Robotic Grasping from Supermarket Shelves using Visual Servoing

MSc. Robotics Thesis

Stijn Lafontaine

Delft University of Technology

TU Delft

Robotic Grasping from Supermarket Shelves using Visual Servoing

by

Stijn Lafontaine

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday October 25, 2023 at 10:00 AM.

Student number:	4908457	
Project duration:	April 1, 2023 – October 25, 2023	
Thesis committee:	Prof. dr. ir. M. Wisse,	TU Delft, supervisor
	Dr. ir. H. Caesar,	TU Delft, supervisor
	Ir. C. Salmi,	TU Delft, supervisor
	Dr. ir. N. Tömen	TU Delft

Cover: Image from AIRLab in Robohouse with the robot used in thesis

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

This thesis focuses on closed-loop product grasping from supermarket shelves. The case is studied where the robot is in front of a shelf in an Albert Heijn supermarket and is tasked to pick a desired product from that shelf. Enabling a robot to achieve the product-picking task, however, is challenging. While many other robotic picking methods are centered around table-top environments, the complex geometry of supermarket shelves presents a challenge in itself. Additionally, the Albert Heijn supermarket is a dynamic environment, where other agents can change the shelves, move products, and can introduce lighting changes. Where the table-top environment allows other methods to pick objects with an open-loop controller, approaching the supermarket environment with a closed-loop picking strategy can be beneficial in overcoming the challenges introduced by the dynamic environment. Therefore, this thesis proposes a product grasping pipeline, where the goal is to discover what combination and optimization of the required robotic skills results in a system that enables the robot to consistently and robustly perform the product-picking task. To close the loop, in-hand position-based visual servoing is used that enables the robot to account for detection mistakes as it picks a product. The three robotic skills that are required for the product-picking pipeline are product detection, product grasp pose estimation, and product tracking. Because of visual servoing, each of these robotic skills must run in real-time. The product detection is achieved by pre-training a YOLOv6 object detector on the SKU-110K dataset and fine-tuning it to the new Albert Heijn Supermarket dataset. The Albert Heijn Supermarket dataset is created to detect 36 products in the supermarket, where the challenges of distinguishing similar products and detecting relocated products are included. To enable detections during visual servoing, the distance, directions toward the shelf, and lighting are varied. The product grasp poses for the suction cup can be estimated using a plane fit on the estimated pointcloud for each product. The pointcloud of the product is estimated by randomly sampling the depth data from the product. Each product pose is then tracked through time via Kalman Filtering, to enable temporal reasoning about the products. Because of this, the grasp pose of the desired product can be refined as the manipulator moves toward the shelf. The proposed system achieved success rates of 90%-100% during experiments on a real robot with a suction cup gripper. While robust picking on a set of 36 products has been achieved, exploring a wider variety of product shapes with other robotic grippers is a compelling research direction to overcome the challenge of picking oddly shaped products. Furthermore, a separate few-shot-learning classifier for the product classification might be used to overcome the challenge of adding new products to the inventory or product re-branding. Next, considering other picking scenarios in the supermarket, like picking from hooks or refrigerated shelves with doors, is important for deployment as well. Finally, interacting with humans and reacting to human behavior during the picking process to ensure safety is another crucial challenge that must be overcome to move toward the integration and deployment of robotics in supermarkets.

Contents

1

Introduction

In an era characterized by rapid technological advancements and evolving consumer demands, the retail industry finds itself at the crossroads of innovation and efficiency. As supermarkets strive to meet the growing expectations of customers for seamless and timely order fulfillment, the integration of robotics has emerged as a compelling solution. This thesis delves into the implementation of a robotic system in a supermarket environment to streamline the process of order picking.

Specifically, the supermarkets that are considered in this thesis are the Albert Heijn supermarkets from the Ahold Delhaize company, with more than 895 different stores in the Netherlands [1]. The rapid increase of online shopping and delivery at home, accelerated by the recent competition on *'Flitsbezorging'*[1], has intensified the need for retailers like Albert Heijn to optimize their operations, especially in the critical aspect of order fulfillment. Robotics technology offers a promising alternative to the way supermarkets manage their inventory, retrieve products, and assemble orders.

This thesis will explicitly focus on the order-picking process. The case is studied where the robot is in front of a shelf in an Albert Heijn supermarket and is tasked to pick a product that is located on this shelf. The goal of this thesis is to describe and motivate the design decisions that were made to develop a product-grasping pipeline that enables the robot to fulfill the product-picking task. To achieve the goal of picking a desired product from a shelf, the robot must have multiple skills.

Firstly, the robot must be able to detect the products on the shelf with a camera. It must know what products it is looking at, called classification, and where the products are relative to itself, called localization. The process of locating and classifying products is called **product detection**.

Secondly, the robot must be able to grasp the products. To achieve a successful grasp the robot must determine where to grasp a product, further referenced as **product grasp pose estimation**. The grasp pose should consist of all 6 degrees of freedom for the robot to know how to approach the product for a successful pick.

Thirdly, to smoothly control the arm when executing the grasping plan and account for mistakes, a robotic controller must be implemented. To be able to react to changes in the environment and to account for mistakes in the product detection or the product grasp pose estimation, **visual servoing** is applied. Visual servoing is a type of closed-loop control that makes use of visual information as feedback. Specifically, the type of visual servoing that is implemented is position-based visual servoing (PBVS), which means that the estimated grasp poses of the products are used as feedback for the controller. Image-based visual servoing (IBVS), in contrast to PBVS, uses the image features as feedback directly. Because visual servoing is done, reasoning about detections and product poses through time becomes relevant to ensure that only correct detections are used, and to react to changes in the environment. This is called **product tracking**.

[1] 'Flitsbezorging' is a Dutch word for delivering groceries within 10 minutes after placing your order

It is important to note, however, that, while visual servoing makes the robot able to react to the environment and its changes, it also produces a difficult requirement for all skills: the operations of each skill must happen in real-time. Product detection, product grasp pose estimation, and product tracking must all happen in real-time. Real-time means that all calculations must be fast enough to be reactive to the surroundings.

Mastering each of these skills as a robot in the supermarket is vital to product-picking success. Section 1.1 describes what work is already done regarding the robotic skills and what can be learned to create a product grasping pipeline. Thereafter, Section 1.2 states the research question of this thesis specifically and sets out the challenges that must be overcome for each robotic skill. It also gives an overview of the entire robot and the modules that apply the discussed skills. Section 1.3 describes the robot that is used in the supermarket to grasp products from shelves.

1.1. Related Work

Through the years a lot of work has been done on both object detection and object grasp pose estimation, both important for the **product detection** and **product grasp pose estimation** skills respectively. Most work tries to implement a framework where both are done simultaneously with the use of deep learning.

The grasp detection methods that perform object grasp detection can be split between object-oriented grasp detectors and scene-oriented grasp detectors [42]. Scene-oriented grasp detectors are most interesting for the product grasping pipeline since they can detect grasps on objects in a scene instead of only on isolated objects, where only one object appears per image. Often, however, the scene-oriented methods do not consider specific object grasp detection, meaning that the classes of the products are not considered. In other words, the methods do not consider the goal of picking a specific desired object, so if one of these methods were to be used in the product grasping pipeline, a separate object detector must be implemented as well to determine the classes of the objects.

The methods that do consider the product classes perform grasp pose estimation in a table-top environment. This means that all products that must be picked are positioned on a flat platform. Supermarket shelves differ quite from this scenario because they have a complex geometry that can interfere with the grasp proposals for the objects.

Additionally, because the methods use deep learning, they require a dataset to be trained on. Such a dataset must include the products of Albert Heijn, labeled with the product classes, and, for grasp pose estimation, with 6 degrees of freedom (6DoF) grasp poses. A dataset of Albert Heijn products is not available and must be made. Labeling a dataset with grasp poses, however, is a difficult process, because the grasp poses depend on the used gripper, and the optimal grasp pose for a certain object is often disputed.

Therefore, because of these difficulties, the decision is made that the **product detection** and **product grasp pose estimation** are done separately. The dataset synthesis can therefore be strictly defined by labeling bounding boxes, which must surround the object, contrary to grasp poses that have a less strict definition. Additionally, the use of bounding box labels allows the use of object detectors that might be more robust and faster in detecting products than the end-to-end scene-oriented object grasp pose estimators.

Next, work on **visual servoing** for robotic manipulators has been done as well. Hutchinson et al. [17] provides an overview of the methods in the field of visual servoing and tracking, where an example is given where position-based visual servoing is used with a Kalman Filter [21] operating at 30 Hz. Wilson [47] proposed a similar method, where position-based visual servoing was used with an Extended Kalman Filter, operating at 60 Hz. Finally, Vincze et al. [39] claimed that the operation rate should at least be 50-100 Hz to achieve good dynamic responses from the system. Similarly, the visual servoing system by Espiau et al. [10] operates at 50 Hz. Together, these methods give insight into what operation rates are necessary to achieve a visual servoing system, and thus the real-time requirement for

visual servoing is likely between 30-100 Hz.

Finally, recent work on grasping products from supermarket shelves on a system level has been done by Bajrackarya et al. [3]. They propose an entire system for a supermarket order-picking robot. Their perception pipeline consists of five modules, yet their method does not implement visual servoing. Their first module implements a YOLOv5 detector [49] trained on the SKU-110K dataset [16]. The second module segments the product in the detected bounding box and overlays the segmentation with RGB-D data from a learned stereo module to obtain the product pointcloud. The third module uses metric learning to compare the detections with a database of products. The fourth module uses a PointNet [34] based network and the product pointcloud to obtain the grasp pose for the product. The fifth and final module uses a network that estimates how to extract the product (e.g. from a hook, shelf, or box).

1.2. Research Question

Taking all this into consideration, it is clear that for successful grasping in the supermarket, multiple robotic skills are necessary. The performance of one skill, however, often depends on the performance of another. Combining the skills and optimizing them together is therefore of great importance. This translates to the main research question of this thesis:

What implementation of robotic skills and combined optimization results in a grasping pipeline that is best in consistently and robustly grasping desired products from supermarket shelves in Albert Heijn?

Additionally, challenges lie in how each robotic skill is achieved separately, which are listed below:

1. Product detection requires a **dataset** to train on, which must include the products from the Albert Heijn supermarket. Such a dataset must contain enough data that represents the Albert Heijn supermarket to make the product detector succeed at detecting products after training.
2. To achieve the **product detection** skill, a suitable object detector has to be chosen and trained to detect the products in the Albert Heijn shelves at a rate of 30-100 Hz for visual servoing.
3. **Product grasp pose estimation** has to be done to enable the robot to pick the detected products from the shelves. Like the product detector, the product grasp pose estimation should run at 30-100Hz for visual servoing but also be accurate enough to make the product grasping pipeline succeed.
4. **Visual servoing** and **product tracking** must be done to keep track of products through time to account for product detection and product grasp pose estimation mistakes, and changes in the environment. Again, the product tracker must track the products accurately through time, and at a rate of 30-100 Hz for visual servoing.

Because of these different robotic skills and their challenge in design on their own, the product grasping pipeline is separated into modules that each describe how one robotic skill is achieved. An overview of the entire grasping pipeline is visualized in Figure 1.1. The modules that each implement a specific robotic skill in the picking process are visualized with the green blocks. The modules, while being separate, are optimized together to increase the picking performance of the robot. The dataset synthesis is also separately discussed due to its challenges. Details on what methods are used are in Sections 2, 3, 4, and 5. Finer details on the implementation of each module and their combination are stated in Section 6. Thereafter, Section 7 tests the performance of each module and their combination. Finally, Section 8 gives a conclusion on the thesis, while Section 9 describes what future work can be done regarding product grasping from supermarket shelves.

1.3. Hardware

The robot that is used in this task consists of a base that makes the robot able to move around and a Franka Emika Panda robot arm, equipped with a suction cup as an end-effector and a realsense D435i camera mounted on top of the end-effector. Since the robot does not have an onboard GPU, a laptop is used that is connected to the robot during operation and is equipped with an NVIDIA GeForce RTX

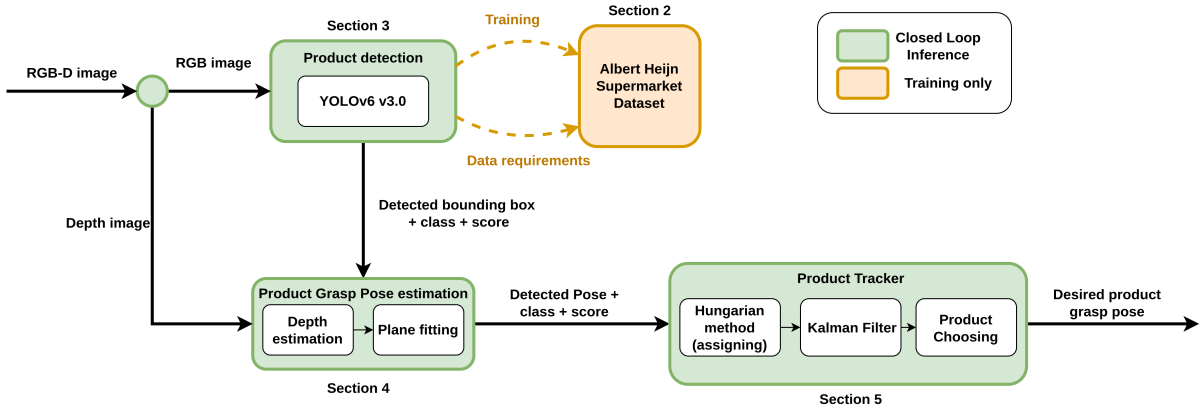

Figure 1.1: shows a visualization of the grasping pipeline. The RGB-D data from the realsense camera is converted into a grasping pose for the desired product. The 'Closed Loop Inference' (green) parts perform the picking sequence, estimating the desired product grasp pose with the use of visual servoing. The 'Training Only' part is executed during product detector training. Each green block represents a module that implements a specific robotic skill.

3080 GPU. During the picking process, the base of the robot is assumed not to move. The robot is visualized in Figure 1.2.

Figure 1.2: shows an image of the supermarket robot 'Albert', with the Franka Emika Panda robot arm, realsense D435i, and suction cup gripper.

2

Dataset

Section 1 concluded that a new dataset must be created to train an object detector to detect products from the shelves in the Albert Heijn supermarket. It is important to create a new dataset in such a manner to prevent pitfalls like overfitting, biased results, or learning shortcuts. This is achieved by creating a dataset that contains examples of the object in all scenarios that can occur during inference. So, the goal of a dataset in the context of object detection is to make the object detector succeed in the specified detection task.

So, to make an object detector succeed at the task of detecting products in the supermarket, a dataset is required that resembles the Albert Heijn supermarket and its products. The two requirements that must be met by the dataset are defined and explained in Section 2.1. The final proposed dataset, the Albert Heijn Supermarket dataset used to train the product detector, is discussed in more detail in Section 2.2, where it is shown what is done to meet the requirements. Also, since determining whether the dataset contains *enough* data is difficult to measure directly, other datasets that resemble supermarkets are explored that could be used to pre-train an object detector to try to increase its performance via transfer-learning. Why transfer learning could be of use is described in Section 2.3, as well as the use of the SKU-110K dataset [16] as the source domain.

2.1. Requirements

Two requirements must be met by the new dataset so that it can make an object detector succeed at the product detection task. They are stated below:

- The dataset must be **complete**, containing examples of all the products that the object detector must be able to detect.
- The dataset must be **diverse**, containing enough examples of each product to make the object detector detect each product in all occurring scenarios, like
  - different places on the shelf, surrounded by different products
  - different lighting falling on the product
  - different distances from the shelf
  - different directions towards the product

As the first requirement stated, the dataset must be **complete**, and so it must at least contain examples for each product that must be detected in the supermarket. Completeness, however, does not guarantee detection success. Not only must the object detector detect products on the shelves of the supermarket, it must preferably do so without making mistakes like classification errors, bounding box misplacements, not detecting products that are present, called False Negatives (FNs) or detecting products that are not present, called False Positives (FPs).

The dataset must enable the object detector to maximize correct predictions, known as True Positives (TPs), and minimize bounding box over- or underestimation and classification errors, that result in False Positives (FPs) and False Negatives (FNs). A perfect object detector places the detected bounding boxes exactly around each object of interest that appears in the image and always classifies the bounding box as the object that is inside of it. So, as stated by the second requirement, the new dataset must be **diverse** to enable the object detector to maximize TPs and minimize FPs and FNs.

Capturing all possible scenarios that can occur during inference is therefore important. In the context of the Albert Heijn supermarket, the dataset must contain images from different distances and directions, and with different backgrounds and surroundings. Furthermore, the Albert Heijn supermarket is a dynamic environment, where customers walk around and pick and place products on the go. This results in misplaced products and different lighting scenarios during run-time, which should all be included in the dataset, must the object detector succeed at detecting products robustly.

While the first requirement can easily be verified, the second requirement regarding diversity is not possible to verify directly. Verifying whether the dataset has *enough* examples and *enough* different scenarios cannot be done by looking at dataset numbers, but it can be measured indirectly by measuring the performance of the object detector in the grasping pipeline, and the influence of transfer learning on the performance of the object detector. The success of the object detector and the dataset are therefore coupled. If the dataset is diverse enough, transfer learning should not have an impact on the object detector's performance, because all required features for successful product detection can be learned using the proposed dataset. If an increase in performance is observed when transfer learning is applied, the proposed dataset can be improved by incorporating more data from different scenarios, because the object detector was able to learn features from the other dataset used as the source domain that it did not successfully learn when trained on just the proposed dataset. The object detector's performance with and without transfer learning, and thus whether the dataset is diverse enough to train the best-performing object detector, is tested in Section 7.1.2. The success of the object detector in the product grasping pipeline is tested in Section 7.3.1, where it is verified whether training with or without transfer learning is able to produce the best-performing product grasping pipeline.

2.2. Proposed Dataset

The proposed dataset, the Albert Heijn Supermarket dataset, is used to train the object detector that has to detect the products in the supermarket. The dataset is labeled by hand, where it is required to include all pixels related to the product within the bounding box, and the goal is to minimize the amount of environmental pixels (pixels unrelated to the product) in the bounding box. Two requirements are established that must be met by this dataset.

The first requirement states that the dataset must be **complete**, so all products that the object detector must be able to detect must be included in the dataset. According to [2], the Albert Heijn supermarket chain sells over 46000 products in total. Incorporating all products in one dataset is beyond the scope of this thesis, and therefore a selection of products is made. The selection of products in the Albert Heijn Supermarket dataset counts 36 different products. The products that are included vary in shape and size but are all rigid and non-porous so that they can be lifted by a suction cup gripper.

Detecting products in a supermarket, however, is challenging, because products often look almost identical. Each product type has a multitude of brands that each sell that type of product, resulting in similar products that are the same intrinsically but differ in brand. For example, the product detector should be able to tell the difference between a type of product from 'brand A' and that same type of product from 'brand B'. Furthermore, different products that are of the same brand often look alike as well, due to the similar labeling a brand uses to represent itself. Here the product detector should also be able to tell the difference between the products. The problem of similarity between classes is intrinsic to all Albert Heijn supermarkets, and therefore it is important to prove that the object detector can distinguish between similar products. So the similarity between products in the supermarket is taken into account by picking products that look similar because of their type or because of their brand. The

object detector must be able to tell these similar products apart from each other to succeed in picking the right products from the shelves in a supermarket. Some products that are similar to each other in the dataset are visualized in Figure 2.1.

Figure 2.1: shows examples of products that look similar. Telling the difference between similar products due to equal brand or equal type is important because it is intrinsic to the Albert Heijn supermarkets. The left two images show products that are of different types but of the same brand; The right two images show products that are of the same type but of different brands.

The second requirement of the dataset states that the dataset must be **diverse**, and thus must contain different examples of each product to make the object detector succeed in all occurring scenarios. Whether the dataset does contain enough examples is tested in Section 7.1.2 and 7.3.1. In total, the Albert Heijn supermarket dataset contains 1166 images of shelves with products, in which there are a total of 15848 examples of products. The number of specific product examples ranges from 146 to 1132, with an average of 440 examples per product. Equally important is how the number of examples per product (class) is balanced, referred to as the class balance. If a class does not occur often enough, an object detector can ignore that low-occurring class during training and still obtain a high score on the object detection performance metrics. For a perfect class balance in a dataset, each class should occur as often as all others. The class balance of the proposed dataset is shown in Figure 2.6. Here it can be seen that not all products occur as often as the others. Whether the class imbalance causes certain low-occurring products not to be detected is tested in Section 7.3.1.

Furthermore, the second requirement regarding dataset diversity stated that examples of different scenarios should occur in the dataset. The dataset contains images taken from different distances to the shelf, ranging from 20 cm to 100 cm. Furthermore, different directions toward the shelves are taken into account, with images taken at angles of at most 45°compared to the shelf, from above/below as well as from left/right. Dark and light lighting scenarios are incorporated, and products are shuffled to make sure that they occur in other locations on the shelf and are surrounded by other types of products. Examples of images in different scenarios are visualized in Figures 2.2, 2.3, 2.4, and 2.5.

To ensure that the object detector performs as well as possible, transfer learning can be used which might increase the performance of the object detector if the dataset does not contain enough examples of products or occurring scenarios. Section 2.3 explains the basic principles of transfer learning and argues why the SKU-110K dataset [16] can be used. The effect of transfer learning is tested in Section 7.1.2.

2.3. Transfer Learning

Transfer learning is known as the use of other datasets to pre-train an object detector so that it can learn features that are similar between the datasets. The goal of transfer learning is to transfer features from a source domain (pre-training dataset) to a target domain (Albert Heijn Supermarket dataset). This is a proven method to help increase the performance of machine learning methods [32] when the target domain does not have enough data to allow the object detector to learn to detect the objects in all occurring scenarios.

Figure 2.2: shows dataset examples of images at multiple distances from the shelf. Detecting images at various distances is important because the arm moves closer to the shelf during visual servoing. The left image is at a distance of 20 cm from the shelf; The middle image is at a distance of 60 cm from the shelf; The right image is at a distance of 100 cm from the shelf.

Figure 2.3: shows dataset examples of images from multiple directions toward the shelf. Important because the robot can be in any orientation compared to the shelf and should therefore detect the products in any orientation. The upper left image is an image from below at an angle of 15°; The upper right image is an image from above at an angle of 45°; The lower left image is an image from right at an angle of 45°; The lower right image is an image from left at an angle of 45°.

Figure 2.4: shows dataset examples of images with various lighting conditions. Important because lighting can vary due to the environment, locations of people walking by, or the position of the robot compared to the lights in the store. The left image is dark and the right image is light.

Figure 2.5: shows dataset examples of images where products are relocated. Important because in different stores the products can be ordered differently, and other agents can misplace products. The upper images show a shuffling example and the lower images show another shuffling example.

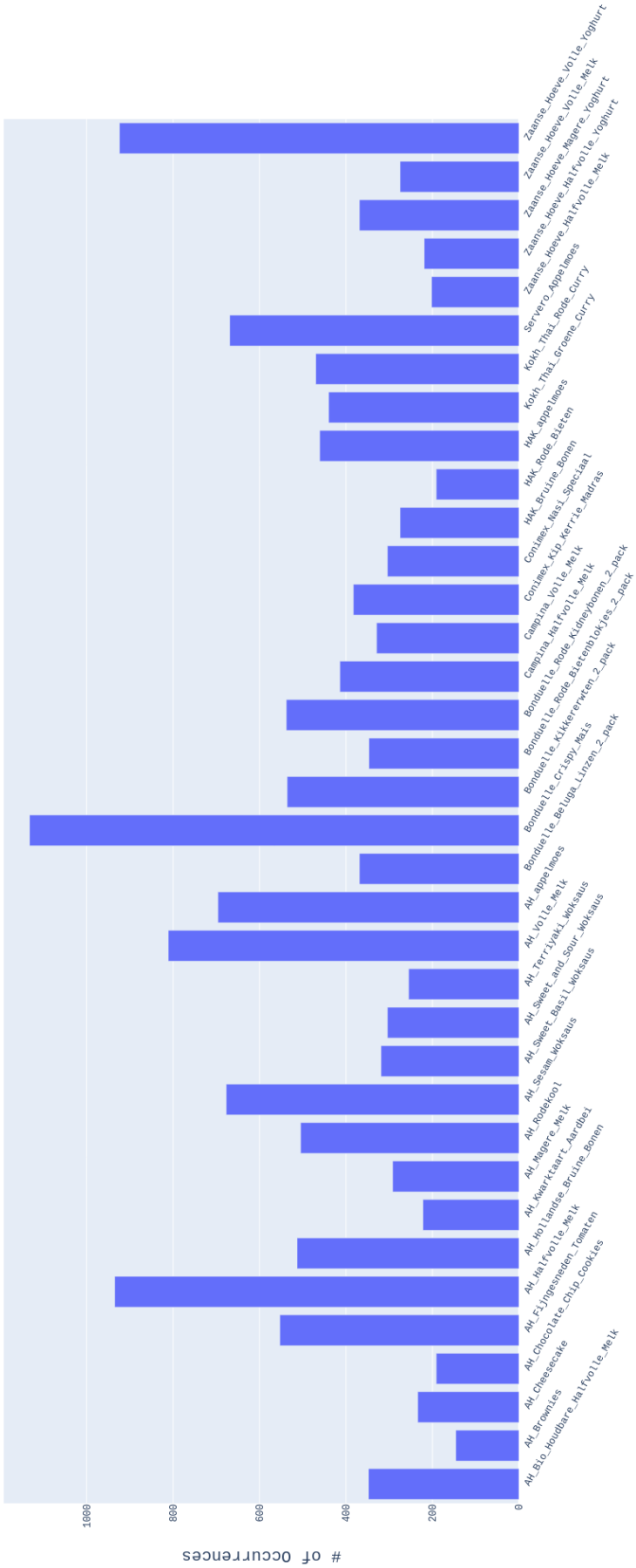

Figure 2.6: shows a visualization of the number of occurrences per class, known as class balance, in the Albert Heijn Supermarket dataset. The ideal dataset has an equal number of occurrences per class, to ensure that an object detector cannot ignore a certain class during training and achieve a high score on the evaluation metrics. Whether the imbalance has a negative influence on the product grasping pipeline is tested in Section 7.3.1.

In the case of product detection in the Albert Heijn, pre-training an object detector on an already existing supermarket dataset (source domain) might improve the detection performance in the Albert Heijn supermarket (target domain). Because verifying whether the proposed Albert Heijn Supermarket dataset contains enough examples of products in different scenarios is difficult, transfer learning could be used to either increase the performance of the object detector or verify that the proposed dataset has enough examples and thus the performance cannot be increased by transfer learning.

Dataset	Type	Environment	# Images	# Categories
SKU-110k Dataset [16]	**bbox**	**Shelf**	**11762**	**1**
RP2K [33]	cls	Shelf	500000	2388
Freiburg Groceries Dataset [20]	cls	Shelf	5021	25
GroZi-120 [30]	cls	Shelf	11870	120
GroZi-3.2k [14]	cls	Shelf	80	9030
Grocery Store Dataset [23]	cls	Shelf	5125	81
Cigarette Dataset [43]	bbox + cls	Shelf	354	10
RPC [46]	bbox + cls	Checkout	83739	200
D2S [12]	inst. seg.	Checkout	21000	60

Table 2.1: shows existing supermarket datasets that might be used as source domains for transfer learning; Due to the similar environment, used labels, and dataset size the SKU-110K dataset is the most viable dataset as a source domain for training an object detector.

Therefore, to successfully perform transfer learning, a viable source domain (dataset) must be found. Currently, there are already datasets available that contain supermarket products or resemble a retail store, which is summarized in Table 2.1. These datasets consider two different environments, namely products stacked on shelves [16, 33, 20, 30, 30, 14, 23, 43] and products at the checkout counter [46, 12]. Since the products must only be detected on supermarket shelves, the checkout counter datasets will not be of use for transfer learning because their environment is too different, making it unlikely that an object detector can learn any features that are of use for detecting products on the shelves in the Albert Heijn supermarket.

Furthermore, the labels for each of the datasets that consider products stacked on shelves differ. Some consider bounding boxes only [16, 43], with or without considering the classes of the products. Most datasets [33, 20, 30, 14, 23], however, only consider the class of the product and assume that only one class product is present per image. These datasets are also not of use, because they cannot be used to train an object detector.

So the SKU-110K and the Cigarette Dataset are the only datasets that are viable as source domains for transfer learning. The Cigarette Dataset contains 3954 images with 10 different classes, while the SKU-110K dataset contains 11.738 images with only one class. The SKU-110K dataset, however, contains a wider variety of products than the Cigarette Dataset, which only contains cigarettes. Also, the SKU-110K dataset is substantially larger than the Cigarette Dataset, containing more examples of different products and shelves. This could help an object detector to learn the features of a supermarket better, which might result in better generalization toward other supermarkets like the Albert Heijn supermarket. For that reason, the SKU-110K dataset is used as the transfer learning source domain. The effect of pre-training an object detector is studied in Sections 7.1.1, 7.1.2, and 7.3.1.

3

Product Detection

In recent years the research on data-driven object detection has grown massively, both due to the advancements in computer capabilities and due to the vast amounts of available data [51]. Deep learning-based methods are popular nowadays due to their accurate predictions and available resources. Because of this, using a deep-learning-based object detector can be an effective way to build a product detector as described in Section 1.

There exist numerous object detectors that all have their properties and implement other methods to achieve the goal of accurately detecting objects in their field of operation. To make the product grasping pipeline as successful as possible, the used object detector must be able to predict bounding boxes for each product in the supermarket environment and classify each bounding box correctly, as stated in Section 2. The goal of an object detector in the grasping pipeline is to do this as accurately as possible, with as few mistakes as possible.

Moreover, due to the use of visual servoing as stated in Section 1, the object detector should be able to detect the products on the shelf in real-time. Real-time means that the detections must occur at such a rate that the robot can be reactive in the supermarket environment. Other visual servoing methods [17, 47, 39, 10] operate at 30-100 Hz. The definition of real-time, however, heavily depends on the situation and might be different in the supermarket. Section 7.3.4 explores what operation rate is necessary for successful product grasping. The real-time detections should be achieved on a laptop with hardware that is described in Section 1.3.

Taking the real-time constraint of 30-100 Hz and the desired high performance in the supermarket into account, Section 3.1 sets out the available data-driven object detectors and argues why the YOLO models are considered the best option for the supermarket product detector. Section 3.2 describes the mean Average Precision (mAP) metric and explains why it is a relevant performance metric for product detection in the supermarket. Section 3.3 compares the available YOLO models and reasons why the YOLOv6 V3.0 models are used as product detectors.

3.1. Object Detectors

Through the years, more accurate and faster object detectors have been proposed. New methods were developed that dealt with learning better features, mitigating the drawbacks of training on unbalanced datasets, and relating pixels and image sections to each other.

The first methods that managed to achieve object detection at reasonable operation rates were Fast R-CNN [15] and Faster R-CNN [38]. These are two-stage detectors, where one stage proposes bounding boxes and the other stage tries to classify them. In Faster R-CNN, the use of the bounding box proposal networks, known as Region Proposal Networks (RPN), made them faster than the previously used sliding window approaches. Thereafter, to decrease the inference time of these first methods, the

Single Shot Multibox Detector (SSD) [29] and You Only Look Once (YOLO) [37] were proposed. These methods combined the two stages into one and managed to increase the operation rate significantly. After this, RetinaNet [27] and EfficientDet [41] worked on increasing the accuracy of object detectors, by introducing Focal Loss and Bi-directional Feature Pyramid Networks (BiFPN) respectively. Focal Loss enabled the networks to be less reliant on perfectly balanced datasets, while Feature Pyramid Networks enabled the networks to become scale-invariant. Next, after the success of transformers in the field of Natural Language Processing (NLP), DETR [7] and ViT [9] were proposed. These large models were able to relate pixels and sections of the image to each other even better, which resulted in a better understanding of the entire scene. Vision transformers therefore brought the object detection field forward in terms of model accuracy and precision.

Currently, object detectors that use transformers like DETR [7] and ViT [9] are becoming increasingly popular due to their high performance in scene understanding. They, however, require very large datasets[1] to train on, which are not available for the Albert Heijn supermarket. Additionally, transformer models are often large and therefore require GPUs with large amounts of memory that are not always available on robots and laptops. The size of the transformer models and the fact that they are required to train on very large datasets makes them ineligible as product detectors with the current constraints in terms of hardware and available data.

The latest YOLO models, however, are gaining popularity as well. This is, contrary to the transformer-based models, due to their high-speed detections. Visual servoing requires real-time detections at 30-100 Hz, as stated in Section 1, to ensure reactivity. According to Li et al. [25], the YOLO models operate at 17-779 Hz on an NVIDIA Tesla T4 GPU, making most models viable for visual servoing.

For that reason, the YOLO models are explored as potential product detectors in the supermarket. These models seem most promising in ensuring that visual servoing can be achieved, and, due to the latest iterations, seem to be able to do so with considerable accuracy as well. Section 3.3 describes the YOLO models in more detail and explains the final choice of using YOLOv6 v3.0 based on the metrics described in Section 3.2.

3.2. Performance Metrics

The goal of an object detector is to propose bounding boxes perfectly around the objects of interest and always classify them as the object inside. To measure how well an object detector performs, metrics are used that describe how well a bounding box is placed and whether it was classified correctly. The correct placement of a bounding box in combination with the bounding box being classified correctly is called a True Positive (TP). The misplacement of bounding boxes and wrong classifications are undesired. Such wrong detections are referred to as False Positives (FPs). Not detecting objects that are present in the image is also undesired and referred to as False Negatives (FNs). The best-performing object detectors maximize TPs and minimize FPs and FNs.

To obtain metrics like TPs, FPs, and FNs, the (correct) placement of bounding boxes must be measured. A metric used to measure bounding box placement is the Intersection over Union (IoU). The IoU is a ratio that describes how much overlap there is between the predicted bounding box and the ground truth bounding box. Equation 3.1 describes the IoU.

$$IoU = \frac{area\_of\_overlap}{area\_of\_union} \tag{3.1}$$

The IoU is a ratio, and therefore a threshold between 0 and 1 must be set that determines whether a bounding box is predicted correctly or not to obtain metrics like TPs, FPs, and FNs. This threshold is further referenced as the IoU threshold. An IoU threshold = 0.5 means that for a bounding box to be considered as correctly placed, the total overlap between the predicted and ground-truth bounding box must be at least 50% of the total area of the union of the predicted and ground-truth bounding box.

[1] ViT is trained on ImageNet (>1M images), ImageNet21k (>14M images), and JFT (>303M images)

Additionally, the classification of each detection has a score associated with it, ranging from 0 to 1. This score is a measure of how confident the object detector is about its predicted class. For example, the object detector is more certain about the detection with a score of 0.75 than about a detection with a score of 0.25. Similar to the IoU threshold, a confidence threshold is used to filter the detections based on the quality of the detected classes. Hence, the confidence threshold influences the TPs, FPs, and FNs as well. A higher threshold filters away more FPs, but also filters away TPs and thus increases the FNs. A lower confidence threshold leads to a decrease in FNs but increases the FPs. Two metrics are used to gain insight into this trade-off: precision and recall. Precision measures the ratio of predicted positives, as described in Equation 3.2.

$$precision = \frac{TPs}{TPs + FPs} \tag{3.2}$$

The closer the precision metric is to 1, the fewer FPs occur, especially relevant when FPs are undesired, and thus the cost of wrongly predicting an object is high. The recall metric measures the ratio of correctly detected objects, as described in Equation 3.3.

$$recall = \frac{TPs}{TPs + FNs} \tag{3.3}$$

The closer the recall metric is to 1, the fewer FNs occur, which is especially relevant when FNs are undesired, and thus the cost of missing an object is high. Both precision and recall depend on the confidence threshold, and as the confidence threshold is changed one is increased and the other is decreased. Precision and recall can therefore be combined in one curve, which depends on the confidence threshold. Integrating this curve can give insight into how well the object detector performs overall, reducing the dependence on the confidence threshold. This measure is known as the Average Precision (AP), shown in equation 3.4.

$$AP = \int PR(th)\, dth \tag{3.4}$$

where $PR(th)$ is the precision-recall curve that depends on th, the confidence threshold. The AP captures the performance of the object detector in detecting one single class because TPs, FPs, and FNs are calculated per class. So to gain insight into the performance of the object detector on all classes, the precision-recall curves for each class must be determined and integrated. Combining all these APs leads to a final metric that captures the performance of an object detector, called mean Average Precision (mAP), obtained by taking the average of all APs. The mAP metric is stated in equation 3.5.

$$mAP = \frac{1}{N} \sum_{n=0}^{N} AP_n \tag{3.5}$$

where N is the total number of classes. A higher mAP means that the detector is on average better at detecting all the objects in the scene. The mAP metric is used to compare the performance of the YOLO models in Section 3.3 and to compare the performance of the trained models in Section 7.1.

3.3. YOLO models

The goal of the YOLO models has been to achieve high accuracy on benchmark datasets like COCO [28] while maintaining the lowest inference time in the field of object detection. A summary of the most well-known YOLO models is visualized in Table 3.1. The (mean) Average Precision of YOLO models on the COCO dataset [27] has increased through the iterations, which is shown in Figure 3.1 from [25].

The most recent YOLO iterations, YOLOv7, YOLOv6, YOLOv8, and YOLOv6 V3.0, are also the best performing YOLO models. YOLOv6 V3.0 is an improvement of YOLOv6 by the same authors, and will from now on be referenced as YOLOv6. All these YOLO models come in various sizes (nano (N), small, (S), medium (M), large (L)), which allows a more precise trade-off between speed and accuracy. In Figure 3.1 it can be seen that the YOLOv6 models (N,S,M,L) perform slightly faster than

the YOLOv8 (N,S,M,L) models while maintaining the same (mean) Average Precision ((m)AP)[2] on the COCO dataset [28].

The YOLOv6 models, however, also have two larger models, namely the medium (M6) and large (L6) expanded YOLOv6 models. These are more accurate but also slower than the other YOLOv6 (N,M,S,L) models. In terms of (m)AP and speed, the L6 model outperforms the previously state-of-the-art performance by the largest YOLOv7-E6E model, which was the largest and slowest available YOLO model.

The YOLOv6-L6 [25] and YOLOv7-E6E [44] models, however, have 140.4M and 151.7M parameters respectively and therefore require a larger GPU than the smaller models like the YOLOv6-N and YOLOv6-L models, which have 4.7M and 59.6M parameters respectively. The smaller models can run on a smaller GPU, often present in a laptop.

Given the high speed and hardware constraint, the nano (N), small (S), medium (M), and large (L) models of YOLOv6 seem to be the most promising models to train a product detector. These models are most likely to make sure that visual servoing can be achieved.

Method	Year	Author
YOLO [37]	2016	Redmon et al.
YOLOv2 [35]	2016	Redmon and Farhadi
YOLOv3 [36]	2018	Redmon and Farhadi
YOLOv4 [6]	2020	Bochkovskiy et al.
YOLOv5 [49]	2020	Jocher et al.
YOLOX [13]	2021	Ge et al.
PPYOLOE [48]	2022	Xu et al.
YOLOv7 [44]	2022	Wang et al.
YOLOv6 [26]	2022	Li et al.
YOLOv8 [18]	2023	Jocher et al.
YOLOv6 V3.0 [25]	2023	Li et al

Table 3.1: shows the history of the YOLO models, each model tried to improve the accuracy of the previous model while maintaining a comparable inference time.

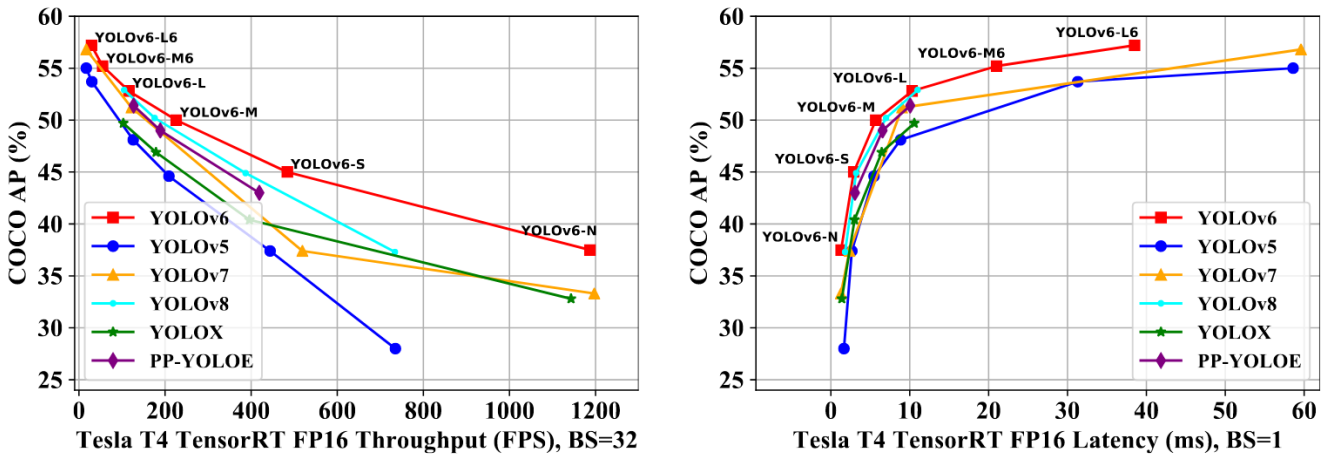

Figure 3.1: shows a comparison of best-performing YOLO models from [25]. The left image shows the (mean) Average Precision (AP) on the COCO dataset [28] vs. the achieved Frames Per Second (FPS) on a Tesla T4 GPU and a batch size (BS) of 32 (so 32 images are evaluated in parallel). The right image shows the same (m)AP, only then versus the inference time in milliseconds (ms) with a batch size (BS) of 1 (so 1 image per evaluation). Note that YOLOv6 means YOLOv6 V3.0

[2]The COCO authors refer to mean Average Precision (mAP) as Average Precision (AP)

4

Product Grasp Pose Estimation

While detecting products in an image in real-time is a first step towards robotic product grasping in the supermarket, a bounding box is not enough for a robot to pick a product from a shelf. Each product requires to be placed in the world relative to the robot to make it able to pick them up, effectively estimating a grasp pose for each product on the shelf. It is especially relevant that to estimate a grasp pose it is required that all 6 Degrees of Freedom (6DoF) of the robot arm are estimated to pick the product. Therefore, the goal of the pose estimation in the grasping pipeline is to convert the bounding boxes for each product given by the object detector to 6DoF grasp poses.

Additionally, the ideal grasp pose heavily depends on the gripper used to pick the product. Since a suction cup gripper is used, the best grasping location can often be found at the center of the front of the product, normal to its surface. Using this assumption, the grasp pose of each product is directly linked to the pose of each product, and so the grasp pose estimation and product pose estimation can be done together. To obtain the product poses, the bounding boxes of the product detector in combination with the depth data from the realsense camera are used. These product poses consist of two parts, namely a position (x, y, z) and an orientation (θ, ϕ, ψ). Both are estimated separately. The entire grasp pose estimation module is visualized in Figure 4.1.

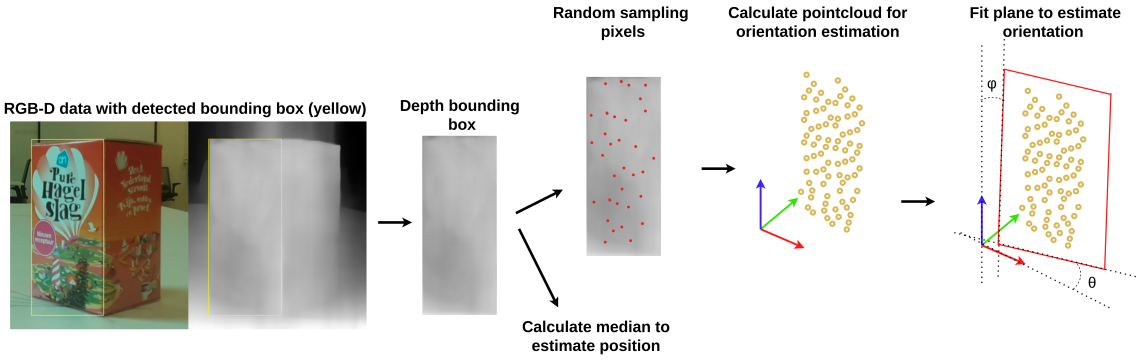

Figure 4.1: visualization of the grasp pose estimation. The detection from the RGB image is used to obtain the depth data of the product, which is then used for the position and orientation estimation. The orientation estimation is done by calculating the pointcloud of the product, which is done by randomly sampling from the depth data to speed up the calculation.

4.1. Position Estimation

To obtain the position of each product, the distance relative to the camera is estimated by using the median value of the depth bounding box. The camera pinhole model is then used to estimate the Cartesian coordinate of a pixel in the image. The equation that shows how a Cartesian coordinate is

converted into pixel coordinates is shown in Equation 4.3:

$$\begin{bmatrix} s*u \\ s*v \\ s \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.1}$$

where u and v are the pixel coordinates, s is a scaling factor, K is the camera intrinsic matrix and x, y, and z are the position of the pixels in the 3D world. The camera intrinsic matrix is shown in Equation 4.2.

$$K = \begin{bmatrix} f_x & 0 & o_x \\ a & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where f_x and f_y are the focal lengths and o_x and o_y are the optimal center offsets. The position of the product is estimated by using the center pixels of the bounding box. In combination with K, all variables are known except s, which can be estimated since z is assumed to be equal to the median distance of the bounding box. Converting pixels and depth estimations to Cartesian coordinates is shown in Equation 4.3.

$$s \begin{bmatrix} x \\ y \\ z \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.3}$$

4.2. Orientation Estimation

The orientation of each product is estimated by calculating the pointcloud that is associated with each product using the depth bounding boxes and the camera pinhole model, which uses the same equation as for the position estimation for every pixel in the bounding box, namely Equation 4.3. Because calculating a pointcloud can be an expensive operation, randomly sampling pixels from the depth bounding box is done, of which the loss of accuracy and speed increase are quantified in Section 7.2. After calculating the pointcloud, it is filtered with a band-pass filter on outliers based on the prior knowledge of product sizes. Since only the front of the products is available and most products have a flat surface, the θ and ϕ angles can be estimated by fitting a plane on the pointcloud. The third angle, ψ, is of no use to estimate because a round suction cup gripper is used, and is therefore not estimated. The resulting orientation estimation is thus $[\theta, \phi, 0]$.

5

Product Tracking

While detecting products and their poses relative to the robot in real-time is a first step towards robust robotic product grasping in the supermarket, without using the information of already seen detections during the picking sequence, the closed-loop strategy is not fully exploited. Due to the use of visual servoing, the robot must be able to relate objects to each other through time. Such temporal reasoning is essential in enabling the robot to make choices based on all available information of the past and present. Temporal reasoning is achieved by using a product tracker that keeps track of all detected products. The goal of the tracker is to keep track of the seen products over time and improve the accuracy of their positions, orientations, and estimated classes. By refining the product poses and classifications, the grasp location is also refined through time, increasing the chances of a successful pick. Moreover, using product tracking can help the robot become less affected by detection errors. Tracking is therefore important because temporal reasoning enables the robot to spot these errors and ignore them. When classification errors and positioning errors have a lower impact on the entire system, the robot can plan its picking strategy according to more reliable information.

To track all products through time, three major challenges must be overcome. The first challenge lies in how a single product is tracked through time, which is discussed in Section 5.1. Then, Section 5.2 discusses how the single object tracker can be used for each product in the scene so that all products can be tracked through time in a multi-object tracker. Finally, Section 5.3 explains how two methods can be used to overcome the challenge of choosing a product to pick from a set of available desired products.

5.1. Single-object tracker

The goal of the single-object tracker is to keep track of the classification, score, and pose of the product. The considered method for pose tracking is the Kalman Filter [21]. The Kalman Filter uses a linear state space model to estimate the future positions of the tracked objects. This means that the motions of the objects are assumed to be linear. After the prediction step, the predicted states of these objects are updated using the measurements to make the tracking more precise and account for prediction mistakes. Gaussian distributed noise is used to account for both the mistakes by the linear model and the measurements. This leads to another assumption of the method, namely that the mistakes by the linear model and measurements follow a Gaussian distribution.

It is important to note that, as stated in Section 1, the base of the robot does not move, and that, without intervention by other agents during the picking sequence, the products on the shelf do no not move as well. Combining these, if the product's poses are tracked relative to the non-moving base of the robot, the products do not move even if the arm moves around during the picking sequence. This allows the use of a simple linear prediction model, and therefore also the use of the Kalman Filter as a tracker, assuming that the measurement and model mistakes follow a Gaussian distribution. The linear state space model used to predict the poses of the products is stated in Equations 5.1, 5.2, 5.3, 5.4, and 5.5.

$$\overline{x} = [x, y, z, \theta, \phi, \psi] \tag{5.1}$$

$$\overline{x}[i+1] = A\overline{x}[i] + Bu[i] \tag{5.2}$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ due to } u = 0 \tag{5.4}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

The predict and update functions of the Kalman filter estimate the future poses of the products and update these predictions based on the new detections, respectively. The prediction functions are stated in Equations 5.6 and 5.7, while the update functions are stated in 5.9, and 5.10. Equation 5.8 states how the Kalman gain is derived.

$$\hat{\overline{x}} = A\overline{x} \tag{5.6}$$

$$\hat{\Sigma} = A\Sigma A^\intercal + Q \tag{5.7}$$

$$K = \hat{\Sigma} H^\intercal (H\hat{\Sigma}H^\intercal + R)^{-1} \tag{5.8}$$

$$\overline{x} = \hat{\overline{x}} + K(z - (H\hat{\overline{x}})) \tag{5.9}$$

$$\Sigma = (I - KH)\hat{\Sigma} \tag{5.10}$$

with $z =$ detection, $R =$ measurement variance and $Q =$ process variance

To track the classification and the score of the product, the single-object tracker counts the number of times a measurement contains a certain class and computes the mean score of each of the classes that have been measured. The classification that has been measured most often is considered to be the classification of the tracked product. The score of the product is the mean score that is associated with the most occurring class. Section 6.5.3 provides details on the implementation of the classification and score tracking.

5.2. Multi-object tracker

Tracking one object through time is done with the use of a Kalman Filter. Tracking multiple objects through time therefore requires the use of multiple Kalman Filters, one for each object. Each time step, all detected poses must be assigned to a Kalman Filter that tracks them through time. The assignment must be correct because the assigned pose is used in the Kalman Filter as a measurement for the update function.

The Hungarian algorithm [24] is used to assign the detected poses to Kalman filters. The costs in the Hungarian algorithm determine how the detections are assigned to the Kalman Filters. The distance between the positions of the detections and the predicted state of the Kalman Filters is used as the cost for assignment. If a detection is not assigned to a Kalman Filter, a new filter is created to keep track of that detection and the following detections. If a Kalman Filter does not receive an assigned detection, the update function is not called. If a Kalman Filter does not get assigned a new detection for multiple consecutive time steps, it gets removed from the multi-object tracker and therefore this object is no longer kept track of. Further details on the implementation of the multi-object tracker are in Section 6.5.

5.3. Product Choosing

Product choosing is required to grasp a desired product from a supermarket shelf where it is likely that multiple desired products reside. Choosing what product to pick is done in two ways, which are tested in Section 7.3.3. The first method selects the single-object track, described in Section 5.1, that has the most measurements that classify this track as the desired product class. The second method selects the single-object track that is classified as the desired product class and has the highest mean score. The implementations of both methods are discussed in Section 6.5.4

6

Implementation

6.1. Pipeline

To integrate all modules in the robot's system, three ROS nodes are made that implement the modules discussed in the previous sections. These three nodes work together with two other already implemented nodes, namely the realsense camera node and the Optimization Fabrics [40] node that is used to generate a collision-free trajectory to the desired grasping pose. The integration in ROS is visualized in Figure 6.1.

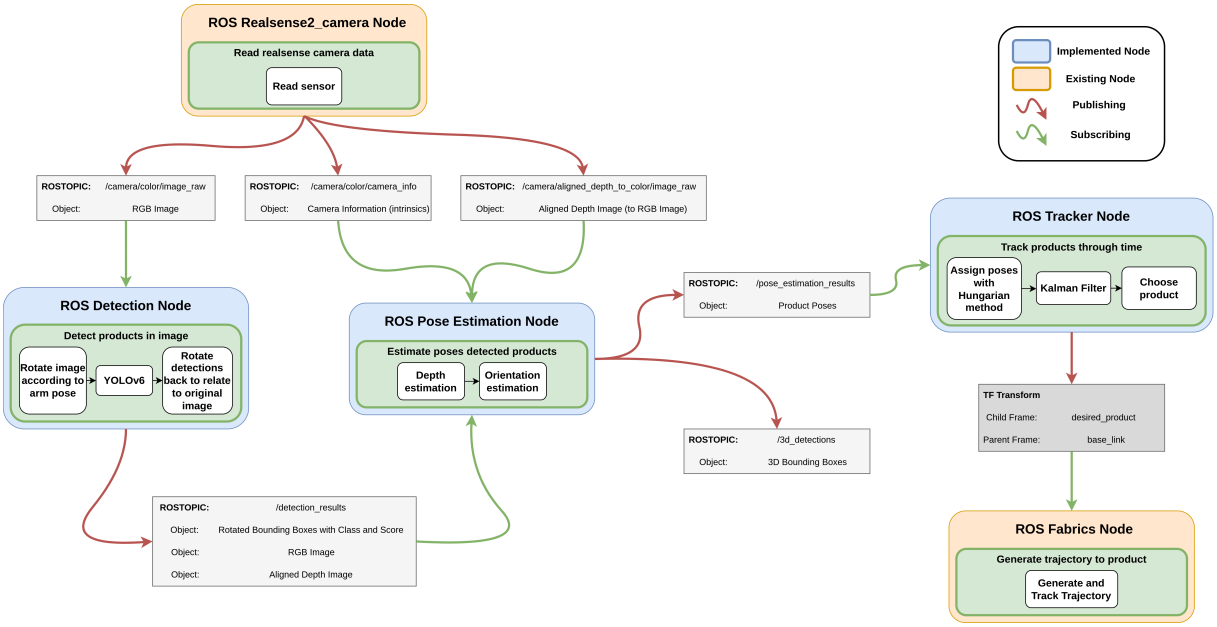

Figure 6.1: shows the ROS pipeline, visualizing the ROS nodes and the topics they use to communicate with each other. The blue ROS nodes are proposed by this thesis, the orange ROS nodes are existing nodes. The grey boxes represent ROS topics that are used to communicate between nodes. Red and green arrows indicate outgoing and incoming messages, respectively.

6.2. Dataset

The proposed Albert Heijn Supermarket dataset is split into three separate sets: a train, a validation, and a test set containing 816, 230, and 120 images, respectively. The splitting is done randomly and the splits are checked manually on whether the product classes are evenly distributed. These splits are important for the successful training of an object detector. The train and validation splits are used to train the model and to validate how well it works on unseen data. The validation set can be used

to detect overfitting during training and to tune the object detection model's hyperparameters to try to increase its performance. The test set is used after training as a final test, to test the models on how well each of their tunings works on unseen data.

The SKU-110K dataset that is used to train the object detector is augmented because the dataset contains images at a fixed distance from the shelf and the object detector must be able to detect images at various distances from the shelf. To simulate the camera being at different distances from the shelves, cropping and resizing on the images were done. The resulting dataset is referred to as the SKU-110K-VS dataset. Examples from both the SKU-110K and SKU-110K-VS dataset are shown in Figure 6.2.

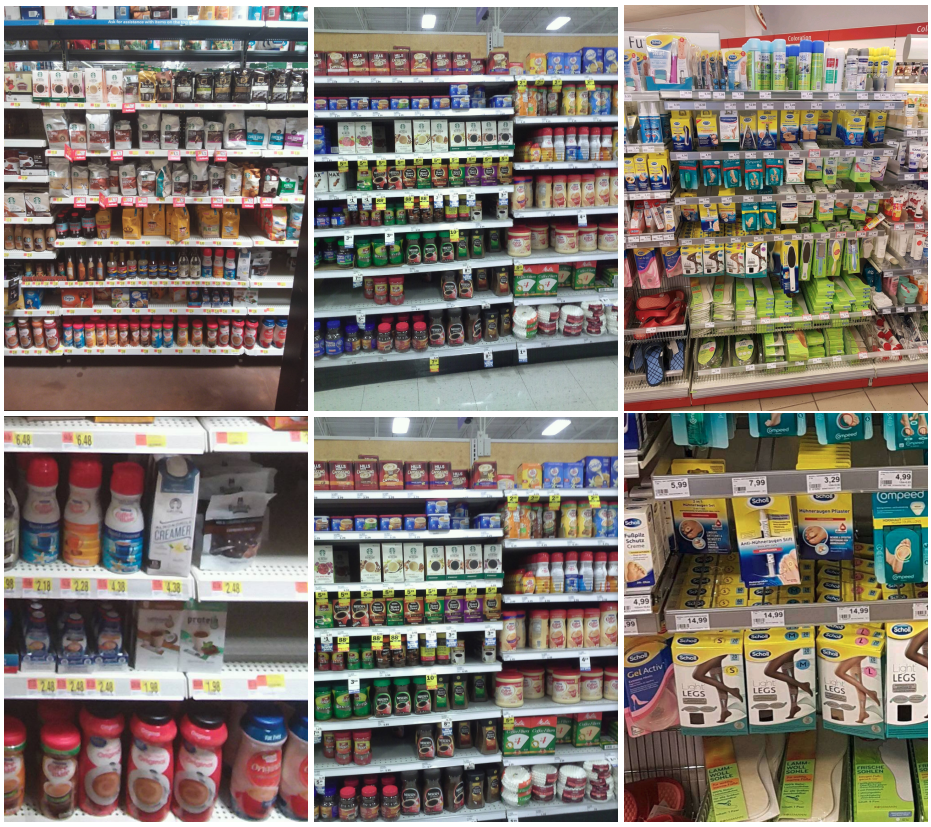

Figure 6.2: shows examples from the SKU-110K dataset and the SKU-110K-VS dataset. The top images are from the original SKU-110K dataset, while the bottom images are the same images, but then from the SKU-110K-VS dataset, augmented to have images from various distances to the shelf.

6.3. Detection

The YOLOv6 models [25], chosen as the most viable object detectors for the product detection task, must be trained on the Albert Heijn Supermarket dataset to detect products. The details regarding the training and pre-training on the SKU-110K and SKU-110K-VS datasets are discussed in Section 6.3.1.

Moreover, during inference, optimization steps are undertaken and fine-tuning is done to optimize the detection performance during the picking sequence. The details regarding these steps are stated in Section 6.3.2.

6.3.1. Training Details

Many hyperparameters must be tuned when training object detectors. Each model is trained for 200 epochs with a patience of 10 epochs, which means that training is stopped early if no improvement has been observed on the validation set for 10 epochs or more. Mosaic data augmentation is used, to

make detections less dependent on context [6]. Other data augmentation includes color adjustment by adjusting HSV values, left-right image flipping, and translating images. Adam [22] is used as the optimizer. Training is done on a computer with an NVIDIA GeForce RTX 2080Ti GPU with a batch size of 8.

Also, to transfer the models from the source (SKU-110K or SKU-110K-VS) domain to the target (Albert Heijn Supermarket) domain, fine-tuning is done after training on the SKU-110K(-VS) dataset. Fine-tuning is done either by training with frozen layers or without frozen layers. To determine the best transfer method, the number of frozen layers during training is varied between 3, 5, 7, and 9 frozen layers. The results are shown in Section 7.1.2.

6.3.2. Inference Details

To ensure minimum reliability of the detection results, non-maximum Suppression (NMS) is used after inference to consider only the highest-scoring detections and suppress the other proposed products by looking at how much overlap there is between proposals. Class-agnostic NMS and intra-class NMS are both implemented and tested in Section 7.1.3. Class-agnostic NMS suppresses the bounding boxes only based on their overlap, independent of the associated class with the bounding box. Intra-class NMS only suppresses bounding boxes that have the same class and more overlap than the threshold. Section 7.1.3 shows the effect of both non-maximum suppression methods on mAP on the Albert Heijn Supermarket test set, while Section 7.3 shows the effect of both methods on the grasp success. A visualization of the use of NMS is shown in Figure 6.3.

Figure 6.3: shows the effect of Non-Maximum Suppression on product detection during Inference. The left image shows an IoU threshold = 0.1, the middle image shows an IoU threshold = 0.7, and the right image shows an IoU threshold = 0.99. From right to left it can be seen that fewer proposed bounding boxes remain.

Furthermore, to maintain a high detection performance while moving to the shelf, rotation compensation of the robot arm is applied to the images of the camera. With this, the orientation of the products in the image stays aligned with the bounding boxes proposed by the object detector. Tests are done in Section 7.3 to verify the use of rotation compensation in the product grasping pipeline. Figure 6.4 shows the effect of rotation compensation during inference.

The workings of the ROS node that implements the product detection node are shown in Algorithm 1. The node implements the YOLOv6 detector with non-maximum suppression, along with rotation compensation for the Franka Emika Panda arm on the images.

Algorithm 1 Detection ROS node

1: **if** realsense camera image available **then**
2: Rotate image according to transform of robot arm to robot base
3: Predict bounding boxes for rotated image with NMS
4: Rotate bounding boxes according to inverted transform of robot arm to robot base
5: Publish resulting bounding boxes, classification, and score
6: **end if**

Figure 6.4: shows the effect of rotation compensation on product detection during Inference. The left images are not compensated, and the right images are. The upper images show a rotation of approximately 30°, while the lower images show a rotation of approximately 45°. Between left and right it can be seen that some products do not get detected at all on the left and that those that get detected have overestimated bounding boxes due to the rotation. Rotation compensation mitigates both problems.

6.4. Pose estimation

The inner workings of the ROS node that implements the pose estimation for each detection are shown in Algorithm 2. This node also performs the pose transformation from the camera frame to the robot base frame necessary for the tracker, as mentioned in Section 5.1.

Algorithm 2 Pose Estimation ROS node

```
if detection data available then
    for all detection in detections do
        Estimate distance detection with median of depth bounding box
        Estimate position with camera pinhole model, bounding box center, and median of depth data
        Randomly sample depth data for pointcloud calculation
        Calculate pointcloud of detected product from sampled points
        Filter pointcloud on outliers via band-pass
        Estimate orientation product with pointcloud by using plane fit
        Transform the estimated pose from camera frame to base frame
        Publish resulting transformed poses, classifications, and scores
    end for
end if
```

6.5. Multi-Product Tracker

The detected products are assigned to a track using the Hungarian Algorithm [24] that uses the distances between poses of the tracks and the detections as costs. Details on how the assignment is done are stated in Section 6.5.1. Then for each track, a Kalman Filter [21], as described in Section 5, is used to track the pose of the product through time. The Kalman Filter, however, requires the variances that describe the Gaussian noise of the model and measurements. How these are determined is explained in Section 6.5.2. Next, Section 6.5.3 describes how the classifications and scores of each track are tracked. Finally, a tracked product must be chosen from the set of tracked products that are of the desired product class. Section 6.5.4 explains the details of the product-choosing implementation. The

ROS node that implements the product tracker is shown in Algorithm 3.

Algorithm 3 Tracker ROS node

```
1: if product pose data available then
2:     Update Multi-object tracker with all detected products
3:     Choose a product to pick
4:     if Chosen Product != None then
5:         Publish grasp location
6:     end if
7: end if
```

6.5.1. Detection assignment

To assign the detection to the tracks, the Hungarian algorithm [24] is used. The algorithm is used to solve assignment problems by minimizing cost. In the case of assigning the detections to existing tracks that means that the distance between the pose of the detections and the pose of the current tracks has to be minimized. The implementation of the assignment algorithm is shown in Algorithm 4, where each detection represents a detected product with pose, classification, and score.

Most noteworthy is that the algorithm makes use of two hyperparameters, namely the max_skipped_frames and distance_threshold parameters. The max_skipped_frames parameter determines how long each product is tracked. If it is set to 60, for example, and the tracker operates at 30 Hz then each object will be tracked for 2 sec if the track gets no new detections assigned. The distance_threshold parameter determines whether the detection and its assigned track are close enough to be considered the same product. If the distance is too large, the detection will not be assigned to the track and will get a separate new track since it is assumed that it is a newly detected product. The track will have no measurement in this frame and will therefore have its skipped_frames incremented with 1.

6.5.2. Kalman Filter tuning

The Kalman Filter has two variances that must be tuned for it to make accurate estimations: the measurement variance R and the process variance Q, stated in Equations 5.8 and 5.7. These variances are associated with how accurate the detections and state space model are respectively.

First, the measurement variance is estimated by measuring the variance of the pose estimation at a 50 cm distance. This is a simplification of the real measurement variance since the pose variance depends on the distance of the product to the camera, which is shown in Section 7.2.

Estimating the process variance, however, is more difficult. The process variance is associated with how well the state space model represents the actual behavior of the positions of the products. Since measuring it includes the measurement noise as well, the process variance is hand-tuned until a desirable tracking performance is reached. Desirable tracking performance means that the Kalman Filter can track the product poses through time even when False Positives or localization errors occur. Undesired tracking performance is especially visible when the initial state of the product is wrong due to a wrong detection. Then the state of the Kalman Filter is updated slowly to the correct state. The process variance is tuned to where the Kalman Filter updates well enough to account for initial state mistakes but also can ignore measurement errors after initializing the Kalman Filter.

6.5.3. Class and Score tracking

Each track as described in Algorithm 4 consists not only of a pose but also a classification, occurrence, and score. For each track, the number of times the track has been classified as a specific class is tracked in the occurrence parameter, and the mean score of that specific class is tracked in the score parameter. The most occurring class is stored in the classification parameter. Algorithm 5 describes

Algorithm 4 Update Multi-product Tracker

if length of tracks $= 0$ **then**
  for all detection in detections **do**
    Create a new track with detection
    Append new track to tracks
  end for
end if
if length of detections > 0 **then**
  Create an empty array dists
  for all track in tracks **do**
    Calculate the Euclidean distance between detections and track
    Append the calculated distance array to diss
  end for
  Apply the Hungarian algorithm to find the optimal assignments of detections to tracks using dists
  Create an empty list true_assignments
  for all assignment pair in assignments **do**
    if distance assignment pair $<$ distance threshold **then**
      Append assignment pair to the true_assignments list
    end if
  end for
  for all track, detection in true_assignments **do**
    $Update$ the track's Kalman Filter with detection
    Set track.skipped_frames to 0
  end for
  for all detection in detections **do**
    if detection is not in assigned **then**
      Create a new track
    end if
  end for
  for all track in tracks **do**
    Increment track.skipped_frames by 1
  end for
end if
for all track in tracks **do**
  if track.skipped_frames < max_skipped_frames **then**
    $Predict$ the next position using Kalman Filter
  else
    Delete track
  end if
end for

how the new detection is added to the track in terms of classification and score. To update the score for the track, the new mean score is calculated with the current mean score and the new score of the assigned detection. The score update function is shown in Equation 6.1.

$$track\_score_i = (track\_score_{i-1} + detected\_score)/(track\_occurrence_{i-1} + 1) \tag{6.1}$$

Algorithm 5 Classification and Score tracking

```
for all tracked_classification in tracked_classifications do
    if classification == tracked_classification then
        Increment classification.occurrence by 1
        Update mean score
        Tracked = True
    end if
end for
if Tracked != True then
    Append classification to tracked_classifications
    classification.occurrence = 1
    classification.score = score
end if
```

6.5.4. Product Choosing

Selecting the right product to pick is an important aspect of the picking process. The shelves in supermarkets are always stacked with multiple items of the same category next to each other, of which only one must be picked. Two methods of picking products are proposed: based on the number of detections ('occurrence' parameter) and based on the mean score of the detections ('score' parameter). Algorithm 6 shows how the product is chosen for picking.

Algorithm 6 Choose product

```
if Product already chosen then
    if Chosen product is tracked then
        return track of chosen product
    end if
end if
Create an empty Occurrence list
Create empty Scores list
for all Track in Tracks do
    if Track.classification == desired product then
        Append track.occurrence to Occurrences list
        Append track.score to Scores list
    end if
end for
if Occurrences list is not empty then
    if Occurrence based product choosing then
        return track with maximum occurrence
    else
        return track with maximum score
    end if
else
    return None (no product tracked with the desired classification)
end if
```

7

Experiments

This section covers the experiments done to verify the methods described in Sections 2, 3, 4, and 5 and the implementation of the methods described in Section 6. Section 7.1 explains what experiments are done regarding the **product detection** in isolation, and shows their results. Section 7.2 discusses the experiments and results of the **product grasp pose estimation** in isolation. Finally, Section 7.3 performs an ablation study of all described modules by performing real-world experiments with the entire proposed product grasping pipeline in the robot described in Section 1.3.

7.1. Dataset and Product Detection

This section evaluates the performance of the trained YOLOv6 models on the test set of the Albert Heijn Supermarket dataset. All models are trained according to the training details described in Section 6.3.1, and their performance is measured in terms of mean Average Precision (mAP), as described in Section 3.2.

As stated in Section 2, it cannot be directly measured whether the Albert Heijn Supermarket dataset contains enough data. The effect of pre-training on the performance of the models, however, can give insight into whether the proposed dataset contains enough data. Therefore the YOLOv6 [25] models are pre-trained on either the SKU-110K [16] dataset or the augmented SKU-110K-VS dataset, as stated in Section 6.2. Section 7.1.1 and Section 7.1.2 indicate that training solely on the Albert Heijn Supermarket train set does not yield the highest mean Average Precision (mAP) for both localizing products and localizing and classifying products.

Additionally, YOLOv6 models with different fine-tuning methods after pre-training are tested, and different non-maximum suppression (NMS) tunings are explored. Section 7.1.2 concluded that pre-training on the SKU-110K dataset and fine-tuning with 0 frozen layers results in the highest mAP for all model sizes (nano, small, medium, and large). Section 7.1.3 showed that there is no clear difference in the use of class-agnostic or intra-class NMS and that IoU thresholds ranging from 0.3 to 0.7 result in equal performance in terms of mAP. An IoU threshold of 0.1, however, suppresses too many bounding boxes and therefore results in a lower mAP.

7.1.1. Localization Results

The goal of the localization test is to quantify how well the trained models can localize products on the shelves of the Albert Heijn Supermarket without fine-tuning. The classes of the products are therefore not considered during this test. This gives insight into how well pre-training on either the SKU-110K or SKU-110-VS might impact the localization performance of the detector. Also, it indicates whether the Albert Heijn Supermarket, the SKU-110K, and SKU-110K datasets are alike. Three models are trained according to the details in Section 6.3.1. One set of models is trained on the SKU-110K dataset, one on the SKU-110K-VS dataset, and one on the Albert Heijn Supermarket dataset. The tests are done

on the test sets of the SKU-110K, SKU-110K-VS, and Albert Heijn Supermarket datasets. The classes of the products in the Albert Heijn Supermarket test set are not considered.

Unsurprisingly, the models that are trained on the SKU-110K dataset perform best on the test set of the SKU-110K dataset, as shown in the upper plots in Figure 7.1. Similarly, the models trained on the SKU-110K-VS dataset perform best on the test set of the SKU-110K-VS dataset, visualized in the middle plots of Figure 7.1. The localization test on the Albert Heijn Supermarket dataset test set (bottom plots), however, shows that the models trained on the SKU-110K and the SKU-110K-VS datasets outperform the models trained on the Albert Heijn Supermarket dataset in localizing products on the shelf, showing that there is a possible performance gain to be made by pre-training on the SKU-110K and SKU-110K-VS datasets. The models trained SKU-110K-VS dataset perform best on the Albert Heijn Supermarket dataset test set, showing that the visual servoing data augmentation seems to help the object localization performance. Because of this performance gain it can also be concluded that the Albert Heijn Supermarket dataset does not contain enough data to perform best in localizing products in the Albert Heijn supermarket.

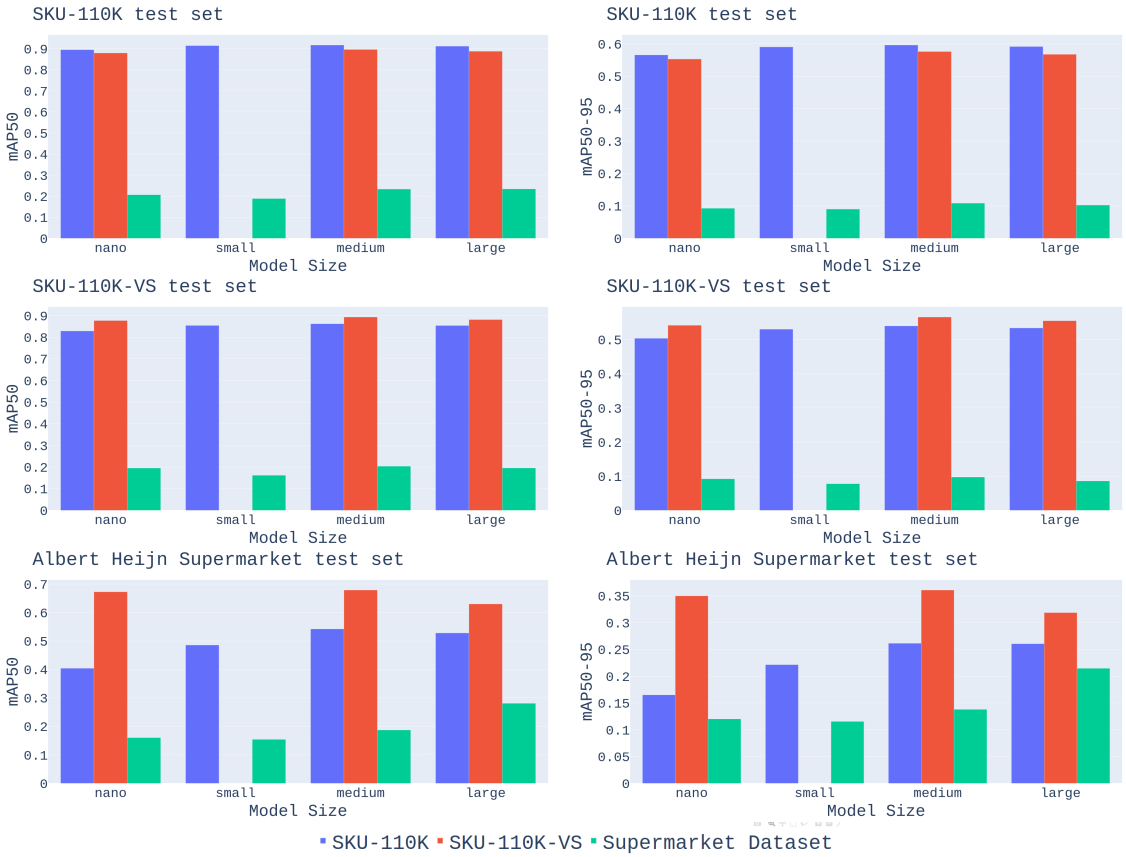

Figure 7.1: shows the product localization results. The left image shows the mean Average Precision (mAP) with an IoU threshold=0.5, while the right image shows the average mAP for IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05. The line color represents a YOLOv6 model trained on a specific dataset. The letters (n, s, m, and l) represent the different model sizes. The upper plots show test results on the SKU-110K test set, where it can be seen that the models trained on the SKU-110K dataset perform best. The middle plots show test results on the SKU-110K-VS test set, where it can be seen that the models trained on the SKU-110K-VS dataset perform best. The bottom plots show test results on the Albert Heijn Supermarket dataset, where it can be seen that models trained on the SKU-110K and SKU-110K-VS datasets outperform the models trained on the Albert Heijn Supermarket dataset. The models trained on the SKU-110K-VS dataset perform best, indicating that visual servoing data augmentation positively impacts localization performance.

7.1.2. Fine-tuning Results

The goal of this test is to discover what training method can reach the highest mAP on the Albert Heijn Supermarket test set. Five fine-tuning strategies are studied, fine-tuning without freezing any layers

and fine-tuning with 3, 5, 7, and 9 frozen layers. Pre-training is done on either the SKU-110K dataset or the SKU-110K-VS dataset, and fine-tuning is done on the Albert Heijn Supermarket dataset.

The results for pre-training the YOLOv6 models on the SKU-110K or SKU-110K-VS and then fine-tuning with or without frozen layers on the Albert Heijn Supermarket dataset are shown in Figure 7.2. Interestingly, training the model on the SKU-110K dataset and then fine-tuning the model on the Albert Heijn Supermarket dataset seems to be the best option. Pre-training on the SKU-110K-VS dataset does not yield the best performance, but does seem to increase mAP50 for the nano and the large model, and mAP50_95 for the large model as well. As can be seen in both sub-figures, the medium model performs best on both mAP50 and mAP5_95, which might be due to the size of the Albert Heijn Supermarket dataset. If the supermarket dataset were larger, the large model might have been able to learn more.

Therefore the conclusion can be drawn that the Albert Heijn Supermarket dataset does not contain enough data to successfully train larger models. Furthermore, because transfer learning increased the performance of all model sizes on the test set of the Albert Heijn Supermarket dataset, it can be concluded that the proposed Albert Heijn Supermarket dataset is not **diverse** enough to achieve the best-performing model. This answers the question stated at the end in Section 6.2 regarding dataset diversity and model performance.

Additionally, the results show that training with more than 3 frozen layers quickly leads to a decrease in performance. Only the small model pre-trained on the SKU-110K-VS dataset managed to outperform the fine-tuning method where no layers were frozen by training with 3 frozen layers. Despite this, the best method to train seems to be with pre-training on the SKU-110K dataset and then fine-tuning on the Albert Heijn Supermarket dataset without frozen layers.

Figure 7.3 summarizes the best models of each size, where it can be seen that the medium-sized model performs best. This model is therefore used as **product detector** in the product grasping pipeline. While the mean Average Precision (mAP) metric is useful in determining the model performance, it still averages over all classes in a dataset, and might not capture how well the model will perform on the final task, namely enabling the product grasping pipeline to grasp all desired products successfully. Therefore the product detector is further evaluated in Section 7.3.1, where it is tested whether the best model enables the product grasping pipeline to pick all desired products.

7.1.3. Non-maxiumum Suppression Tuning

The goal of this test is to find what non-maximum suppression (NMS) tuning makes the product detector detect the products on the shelves of the Albert Heijn the best. Tests are again done on the Albert Heijn Supermarket test set, and the performance is measured using mean Average Precision (mAP). Class-agnostic NMS and intra-class NMS are tested, as well as different IoU thresholds for the NMS. The YOLOv6 models used for evaluation are pre-trained on the SKU-110K dataset and then fine-tuned without frozen layers on the Albert Heijn Supermarket dataset, which reached the highest mAP in the tests conducted in Section 7.1.2.

The results of the test that compares class-agnostic NMS with intra-class NMS are shown in Figure 7.4. The mAP is not influenced by either, so based on that both methods can be used equally well in the product grasping pipeline. Also, the effect of the IoU threshold on the mAP is shown in Figure 7.5, where a threshold of 0.7, 0.5, 0.3, and 0.1 are tested. If the threshold becomes too low (0.1), the NMS suppresses too many boxes, resulting in a performance drop. The difference between an IoU threshold of 0.7, 0.5, and 0.3 is not significant, and therefore either can be used.

7.2. Grasp Pose Estimation

This section tests the accuracy and speed of the grasp pose estimation. The goal is to discover the accuracy and speed of the pose estimation. Random sampling introduces a trade-off between the two for the orientation estimation, and therefore the best sampling method must be determined. Section 7.2.2

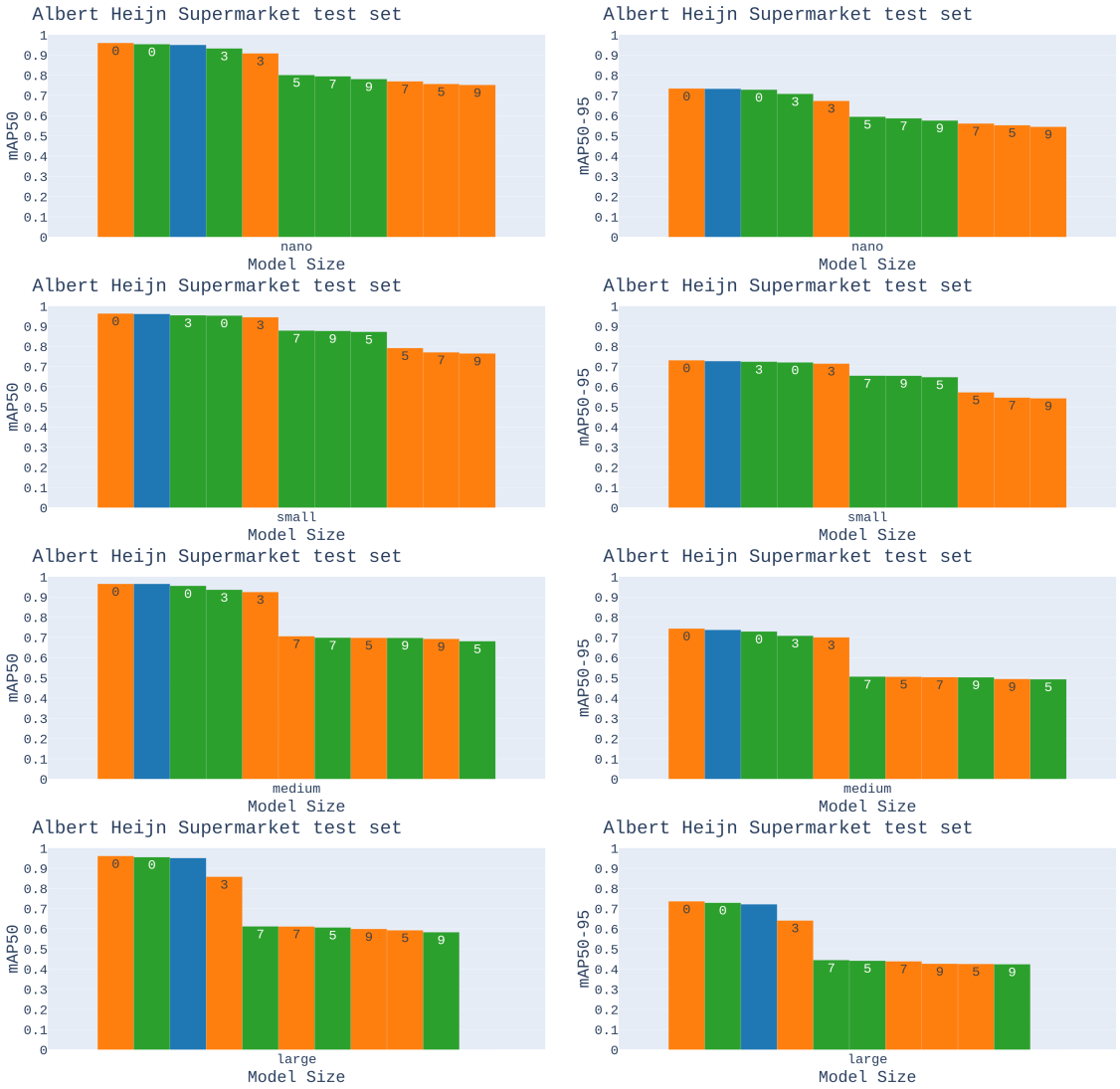

Figure 7.2: shows test results of the fine-tuned models on the Albert Heijn Supermarket test set. It can be seen that for each model size, pre-training with the SKU-110K dataset and then fine-tuning with 0 frozen layers is best. The left plot shows the mean Average Precision (mAP) with an IoU threshold=0.5. The right plot shows the average mAP for IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05. From top to bottom, the plots show results of the different model sizes, ranging from 'nano' to 'large'. In each plot, the bars are sorted on their mAP from left to right. The bar color indicates what dataset is used to pre-train the model. The numbers in the bars, if pre-training is applied, represent the amount of frozen layers during training.

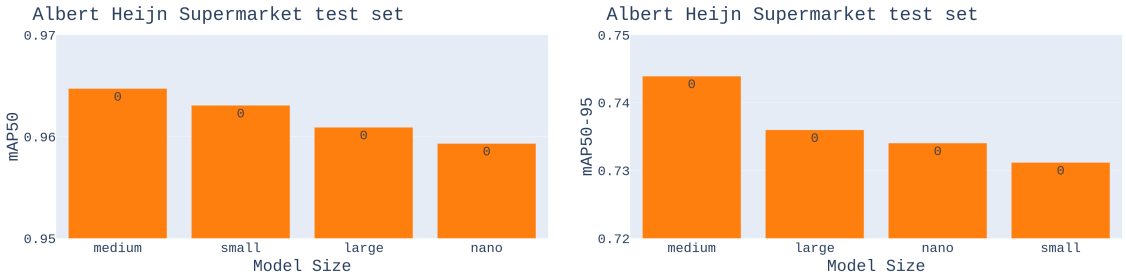

Figure 7.3: shows for each model size the model with the highest mAP on the Albert Heijn Supermarket test set. The medium model performs best based on mAP50 and mAP50_95 and is therefore used as the product detector in the product grasping pipeline. The left plot shows the mean Average Precision (mAP) with an IoU threshold=0.5. The right shows the average mAP for IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05. In each plot, the bars are sorted on their mAP from left to right.

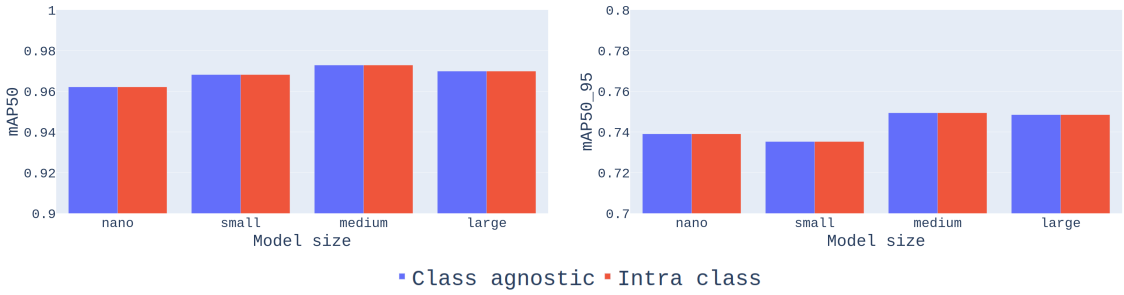

Figure 7.4: shows the effect of using class-agnostic NMS or intra-class NMS on the YOLOv6 models pre-trained on the SKU-110K dataset and fine-tuned on the Albert Heijn Supermarket dataset. The line colors indicate what NMS method is used. The left plot shows the mean Average Precision (mAP) with an IoU threshold=0.5. The right plot shows the average mAP for IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05. The letters (n, s, m, and l) represent the different model sizes.

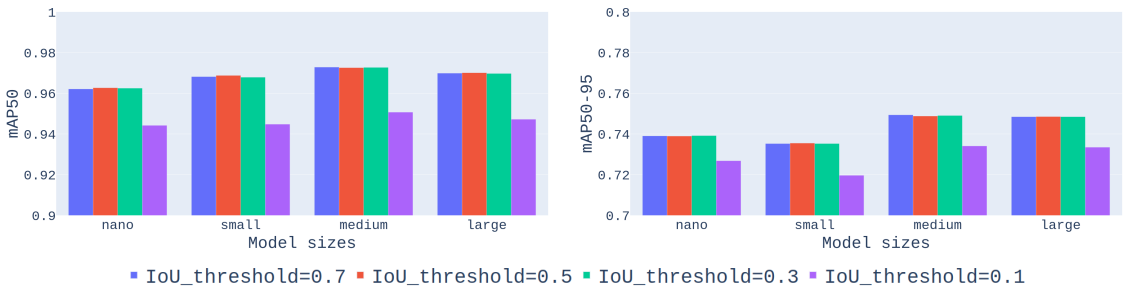

Figure 7.5: shows the results of predicting with different IoU thresholds for the NMS. The tests are done with the YOLOv6 models pre-trained on the SKU-110K dataset and fine-tuned to the Albert Heijn Supermarket dataset with class-agnostic NMS. The line colors indicate different IoU thresholds. The left plot shows the mean Average Precision (mAP) with an IoU threshold=0.5. The right plot shows the average mAP for IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05. The letters (n, s, m, and l) represent the different model sizes.

concludes that randomly sampling 125 points per bounding box is enough to obtain an orientation that is accurate enough to pick products. Section 7.2.1 shows that the use of the median for the distance estimation is most stable and that an underestimation of the bounding box improves the position estimation. The same conclusion follows from Section 7.2.2 for the orientation estimation. Therefore, each bounding box is reduced in size by 50% during inference, to ensure that overestimation is unlikely to occur, and to make underestimation more likely to occur. Lastly, both Section 7.2.1 and 7.2.2 conclude that the pose estimation improves as the camera comes closer to the product. This finding therefore supports the use of visual servoing to refine product poses during the picking sequence. All these tests are done on objects with a flat surface. Appendix A shows the results for these tests on objects with a round surface.

All tests are done by taking 100 measurements and calculating the mean and standard deviation of the predictions. A schematic of the test setup is shown in Figure 7.6.

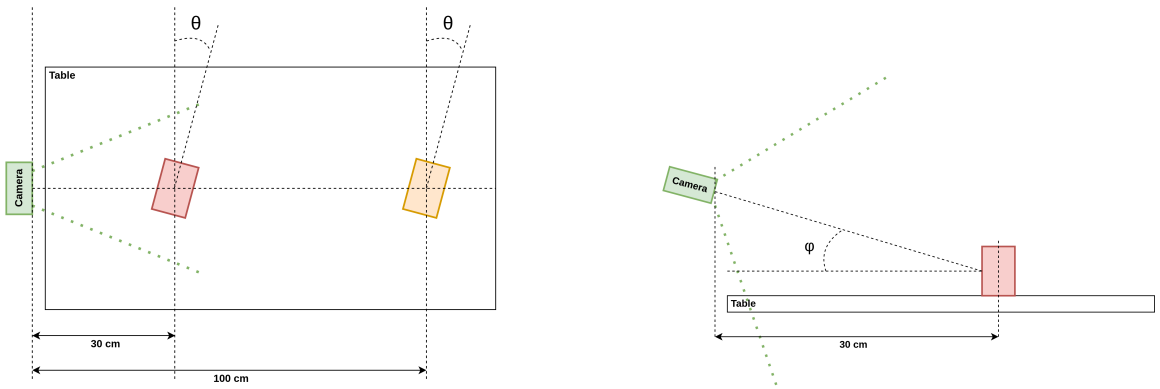

Figure 7.6: shows a schematic top view (left) and a side view schematic (right) of the test setup for the grasp pose estimation. The red object denotes the test at 30 cm from the camera, orange object denotes the test at 100 cm from the camera. Both round and flat objects are tested, as well as the impact of different angles with respect to the camera denoted as θ and ϕ.

7.2.1. Distance estimation results

The goal of this experiment is to discover how accurate the distance estimation of the products is, and what can be done to improve the accuracy of the estimation. Since the **product detector**, as described in Section 3, does not always produce perfect bounding boxes, the effect of over- and underestimation of the bounding boxes on the distance estimation is studied. The distance estimation is done via the median of the depth bounding box. A comparison with the mean of the depth bounding box is shown in Figure 7.7, from which it can be concluded that the use of the median is more stable to variations of the bounding box size.

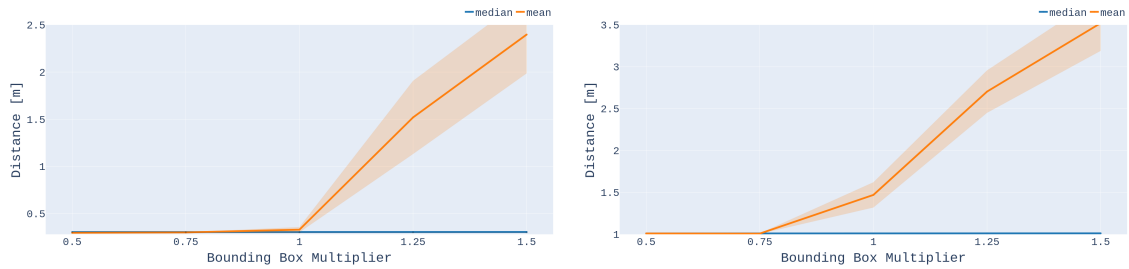

Figure 7.7: shows the results of the distance estimation with the mean and or the median of the depth bounding box 30 cm (left) and 100 cm (right). It can be seen that the mean becomes unstable when the bounding box is overestimated, and therefore the median is used for the distance estimation. The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in mm. ϕ = 0°during this test.

The accuracy of the median distance estimation is tested by looking at how accurate the measured

distance of a flat object is with regard to the known distance at both 30 cm and 100 cm from the camera. The orientation of the object with regard to the camera is also changed to see its impact on the distance estimation. θ is changed from 0°to 45°in steps of 15°. Figure 7.8 shows the results of the distance estimation. The mean prediction of the 100 measurements along with a confidence interval of 95% is shown.

First, the test at 30cm (the left plot in Figure 7.8) shows that overestimation of the bounding box has the largest impact on both the offset and the confidence interval of the results, making the distance estimation less accurate. Underestimation, however, seems to have a slight positive impact on the distance estimation under various θ angles. This is likely because underestimation, when for example $\theta = 45°$, leads to the pointcloud only containing points of the product, and not of the surroundings. The constant offset of roughly 4 mm that is visible in the test at 30 cm is likely due to a measurement error in the test setup, and not because of the used distance estimation.

Next, the test at 100 cm (the right plot in Figure 7.8) shows that the distance estimation becomes less accurate as the distance is increased. The effect of θ on the distance estimation is greater, as it can be seen that the offset increases as θ is increased. This is likely because the bounding box is a lot smaller at 100 cm than at 30 cm, thus having a larger impact on the median value. After all, there are fewer pixels to take the median from. The constant offset of 10 mm at $\theta = 0$ is likely due to measurement errors in the test setup.

In conclusion, Figure 7.7 showed that the median is more stable to variations of the bounding box size, and is therefore used to estimate the distance. Additionally, Figure 7.8 showed that the distance estimation has a smaller confidence interval when the bounding box is underestimated because fewer environmental pixels are considered in the estimation. The position estimation is therefore done by resizing the bounding box to half its size. This makes overestimation unlikely to occur during inference and can help increase the accuracy of the distance estimation during inference.

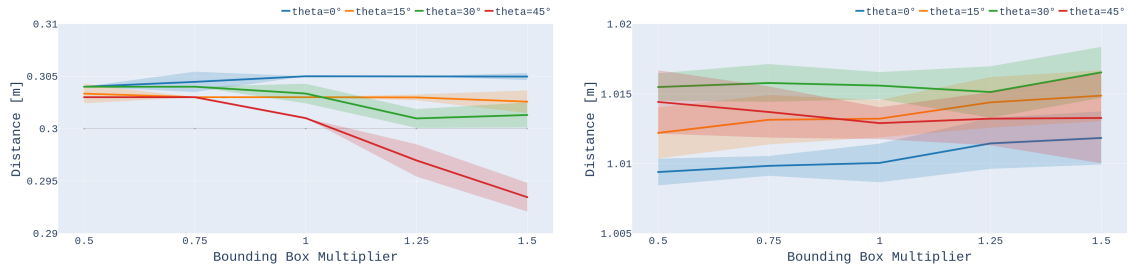

Figure 7.8: shows the distance estimation results for a flat object at a distance of 30 cm (left) and 100 cm (right). The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in mm. Each line represents θ in degrees with respect to the camera. ϕ = 0°during this test. Left and right it can be seen that underestimation improves the confidence interval and on the left, it also lowers the offset in the distance estimation. The constant offset of 4 mm is likely due to a measurement error in the test setup. The constant offset on the right of 10 mm is likely due to a measurement error in the test setup.

7.2.2. Orientation estimation results

The goal of this section is to measure how accurate the orientation estimation (θ and ϕ) is, and what can be done to improve the accuracy of the estimation. Furthermore, the effect of random sampling points on the accuracy and run-time of the orientation estimation is studied. Similar to the distance estimation, the effect of over- and underestimation of the bounding boxes on the orientation estimation is studied. The ground truth θ and ϕ are measured by hand and then compared to the estimated angles. Tests are done for flat objects at 30 cm and 100 cm where $\phi = 0°$ and the θ angle is varied similarly to the distance test. The same test is done at approximately 30 cm with $\phi = 45°$. The results of the θ and ϕ estimations where all points are used are shown in Figure 7.9 and Figure 7.10 respectively. Thereafter, a comparison is done between estimating the orientation with all points against randomly sampling 1000, 500, 250, or 125 points. Figure 7.11 and Figure 7.12 show the effect of random sampling on the accuracy of θ and ϕ respectively. Figure 7.13 shows the effect of random sampling on the estimation

time. Again, each test shows the mean of 100 measurements along with a confidence interval of 95%.

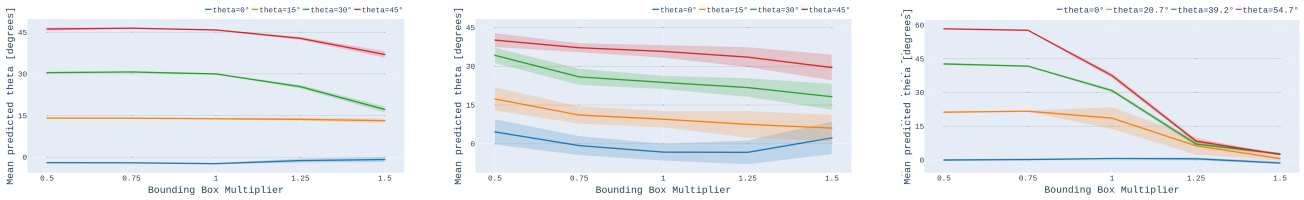

Figure 7.9: shows the theta estimation results for a flat object at a distance of 30 cm with $\phi = 0°$ (left), 100 cm with $\phi = 0°$ (middle), and 30 cm with $\phi = 45°$ (right) where the theta angles are different due to the displacement of the camera. Overestimation increases the confidence interval as well as the offsets of the estimation. The larger distance also negatively impacts the confidence interval and the offset. The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in degrees. Each line represents theta in degrees with respect to the camera.

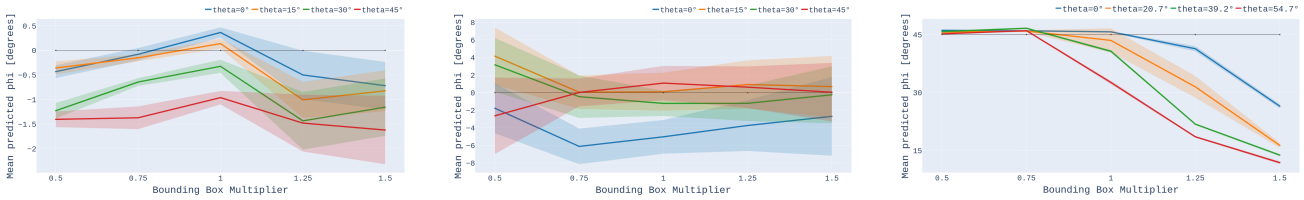

Figure 7.10: shows the phi estimation results for a flat object at a distance of 30 cm with $\phi = 0°$ (left), 100 cm with ϕ = 0°(middle), and 30 cm with $\phi = 45°$(right). At 30 cm, overestimation increases the confidence intervals of the estimation. At 100 cm, the offsets are increased. When $\phi = 45°$, it can be seen that overestimation increases the offset. Also, the theta angles are different due to the displacement of the camera. The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in degrees. Each line represents theta in degrees with respect to the camera.

At all distances and tested angles for θ and ϕ, it can be seen that bounding box overestimation leads to a performance decrease in terms of offset and confidence interval. Underestimation of the bounding box seems to improve the theta estimation, especially visible in the test with $\phi = 45°$. Also interesting is that the confidence intervals and offsets are a lot smaller in the θ tests at 30 cm than in the tests at 100 cm. The offset for the ϕ estimation seems to grow as θ is increased. This effect is smaller in the test at 100 cm, yet both the offset and confidence interval in that test are larger. In the test at 30 cm, where $phi = 0°$, it can be seen that overestimation has an impact on the offset of the estimation of ϕ. Again, underestimation seems to improve the confidence interval, especially when θ is increased. In the test at 30 cm where $phi = 45°$, it can be seen that underestimation lowers the offset significantly in comparison to overestimation.

From this it can be concluded that the θ and ϕ estimations perform better as the product is closer to the camera. Additionally, bounding box underestimation seems to improve both estimations. Therefore the orientation estimation is done by resizing the bounding box to half its size. This makes overestimation unlikely to occur during inference and can help increase the accuracy of the distance estimation during inference.

Figures 7.11 and 7.12 show that the confidence interval increases when less randomly sampled points are used to estimate the product orientation. The confidence interval when sampling 125 points with underestimation at 30 cm for θ is 4.5 times as much and for ϕ 3.3 times as much. While this is significant, the error that may occur is then still bounded to $\pm 1°$ for θ and $\pm 0.5°$ for ϕ, which will not influence the grasp success, as will be demonstrated in Section 7.3.2. At 100 cm, the difference between sampling points and using all points for orientation estimation is not significant. Therefore, it can be concluded that random sampling points for the orientation estimation maintain a tolerable accuracy of the orientation estimation.

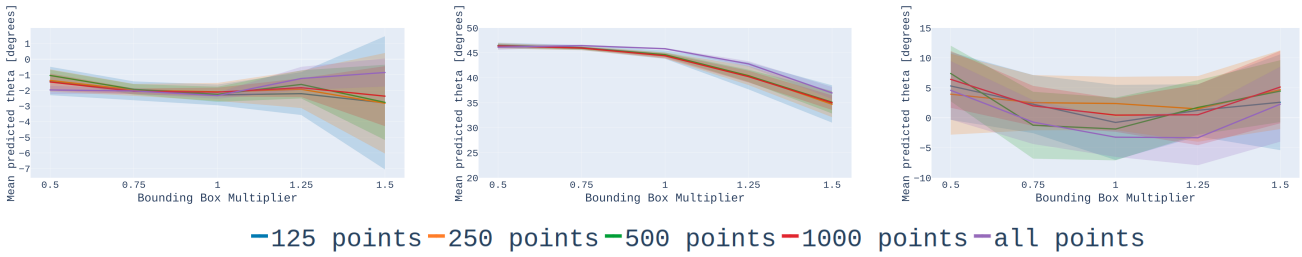

Figure 7.11: shows the theta estimation results for a flat object at a distance of 30 cm with $\theta = 0°$ (left), 30 cm with $\theta = 45°$ (middle), and 100 cm with θ = 0°(right). $\phi = 0$ in every plot. The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in degrees. Each line represents the number of sampled points in the pointcloud, except the 'all points' line, where no sampling is used. Overestimation increases offset and confidence interval for all sampling methods, but the confidence interval increases faster for the sampling methods that use fewer points. Also for underestimation, the confidence interval increases slightly as fewer points are used in the orientation estimation.

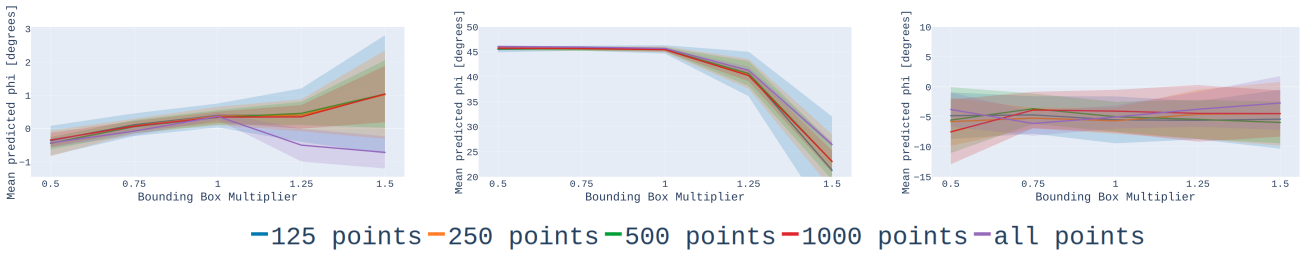

Figure 7.12: shows the phi estimation results for a flat object at a distance of 30 cm with $\phi = 0°$ (left), 30 cm with $\phi = 45°$ (middle), and 100 cm with $\phi = 0°$ (right). $\theta = 0$ in every plot. The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in degrees. Each line represents the number of sampled points in the pointcloud, except the 'all points' line, where no sampling is used. Overestimation increases offset and confidence interval for all sampling methods, but the confidence interval increases faster for the sampling methods that use fewer points. Also for underestimation, the confidence interval increases slightly as fewer points are used in the orientation estimation.

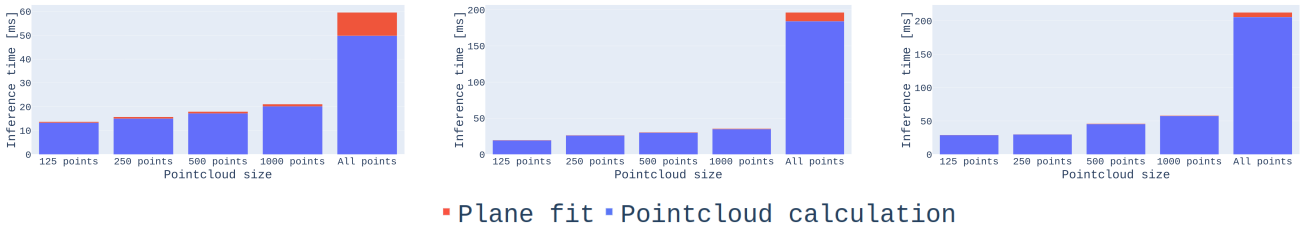

Figure 7.13: shows the orientation estimation times when 5 bounding boxes (left), 15 bounding boxes (middle), or 30 bounding boxes (right) are present in the frame. Random sampling has a large impact on the pointcloud calculation time as using fewer points leads to a faster calculation time. The achieved speed increase, however, decreases as points are halved, suggesting that the calculation time converges to a minimum.

Randomly sampling the pointcloud increases the run-time of the orientation significantly, 3.7-7.3 times faster when evaluating 30 bounding boxes in one frame and 2.8-4.5 times faster when evaluating 5 bounding boxes in one frame, as shown in Figure 7.13. Using random sampling with 125 points the orientation estimation takes 13-26 ms, enabling an operation rate of 38 Hz. In combination with the fact that the accuracy remains similar when underestimating the bounding box, random sampling of 125 points is used in the product grasping pipeline and further tested in Section 7.3.2.

7.3. Ablation Study

An ablation study for the proposed modules and their tuning is done. The goal of the ablation study is to discover what impact the implementations and the tunings of each module have on the success of the product grasping pipeline. Each of the tests that are done consists of 10 grasp attempts by the robot in a supermarket. The 10 grasp attempts are done on 5 different products, of which each covers a different type of grasping scenario. The success rate SR is used for comparison, which is described in equation 7.1.

$$SR = n_{success}/n_{total} \tag{7.1}$$

where $n_{success}$ = the number of successful grasping attempts and n_{total} = the number of grasping attempts. The testing setup of the shelf is shown in Figure 7.15. The five products that the robot must attempt to grasp are shown in Figure 7.14. Appendix B shows examples of successful grasps for each of the products.

Figure 7.14: visualizes the 5 products grasped in the pipeline tests. From left to right the products are: 'Brownie', 'Nasi Speciaal', 'Volle Yoghurt', 'Mais', and 'Sweet Sour Sauce'.

The 'Brownie' product is the easiest case. This product occurs once on the shelf, making decision-making for the robot easy. Furthermore, it is one of the larger products with a flat surface. The 'Nasi Speciaal' product is more difficult. Although it only occurs once on the shelf, is quite large, and has a flat surface, it looks very similar to the product that is located next to it. The 'Volle Yoghurt' product is similar to the first two, only a bit narrower, and it occurs twice on the shelf, positioned next to each other. This setup can test the decision-making of the robot. The decision-making is further challenged by the 'Mais' product, which occurs three times on the shelf, is round, relatively small, and two of them are stacked on top of each other. Finally, the 'Sweet Sour Sauce' product is small and round, forcing the detection to be precise about where to pick.

The tuning remains the same in all tests unless stated otherwise. The detector, pose estimation, and tracking all operate at a rate of 30 Hz and the tracker keeps track of each product for two seconds without new detections (max_skipped_frames=60). The distance_threshold is set to 0.1, meaning that measurements are not assigned to a specific track if they are further away than 10 cm, even when they are assigned to the track by the Hungarian algorithm. The applied non-maximum suppression on the product detector is class-agnostic with an IoU threshold=0.7, but as stated in Section 7.1.3, IoU thresholds of 0.3 and 0.5 and intra-class NMS can also be used. The model used as the product detector is the YOLOv6m model pre-trained on the SKU-110K dataset and then fine-tuned to the Albert Heijn Supermarket dataset. This model yielded the highest mAP on the test set of the Albert Heijn Supermarket dataset, as stated in Section 7.1.2. The occurrence-based product-choosing method is used during the tests to choose the correct desired product.

Figure 7.15: shows the setup of the shelf during all the pipeline tests. Some products have counter-parts in the setup that look similar to the desired products. Other products have multiple instances of them available to pick, forcing the robot to decide. Products differ in size and shape to test their influence on the grasp success rate.

If the chosen product is lost (so no detections > 2 sec), the robot tries to choose another product that is available based on the product-choosing method and the available tracks. This behavior is further referenced as 'switching'. Additionally, if no other track is available, then the fallback behavior is that the robot moves toward the last known location of the product it chose and tries to pick blindly. Due to this fallback behavior, certain grasps can still succeed even if the detection pipeline fails to track any desired product until the product is picked. In this case, the grasp will still count towards the success rate, but its observation will be stated. The **product detection** module is tested in Section 7.3.1. The **product grasp pose estimation** module is tested in Section 7.3.2. Section 7.3.3 tests the implementation of the **product tracker**. Finally, the use of position-based visual servoing (PBVS) is tested in Section 7.3.4.

7.3.1. Detection

Three product detection tests are performed to discover the influence of non-maximum suppression, rotation compensation, and model performance on the success of the product grasping pipeline.

For the first test, the effect of non-maximum suppression (NMS) on the grasp success is measured. The goal is to determine whether using class-agnostic or intra-class non-maximum suppression has an impact on the success rate of the product grasping pipeline. Non-maximum suppression is used after inference to consider only the highest-scoring detections and suppress the other proposed products by looking at how much overlap there is between proposals. Class-agnostic NMS suppresses the bounding boxes only based on their overlap, independent of the associated class with the bounding box. Intra-class NMS only suppresses bounding boxes that have the same class and more overlap than the threshold. The results of the tests are shown in Table 7.1.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
Class-agnostic	1	1	0.5	1	1	**0.9**
Intra-class	1	1	1	1	1	**1**

Table 7.1: shows the ablation results of using class-agnostic vs intra-class non-maximum-suppression. For both methods, the intersection over union (IoU) threshold is set to 0.7 (70% overlap). Intra-class NMS increases the success rate slightly.

In the class-agnostic and intra-class non-maximum suppression test, the intra-class tuning showed the highest success rate. During the test, both methods kept track of a set of desired products at all times and no fallback behavior was observed. The grasps in which 'Volle Yoghurt' was the desired product, however, showed some switching behavior for both the class-agnostic and intra-class settings. The switching occurred twice in the intra-class setting and once in the class-agnostic setting, in which the

latter resulted in a failed pick because the switching occurred just before grasping the product. Intra-class NMS shows slightly better performance in providing detections for the tracker.

The second detection test studies the effect of rotation of the robot arm on the grasp success rate. The goal is to see the effect of rotation compensation on the grasping success rate. The realsense camera is mounted on the manipulator, so the camera moves and rotates as the arm moves toward the shelf. Rotation compensation is used to enable the product detector to detect products under the same orientation, regardless of the position of the manipulator. The test measures the grasp success of the robot with and without the camera rotation compensation. The results of the test are shown in Table 7.2.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
With Rotation Compensation	1	1	0.5	1	1	**0.9**
Without Rotation Compensation	1	0	1	0.5	0	**0.5**

Table 7.2: shows the ablation results of using rotation compensation for the orientation of the camera during the picking sequence. Rotation compensation has a large impact on grasping success. Without rotation compensation, the detector cannot accurately detect the product, resulting in both localization and classification errors.

The rotation compensation results show a clear pattern. Compensating for the rotation of the arm makes the grasping pipeline far more reliable. Without the rotation compensation, the picking success rate drops from 0.9 to 0.5. Not all picking failures, however, are due to the detector not detecting the products. Observing the output of the detector showed that, apart from a lower detection performance, the assumption that the grasp location for the products should be in the center of their bounding box does not hold. The product detector predicts bounding boxes that are upright, and if the products are rotated, the bounding box center does not align with the product's center anymore, leading to grasp failure. This behavior is visible in its failure for the 'Sweet Sour' product, which is small and therefore requires a precise grasp pose. Also, the classification between similar products seems to fail when no rotation compensation is used because the 'Nasi Speciaal' product got misclassified for both picks.

The third experiment tests the effect of the model performance on the grasp success of the product grasping pipeline. The best-performing model in terms of mAP, as concluded from the experiments in Section 7.1.2, is compared to the best-performing model trained without pre-training (YOLOv6 medium model), to capture the influence of pre-training on the grasp success. Furthermore, the worst-performing model in terms of mAP is tested as well, to confirm that the use of the best model is crucial. The experiments are summarized in Table 7.3.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
Best	1	1	0.5	1	1	**0.9**
No pre-training	1	1	1	1	0	**0.8**
Worst	0	0	0	0	0	**0**

Table 7.3: shows the ablation results of using other models than the best-performing YOLOv6 model. The worst model is not capable of detecting products well enough for successful product grasping. The model without pre-training managed to detect all products except the small 'Sweet Sour' product, resulting in failed grasps for that specific product.

The worst performing model cannot classify the products accurately enough to obtain any successful grasps. Either the product was not found, or a product of a different class was picked. The model without pre-training, however, performed well on most products. However, it was not able to detect the 'Sweet Sour' product and therefore failed to pick this product twice. It can therefore be concluded that mean Average Precision (mAP) can be used as an estimate for how well the product detector can predict products, and that pre-training actually improves grasp success, and not just the mAP metric.

Furthermore, the model without pre-training was not able to enable the product grasping pipeline to pick all desired products. From this, it can also be concluded that the Albert Heijn Supermarket dataset is not diverse enough to create the best-performing product grasping pipeline. This answers the question stated at the end of Section 2.1 regarding dataset diversity and product grasping pipeline success.

Next, the impact of the class imbalance, mentioned in Section 2.2, on detection failures is low, because the lowest occurring object, namely the 'Brownie' product, has been picked successfully during all tests. Therefore it can be concluded that the class imbalance of the dataset did not negatively impact the product detector enough to lead to grasp failures.

Lastly, Section 2.2 discussed why the challenge of distinguishing similar products from each other must be included. The 'Nasi Speciaal' and 'Volle Yoghurt' products were often picked successfully, and the occurring failures were not due to mixing up similar products, except when grasping without rotation compensation. Therefore it can be concluded that the challenge of similarity between products has been overcome, at least for the set of 36 products in the Albert Heijn Supermarket dataset.

7.3.2. Product Grasp Pose Estimation

The orientation estimation is tested in this section to uncover its effect on the grasp success. Random sampling 125 points for each pointcloud as described in Section 7.2 is used here. The test setup is changed slightly, to test the effect of different orientations per product in the shelf. The changes to the test setup are shown in Figure 7.16.

Figure 7.16: shows the changes to the test setup during this specific experiment. The "Brownie" object is rotated to approximately $\theta = 35°$. The "Nasi Speciaal" and "Volle Yoghurt" objects are rotated to approximately $\phi = 30°$.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
no orientation	1	1	0.5	1	1	**0.9**
orientation	1	1	1	1	1	**1**

Table 7.4: shows the ablation results of using orientation estimation and not using orientation estimation. Orientation estimation increases the success rate slightly.

Using orientation estimation increases the grasp success rate slightly, as can be seen in Table 7.4. Therefore, as stated in Section 7.2, it can be concluded that the confidence interval of $\pm 1°$ for θ and $\pm 0.5°$ for ϕ when using 125 points per pointcloud does not influence the grasp success, because even when no orientation estimation is used and the angle between gripper and product is off by 35°the grasp can succeed. The effect of the orientation estimation, however, is evident in Figure 7.17, which shows the grasps with and without orientation estimation. It can be seen that most grasps are successful without orientation estimation thanks to the flexibility of the suction cup. If a less flexible gripper is used, the success rate for the picking without orientation estimation is not guaranteed, and therefore the use of the orientation estimation is recommended.

Furthermore, the test where the orientation estimation is used also grasps all round objects successfully. Therefore it can be concluded that the orientation estimation also works well enough to enable successful grasps for products with a round surface.

Figure 7.17: shows grasp examples without orientation estimation (top) and with orientation estimation (bottom). Although both grasps are successful, the grasp with orientation estimation managed to not deform the product and relied less on the flexibility of the suction cup.

7.3.3. Tracking and Product Choosing

The product tracking and product choosing, as described in Section 5 and Section 6.5 are evaluated in this section. The goal is to find the effect of using a product tracker at all, using occurrence- or score-based product choosing and tracking for 2 or 10 seconds without new detections on the product grasping success. No tracking means that while in each frame there still are detections to make decisions on, the previous detections are not taken into account in that decision. Also, because the products are not tracked, the robot is forced to choose what product to pick in each frame that products are detected. Therefore, the occurrence-based product choosing cannot be used, since without tracking the occurrences that a product has been detected as a certain class are not counted. The results of the tests are shown in Tables 7.5 and 7.6.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
Tracking Occurrence based	1	1	0.5	1	1	**0.9**
Tracking Score based	1	1	0.5	1	1	**0.9**
No Tracking Score based	1	1	0	1	1	**0.8**

Table 7.5: shows the ablation results of using a product tracker with score-based and occurrence-based product choosing. Using no tracking at all has been tested as well. Both product-choosing methods perform equally well. Tracking products with a Kalman Filter improves the grasp success slightly.

The tracking and product choosing test show that choosing a product based on its occurrences or its score does not have a significant impact on the overall success of the pipeline. However, the success rate is decreased without tracking, namely from 0.9 to 0.8. During the experiment without a tracker, switching and fallback behavior was observed several times. Switching too often can cause grasp failures, and fallback behavior is undesirable since the robot does not reason about the desired product anymore. Therefore, it can be concluded that the use of the tracker helps the grasping pipeline succeed

by allowing the pipeline to make decisions without picking blindly.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
Tracking 2 sec	1	1	0.5	1	1	**0.9**
Tracking 10 sec	1	1	1	1	1	**1**

Table 7.6: shows the ablation results of using a product tracker with different max_skipped_frames parameters, allowing products to be tracked for 2 and 10 seconds without detections, respectively. Tracking for 10 seconds improves the grasping success slightly.

Tracking for 10 seconds seems to improve the grasp success rate. This is because no switching occurs when tracking for such a long period. Tracking for a longer period without measurements means that the grasping pipeline becomes more dependent on the choice it makes at the beginning of the picking sequence. While this improves the success rate of this test, it might not be best to track without measurements for a longer period. After all, the failure in the case of 2-second tracking is due to a switch at the end of the picking sequence. The switch occurs because the detector is not able to detect the product anymore because the gripper is in the way of the camera. A better way to mitigate this problem would be to lock the product choosing when within a certain range to the desired product, to ensure that failure because of switching does not occur. Switching is a desired feature that enables the product grasping pipeline to correct mistakes.

7.3.4. Visual Servoing

This section compares visual servoing, and thus closed-loop control, to choosing a product once at the beginning and picking blindly after that, known as open-loop control. The goal is to uncover the effect of visual servoing on the product's grasping success. Furthermore, the influence of the operation rate on the grasping success rate during visual servoing is also studied, of which the results are shown in Table 7.8. The results of the closed-loop vs. open-loop tests are shown in Table 7.7.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
Closed-Loop	1	1	0.5	1	1	**0.9**
Open-Loop	1	0	1	0.5	0	**0.5**

Table 7.7: shows the ablation results of using visual servoing (closed-loop) and not using visual servoing (open-loop). Visual servoing has a large impact on the success of the product grasping pipeline, making it crucial for product picking success.

The difference between open-loop and closed-loop grasping is significant. Open-loop control drops the success rate from 0.9 to 0.5. The major reason for the failure of the grasping pipeline during open-loop is due to the distance the detection is performed. Since for open-loop grasping the product detector is furthest away from the shelf, the classification and localization are more difficult to perform accurately, leading to grasp failures. When the 'Nasi Speciaal' product test was performed, another similar product to 'Nasi Speciaal' product was misclassified and the wrong product was picked. The 'Sweet Sour' product was classified correctly, but slightly misplaced in the world, which must be precise for a small product like this, which resulted in a grasp failure.

Method	Brownie SR	Nasi Speciaal SR	Volle Yoghurt SR	Mais SR	Sweet Sour SR	**Total** SR
30 Hz	1	1	0.5	1	1	**0.9**
10 Hz	1	1	0.5	0.5	1	**0.8**

Table 7.8: shows the ablation results of changing the operation rate of the product grasping pipeline. Lowering the operation rate from 30 Hz to 10 Hz has a negative impact on the grasping success rate, from which it can be concluded that operating at 30 Hz is better.

The detection operation rate at 30 Hz and 10 Hz shows that the success rate drops slightly as the detection operation rate is lowered. Nevertheless, the grasping pipeline still succeeded 80% of the time. Switching, however, was observed far more often at 10 Hz than at 30 Hz. Since switching too often can lead to failures, an operation rate of 30 Hz is more desirable than an operation rate of 10 Hz.

7.3.5. Discussion

The fallback behavior used as a last resort to pick products resulted in pick successes while they would otherwise have failed. Such behavior, however, can only be used when a human operator is supervising the robot and can stop the robot when collision with the environment is inevitable. Therefore, conclusions based on the fallback behavior success must be taken with caution. Other fallback behavior could be more desirable, like forcing the robot to move back and look again for the product instead of blindly moving forward. Such behavior makes the robot able to operate more autonomously.

Furthermore, the shelf setup remained the same during all tests. Therefore, the success or failure cases were often similar and most tests resulted in the same detections, and thus the same movement of the robot arm, which in turn led to similar decisions. More tests with various locations of the products could help understand the robustness of the product grasping pipeline even more.

Next, in the current product choosing implementation there is no reasoning about stacked products. Therefore, the robot might pick the lower products of two stacked products, which is undesirable. During these tests picking lower products counted toward success, but taking stacked products into account in the decision-making is a necessity for supermarket deployment.

Lastly, the switching and fallback behavior that occurred during the tests is due to the product grasping pipeline losing track of the product it chose. Most of the time, this happens just before the pick, when the suction cup of the robot arm is in front of the product. The robot moves slowly at the end of the pick to ensure precision, yet this also leads to losing track of the desired product if *max_skipped_frames* is too low. Simply increasing this parameter might be undesirable, because it can lead to tracks being tracked for too long. Other solutions might be to increase the speed of the robot arm at the end of the pick or to lock the product by choosing not to refine the pose anymore and pick blindly after being below a certain distance threshold to the product, as suggested in Section 7.3.3.

8

Conclusion

This section concludes this thesis by summarizing the findings from each section and answers the main research question from Section 1:

What implementation of robotic skills and combined optimization results in a grasping pipeline that is best in consistently and robustly grasping desired products from supermarket shelves in Albert Heijn?

Section 1 stated that three robotic skills, **product detection**, **product grasp pose estimation** and **product tracking**, must be achieved to realize a product grasping pipeline, and that a new **dataset** is necessary to train the product detector. Furthermore, Section 1 stated that each robotic skill and the dataset had challenges that had to be overcome. These challenges are repeated below:

1. Product detection requires a **dataset** to train on, which must include the products from the Albert Heijn supermarket. Such a dataset must contain enough data that represents the Albert Heijn supermarket to make the product detector succeed at detecting products after training.
2. To achieve the **product detection** skill, a suitable object detector has to be chosen and trained to detect the products in the Albert Heijn shelves at a rate of 30-100 Hz for visual servoing.
3. **Product grasp pose estimation** has to be done to enable the robot to pick the detected products from the shelves. Like the product detector, the product grasp pose estimation should run at 30-100Hz for visual servoing but also be accurate enough to make the product grasping pipeline succeed.
4. **Visual servoing** and **product tracking** must be done to keep track of products through time to account for product detection and product grasp pose estimation mistakes, and changes in the environment. Again, the product tracker must track the products accurately through time, and at a rate of 30-100 Hz for visual servoing.

To overcome the first challenge, The proposed **Albert Heijn Supermarket dataset** is used to train the product detector in the product grasping pipeline, as discussed in Section 2 and Section 6.2. The dataset is annotated by hand and consists of 1166 images of a shelf with products labeled with bounding boxes. A total of 36 different products are considered in the Albert Heijn Supermarket dataset. The product selection consists of sets of products that look similar due to their brand or their type to incorporate the challenge of distinguishing similar products in the supermarket. Additionally, lighting is taken into account by having dark and light images, the distance is taken into account by taking images at distances varying from 20 cm to 100 cm, directions are taken into account by taking pictures with directions varying from -45°to 45°with respect to the shelf, and product misplacement is taken into account by shuffling the products on the shelves. Directly verifying whether the dataset contained enough data was not possible, and is measured indirectly by measuring the performance of YOLOv6 models trained only on the proposed dataset and comparing them to the same models that are fine-tuned to the proposed dataset while first being pre-trained on either the SKU-110K dataset or the augmented SKU-110K-VS dataset.

Pre-training on the SKU-110K dataset achieved the highest mAP on the Albert Heijn Supermarket test set, demonstrated in Section 7.1.2. Therefore, it can be concluded that the train set of the Albert Heijn Supermarket dataset does not contain enough data to achieve the highest mAP on the test set of the Albert Heijn Supermarket dataset, because pre-training allowed the models to learn other features leading to a higher mAP compared to the models trained on solely the Albert Heijn Supermarket dataset. The results of the experiment in Section 7.3.1 show that training solely on the Albert Heijn Supermarket dataset does not yield the highest success rate in the product grasping pipeline. The experiment compared the YOLOv6 medium model pre-trained on the SKU-110K dataset with the YOLOv6 medium model without pre-training, where classification errors resulted in a lower success rate for the latter. From this experiment it can therefore also be concluded that the Albert Heijn Supermarket dataset alone does not contain enough data to achieve the highest performance. Pre-training on the SKU-110K dataset and fine-tuning on the Albert Heijn Supermarket dataset, however, proved effective, and enabled the proposed product grasping pipeline to reach a success rate of 100% in some experiments from Section 7.3. Consequently, it can be concluded that the Albert Heijn environment and the environment in the SKU-110K dataset are similar. To sum up, the Albert Heijn Supermarket dataset does not contain enough data alone, but when combined with the SKU-110K dataset it enables the YOLOv6 models to detect the products in the Albert Heijn supermarket, which in turn enables the robot to pick products successfully.

The second challenge is addressed in Section 3 and Section 6.3, that describe how **product detection** is achieved. Section 3 and Section 6.3 discuss what object detector is used for the product detector and how the product detector is implemented, respectively. The experiments in Section 7.1.1 showed that training a model on the SKU-110K-VS dataset achieved the highest mAP on the Albert Heijn Supermarket test set without taking classification into account. From this, the conclusion can be drawn that training a model on solely the SKU-110K-VS dataset is best for product localization in the Albert Heijn supermarket and that augmenting the SKU-110K dataset to simulate visual servoing helps in increasing model localization performance. However, when the classification task was introduced, Section 7.1.2 showed that pre-training on the SKU-110K dataset achieved the highest mAP on the Albert Heijn Supermarket test set (with product classification). Fine-tuning without freezing any layers yielded the highest mAP on the Albert Heijn Supermarket test set, compared to training with 3, 5, 7, or 9 frozen layers. Finally, Section 7.3.1 showed that mean Average Precision (mAP) can indicate the grasp performance by confirming that the pre-trained YOLOv6 medium model enabled the product grasping pipeline to achieve a higher success rate than the YOLOv6 medium model that is trained solely on the Albert Heijn Supermarket dataset.

Furthermore, rotation compensation and non-maximum suppression (NMS) are used to increase the performance of the product detector. Rotation compensation is used to make the detection performance of the product detector independent of the orientation of the manipulator. The experiments in Section 7.3.1 concluded that the impact of rotation compensation on the success of the grasping pipeline is large, increasing the success rate from 0.5 to 0.9. Non-maximum suppression (NMS) is used to suppress predictions with low importance. Experiments in Section 7.1.3 showed that both class-agnostic and intra-class NMS result in the same mAP on the Albert Heijn Supermarket test set, while Section 7.3.1 suggests that intra-class NMS compared to class-agnostic NMS improves the success of the product grasping pipeline slightly. Therefore intra-class NMS is best to use in the product grasping pipeline.

The method and implementation used to overcome the third challenge, regarding **product grasp pose estimation**, are addressed in Section 4 and Section 6.4, respectively. The product grasp poses are estimated by assuming that the optimal suction cup grasp location for supermarket products is in their center, normal to their surface. Position estimation is done by using the depth estimation of the realsense camera and by using the pinhole camera model. The orientation estimation for each product is done by fitting a plane over the pointcloud of the product. Both the position and orientation estimation performed better when the products were closer to the camera, which verifies that visual servoing can refine the product poses during the picking sequence. Furthermore, bounding box underestimation improved both the position and orientation estimation as well, because it ensures that no environmental pixels are considered in the pose estimation. Therefore, the bounding boxes used for the pose estima-

tion are 50% smaller to make overestimation unlikely, and underestimation more likely.

Section 7.2.2 showed that calculating the entire pointcloud for each product leads to high inference times per frame, ranging from 28.5 ms (one bounding box per frame) to 213 ms (30 bounding boxes per frame), meaning that the operation rate of the grasping pipeline is at best 35 Hz, but can drop to 5 Hz, leading to a lower success rate as demonstrated by the test in Section 3. Therefore, random sampling of the depth data is proposed to efficiently calculate the pointcloud, making the number of pixels evaluated independent of the size of the bounding box. Sampling 125 pixels per bounding box was shown to maintain a tolerable ($\pm 1°$ at 30 cm with 95% confidence) orientation accuracy, while increasing calculation speed significantly, from 28.5 ms to 10.9 ms for a frame with a single bounding box and from 213 ms to 28.7 ms for a frame with 30 bounding boxes. The random sampling approach enabled the product grasping pipeline to operate at least 30 Hz at all times. As demonstrated by Section 7.3.4, operating at 30 Hz increases the success rate of the grasping pipeline from 0.8 to 0.9.

Lastly, **visual servoing** and **product tracking** are used to perform temporal reasoning, and thus to overcome the fourth challenge stated in Section 1. A Kalman Filter [21] is used to track each product separately, and the detected poses in each frame are assigned to the Kalman Filter of a product using the Hungarian Algorithm [24]. The classifications and scores are also counted and kept track of separately. The product choosing is done in combination with the tracker, for which two methods are proposed, of which both performed equally well during the ablation study. Experiments in Section 7.3.4 showed that visual servoing is vital to the success of the product grasping pipeline, increasing the success rate from 0.5 to 0.9. The experiments in Section 7.3.3 showed that the use of the product tracker improved grasp success as well, increasing the success rate from 0.8 to 0.9. Additionally, Section 7.3.3 showed that choosing products based on score or occurrence does not change the success rate of the pipeline. Finally, while most work on visual servoing [17, 10, 47, 39] suggests an operation rate of 30-100 Hz, an operation rate of 10 Hz already enabled the product grasping pipeline to successfully grasp products, with a slight decrease in success rate compared to an operation rate of 30 Hz.

While one supermarket shelf grasping method exists [3], comparison with the method in terms of picking success is not possible. The proposed grasping pipeline managed to enable the robot to robustly and consistently pick the correct products from the shelf with the proposed product grasping pipeline, answering the main research question stated at the beginning of this Section.

9

Future Works

This section dives into the future work that can be done regarding the development of a robotic product grasping pipeline in the supermarket.

The proposed Albert Heijn Supermarket dataset consisted of images containing stacked shelves that are annotated with bounding boxes. Because transfer learning succeeded in improving the performance of the YOLOv6 models, it can be concluded that the current dataset does not contain enough data to optimally train the YOLOv6 models. Also, in the current dataset, not all products that occur on the shelves are annotated due to their low occurrence. Making sure that all products are annotated can enable an object detector can learn what products are, and does not punish one if it correctly detects unlabeled products. However, if the product localization and classification are split, the product locator could be trained on just the SKU-110K-VS dataset, as demonstrated in Section 7.1.1. Then, the need to annotate every object on the shelf dissipates, because only the products themselves are necessary to train a product classifier. Alternatively, the labels of the dataset could also be defined differently, representing optimal grasps or multiple possible grasps per product. Using such a dataset enables the use of end-to-end grasp estimation models given they perform the classification task.

The object detector that is used to detect the products on the shelves is a YOLOv6 V3.0 model [25], pre-trained on the SKU-110K dataset and then transferred to the Albert Heijn Supermarket dataset via fine-tuning. While fine-tuning with 3, 5, 7, or 9 frozen layers proved ineffective, training with 1 or 2 frozen layers is not explored in this thesis and might lead to better performance. Additionally, the current implementation of rotation compensation leads to information loss due to the image transformation. It would be optimal if the image transformation would also ensure that the entire original image remains visible. Moreover, the confidence threshold, as mentioned in Section 3.2, is not tuned in this work. Tuning it to another value might improve the product grasping pipeline as well.

Other object detection models, however, could also be implemented. Section 7.3.4 showed that operation at 10 Hz only impacted the grasping pipeline slightly, which might become negligible if the performance of the product detector is increased. Vision Transformers managed to outperform many previously state-of-the-art object detectors, but are larger and slower models. To make transformers smaller and run faster, network pruning [5] can be used. Network pruning is a method where only the most important parameters of a neural network are saved, effectively making the models small and run more efficiently. Pruning is already used to make transformers for Natural Language Processing [4] and Vision Transformers [50] faster. Pruned transformers could be of use in the grasping pipeline and are therefore an interesting research direction.

Additionally, since there exist thousands of products in a real supermarket scenario, other classifiers could also be of interest. In such a system the product locator and classifier are separated. Then, the YOLOv6 medium model trained on the SKU-110K-VS dataset can be used as a product locator, as described in Section 7.1.1. The classifier can be implemented differently. A compelling alternative could be a classifier that uses few-shot learning [45] to overcome the barrier of adding products to the

supermarket collection and re-branding products. Another interesting method of classifying products could be similar to the method proposed by Bajracharya et al. [3], where metric learning is used to compare the detections with a database of products. Using a separate product classifier is likely the most promising method in extending the graspable product selection from 36 products to thousands of products.

To improve the current product grasp pose estimation, the outlier detection via a hand-crafted band-pass filter might be improved by using RANSAC [11] and simultaneously increase the plane fit calculation time. This might improve the current implementation of the orientation estimation. Other methods of estimating the orientations of pointclouds can also be explored. More valuable would be, however, if the object pose could directly be estimated from the depth information, to remove the need for the expensive pointcloud calculation.

Additionally, while the product grasp pose estimation works for the objects in the Albert Heijn Supermarket dataset that are often flat and rectangular, it is more difficult to generalize to other, more oddly shaped objects. Section 7.3.2 demonstrates that round, cylindrical objects can be picked as well, despite the lower accuracy of the orientation estimation for round objects discussed in Appendix A. Other oddly shaped products are not tested, but the pose estimation must likely be adapted to incorporate these.

The first reason is because of the optimal grasp pose for the suction cup assumption. If the product is nonsymmetrical or has an uneven surface, the grasp location may be somewhere else where the object allows a better grip for the gripper. Secondly, the assumption only works for the suction cup gripper, and not for parallel grippers or other grippers. Therefore, the method should be adapted to the other gripper if another one were to be used. Thirdly, the plane fit that is used to estimate the orientation works best for objects with a flat front and might perform worse on other objects that do not have a flat front, as demonstrated for round, cylindrical objects.

To ensure that all types of products can be grasped in the supermarket, oddly shaped products and products that are non-rigid and/or porous must be included as well. Data-driven grasp pose estimators can be used to include oddly shaped products and to generalize to other grippers so that products do not necessarily have to be rigid or non-porous. Methods that do object grasp pose estimation [42] for isolated objects either directly from depth data or RGB-D data are most interesting to incorporate in the current pipeline, because they can remove the need for the pointcloud calculation. After all, such a method can easily replace the grasp pose estimation module without the need for complex adjustments in the other modules. Replacing the module like this also allows the use of grasp pose estimators that do not perform classification.

For product tracking, a Kalman Filter [21] is used with a state-space model that assumes no velocity. While being effective, as demonstrated in Section 7.3.3, without allowing velocity, the tracker is not able to track the products if they are moved by other agents in the supermarket during the picking sequence. To allow the involvement of other agents during the picking sequence, at least another state space model must be used that allows the product velocities to be tracked as well. If the detection rate is high enough the robot might be able to track moving products with a Kalman Filter as well. However, because these motions are likely non-linear, exploring the use of an Extended Kalman Filter [19] can be of interest. If the motion model is unknown and/or the noise is non-Gaussian, however, a method like Particle Filters [8] might be an interesting research direction.

The product-choosing methods proved to be reliable most of the time during the ablation study. The proposed methods, however, do not consider the case where objects are stacked and the top product should be chosen instead of the bottom product. Additionally, the switching behavior at the end of the picking sequence as observed during the ablation study might be overcome if the distance towards the chosen product is taken into account as well. This allows the max_skipped_frames parameter to remain low, to ensure that the early detections during the picking sequence do not gain more influence in the product choosing than the final detections. A method that weighs the distance and the score is proposed by Morisson et al. [31] that implemented a grasp proposal network that also performed visual

servoing for objects on a flat surface.

The use of Optimization Fabrics to generate collision-free trajectories proved to work well in the grasping pipeline. Exploring other methods, however, might be of interest as well concerning improving the grasping pipeline.

Furthermore, the picking scenario for each product is currently constrained to shelves only. Most supermarkets, however, contain hooks, vegetable boxes, and refrigerated shelves with doors as well. These scenarios must be included to ensure that all products can be grasped by the robot in the supermarket. This means that the picking scenario must be detected as well. Bajracharya et al. [3] implemented a module that can detect picking scenarios and can adapt the picking strategy accordingly.

Finally, interacting with humans and reacting to human behavior during the picking process to ensure safety is the final crucial challenge that must be overcome to move toward the integration and deployment of robotics in supermarkets. Making the manipulator compliant when in contact with a human and ensuring that the robot's movements and decision-making are intuitive for humans are examples of challenges that must be overcome regarding human-robot interaction.

References

[1] July 2016. URL: https://nieuws.ah.nl/feiten-en-cijfers/.

[2] Anoesjka Aspeslagh and Stefan Jordan. *Albert Heijn geeft inzicht in keten met wereldkaart*. Aug. 2019. URL: https://nieuws.ah.nl/albert-heijn-geeft-inzicht-in-keten-met-wereldkaart/#:~:text=Het%20assortiment%20van%20de%20supermarkt,dan%2046.000%20producten%20in%20totaal..

[3] Max Bajracharya et al. "Demonstrating Mobile Manipulation in the Wild: A Metrics-Driven Approach". en. In: *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023. ISBN: 978-0-9923747-9-2. DOI: 10.15607/RSS.2023.XIX.055. URL: http://www.roboticsproceedings.org/rss19/p055.pdf (visited on 09/25/2023).

[4] Maximiliana Behnke and Kenneth Heafield. "Losing Heads in the Lottery: Pruning Transformer Attention in Neural Machine Translation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2664–2674. DOI: 10.18653/v1/2020.emnlp-main.211. URL: https://aclanthology.org/2020.emnlp-main.211 (visited on 09/25/2023).

[5] Davis Blalock et al. *What is the State of Neural Network Pruning?* arXiv:2003.03033 [cs, stat]. Mar. 2020. URL: http://arxiv.org/abs/2003.03033 (visited on 09/25/2023).

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv:2004.10934 [cs, eess]. Apr. 2020. URL: http://arxiv.org/abs/2004.10934 (visited on 08/14/2023).

[7] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. arXiv:2005.12872 [cs]. May 2020. URL: http://arxiv.org/abs/2005.12872 (visited on 09/20/2023).

[8] P.M. Djuric et al. "Particle filtering". In: *IEEE Signal Processing Magazine* 20.5 (Sept. 2003). Conference Name: IEEE Signal Processing Magazine, pp. 19–38. ISSN: 1558-0792. DOI: 10.1109/MSP.2003.1236770. URL: https://ieeexplore.ieee.org/abstract/document/1236770 (visited on 09/25/2023).

[9] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929 [cs]. June 2021. DOI: 10.48550/arXiv.2010.11929. URL: http://arxiv.org/abs/2010.11929 (visited on 09/20/2023).

[10] Bernard Espiau, François Chaumette, and Patrick Rives. "A new approach to visual servoing in robotics". In: *ieee Transactions on Robotics and Automation* 8.3 (1992), pp. 313–326.

[11] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://dl.acm.org/doi/10.1145/358669.358692 (visited on 09/25/2023).

[12] Patrick Follmann et al. *MVTec D2S: Densely Segmented Supermarket Dataset*. arXiv:1804.08292 [cs]. July 2018. URL: http://arxiv.org/abs/1804.08292 (visited on 09/04/2023).

[13] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. arXiv:2107.08430 [cs]. Aug. 2021. URL: http://arxiv.org/abs/2107.08430 (visited on 08/14/2023).

[14] Marian George and Christian Floerkemeier. "Recognizing Products: A Per-exemplar Multi-label Image Classification Approach". en. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 440–455. ISBN: 978-3-319-10605-2. DOI: 10.1007/978-3-319-10605-2_29.

[15] Ross Girshick. *Fast R-CNN*. arXiv:1504.08083 [cs]. Sept. 2015. DOI: 10.48550/arXiv.1504.08083. URL: http://arxiv.org/abs/1504.08083 (visited on 09/13/2023).

[16] Eran Goldman et al. *Precise Detection in Densely Packed Scenes*. arXiv:1904.00853 [cs]. Apr. 2019. URL: http://arxiv.org/abs/1904.00853 (visited on 08/12/2023).

[17] S. Hutchinson, G.D. Hager, and P.I. Corke. "A tutorial on visual servo control". en. In: *IEEE Transactions on Robotics and Automation* 12.5 (Oct. 1996), pp. 651–670. ISSN: 1042296X. DOI: 10.1109/70.538972. URL: http://ieeexplore.ieee.org/document/538972/ (visited on 10/13/2023).

[18] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics.

[19] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Vol. 3068. SPIE, July 1997, pp. 182–193. DOI: 10.1117/12.280797. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3068/0000/New-extension-of-the-Kalman-filter-to-nonlinear-systems/10.1117/12.280797.full (visited on 09/25/2023).

[20] Philipp Jund et al. *The Freiburg Groceries Dataset*. arXiv:1611.05799 [cs]. Nov. 2016. URL: http://arxiv.org/abs/1611.05799 (visited on 09/04/2023).

[21] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". en. In: *Journal of Basic Engineering*. Vol. 82. ISSN: 0021-9223 Issue: 1. Mar. 1960, pp. 35–45. DOI: 10.1115/1.3662552. URL: https://asmedigitalcollection.asme.org/fluidsengineering/article/82/1/35/397706/A-New-Approach-to-Linear-Filtering-and-Prediction (visited on 09/25/2023).

[22] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs]. Jan. 2017. DOI: 10.48550/arXiv.1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 09/27/2023).

[23] Marcus Klasson, Cheng Zhang, and Hedvig Kjellström. *A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels*. arXiv:1901.00711 [cs]. Jan. 2019. URL: http://arxiv.org/abs/1901.00711 (visited on 09/04/2023).

[24] H. W. Kuhn. "The Hungarian method for the assignment problem". en. In: *Naval Research Logistics Quarterly* 2.1-2 (1955). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109, pp. 83–97. ISSN: 1931-9193. DOI: 10.1002/nav.3800020109. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109 (visited on 09/20/2023).

[25] Chuyi Li et al. *YOLOv6 v3.0: A Full-Scale Reloading*. 2023. arXiv: 2301.05586 [cs.CV].

[26] Chuyi Li et al. *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. arXiv:2209.02976 [cs]. Sept. 2022. URL: http://arxiv.org/abs/2209.02976 (visited on 08/14/2023).

[27] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. arXiv:1708.02002 [cs]. Feb. 2018. URL: http://arxiv.org/abs/1708.02002 (visited on 08/12/2023).

[28] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. arXiv:1405.0312 [cs]. Feb. 2015. DOI: 10.48550/arXiv.1405.0312. URL: http://arxiv.org/abs/1405.0312 (visited on 09/24/2023).

[29] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: vol. 9905. arXiv:1512.02325 [cs]. 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: http://arxiv.org/abs/1512.02325 (visited on 08/12/2023).

[30] Michele Merler, Carolina Galleguillos, and Serge Belongie. "Recognizing Groceries in situ Using in vitro Training Data". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919. June 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383486.

[31] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Learning robust, real-time, reactive robotic grasping". In: *The International journal of robotics research* 39.2-3 (2020), pp. 183–201.

[32] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1345–1359. ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[33] Jingtian Peng, Chang Xiao, and Yifan Li. *RP2K: A Large-Scale Retail Product Dataset for Fine-Grained Image Classification*. arXiv:2006.12634 [cs]. Sept. 2021. URL: http://arxiv.org/abs/2006.12634 (visited on 09/06/2023).

[34] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[35] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. arXiv:1612.08242 [cs]. Dec. 2016. URL: http://arxiv.org/abs/1612.08242 (visited on 08/12/2023).

[36] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[37] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. arXiv:1506.02640 [cs]. May 2016. URL: http://arxiv.org/abs/1506.02640 (visited on 08/12/2023).

[38] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv:1506.01497 [cs]. Jan. 2016. DOI: 10.48550/arXiv.1506.01497. URL: http://arxiv.org/abs/1506.01497 (visited on 09/13/2023).

[39] "Robust Image Processing and PositionBased Visual Servoing". en. In: *Robust Vision for Vision-Based Control of Motion*. IEEE, 2009. ISBN: 978-0-470-54636-9. DOI: 10.1109/9780470546369.ch13. URL: https://ieeexplore.ieee.org/document/5264748 (visited on 10/13/2023).

[40] Max Spahn, Martijn Wisse, and Javier Alonso-Mora. "Dynamic Optimization Fabrics for Motion Generation". In: *IEEE Transactions on Robotics* 39.4 (Aug. 2023). Conference Name: IEEE Transactions on Robotics, pp. 2684–2699. ISSN: 1941-0468. DOI: 10.1109/TRO.2023.3255587. URL: https://ieeexplore.ieee.org/abstract/document/10086617 (visited on 09/25/2023).

[41] Mingxing Tan, Ruoming Pang, and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. arXiv:1911.09070 [cs, eess]. July 2020. DOI: 10.48550/arXiv.1911.09070. URL: http://arxiv.org/abs/1911.09070 (visited on 09/13/2023).

[42] Hongkun Tian et al. "Data-driven robotic visual grasping detection for unknown objects: A problem-oriented review". In: *Expert Systems with Applications* 211 (Jan. 2023), p. 118624. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.118624. URL: https://www.sciencedirect.com/science/article/pii/S0957417422016736 (visited on 10/04/2023).

[43] Gül Varol and Rıdvan Salih. "Toward retail product recognition on grocery shelves". In: Mar. 2015, p. 944309. DOI: 10.1117/12.2179127.

[44] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *arXiv preprint arXiv:2207.02696* (2022).

[45] Yaqing Wang et al. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Computing Surveys* 53.3 (June 2020), 63:1–63:34. ISSN: 0360-0300. DOI: 10.1145/3386252. URL: https://dl.acm.org/doi/10.1145/3386252 (visited on 09/25/2023).

[46] Xiu-Shen Wei et al. *RPC: A Large-Scale Retail Product Checkout Dataset*. arXiv:1901.07249 [cs]. Jan. 2019. URL: http://arxiv.org/abs/1901.07249 (visited on 09/04/2023).

[47] W.J. Wilson. "Visual Servo Control of Robots Using Kalman Filter Estimates of Relative Pose". en. In: *IFAC Proceedings Volumes* 26.2 (July 1993), pp. 633–638. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)48804-5. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667017488045 (visited on 10/13/2023).

[48] Shangliang Xu et al. *PP-YOLOE: An evolved version of YOLO*. arXiv:2203.16250 [cs]. Dec. 2022. URL: http://arxiv.org/abs/2203.16250 (visited on 08/14/2023).

[49] *YOLOv5*. https://docs.ultralytics.com/models/yolov5/. Accessed: 2023-08-14.

[50] Mingjian Zhu, Yehui Tang, and Kai Han. *Vision Transformer Pruning*. arXiv:2104.08500 [cs]. Aug. 2021. DOI: 10.48550/arXiv.2104.08500. URL: http://arxiv.org/abs/2104.08500 (visited on 09/25/2023).

[51] Zhengxia Zou et al. “Object Detection in 20 Years: A Survey”. In: *Proceedings of the IEEE* 111.3 (Mar. 2023). Conference Name: Proceedings of the IEEE, pp. 257–276. ISSN: 1558-2256. DOI: 10.1109/JPROC.2023.3238524.

A

Round pose estimation

The goal if this section is to estimate whether the pose estimation is accurate enough to estimate the poses for round objects as well. Similarly to the tests with a flat object, the pose estimation is more accurate at a distance of 30 cm than at a distance of 100 cm. Especially the orientation estimation (ϕ and θ) are worse, with a larger offset and confidence interval than for flat objects. This is because the plane fit does not work well on a round surface, leading to a higher confidence interval and offset. So, compared to the flat object, estimating the orientation with a plane fit proves more difficult. For round objects, however the θ estimation might not as relevant as for flat objects, because a round object does not have a specific θ orientation. Section 7.3.2 will conclude that round, cylindrical objects can also be picked with plane fit orientation estimation.

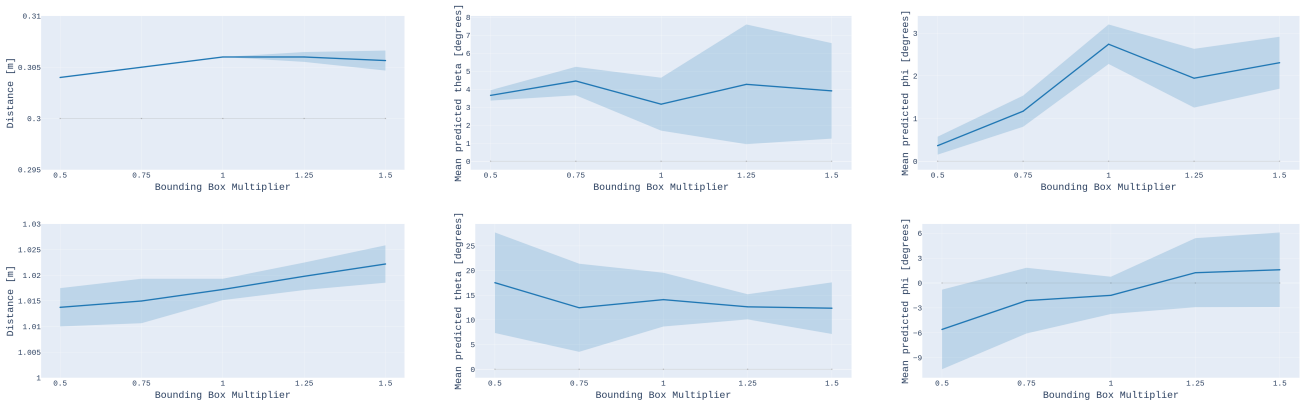

Figure A.1: shows the pose estimation results for a round object at a distance of 30 cm (upper) and 100 cm (lower). The same offsets of 4 mm and 10 mm are visible in the distance estimation at 30 cm and 100cm due to the test setup. in The x-axis shows the bounding box over- or underestimation factor; The y-axis shows the mean with 95% confidence interval ($\pm 1.96\sigma$) in degrees. The left plot shows the distance estimation, the middle plot shows the θ estimation, and the right plot shows the ϕ estimation. In all tests ϕ = 0°

B

Successful Grasp Examples

Examples of grasps for each of the products used in the ablation study are shown here in Figures B.1, B.2, B.3, B.4, and B.5.

Figure B.1: shows an example of picking the 'Brownie' product.

Figure B.2: shows an example of picking the 'Nasi Speciaal' product.

Figure B.3: shows an example of picking the 'Volle Yoghurt' product.

Figure B.4: shows an example of picking the 'Mais' product.

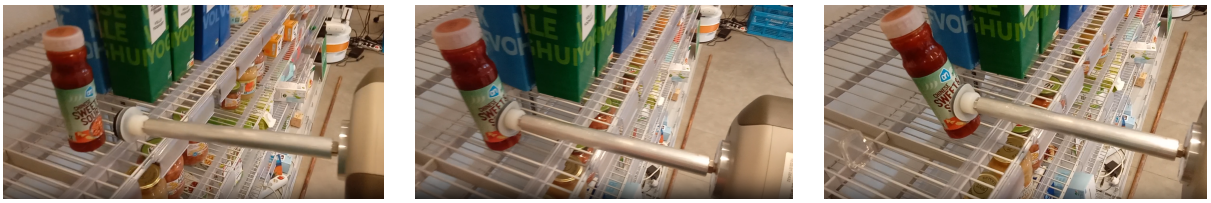

Figure B.5: shows an example of picking the 'Sweet Sour' product. This product is too light to grasp with the current suction gripper due to the low vacuum power.