model also improved the classification, especially in the cases that the objects were occluded.

A more recent approach to the topic is conducted by Wada et al. [16]. Usually, the segmentation datasets for image detection only have visible masks for every instance but their dataset has both visible and occluded masks of every instance. In this way, their model can be trained to predict the visible and occluded masks. In this thesis, we have just one dataset of full shape masks. Full shape mask is the combination of the visible and occluded masks of [16].
Wada et al. [16] don’t use depth cameras and hence don’t know the depth of every instance. In order to know the picking order, they must rely on understanding what is occluded by what. So this approach must study the relationship of the objects from the pictures with the help of the masks. The approach showed that occlusion-aware object recognition can be done without a depth camera. This idea of solving the occlusion problem without depth sensors is great because depth sensors are not accurate enough yet to understand thin instances like paper or transparent objects like a glass bottle. It is also cheaper to have a robot without a depth sensor. At the moment the depth sensor still looks like a simpler way to understand occlusion. Still, without a depth camera, it’s hard to grasp the object properly because the robot will not know the depth to grasp. In the suction hand, it might be easier as the suction hand can just go deeper until it reaches the object.

In the study of [16], Mask R-CNN is used for the occlusion handling but with some changes and additions. Mask R-CNN doesn’t study connections between the objects. Still, the connection is important for understanding which instance is occluded by which another instance. This helps to conclude which instances must be removed and in which order to reach the target. Also, in the study, the sigmoid function in the last layer of the mask branch of mask R-CNN is changed to softmax function. In this way, they can have a multi-class output which gives multiple masks of a single instance (visible, occluded, and out of the instance masks). Moreover, they added a new relook model, which studies the relationship and dependencies among detected instances. This gives them a better understanding of the connections between instances (which instance is occluded by which) and allows them to conclude the picking order. This relook architecture makes the process slower from the original mask R-CNN.

Other studies on the topic include for example [15] and [17]. Ehsani et al. [15] combined two CNNs to create a network that handles occlusion while Purkait et al. [17] fine-tuned an existing CNN for that. Ehsani et al. [15] have synthesized a 3D dataset that is created from 2D images. They train the network to paint the invisible regions of an object based on its visible region. The model has two parts, one doing segmentation and one generation/painting. They use a network called Multipath network for segmentation and the masks taken from this network are then fed with the RGB pictures to another network which paints the invisible
regions of the objects, kind of like create the occluded masks but takes in account the color of the object and tries to generate its full image. The approach is called SeGAN. Purkait et al. [17] fine-tuned the CNN-based U-net for their occlusion handling task by changing the last soft-max layer to the group soft-max layer. In this way, they solve the problem of semantic segmentation networks that they only give one label per pixel. From their group-wise semantic segmentation, a single pixel can get multiple labels, one for every object category.
3 Occlusion-aware object recognition

3.1 Overview of the method

The appointed task of this thesis is to design an application for grasping an object from a pile of heavily-occluded objects. This grasping task is primarily detecting the objects and then understanding their hierarchy. However, we need to recognize the occluded objects in order to understand their hierarchy. As a result, herein, we are trying to solve the occlusion problem in a pile picking system.

Fortunately, there are many ways to recognize the visible parts of the occluded objects in deep learning through segmentation. However, the most challenging part of recognizing the occluded objects is to identify the invisible parts. Nowadays, few ways are currently available to estimate the invisible parts of occluded objects.

Our solution is inspired by the human way to recognize the visible regions of the objects and estimate the invisible regions of it using the previous knowledge about the full shape (the visible and invisible parts) of objects. Our method is based on this idea as well. We want our deep learning model to estimate the outer lines of the invisible region of the object from its visible region based on the knowledge of the full shape of the object. This knowledge we provide for the model by training it with the full shape (the visible and invisible parts of the object) annotations, see figure 7.
Figure 7. The humans can understand the occlusion (upper row left) by their previous knowledge of the objects (upper row middle and right). A deep learning model can learn the full shape of the objects when it is trained with the full shape annotation (lower row left) and gain the knowledge of objects full shape (lower row the middle and right). The knowledge of texture is not needed in this task.

The full shape of objects (visible and invisible parts) can be segmented into a single mask. Then, the intersection between any of these segmented full shape masks in an image represents occlusion. By employing a depth camera, we can identify which object is occluding the other. In our method, Mask R-CNN, as a deep learning model, was used for segmentation.

Unlike other occlusion handling methods (see subsection 2.5), our method implies a dataset that is annotated with the full shape of objects. Visible parts can be easily annotated, but we can not annotate invisible parts because we don’t see it to annotate it accurately. Thus, a synthesized dataset was created to overcome this limitation as explained in 3.3.

Wada et al. [16] reported a different method to deal with the problem of occlusion in the same task using Mask R-CNN. They predicted simultaneously two kinds of masks, one for the invisible part and the other for the visible part. They didn't have to use a depth camera to conclude the hierarchy of the objects, because they possessed up to three different masks, one
for the visible part, one for the invisible part, and one for the background. In addition, they had a deep learning structure to study the relations between objects to conclude their hierarchy which denoted relook-architecture. However, this relook-architecture required additional time of processing.

3.2 Full shape segmentation for solving the occlusion

Normally in a deep learning model, the dataset which is prepared to train the models is annotated for just the visible part of the objects. It is a realistic way to train a model to recognize and segment the scene objects.

We solve the problem of occlusion by annotating the full shape (the visible and invisible parts) of the objects in our training dataset so that our model can segment the full shape of objects in the testing. As a result, we can find the occlusion between object masks and which objects have it. We choose to use the state-of-the-art deep learning model for instance segmentation, Mask R-CNN.

Using a Mask R-CNN for the occlusion handling task could be seen as an unlogical way because it’s employing instance segmentation. Instance segmentation is a way to segment each pixel to a specific object. How can it make segmentation for two objects on the same pixel in the occlusion area? The answer is that Mask R-CNN is making instance segmentation in parallel for every object. This kind of independent segmentation for every object allows us to segment two objects' masks even if they are sharing the same occlusion area.

We are using the same network structure for the Mask R-CNN implementations as on https://github.com/matterport/Mask_RCNN page. The structure is divided into:

1. The backbone (ResNet101 architecture): for feature extraction.

2. The header: for bounding-box recognition and mask prediction.
The backbone, ResNet 101, extracts the features from the images producing the feature maps. The region proposal network (RPN) takes the input from the backbone and proposes regions of interest (ROI) on the feature map. The regions of interest are fed for two branches, the classification branch, and the mask branch, to predict the classes and bounding boxes and the masks in parallel, see figure 8.

Figure 8: The detailed structure of Mask R-CNN from the input image to the classes, boundary boxes, and masks, as drawn by Hui [34].

We are using a pre-trained model trained with the MS COCO dataset. We replaced the pre-trained output layer with a new output layer suitable for our number of objects. Then, we train just the header of Mask R-CNN keeping the backbone (feature extractor) as it is. The Mask R-CNN was trained twice with different kinds of annotations for the same images, visible annotation, and full shape annotations. The Mask R-CNN trained with visible annotation is the state-of-the-art model for instance segmentation and we want to compare our model with it. Because of that, we train the model also with the visible annotations even though our solution is trained only with full shape annotation.
3.3 Data generation

We generated a full shape annotation dataset by the Dataset Generation Tool that we created for this purpose. The tool synthesizes images and creates full shape masks for every object in those images and uses the full shape masks to make the full shape annotation. The parameters for the Data Generation Tool are the number of synthesized images we want to generate and the number of objects in every image. The difference from Wada et al. [16] is that we train the Mask R-CNN with a dataset of the full shape annotation and we predict full shape masks. Wada et al. trained the Mask R-CNN with a dataset that includes two kinds of annotations, the visible and occluded annotation. In order to make the Mask R-CNN able to predict two different masks at the same time, they fine-tuned the mask branch in Mask R-CNN.

We also generated from our dataset the visible masks and the visible annotation. And we train another Mask R-CNN model with them to compare the accuracy of our Mask R-CNN model trained with a full shape annotation dataset with the Mask R-CNN model that is trained with the usual visible annotation dataset.

The two kinds of masks, the visible masks, and the full shape masks, are important in the dataset generation as they are used in the annotation process. They are also used as valuable information by the Dataset Statistical Tool and as ground truths by the Classifier Accuracy Tool. The Dataset Statistical Tool and Accuracy tool are explained in detail later (see chapters 3.4.1 and 3.4.2).

Papers [16 and 4] extract the masks using object recognition networks like ConvNet and FCN respectively, but our solution is to use image processing, the Canny edge detection, for finding the masks for the objects. The solution of [16 and 4] is better in finding the masks for transparent objects. The image processing solution needs more manual checking and optimizing but on the other side, it can give masks nearer to the right shape of the objects. We didn’t add any transparent object to avoid the problem of transparency and we did manual checking for all the masks. Because of that, the quality of our masks could be slightly better.
from the masks of [16 or 4]. It is important because the better masks will give better annotation for a dataset and thus we will have better results from our model.

We generated the datasets according to the following steps:

1. We collected a small dataset of images of the objects first. We chose to have 8–15 images per object and they were from different viewpoints.

2. For every object, we created a corresponding binary mask, where the background is black and the cropped object is white. We used Canny edge detection to detect the edge of the object, then we created an empty mask with the same size as the image and inserted the biggest contour area found by Canny into the mask and filled it in white color. In the end, we used Gaussian blurring for blending. Please note, that sometimes the Canny edge detection didn’t work as wanted (see figure 9) so to ensure the good quality of the masks, every mask was checked manually. The best result for Canny detection can be gained when the images are taken in a one-color background, in a bright environment without shadows, and by a high-resolution camera. We used mostly white backgrounds but for some objects, black or green backgrounds were used, depending on the colors of the objects because the background color must not be the same color with the object (multicolor problem).

![Image](image1.png)

Figure 9. The Canny edge detector’s incorrect results will cause incorrect masks. Because of that, the results were checked manually.
3. Then we used these masks to crop the objects from the images so that we had object images without a background, see figure 10. These cropped object images are used to make synthesized images in step 4.

![Image](image1.png)

Figure 10. The object images (on right) were taken on white background from which they were cut (on left) with the help of masks (in the middle).

4. We gathered 21 different background images for the synthesized images. The backgrounds have different patterns and colors on them. We then generated new images by adding the cropped object images in random positions on the background to simulate the real-life occlusion. We got the random positions by augmentation, that was used to rotate, translate, and flip the object images. Please note that even though we have quite a small number of images of one object, the augmentation process allows us to have almost endlessly different synthesized images. The augmentation result can be different in every generated image so there will not be repeating in the dataset, see figure 11.
Figure 11: The object images on the background after augmentation. After augmentation, the same objects can result in an endless amount of different combinations for the images.

At the same time with the augmentation, we made the full shape masks from every object in the image and saved them in different files with a special naming system. We also created visible masks to use them later to compare our results with it. See figure 12 to understand the difference between the full masks and visible masks.

Figure 12. The synthesized image and its masks, the upper row have the visible masks, the lower row has the full masks.
5. In the last step, we annotated the masks by taking the contour of the masks and saved the results in a JSON file. In this way, we have the training datasets and the corresponding annotation.

3.4 Experimental evaluation

3.4.1 The Dataset Statistical Tool

Because we are creating synthesized images by randomly choosing many objects for every image and randomly, with geometric data augmentation, locating those objects inside the image (which will give us many kinds of occlusions between objects), we don’t really know what kind of dataset we have got. The Dataset Statistical Tool helps to understand many statistical features of our synthesized image dataset and to analyze in more detail the result for our model after training. The Dataset Statistical Tool could be implemented in the Dataset Generating Tool.

The Dataset Statistical Tool tells us:

1. The number of occluded and non-occluded objects in our dataset. By comparing the full shape masks and the visible masks that we created in data generation, we can conclude how many objects are occluded. We compare the sizes of the visible masks to the sizes of the full shape masks. If the size of a visible mask is equal to the corresponding full shape mask, we can understand that there is no occlusion. But if the size of a visible mask is smaller than the size of the corresponding full mask, the object must be occluded. By knowing these numbers the percentage of occluded and non-occluded objects in the dataset can be concluded.

2. How many times every object is presented in the dataset. Because we are randomly adding the objects for the synthesized images, this tool helps us to understand the balance of the existing objects in the dataset. So in case the classification or/and the
segmentation is not accurate for an object, we can check by this tool if that object is presented in the dataset often enough for the Mask R-CNN to learn it.

3. The size of the full shape and visible masks (in pixels) and the ratio between the visible mask and full shape mask for the same object. From this ratio, we can understand how much every object is occluded.

4. The ratio between visible mask size in pixels and the full size of the image in pixels see figure 13. From this ratio, we can understand how much of the object is visible after the occlusion. If the visible part is too small (< 0.75%) compared to the size of the image, it will be counted as a bad (unuseful) visible mask. In that case, the object is mostly occluded and it is obvious that it won’t be recognized in a testing (and even in training).

![Figure 13: The Dataset Statistical Tool tells the size of the object compared to the image size and the object size. The size of the Mint box (on the left) is just 0.25% from all the image size and the size of the shampoo bottle is also less than 0.75% from the image size. Because of that both of them are accounted as unuseful masks.](image)

5. The number of “almost non-occluded” masks. If the visible mask size is > 95% from the full shape mask size, it is an “almost non-occluded” mask, because the object is so
little occluded that its visible mask is similar to the full shape mask, see figure 14. The “almost non-occluded” masks are counted as non-occluded objects by the tool.

Figure 14: The mask size of the Monster can (on the left) is 97.72 % from the full shape mask size so it is counted as a non-occluded object while the Pringles can, which mask size (on the right) is 90.12 % from the full shape mask size, is counted as an occluded object.

3.4.2 The Classifier Accuracy Tool

Because we use synthesized images to create our dataset, we have the ground truths for the dataset including

- the name of the images
- the number and names of the objects inside every image
- the mask shape and position for every object inside the images.
When we test Mask R-CNN, it will predict multiple masks for every single image from the dataset: one mask for every single object inside the image. We will save all those masks and name every one of them depending on the image name that we get from the dataset and the classification that we get from Mask R-CNN. In this way, we can take the results of Mask R-CNN and compare it with the ground truth because they have the same name for the mask.

We will compare the accuracy of two Mask R-CNN models, one that is trained with our full shape annotation dataset method and one that is trained with the usual, visible annotation dataset, which is the state-of-the-art method. We will check two different accuracies, the segmentation accuracy, and the classification accuracy as follows.

1. **Segmentation accuracy** tells how many from the detected objects have correct segmentation. It is counted by dividing the number of correct segmentation by the number of all detected objects in Mask R-CNN. Intersection over Union (IoU) is used to check the correctness. For example, there could be 8 detected objects but the classifier segments just 6 correct full shape masks, so the segmentation accuracy is $6/8 = 75\%$. The undetected objects can not be segmented or classified, and because of that they are not taken in consideration when counting the segmentation accuracy or classification accuracy.

2. **Classification accuracy** tells how many from the segmented objects were classified correctly. It is counted by dividing the number of correct classifications by the number of correct segmentations. For example, there could be 6 correct full masks but the classifier made just 4 right classifications. In this case, the classification accuracy is $4/6 \approx 67\%$.

### 3.4.3 Evaluation results

For the training, we generated 1200 synthesized images with 9 objects per image using the dataset generation tool, described previously. We annotated the images twice in order to train
two models by two kinds of annotation, visible and full shape annotation. We divided the dataset so that 1000 images are for the training dataset and 200 for the validation dataset. We did training for the two models in the same image dataset, the first model is trained with visible annotation and the second with full shape annotation. For the evaluation, we generated 100 synthesized images with 12 objects per image. Finally, we evaluated the results of both models by the accuracy tool, described previously, with IOU threshold 0.5.

The training time for both the full shape annotation model and a visible annotation model was 6 to 7 hours for each. The learning rate was 0.001, learning momentum 0.9, and there were 1000 steps per epoch, and in total 60 epochs. The image size was 504*378 pixels. We have 13 object classes. The weight decay is 0.0001. The Mini-batch has 2 images per GPU and only one GPU was in use. The GPU is NVIDIA’s GeForce 1080 and CPU is Intel i5.

Both of the models were tested by 1200 objects (100 images * 12 objects per image). In segmenting the masks, the full shape annotation model achieved 88.1 % accuracy by IoU 0.5, while the visible annotation model exhibited 89.4 %. Interestingly, the difference is small. It is worth mentioning that the visible annotation should have the same accuracy for the original, state-of-the-art Mask R-CNN.

For the accuracy of the classification, the full shape annotation model achieved 82.95 % while the visible annotation model got 87.07 %. There is no significant difference between them, but the full object shape annotation model had more errors.

When studying the result with the Dataset Statistical Tool, we found that the presentation of 81 objects in the images in the testing dataset is so small that it is not possible to recognize them. Taken that into consideration, the detection accuracy would most probably be better, if we had not allowed these small presentations to be shown in our dataset from the beginning. On the other hand, the results are relatively good even in our situation.

The Dataset Statistical Tool reveals that 481 masks are denoted as bad masks, meaning that those kinds of masks resulted from objects whose visible parts after occlusion is really small compared to the image size, less than 0.75 %. So if we had developed the Dataset generator tool to ignore those masks from the beginning, we would have avoided training the Mask
R-CNN on less than 1429 (size of a training image multiplied with 0.75 %) pixels as an object and we would, most probably, have got a better result.

Paper [4] uses a constraint when synthesizing the image. They allow occlusion or truncation only when a minimum of 25 % of the object’s bounding box must be seen. That means that they aimed to train their model to classify objects where the visible part is 25 % or more. Maybe we should add a similar condition with them in the dataset generation tool. A constraint in the presentation of the object compared to the whole image size is not directly comparable with the presentation of the object compared to its full size because the size of an object or an image can vary. In our experiment, the dataset statistical tool reveals that 24.72 % of a shampoo bottle equals 0.65 % presentation compared to the all image size. Maybe both constraints are needed, the visible area of the object should be more than 0.75 % of all the image size and 25 % of full object size.

<table>
<thead>
<tr>
<th></th>
<th>segmentation accuracy</th>
<th>classification accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>full shape annotation model</td>
<td>88.1 %</td>
<td>82.95 %</td>
</tr>
<tr>
<td>visible annotation model</td>
<td>89.4 %</td>
<td>87.07 %</td>
</tr>
</tbody>
</table>

Table 1. The comparison of the accuracy of the full shape annotation model and the visible annotation model.
4 Occlusion aware object manipulation system

4.1 Overview of the system

The system consists of software and hardware building blocks. The software blocks are robotics middleware ROS (Robot Operating System), instance segmentation model Mask R-CNN, and a decision-maker for the robot actions, which decides if the robot will grasp or sweep. The hardware building blocks are the robot and the computer server. The robot includes a gripping hand and a depth camera, and it is operated by ROS. The computer server runs the Mask R-CNN and the decision-maker, and also ROS to communicate with the robot. Next we will explain the system flow through the building blocks as in figure 15.

1. The robot will take a color image and a depth map by the depth camera that is located in its head, and publish them in the ROS system environment. The color image and the depth map are aligned and synchronized.

2. Mask R-CNN will subscribe to the color image and its depth map and make a prediction for the classification of the objects in the color image and for their masks.

3. The decision-maker is implemented inside Mask R-CNN to reduce the time and the errors which might happen due to publication and subscribing. The decision-maker needs the predictions of Mask R-CNN and the depth map, and the target object in order to make a decision about what the robot should do next: grasp, or sweep. A user will inform the decision-maker about the target object. After receiving all the information and making the decision, it will publish on the ROS environment the order to grasp or to sweep with the needed parameters for the order.

4. The robot will then subscribe to the order and act accordingly.

The grasping and sweeping needs parameters to success. The grasping can be done in two ways. The suction hand needs to have as parameters the coordinates of the center point of the object while the gripping hand requires also the width of the object as well as the rotation
angle \( \alpha \) for the wrist. The sweeping requires depth points A and B, which represent the start point and endpoint of the sweeping motion.

Figure 15. The system flow through the hardware and software building blocks.

4.2 Deciding the robot action

4.2.1 Decision-maker

After the frame and depth map are published in ROS, Mask R-CNN will subscribe to them. Mask R-CNN will perform a prediction for the classes and masks for all objects in the scene. Every scene has multiple objects so there are going to be multiple masks. The user will input the targets which we need to find and grasp. All this information is forwarded to the decision-maker. The decision-maker uses this information to find the occlusion between the objects, to decide if the robot should pick up something and in which order or if it should first sweep the scene and pick up after that.

The decision-maker follows these steps:
1. Identify if the target is present in the scene by checking the classification vector produced by Mask R-CNN. If the target exists, go to step (2). If it’s not present, go to step (3).

2. Grasp an object: Check by using the masks and the depth map if the target object is not occluded by anything. If it is, go to step (b). Otherwise, go to step (a).
   a. Grasp in order all the objects that occlude the target.
   b. Execute grasping the targets and end the processing.

3. Sweeping hand motion:
   a. Estimate the sweeping area and sweeping direction.
   b. Execute sweeping.
   c. Return to step (1).

4.2.2 Grasping order

In case that the target object is occluded by other objects, we need to make a decision about how to grasp the occluding objects in order to reach the target object. To make that decision, at least two ways can be used, the long term decision (in future LTD) and the short term decision (in future STD). The main difference between these two methods is that the LTD gives an order to pick up a queue of objects to reach the target at once, so the robot executes multiple picking actions until picking the target with just one recognition result. The STD gives an order about which one object to pick up next and after the picking action, it will make a new recognition and decision for the new situation. The problem in the LTD is that it is not aware of any unexpected changes that the picking action might cause in the scene. The STD, on the other hand, could be more time consuming as it needs to make a new decision after every picking action. Next, we will go through the two decision making models more in detail.
Long term decision (LTD): Long term decision can be understood as a full path solution because the result of this algorithm is a sequence of all objects occluding the target with the target itself in a queue. When we grasp the objects in this queue one after another, we actually reach to pick up the target object with a single decision. As a result, we will send this queue only once to the robot which will grasp the objects one by one without sending other frames and depth maps to Mask R-CNN or making new decisions.

On the other hand, making this decision can be time-consuming because finding the occluding objects requires counting the intersections between the occluded object’s mask and all the possibly occluding objects’ masks. The more objects there are in the scene, and the deeper the occluded target is under the occluding levels, the longer the decision-making takes.

The LTD algorithm:

1. Put the target into a queue and check if the target is occluded by others.

   If yes, make all the occluding objects as targets and repeat step 1 for all of them until they are all free to pick up (i.e. they are not occluded anymore). Go to step 2.

   If no, go to step 2.

2. Pick up all the objects in the queue.

Short term decision (in future STD): This algorithm gives only one object to grasp, either one non-occluded object from all the objects occluding the target or the target itself. If it is not the target itself, after the grasp, the target object will be less occluded by other objects. Because of that, the short term decision can be understood as a single grasping step that simplifies the solution. The algorithm must be repeated until the target object is grasped which implies that we also have to send new frames and depth maps to Mask R-CNN and to make a new decision from the beginning. The repeating of the algorithm is time-consuming.

The STD algorithm:

1. Check if the target is occluded.

   If yes, change the occluder to be the target and repeat this step.
If no, go to step 2.

2. Check if the object is the original target. If yes, grasp it and end the process. If no, it is the occluder. Grasp it and return to step 1.

It looks like the LTD is faster because it will give the full solution path with just one decision while STD needs to repeat all the decision-maker algorithm as many times as there are objects occluding the target just to be able to grasp the target. Still, when we consider our main task where there is a pile with heavily occluded objects, we will understand that the hierarchy of objects could be changed after grasping any object. Either the hand of the robot could mistakenly touch another object during the grasping and move it from its position or some objects might slip from their positions after grasping other objects.

As a result, the reliability for the LTD is lower than STD, because in an LTD the robot grasps multiple objects without checking if the hierarchy changed in between the picking sequence. If we want to increase the reliability of LTD, we need to check positions and hierarchy for every object after a grasp and that will reduce the speed of the system. Moreover, if we find that the hierarchy of the objects has changed, we need to send another frame and depth map to Mask R-CNN and to make another LTD that will slow down the system again.

Compared to LTD, the STD is more reliable because we are making a new decision with a new frame and depth map after every grasp. In addition, we don’t need to find all objects occluding the target from first. Instead, we need just one of them with confirmation that it is free to grasp. While LTD is trying to find all the objects occluding the target or any of its occluders in all the levels of hierarchy, for STD finding just one object per hierarchy level is enough. This minimizes the work of this algorithm because finding the occlusion by checking the intersections between the masks is the most time-consuming part of both algorithms. One more benefit of the STD algorithm is that every time when we make a new decision, our objects occluding the target will be reduced by one, which makes this algorithm faster every time we call it.

We were not able to compare the two algorithms in real experiments and because of that, it is hard to know which one of them really performs better. The LTD might still perform better at
the end because if the target is really heavily occluded, it is most probably also not seen and in that case, the robot will sweep the scene. We chose the short-term decision because it’s more reliable. See figure 16 for the decision-maker algorithm with short-term decision for grasping.

Figure 16. The decision-maker algorithm with short-term decision. See sections 4.1 and 4.2 for details.

4.3 Hand motion strategies

4.3.1 Sweep estimation

Sometimes the target might not be found in the scene. This situation represents one of the three scenarios:

1. The target object is occluded by another object so we cannot see it.

2. The target object exists in the scene but there is a problem with the recognition and it gives us the wrong classification so it is not found.
3. The target object does not exist in the pile.

In these scenarios, we will use sweeping motion in order to get a new viewpoint to the objects and to be able to see the objects at the bottom of the pile. The sweeping motion is executed as follows:

1. We find first the most crowded mask area. It’s a kind of cluster of occluded masks in the scene. The biggest area will be the most likely area to find the target under it. We can find the crowded mask area by adding all the masks, which belong to the same image, above each other. After that, we make little erosion so that in case there are other objects connected to the crowded mask area through a little occlusion, they will be cut away from the crowded mask. We will find all contours and keep the biggest contour of them and we will make dilation for it to return to the original size. This is the crowded mask area, see figure 17.

![Figure 17](image)

Figure 17. The crowded mask area (at right) has the two grey circles on the rectangle, they represent the A and B points between which the sweeping goes. The sweeping direction will be from left to right as the center of the mass is nearer to the left side.

Then we will calculate the center of mass for the crowded mask area. After that, we will fit the smallest rectangle around the crowded mask area, and we will find points A and B, which represent the left and right side of the rectangle in the same height of the center point.
If the center point is nearer to the right edge of the image, the sweeping direction will be from right to left, and in the opposite situation vice versa. We are using the center of mass of the crowded mask as a pointer for sweeping direction from left to right or vice versa, avoiding a random sweeping or sweeping in the same direction multiple times which could collect all the objects on one side. Finally, we will take the depth coordinate for A and B because the sweeping motion goes between A and B. The sweeping is always horizontal.

2. We will repeat the step (1) N times to avoid the open-loop which can happen from the scenario when the target object does not exist in the pile.

4.3.2 Grasp estimation

When we want to grasp an object, we need to know the position of that object between all objects in the scene, and we need to estimate its pose to know from where to grasp it. We can know the position from the mask produced by Mask R-CNN. To calculate the pose for the grasping, we need to know the center of the mass for the objects, it’s height, width, and the rotation angle ($\alpha$), see figure 18.

![Figure 18. The grasp estimation parameters W (width), H (height), $\alpha$ (rotation angle) and the center point for the suction hand.](image)

Figure 18. The grasp estimation parameters W (width), H (height), $\alpha$ (rotation angle) and the center point for the suction hand.
Calculation of the height, width and rotation angle is simple because we have the binary mask that has the white area representing the object and black area representing the background. So we need just to calculate the contour of the white area and to fit the smallest rectangle around it to get height and width from it. After that, we can calculate the rotation for that rectangle from X-axis, to get the angle. We can calculate the center of mass from the contour but it will be without depth information. Because of that, we need the depth camera to have a 3D point for that center. The center point represents the place where to grasp if we will use the suction hand robot technique.

If we are using a gripping hand, we will need to know the same parameters as in the suction hand and also the width of the objects to know how much to open the hand and the rotation angle $\alpha$ for the wrist, see figure 19.

![Diagram of gripper hand](image)

Figure 19. The gripper hand will first open the hand according to the width (W) and then rotate the wrist according to the angle $\alpha$ after which the hand will be moved to the center point P and it will grip the object from the G-areas.

4.4 Experiments and evaluation

4.4.1 The Dataset Generation Tool for Decision-Maker

In order to test the decision-maker, we need to create a depth map for every synthesized image. This depth map will give the depth for every pixel in the image so it is showing the
third coordinate Z for grasping. At the same time, we need to prepare a ground truth for every image so that after we run the decision-maker we can compare the ground truths with the decision-maker results to check its accuracy. The Dataset Generation Tool gives us the following things (see also figure 20):

Figure 20. The Dataset Generating Tool for Decision-maker synthesizes images by giving them a value according to their level in the hierarchy. From this knowledge, the depth map (up in the middle) is created. The tool also gives the ground truth for the sweeping of crowded areas (upper right), the ground truths of the target object’s full shape (lower left), and the free objects occluding the target or its occluders (lower middle and right) with possible grasping points (the grey circles).

1. Synthesized images. The tool is using the same object masks and cropped object images that we have from the dataset generating tool that is explained in chapter 3.3. It is synthesizing the images in the same way as the Dataset Generation Tool does. The only difference is that the objects get a value according to their level in the hierarchy. This value is later used in creating the depth map.

2. Corresponding depth map for every synthesized image. We are trying here to simulate the depth map for a depth camera. To do this depth map, we calculate the levels of the objects to understand which object is above which other object and we will find at the
end the highest level of the images. We give color codes to the objects on different levels. The black color is the background of the image, and the lighter the object is, the higher it is in the object hierarchy. The white objects are the highest objects.

3. Ground truth for sweeping on the crowded mask area. The ground truth includes the correct crowded mask area, the right direction for sweeping, and the possible area for sweeping coordinates, i.e. the points from where to where to sweep. We made the ground truth of the sweeping coordinates slightly flexible by drawing a circle with radius 100 pixels around the coordinates. Every point in this circle is counted as a correct sweeping point.

4. Ground truth of the full shape mask of the target object. It is needed to check the accuracy of finding a target.

5. Ground truth of free objects that are occluding either the target or its occluders, and the possible grasping points of them. We are calculating the grasping point by finding first the center point of the object and drawing a circle around it. The radius of the circle is a quarter of the length of the object.

4.4.2 The Decision-maker Accuracy Tool

This tool helps us to check the accuracy of the decision-maker and the whole system. We don’t take in consideration the situation when the target object is not found by Mask R-CNN, because the robot will sweep in that case. We discard that kind of error in recognition because we want to know if the decision-maker is accurate. This tool gives us:

1. The accuracy of finding the target. This accuracy tells us how well the Mask R-CNN detects, classifies, and segments the target object in the image. It is counted by dividing the number of correctly found targets by the number of all the images i.e. the amount of all targets as there is one target in every image.
2. The accuracy of finding the object to pick up and its grasping point. The object must be the target or occluding the target, and it must be free to pick up. The grasping point is calculated for a suction hand but it could be calculated also for the gripping hand. The accuracy is counted by dividing the number of correctly found objects to pick by the number of correctly found targets (see step 1.).

3. The accuracy of deciding the correct sweeping direction. It is counted by dividing the number of correct decisions on sweeping direction by the number of sweeping actions.

4. The accuracy of finding the sweeping coordinates. The sweeping coordinates are two points between which the sweeping will occur. The accuracy of finding them is counted by dividing the number of correct sweeping coordinates by the number of sweeping actions.

<table>
<thead>
<tr>
<th>Finding the target accuracy</th>
<th>Finding the free object to pick and its grasping point accuracy</th>
<th>Sweeping direction accuracy</th>
<th>Sweeping coordinates accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>66.2 %</td>
<td>68.3 %</td>
<td>79.7 %</td>
<td>64.2 %</td>
</tr>
</tbody>
</table>

Table 2. The accuracy results of the decision-maker application depend on the Mask R-CNN results.

4.4.3 Evaluation results for Decision-maker

We generated 1000 images to test our decision-maker. Every image has around 15 objects from which one is the target. The target is the first object that we add when we synthesize the image and we add the rest of the objects into the image after that. In this way, we ensure that it is occluded most of the time. The decision-maker results rely on the object recognition results so the Decision-maker Accuracy Tool is actually evaluating our full system. The final
speed for the full system, the decision-maker with the Mask R-CNN, was one frame per two seconds by using NVIDIA's GeForce 1080 GPU and Intel i5 CPU.

In our experiment, the accuracy to find the target by the decision-maker using the Mask R-CNN model is 66.2 %. If the target is not found, the robot will sweep the scene and search for the target again. The sweeping motion thus gives a better probability to find the target. The accuracy for the correct sweeping direction is 79.7 % and the sweeping coordinates are 64.2 %. When the target object is found, the decision-maker will search for the first free occluding object to grasp. The accuracy for the decision-maker to find the correct object to grasp is 68.3 %, see table 2.

The decision-maker results depend on the Mask R-CNN predictions because it is using the predicted masks to make its decisions. When we check the intersections of the masks or add them to the crowded mask area, the undetected objects are the biggest reason for the errors. The undetected objects do not have masks and because of that, the decision-maker cannot take them into consideration. This might cause errors in creating the crowded mask area, the undetected object masks can make the predicted crowded mask area smaller than the ground truth, see figure 21. In finding the free object to grasp, the decision-maker needs to find a sequence of the objects occluding each other. If anyone of the objects in that sequence is undetected, the decision-maker will not choose the right free object to grasp, see figure 23. This sequence of objects accumulates the prediction errors of the objects.

Figure 21. This figure shows a sweeping error caused by undetected objects. The masks U (on the left) are undetected by Mask R-CNN and because of that part of the masks in the crowded mask area are missing (in the middle). The ground truth for the sweeping is on the right.
Figure 22. This figure shows how an undefined object U (on the left) makes the crowded mask area smaller from up (the predicted crowded mask area in the middle and the ground truth of it on the right) but still does not affect on the sweeping points or sweeping direction.

The error in the crowded mask area can cause an error in the sweeping direction accuracy and sweeping point accuracy as the crowded mask area is different from the ground truth crowded mask. On the other hand, this problem is not severe because the sweeping is still on one of the crowded mask areas, just not on the most crowded mask area. The sweeping is not random and not just in one direction, which is the goal in order to not make all the objects in one side of the area and to have reasonable sweeping motion. And if needed, it can be repeated. Also sometimes, the error in a crowded mask area does not cause an error for the sweeping direction or sweeping points. If the undetected object is a little up or down from the crowded mask area, it might not affect the sweeping motion because it is always horizontal, see figure 22. This together with the accumulated error in the accuracy of finding a free object to grasp cause the difference between the accuracy of the finding free object to grasp and the accuracy of the sweeping direction. Also, a small error in a crowded mask area might not affect the sweeping direction but it might still affect on estimating the sweeping coordinates. Thus the accuracy for sweeping direction is better than the coordinates for sweeping.
Figure 23. An undetected object U (on the left) causes a decision-maker error in finding the correct free object to grasp. The decision-maker proposes to pick up the bigger coca-cola can (the mask in the middle), while the correct pick is the undetected smaller coca-cola can (the ground truth mask in the right).
5 Conclusion

The task of this thesis was to handle occlusion in a robotic picking task. We solved the occlusion problem by training the state-of-the-art instance segmentation model Mask R-CNN with full shape annotation of the objects. The training with full shape annotated images simulates the learning process of a human when s/he learns the full shapes of objects. This knowledge of the full shapes is used to predict the full shape of an occluded object. The full shape annotation dataset was created by synthesizing images by a Dataset Generating Tool. The Dataset Statistical Tool was created to understand the features of the dataset and the Accuracy tool was created to check the accuracy of the Mask R-CNN after training.

The results of the Mask R-CNN trained with the full shape annotation dataset were compared with the results of the Mask R-CNN trained with the same dataset but with visible part annotation. The Mask R-CNN result dropped only a little bit. The segmentation accuracy dropped from the 89.4 % of the visible part annotation to 88.1 % of the full shape annotation. Knowing the occluded parts of the object, we can know which objects are occluded or occluding other objects by finding the intersections between the masks. By implementing the depth map to this scenario, we can conclude which one from the objects is on the top of the pile and can be picked up safely.

We created a decision-maker application that uses the results of Mask R-CNN together with a depth map and decides according to them about the robot’s grasping and sweeping actions. We created a depth map dataset for the decision-maker application to simulate a depth camera by a Dataset Generating Tool for Decision-maker. We also created an Accuracy tool to check the accuracy of our system.

We showed that Mask R-CNN can solve the occlusion problem when it is trained by the full shape annotation dataset. We showed that our decision-maker application can decide to sweep when needed and can find an object to pick to reach the occluded target. The accuracy for the correct sweeping direction reached 79.7 % and the accuracy to find a correct object to pick up reached 68.3%. The decision-maker application bases its decision on the Mask R-CNN
results. Because of that, the results of the decision-maker would improve if the Mask R-CNN predictions improve.

We tested our system on synthesized images, not on real images because of the difficulties to annotate real images with full shape annotation. When we created our synthesized image dataset, we used only geometric augmentation. The dataset could be developed by doing color augmentation to the images to simulate the real-world changes in lighting. After training the Mask R-CNN with this kind of new synthesized dataset it could be tested in the real scene.

We also didn’t use any constraints when we created our dataset. The dataset could be improved by adding the following two constraints for the size of occlusion for an object in the Dataset Generation Tool: 1. The visible part of the object must be more than 25 % from the object’s full size, as in [4]. 2. The visible part of the object must be more than 0.75 % of the image size. Using these improvements the recognition model will not be trained with a really small number of pixels and the recognition result will most likely improve.

We chose the short-term decision algorithm for the decision-maker because it seems to be more reliable. We didn’t test it or the long-term decision algorithm with a robot to see how they perform in a real situation. In future, the decision-maker application could be tested with the long-term decision algorithm and short-term decision algorithm to see how they perform in practice.
References


https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272
(cited: 10/8/2020).