



DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

DIOGO LÚCIO FINO

BSc in Electrical and Computer Engineering

ROBOTIC MANIPULATION OF SOFT-TISSUES IN AGRI-FOOD PROCESSING

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING
SPECIALIZATION IN ROBOTICS AND INTEGRATED MANUFACTURING

NOVA University Lisbon
September, 2025



ROBOTIC MANIPULATION OF SOFT-TISSUES IN AGRI-FOOD PROCESSING

DIOGO LÚCIO FINO

BSc in Electrical and Computer Engineering

Adviser: André Rocha

Assistant Professor, NOVA University of Lisbon

Co-advisers: Nelson Freitas

Master, NOVA University of Lisbon

Miguel Arvana

Master, NOVA University of Lisbon

Examination Committee

Chair: Luís Bernardo

Associate Professor, NOVA University of Lisbon

Rapporteur: José Barata de Oliveira

Full Professor, NOVA University of Lisbon

Adviser: André Rocha

Assistant Professor, NOVA University of Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING
SPECIALIZATION IN ROBOTICS AND INTEGRATED MANUFACTURING

NOVA University Lisbon
September, 2025

Robotic manipulation of soft-tissues in agri-food processing

Copyright © Diogo Lúcio Fino, NOVA School of Science and Technology, NOVA University Lisbon.

The NOVA School of Science and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

This document was created with the (pdf/Xe/Lua) $\LaTeX$ processor and the [NOVAthesis](#) template (v7.5.0) .

Lourenço, J. M. (2021). *The NOVAthesis $\LaTeX$ Template User's Manual*. NOVA University Lisbon. <https://github.com/joaomlourenco/novathesis/raw/main/template.pdf>

Aos meus Pais e Amigos

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my advisors, Professor André Rocha, Nelson Freitas, and Miguel Arvana, for their guidance, constructive feedback, and encouragement during this work development and research. Their expertise was fundamental to the completion of this work. I would like to extend my gratitude for Nova School of Science and Technology for providing the resources and tools for this journey. Finally, my heartfelt thanks go to my family and friends.

To all of you,

Thank you

”

“A experiência é a professora de todas as coisas”

— **Caio Júlio César**, *De Bello Civili*
(Estadista e General Romano)

ABSTRACT

Many challenges in the food industry have arisen from the rapidly growing population increasing the food demand and, by consequence, creating a need to produce higher quantities of food products. Additionally, the growing concerns about climate change and the increased environmental awareness among the population have created requirements for the industry in the food sector to adopt more efficient practices with lower environmental impacts. To combat these problems, many businesses have deployed robotic systems, which have already proven to be an effective way to boost productivity rates and reduce the creation of waste. In the food industry there is also the concern with hygiene and potential contamination problems, these concerns create some challenges for the employed robotic systems. The proposed work for this dissertation is the creation of a robotic system to manipulate food products, specially smoked and cured meats like *chouriço*. The system is composed of two parts, one dedicated to the manipulation and the other dedicated to the vision capabilities of the system in order to identify and locate the various food items. The manipulation part of the system is composed of fin ray effect gripper, 3D printed using TPU filament. This type of gripper is capable of deforming around object creating a more secure grasp on the various products, while limiting the risk of damage to them. The developed object pose estimation algorithm, for the vision part of the system, makes use of certain keypoints on the products to calculate their pose in relation to the camera system. This approach does not require complex data about the objects to be detected, like point clouds. It just the requires a general knowledge about the shape of the products. The developed fin ray effect gripper proved to be adequate in handling the three chosen test products. The developed object pose estimation algorithm archived a success rate of around 80% for products that lay flat on the workspace table. Around the same success rate for the products that present an angle against the table. And a lower rate, of around 72% in cases of overlapping products.

Keywords: Agri-food · Soft manipulation · Soft gripper · Computer vision · Robotic systems · Automation

RESUMO

Vários desafios na indústria alimentar surgiram devido ao rápido crescimento da população, que criou um aumento na procura de alimentos e, por consequência, criou uma necessidade de aumentar a produtividade. A crescente preocupação com as alterações climáticas e uma maior consciência ambiental entre a população exigem que a indústria do setor alimentar adote práticas mais eficientes e reduza impactos ambientais. Para enfrentar estes problemas, muitas empresas implementaram sistemas robóticos, que provaram ser uma forma eficaz de aumentar a taxa de produtividade e reduzir o número de resíduos. Na indústria alimentar existe também uma preocupação com a higiene e potenciais focos de contaminação, o que adiciona mais desafios sob os sistemas robóticos utilizados. O trabalho proposto por esta dissertação é a criação de um sistema robótico para manipulação de produtos alimentares, em específico carnes fumadas e curadas, como chouriço e outros enchidos. O sistema é dividido em duas partes, uma para a manipulação dos produtos alimentares, e outra o sistema de visão computacional para identificação e localização dos produtos alimentares. A parte de manipulação de produtos utiliza uma gripper "fin ray effect", construída através de impressão 3D, com a utilização de TPU. Este tipo de gripper tem a capacidade de deformar-se em torno no object, o que cria uma pega à volta dos produtos mais segura, ao mesmo tempo que limita a possibilidade de danos e estragos feitos aos mesmos. O algoritmo de "*object pose estimation*", desenvolvido para a parte de visão do sistema robótico, utiliza certos pontos chave dos produtos para o cálculo da sua pose em relação à câmara do robot. Este método não necessita informação previa complexa, como point clouds, apenas é necessário saber o formato geral dos produtos. A gripper de efeito "*fin ray*" desenvolvida provou ser adequada para a manipulação dos três produtos escolhidos para teste. O algoritmo de "*object pose estimation*" desenvolvido atingiu uma taxa de sucesso de por volta de 80% para produtos que se encontrem planos à mesa de trabalho do robot, uma taxa similar para casos de produtos que se encontrem em angulo, mas uma taxa reduzida de por volta de 72% para casos de sobreposição.

Palavras-chave: Agri-food · Soft manipulation · Soft gripper · Visão computacional · Sistemas robóticos · Automação

CONTENTS

Acronyms	xiv
1 Introduction	1
1.1 Context and Motivation	1
1.2 Proposed work and Goals	2
1.3 Document structure	3
2 State of the art	4
2.1 Agri-food robotics	4
2.1.1 Manipulation, hygiene and safety	7
2.2 Soft tissue manipulation	8
2.2.1 Grippers	8
2.3 Computer vision	16
2.3.1 Computer vision applied in industry	25
2.4 State of the art summary	26
3 Proposed approach	27
3.1 Case Study scenario description	27
3.1.1 Case study objective	28
3.2 Proposed solution	28
3.2.1 Gripper subsystem	29
3.2.2 Vision subsystem	29
3.2.3 Overall robotic system	30
3.3 Evaluation methodology	31
4 Soft-Tissues Gripper Development	33
4.1 Fin ray effect grippers gathering	33
4.2 Fin ray effect gripper requirements	34
4.3 Fin ray effect gripper development	35
4.4 Gripper module	41

4.5	Gripper's digital interface	41
5	Pose Estimation Algorithm Development	44
5.1	Algorithm pipeline	44
5.2	Camera system hardware	45
5.3	Depth and color images preprocessing	45
5.4	Algorithm development	46
5.4.1	First stage - Object detection and segmentation	47
5.4.2	Second stage - Position calculation	52
5.4.3	Third stage - Orientation calculation	54
5.5	Chosen approach and limitations	63
6	Robotic system development	66
6.1	Robotic system hardware and software	66
6.2	Robot's URDF, SRDF and manipulator configurations	67
6.3	Overall ROS controller architecture	68
6.4	Robotic system ROS peripheral nodes	69
6.4.1	Vision system communication node	69
6.4.2	Gripper controller interface node	70
6.4.3	Path planning and move group interface node	70
6.4.4	RViz object visualizer node	71
6.5	Controller nodes	72
6.5.1	Manual controller	73
6.5.2	Automatic controller	76
6.6	Robotic system limitations	77
7	Validation and tests	79
7.1	Fin ray effect gripper validation test	79
7.1.1	Test description	79
7.1.2	Test results	79
7.2	Object pose estimation algorithm validation tests	80
7.2.1	First object pose estimation algorithm test	81
7.2.2	Test results	82
7.2.3	Second object pose estimation algorithm test	83
7.2.4	Test results	83
7.2.5	Third object pose estimation algorithm test	85
7.2.6	Test results	86
7.2.7	Fourth object pose estimation algorithm test	87
7.2.8	Test results	89
7.3	Validation tests conclusion	90
8	Conclusion	93

Bibliography	98
---------------------	-----------

Appendices

A Object pose estimation algorithm's tests values	105
A.1 Values obtained during the first test	105
A.2 Values obtained during the second test	107
A.3 Values obtained during the third test	110
A.4 Values obtained during the fourth test	111

LIST OF FIGURES

2.1	Generic diagram of a pneumatic soft fingers gripper's finger	10
2.2	Generic diagram of a fin ray effect gripper	11
2.3	Generic diagram of an inflatable hook type gripper	12
2.4	Generic diagram of a deformable fingertips gripper	12
2.5	Generic diagram of a bernoulli type gripper	14
2.6	Generic diagram of a vacuum / suction cup gripper	15
2.7	Flowchart of the previously described algorithm [adapted from Gupta et al., 2019]	19
2.8	Flowchart of the previously described algorithm [adapted from Höfer et al., 2021]	20
2.9	Flowchart of the previously described algorithm [adapted from the descriptions in Yun et al., 2017]	20
2.10	Flowchart of the previously described algorithm [adapted from Chen et al., 2020]	21
2.11	Flowchart of the previously described algorithm [adapted from the descriptions in Wu et al., 2022]	22
2.12	Flowchart of the previously described algorithm [adapted from Wong et al., 2023]	23
2.13	Flowchart of the previously described algorithm [adapted from Tran and Lin, 2020]	24
2.14	Flowchart of the previously described algorithm [adapted from the descriptions in Taryudi and Wang, 2017]	24
3.1	Packaging line	27
3.2	Diagram of the robot system thats going to be used. Red for rotations on x-axis, green for y-axis, blue for z-axis.	30
4.1	Dimensions of the first developed fin ray effect gripper finger (in millimeters)	35
4.2	Dimensions of the first developed holder for gripper finger (in millimeters)	36
4.3	Image of the complete assembly of the first version of fin ray gripper assembly	36

4.4	Solution attempt to the stiffness problem on the ends of the finger. left - before, right - after	37
4.5	Dimensions of the second developed fin ray effect gripper finger (in millimeters)	38
4.6	Dimensions of the redesign extension part of the holder (in millimeters) . .	39
4.7	Dimensions of the second version of the developed holder for gripper finger (in millimeters)	39
4.8	Image of the complete assembly of the second version of fin ray gripper assembly	39
4.9	Dimensions of the third version of the developed holder for gripper finger (in millimeters)	40
4.10	Image of the complete assembly of the third version of fin ray gripper assembly	40
4.11	Image of the complete 3D module of the gripper	42
4.12	Image python script for testing the gripper HTTP request services	42
5.1	High level flowchart of the developed algorithm	45
5.2	Intel - Depth Quality Tool for Intel RealSense Cameras	46
5.3	Results of the training of the first model	48
5.4	Results of the training of the second model model	49
5.5	Results of the training of the third model model	50
5.6	Speed test for the first three trained models (resolution 640x480)	50
5.7	Results of the training of the fourth model model	51
5.8	Results of the training of the final model model	51
5.9	Result of the model for a images with all products. The right image shows an example with one of the products being occluded by another	52
5.10	Simplified pipeline for the average depth method for the calculation of the center point of the object	53
5.11	Simplified pipeline for the point cloud method for the calculation of the center point of the object	54
5.12	Comparison between the depth values method and the point cloud method	55
5.13	Visual representation of the Principal Components Analysis (PCA) method (source: Nicoguardo, 2016)	55
5.14	Results of orientation calculation method that uses PCA on the segmentation mask. On the right image the product has some inclination.	56
5.15	Results of orientation calculation method that uses PCA on the object's point cloud. On the right image the product has some inclination.	58
5.16	Results of orientation calculation method that uses Ordinary Least Squares (OLS) on the object's point cloud. On the right image the product has some inclination.	59
5.17	Results of orientation calculation method that uses the combined OLS on the object's point cloud with PCA on the segmentation mask pixels. On the right image the product has some inclination.	60

5.18	Location of the keypoints on a product.	61
5.19	Results of orientation calculation method previously discussed (keypoints method). On the right image the product has some inclination.	63
5.20	High level flowchart of the complete vision algorithm	64
6.1	Robotic system connections	67
6.2	Robotic system architecture	69
6.3	Three robot's hard coded positions (from left to right: home position, upper home position, calibration position)	72
6.4	Manual controller Graphical User Interface (GUI)	73
6.5	Automatic controller flowchart	78
7.1	Successful fin ray effect gripper agri-food grasp	80
7.2	Graphs containing the average absolute errors for each axis	86
7.3	Graphs containing the average absolute errors for each axis	87
7.4	Example of placed products for the fourth test	88
7.5	Edited image with the position error demonstrated	92

LIST OF TABLES

7.1	Fin ray effect gripper test results	80
7.2	List of the various poses chosen for the first test	81
7.3	First test pick and place operation results. Yes* - represents a successful pick but with an unstable grasp. Yes** - represents a successful pick but the product was heavily adjusted	82
7.4	List of the various poses chosen for the second test	84
7.5	Second test pick and place operation results	85
7.6	Second test pick and place operation success rates for the combined poses .	86
7.7	List of the various poses chosen for the third test	87
7.8	Second test pick and place operation results	88
7.9	List of the various poses chosen for the fourth test	89
7.10	Fourth test pick and place operation results	89

ACRONYMS

ANFIS	Adaptive Neuro Fuzzy Inference System (<i>p. 24</i>)
CAD	Computer Aided Design (<i>p. 19</i>)
CNN	Convolutional Neural Network (<i>pp. 16, 21</i>)
DNN	Deep Neural Network (<i>p. 25</i>)
DoF	Degrees of Freedom (<i>pp. 18, 30, 66, 93</i>)
GUI	Graphical User Interface (<i>pp. xii, 73, 74, 82</i>)
HSV	Hue Saturation Brightness (<i>pp. 24, 26, 30</i>)
ICP	Iterative Closest Point (<i>pp. 19, 23</i>)
IOT	Internet of Things (<i>pp. 4, 5</i>)
IR	Infrared (<i>p. 45</i>)
ISO	International Organization for Standardization (<i>pp. 7, 13, 15</i>)
LED	Light-Emitting Diode (<i>p. 41</i>)
LiDAR	Light Detection And Ranging (<i>p. 17</i>)
NAT	Network Address Translation (<i>p. 66</i>)
OLS	Ordinary Least Squares (<i>pp. xi, 55, 57–60, 63, 64, 95</i>)
PCA	Principal Components Analysis (<i>pp. xi, 54–61, 63–65, 95</i>)
PLA	Polylactic acid (<i>p. 41</i>)
PVC	polyvinyl chloride (<i>p. 9</i>)
R-CNN	Region Based Convolutional Neural Network (<i>pp. 16, 18, 23</i>)

RANSAC	Random Sample Consensus (<i>p. 20</i>)
RGB	Red Green Blue (<i>pp. 17–19, 22, 23, 26, 29, 30, 44–47, 94</i>)
RGB-D	Red Green Blue Depth (<i>pp. 17, 18, 21–23, 26, 29, 30, 45</i>)
RNN	Recurrent Neural Network (<i>p. 16</i>)
ROS	Robot Operating System (<i>pp. 30, 31, 43, 57, 66–70, 73, 76, 89–92, 95</i>)
SCARA	Selective Compliance Assembly Robot Arm (<i>p. 6</i>)
SRDF	Semantic Robot Description Format (<i>p. 68</i>)
SSD	Single Shot Detector (<i>p. 16</i>)
SVD	Singular Value Decomposition (<i>pp. 23, 55</i>)
TCP	Transmission Control Protocol (<i>pp. 67, 69</i>)
TPU	Thermoplastic polyurethane (<i>pp. 13, 34</i>)
URDF	Unified Robot Description Format (<i>pp. 67, 78</i>)
USB	Universal Serial Bus (<i>pp. 45, 49, 67</i>)
YOLO	You Only Look Once (<i>pp. 16, 18, 30, 47, 48, 51, 52, 56, 64, 67, 88, 91, 94</i>)

INTRODUCTION

This first chapter presents the context and motivation for the underlying work, with emphasis on the current trends and challenges of the agri-food sector, and the ongoing process of automation within food industry. This work will focus on the development of a robotic system designed for the needs of this industry sector and on the difficulties associated with implementing such technologies. The objectives and goals of the work, as well as the overall document structure, are also detailed here.

1.1 Context and Motivation

The rapidly growing population, combined with the increasing demand for higher food production, has intensified the pressure on the available existing farmland (Raffik et al., 2023). Additionally, heightened environmental awareness and better understanding of the impacts of climate change among the population, have compelled the agri-food industry to increase its productivity while adhering to sustainable practices (Derossi et al., 2023). These factors, along with the shortage of workers in the sector (Raffik et al., 2023; Derossi et al., 2023), create significant obstacles for the agri-food sector to maintain adequate levels of production.

There are already several robots introduced in the agri-food sector, proving that automation is an effective way to boost the productivity and reduce resource wastage, thus contributing to more sustainable solutions (Raffik et al., 2023; Derossi et al., 2023; Konfo et al., 2023; Duong et al., 2020). However many areas of the food industry still rely heavily on manual human labor (de Medeiros Esper et al., 2021). This reliance stems from the challenges associated with the automation of this sector. Particularly those related to the safe manipulating agri-food products, due to their irregularity, fragile nature, and the possibility of contamination (Raffik et al., 2023).

The main problem addressed in this work concerns the manipulation of agri-food products mainly composed of soft-tissues, such as smoked sausages, using a robotic manipulator. The proposed robotic system is designed for a real life scenario of food

industry. In this context the system has the task of manipulating three types of agri-food products without damaging or contaminating them.

1.2 Proposed work and Goals

The aim of this work is to create an autonomous robotic system for manipulation of smoked and cured meats, such as *chouriço*, sausages, and similar items, moving them from a storage place, a bin for example, to another designated destination. To achieve this purpose, the system necessitates vision capabilities to detect and locate the products, as well as a gripper capable of manipulating food products without damaging them. The three selected products for testing the developed robotic system are *chouriço*, *farinheira*, and *morcela*.

The objectives of the proposed work are the following:

- Conduct a study of different types of grippers for the manipulation of these types of food products.
- Develop and test a gripper capable of safely manipulating the three chosen products.
- Analyze different computer vision algorithms for the identification, detection, and localization of the products.
- Develop a computer vision algorithm for the identification, detection, and localization of the three chosen products.

The developed gripper should be capable of manipulating the three chosen products without damaging them, or leaving permanent marks on the surface of the products. The computer vision algorithm should be capable of detecting and the products' orientation and position (pose) on a environment where there might be some occlusions and product overlapping.

After the previous highlighted goals, the work begins with a study of the current state of the art regarding the various available grippers and computer vision algorithms applied to the agri-food sector. This review provides a foundation for understanding of the current main challenges for soft-tissue manipulating and product localization, different designs of grippers, and various methods for the development of computer vision algorithms. After a review of the state of the art, the development of the gripper follows, with its experimental results guiding the subsequent direction of development of the vision algorithm. After both components are designed, the testing of the overall robotic system is performed. The results obtained during the testing phase will incur the iterative improvements for both the gripper and the computer vision algorithm.

1.3 Document structure

This document has the following structure:

- Chapter 2 presents the current state of the art regarding agri-food robotics, soft tissue manipulation grippers, and computer vision.
- Chapter 3 describes the use case scenario, proposed solution, and the proposed approach.
- Chapter 4 details the development of the gripper and the manipulation system.
- Chapter 5 presents the development of the computer vision algorithm for the detection and localization.
- Chapter 6 describes the development of the robotic system, including its controller and configurations.
- Chapter 7 reports the results of the various tests conducted on the gripper, vision algorithm, and robotic system.

STATE OF THE ART

This chapter presents a review of the current state of the art regarding the topics of agri-food robotics, soft tissue manipulation grippers, and computer vision. The first section provides an overview of the general state of agri-food automation and robotic systems, the technologies involved, and the safety and health concerns to be regarded. In the second section, the topic of grippers designed for the manipulation of soft and fragile objects is explored. The third section discusses the topic of computer vision, focusing primarily on object pose estimation algorithms.

2.1 Agri-food robotics

The continuous growth of the global population has intensified the demand for food products, increasing the pressure on the available agricultural land. At the same time, rising concerns regarding environmental sustainability and impacts of climate change impose additional challenges to the food industry (Raffik et al., 2023). Addressing these demands and challenges while maintaining, or even improving, the original quality of the food products requires new sustainable solutions that combine high efficiency, precision, and wastage control. In this context the integration of digital technologies and robotics into the food industry plays a central role in improving the productivity and working conditions while reducing the environmental impact of industry (Raffik et al., 2023; Derossi et al., 2023; Konfo et al., 2023; Duong et al., 2020).

Konfo et al., 2023 highlights a series of technologies, including the Internet of Things (IOT), big data analytics, artificial intelligence with machine learning and computer vision, robotics, automation, and blockchain, which collectively contribute to increased productivity, improved safety, and enhanced sustainability in the food industry. According to the study, the application of artificial intelligence in the food industry, specifically the field of computer vision, allows the automation of tasks such as detection, classification, and quality inspection. The contribution of robotics in this industry, allowing the automation of repetitive and labor intensive tasks, improves the productivity and efficiency in both

industrial and agricultural environments (Konfo et al., 2023).

Derossi et al., 2023 emphasize the potential of precision manufacturing in creating new sustainable solutions. Allowing the manufacturing of products with higher precision, repeatability, and consistency, diminishing the wastage of resources, and improving the overall quality of the product. Similarly to the study by Konfo et al., 2023, it discusses the use of technologies like artificial intelligence, big data, IOT, digital twins, smart sensors, robotics, and others to assist with precision manufacturing, as well as improve the quality, efficiency, and productivity.

The paper by de Medeiros Esper et al., 2021 discusses the paradigm of cell concept for the industry, where instead of a serial production line with multiple stations, one for every task, there are a multitude of multitasking cells in which all operations are performed, considered more hygienic.

The article by Duong et al., 2020 underlines the historical reliance of the food industry on manual labor, with limited machinery use. However, with the increased demand on food products and the advancement of technology, robotic systems are beginning to be employed in the agri-food sector. Some sectors that have already been automated are the dairy sector, using automated milking machines, and agriculture with soil moisture monitoring systems.

Aly et al., 2023 present a study on the red meat processing industry, highlighting the difficult work conditions, having both physically and mentally demanding tasks. Factors that contribute to these work conditions include the cold ambient temperatures of the work environment and repetitive tasks. The health hazard of contamination by human operators is also a considered factor. The authors propose that the solution for these problems lies in perception technology in order to give robots capabilities similar to humans, assisting in the required operations for the industry.

The robots in the food industry are employed in a wide range of operations, including pick and place handling, packing and palletizing, agricultural tasks such as seeding and spraying water, and processing activities like harvesting, cutting, and slicing (Iqbal et al., 2017). One of the most common tasks for robotic manipulators in the food industry is pick and place. Robots have several advantages over humans in this task. Robots are faster, have better precision, and reduce the potential food contamination problems by limiting human interaction with the items (Raffik et al., 2023). Another common task is packaging and palletizing, preparing the products for storage and distribution. The automation of this task improves the productivity of the factories (Raffik et al., 2023). In processing operations, robots can also perform cutting and slicing of vegetables or meat products. These tasks are challenging due to the natural variability and deformable morphology of

food items, as they interfere with the cutting and compromise the desired final geometry. (Raffik et al., 2023).

The robots used in the food industry can be classified into two main categories, serial manipulators and parallel manipulators (Iqbal et al., 2017). Serial manipulators, which constitute the majority of the robots employed in the industry, consist of a single kinematic chain of sequentially connected links and joints. In contrast, parallel manipulators are composed of a multitude of kinematic chains connected in a parallel format (Yeshmukhametov et al., 2017). Selective Compliance Assembly Robot Arm (SCARA) manipulators are an example of a serial robot widely employed by the food industry. These manipulators are usually used for tasks such as food handling, packing, sorting, and quality inspection. They are also frequently employed in laboratories for food analysis (Raffik et al., 2023). Delta robots, which fall under the category of parallel manipulators, because of their speed are used for high-speed pick and place tasks, assembly and disassembly, and food decoration (Raffik et al., 2023). Another important class of robot is the collaborative robot, designed to safely operate alongside human workers in a shared workspace. These manipulators are usually employed on tasks like pick and place, packing, labeling, and palletizing operations (Raffik et al., 2023).

The article by Duong et al., 2020 identifies several subsectors within the agri-food industry and discusses their current level of automation. In the agriculture sector, there are some developed robotic systems equipped with computer vision subsystems and with picking capabilities for harvesting operations. One example is an autonomous robot for tomato-harvesting operations developed by Yaguchi et al., 2016, which employs a plucking technique and a stereo camera vision system to harvest the product without damaging the plant. A similar example of autonomous system design for the agri-food industry is introduced by Hu et al., 2021, featuring a vision system composed of a depth camera and a specialized gripper designed for citrus harvesting. The employment of autonomous systems is already more established in the livestock sector. The use of milking systems in dairies, as well as food delivering robots. These systems contribute to a more hygienic environment, minimizing contamination by reducing human contact and preventing potential health hazards. Duong et al., 2020 further highlights potential opportunities in the aquaculture sector, where harsh environmental conditions and high operational costs make robotic solutions particularly attractive.

The paper by de Medeiros Esper et al., 2021 examines the integration of robotic systems in the meat processing industry, particularly within slaughterhouses. Slaughterhouses have inherently hazardous working environments due to their low temperatures, required to preserve the meat, wet and noisy conditions, and frequent use of sharp tools. These factors, combined with the repetitive nature of many operations, contribute to a high incidence of worker injuries. The automation of the sector through robotic systems can mitigate these

issues, improving workplace safety, enhancing working conditions, and increasing overall productivity. Moreover, automation of this sector can bring traceability, supporting the identification of potential risks and improving the response for contamination events.

The study Özentürk et al., 2024 examines the use of robotics in poultry farming, which faces several operational challenges, including animal health, poultry house management, and human contamination. The transition towards smart automated systems is driven by the desire for better economic profitability. According to the study, robotic systems are employed in tasks such as monitoring, management, environmental control, and collection of eggs and deceased animals. Maintaining proper animal health conditions is crucial, since disease outbreaks represent significant financial risks and may escalate into large-scale epidemics. Ensuring optimal environmental conditions is also essential for product quality and continuous production. For this purpose robots can be equipped with sensors in order to monitor factors like temperature, humidity, lighting, and ventilation, as well as perform sanitation tasks such as removing feed residues and animal feces, thereby improving health conditions.

2.1.1 Manipulation, hygiene and safety

Food products often present fragile structures, irregular geometries, variable weights, and slippery or sticky surfaces. These characteristics require manipulation systems equipped with sophisticated sensing capabilities and compliant tools to ensure that the products can be handled without causing permanent deformations or damage (Raffik et al., 2023). When designing a robotic manipulator for such items, parameters such as size, shape, dimensions, weight, and the required lifting force must be considered. However, exact precise measurements for these parameters can be difficult to obtain, given the natural variability of food products, especially meat products (Aly et al., 2023).

Hygiene is another important factor to take into consideration when developing a robotic system for the food industry. All components that come into contact with food, such as grippers, end effectors, and auxiliary tooling, must be designed to prevent the contamination and spreading of bacteria and germs (Iqbal et al., 2017). The materials used for these components must be manufactured from materials capable of withstanding rigorous cleaning and sanitation procedures (Iqbal et al., 2017). Contamination risk may arise not only from microorganisms but also from plastic fragments and other foreign bodies (Aly et al., 2023). To mitigate such risks, Aly et al., 2023 cite several International Organization for Standardization (ISO) for safety and hygiene considerations, including:

- *ISO 14159* - Standard that specifies hygiene requirements for machines where contamination of the product might occur (*Safety of machinery — Hygiene requirements for the design of machinery*, 2002).

- *ISO 22000* - Standard creates a framework and specifies requirements for managing food safety (*Food safety management systems — Requirements for any organization in the food chain*, 2018).
- *ISO 10218* - Standard provides guidelines for the creation of safe designs and protective measures for industrial robots (*Robots and robotic devices — Safety requirements for industrial robots*, 2011).

2.2 Soft tissue manipulation

Food products such as smoked and cured meat products, including *chouriço* and similar items, require careful handling during manipulation to prevent surface damage or deformation. These products exhibit complex, nonlinear mechanical properties due to the heterogeneous composition of meat, which include fat, muscle, and connective tissues. As such their mechanical response combines deformable, viscous, and elastic characteristics. In addition, many agri-food products present irregular geometries, variable stiffness, slippery surfaces. These factors can compromise grasp stability and increase the risk of damaging the product, particularly due to uneven pressure distribution or excessive contact forces (Aly et al., 2023).

The authors of Ross et al., 2022 review several gripper designs based on criteria such as hygiene standards, damage prevention (both visual and physical damage), load capability, and maintenance. Their analysis concludes that grippers that rely on magnetic forces, needles, or adhesives are unsuitable for food handling, as they can cause product damage or fail hygiene requirements. On the other hand, the authors note that vacuum-type grippers present a promising alternative, although the authors also note that further investigation is required. Similarly, Raja et al., 2024 highlights the importance of strict hygiene requirements and damage prevention measures when designing grippers for delicate products. According to Aly et al., 2023, the red meat sector commonly employs hook and clamp grippers. However, hook grippers have the problem of inherently having to pierce the product and therefore causing damage. The study also reports the existence of underactuated finger grippers that deform on contact with the grasped object, increasing the contact area. Despite this advantage, this type of gripper exhibits low load capacity and presents cleanliness and durability problems, as they are often assembled using 3D-printed components, which tend to be fragile and can degrade during industrial cleaning processes.

2.2.1 Grippers

Grippers that make use of elastic polymers, elastomers, as soft actuators have the advantage of deforming on collision with the object and thus absorbing energy from the impact, something that all rigid grippers cannot do (Guo et al., 2017).

2.2.1.1 Soft-fingers pneumatic grippers

A common type of gripper for the manipulation of soft and delicate objects is the pneumatically actuated design that consists of two or more soft, deformable fingers made from elastic polymers. These fingers bend when air pressure increases within the internal channels, enabling the gripper to open or close and thus grasp soft objects without causing damage (Low et al., 2022; Salem et al., 2018; Low et al., 2017). Some grippers of this type are also capable of adjusting the configuration of their fingers into different poses, allowing the execution of multiple grasping techniques for delicate objects, as demonstrated in Low et al., 2022.

The gripper developed in the study Low et al., 2022 incorporates a food-safe fabric sleeve in order to prevent direct contact of the food with the plastic fingers. However, this protective layer limits the maximum bending angle achievable by the gripper fingers. In addition, the gripper includes a tactile sensor integrated into the fingers, enabling measuring of the force during grasping.

Like the previously mentioned gripper, the grippers presented in Salem et al., 2018; Low et al., 2017 also employ inflatable air channels that induce the bending on the gripper's fingers when pressurized. Another common characteristic is the use of 3D-printing techniques for manufacturing the fingers. The gripper described in Salem et al., 2018 features four fingers arranged in a cross configuration. Unlike the design in Low et al., 2022, it does not allow for the reconfiguration of the finger positions. As a result, its versatility is reduced, as it cannot adapt the grasping technique to accommodate for different object geometries.

In the article by Low et al., 2017, the authors note that 3D-printing offers several advantages against traditional silicone-based fabrication approaches. 3D-printing allows for a simpler manufacturing process, requiring typically just a printing step, whereas silicone-based approaches require multiple processes. Another benefit highlighted is the increased level of customization that 3D-printing allows. Furthermore, the paper by Hashanjana et al., 2023 presents a variation of this type of gripper that, instead of using elastomers for the soft actuator fingers, utilizes an "*origami inspired*" construction made from polyvinyl chloride (PVC) and polyethylene. This design also incorporates a reconfigurable plate to assist the soft actuators during the grasping motion.

The figure 2.1 illustrates a generic diagram of a finger from a pneumatic soft finger type gripper, actuated via an internal air channel. The diagram is inspired by the descriptions and images provided in Low et al., 2022; Salem et al., 2018; Low et al., 2017. When air is supplied to the air channel, it inflates and induces deformation in the fingers structure, closing the gripper's fingers.

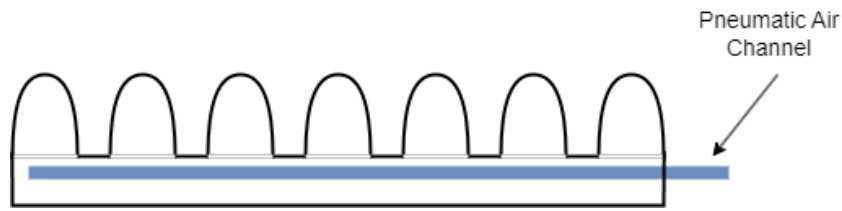


Figure 2.1: Generic diagram of a pneumatic soft fingers gripper's finger

2.2.1.2 Fin ray effect grippers

Another type of gripper developed for the task of handling soft and delicate objects is the fin ray effect gripper. Gripper fingers that make use of the fin ray geometry are able to deform and conform naturally to the surface of the object being grasped, regardless of its shape. This behavior increases the contact surface area and creates a more stable grip while avoiding any damage (Yang et al., 2021; Liu and Adelson, 2022).

In the paper by Yang et al., 2021, the authors propose a three finger gripper configuration employing the fin ray effect. This arrangement has the advantage of creating a more balanced grasp position and increasing the contact surface area. However, it may have the shortcoming of being less suitable for longer objects, as the third finger can interfere with the desired grasping pose. The paper identifies two main actuation principles. The first is pneumatic or hydraulic actuation, in which the fingers have internal hollow channels that deform when pressurized gas or liquid is pumped inside. This type of action requires external equipment such as compressors or pumps, resulting in heavier and noisier systems. The second is to make use of "smart materials" for the gripper fingers, which deform when an external stimuli is applied (Yang et al., 2021). The gripper was fabricated in one piece using 3D-printing. Its structure has a hollow region in its center with a plate connecting all three fingers. When a force is applied to this plate, the fingers bend. To control the force applied to grasped objects, the authors integrated a piezoresistive pressure sensor in the inner side of the gripper in one of its fingers. This allows the measurement of the force applied on the target object. To close or open the gripper, a servo motor is employed. This motor has attached a cord that can pull the plate that connects all fingers (Yang et al., 2021).

The gripper developed in Liu and Adelson, 2022 consists of two fingers that make use of the fin ray effect and incorporate a built-in tactile sensor. The fingers were constructed using 3D-printed parts combined with a laser cut fabric layer. The sensing system consists of a silicone pad that is painted and patterned with laser-etched dots, a camera to observe the dot displacement, and LEDs for illumination. The fingers were attached to a WSG 50-110 parallel gripper for actuation. The camera of the tactile sensor observes the deformation of the fingers by observing the dot displacement on the silicone pad.

The figure 2.2 presents a generic diagram of a fin ray effect gripper, inspired by the descriptions and illustrations provided in Yang et al., 2021; Liu and Adelson, 2022. The fin ray structure enables each finger to deform and conform to the geometry of the grasped object.

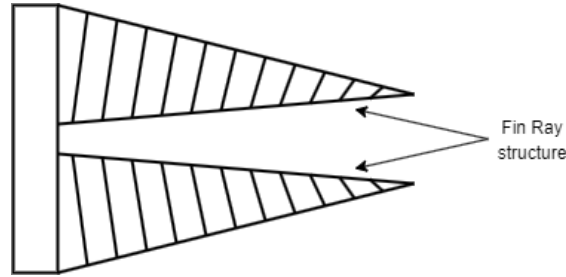


Figure 2.2: Generic diagram of a fin ray effect gripper

2.2.1.3 Hook type inflatable grippers

Another type of gripper developed to manipulate delicate objects is the hook type with an inflatable gripping element to secure the object. This type of gripper is more commonly employed for robotics in surgical environments (Guo et al., 2017, 2019). The gripper developed in the article by Guo et al., 2019 consists of a rigid hook structure with an inflatable gripping element. When compressed air is supplied to the gripping element, it inflates and contacts the object, securing it on the gripper. The hook part of the gripper in this work is made of stainless steel and as such has the disadvantage of damaging the product if it was impacted with that part. In the paper by Guo et al., 2017, another type of hook-like gripper with an inflatable element is presented. However, this one is completely made of plastic, and instead of having just one inflatable element, it has three, and as such the object being grasped does not come in contact with the hard shell, avoiding damaging the delicate object. These two previously presented grippers are designed with surgical environments in mind, and because of that, the grippers would need to be resized in order to grasp larger objects typically found in agri-food environments, which might not be feasible.

The figure 2.3 illustrates a generic pneumatic hook type gripper with three pneumatic actuated inflatable gripping elements, inspired by the descriptions and images in Guo et al., 2017, 2019. When compressed air is pumped to the internal channels, the gripper elements inflate and secure the grasped object.

2.2.1.4 Deformable fingertips grippers

A different type of gripper suitable for grasping fragile and delicate objects is one that employs the use of deformable fingertips containing cavities filled with an incompressible fluid. This design offers the benefit of uniform distribution of pressure on the surface of

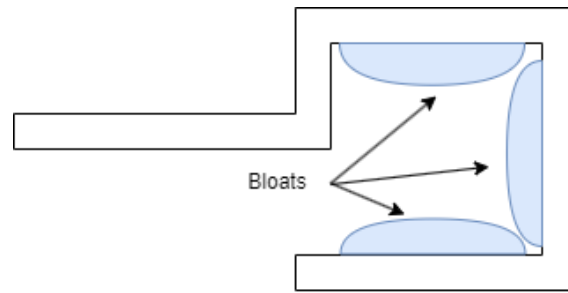


Figure 2.3: Generic diagram of an inflatable hook type gripper

the fragile object, thereby reducing the likelihood of fractures (Maruyama et al., 2013). The gripper presented in Maruyama et al., 2013 consists of a two-finger configuration equipped with the aforementioned deformable fingertips. These fingertips consist of an outer rubber layer and an inner silicon layer filled with a fluid. The system also incorporates two pressure sensors to measure the fluid pressure and the applied force on the silicon layer, enabling the control of the applied pressure on the delicate object.

A variation on the aforementioned gripper type was developed by He et al., 2020. The authors employed soft deformable fingertips filled with compressed air, instead of an incompressible fluid. This allows the fingertips to be pressurized when grasping the delicate object and act as a closure securing it to the gripper, preventing the object's fall (He et al., 2020). The developed cavities filled with air also serve as tactile sensors. When the air cavities deform in contact with the object, the pressure inside them changes, giving feedback in the grasping task (He et al., 2020).

The figure 2.4 illustrates a generic deformable fingertip-type gripper, inspired by the descriptions and images in Maruyama et al., 2013; He et al., 2020. In the figure, the blue half-circles represent the deformable fingertip elements. These fingertips deform around the grasped object, making the gripper less prone to cause damage.

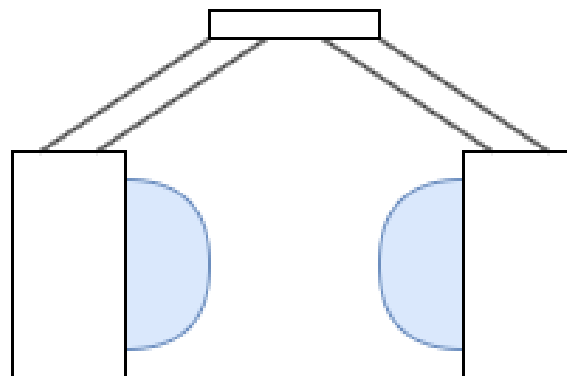


Figure 2.4: Generic diagram of a deformable fingertips gripper

2.2.1.5 Hybrid pneumatic-tendon grippers

A type of gripper specially developed for multiple grasping modes is presented in work by Godaba et al., [2020](#). This design employs flexible fingers incorporating two flexure hinges, actuated through a tendon mechanism. The hinges are joined together by a rigid link, and each hinge contains flexible flaps that fold using pneumatic actuation, allowing control of the rigidity of the structure. The fingers are manufactured using 3D-printed Thermoplastic polyurethane (TPU), a material that provides both flexibility and mechanical robustness. The pneumatic pouches integrated into the fingers are likewise produced from TPU. The tendon mechanism is actuated through a servo motor. The combination of pneumatic and electric actuation in different sequences allows the gripper to assume multiple configurations, enabling it to grasp a wide variety of delicate and fragile objects.

2.2.1.6 Bernoulli effect grippers

Grippers based on Bernoulli's principle provide a non-contact alternative to physical grasping. These grippers operate by generating a high-velocity airflow between the gripper plate and the target object, which in turn produces a lifting force, allowing the object to be effectively "grasped" and manipulated (Sam and Nefti, [2011](#); Sam and Buniyamin, [2012](#); Mansor et al., [2017](#)). However, the high-speed airflow might damage delicate objects. A solution to this problem is to integrate flow deflectors on the gripper surface to shield fragile items from the direct air stream (Sam and Nefti, [2011](#)). Another potential source of damage occurs when the center of the delicate object is forced away by the high-velocity airflow pushing on it. This effect can be mitigated by incorporating a central deflector (Sam and Buniyamin, [2012](#)).

The study by Sam and Nefti, [2011](#) notes that, when handling food products with compressed air, as used by the Bernoulli grippers, it must meet certain purity requirements, namely ISO 8573-1 Class 2.2.1 *Compressed air Part 1: Contaminants and purity classes*, [2010](#). Producing compressed air of this quality is expensive, so the gripper proposed in the study aims to reduce the air consumption. This is achieved by using the Bernoulli effect only during the initial lifting of the object and subsequently relying on mechanical fingers to secure the item, allowing the airflow to be switch off during transportation.

The effectiveness of this type of gripper depends on several factors. First, the surface characteristic of the object, as smooth surfaces are more difficult to lift compared to rougher ones (Sam and Buniyamin, [2012](#)). Second, the geometry of the item, as flat objects allow the gripper to lift higher masses than rounded objects (Sam and Buniyamin, [2012](#)). Finally, the diameter of the gripper influences air consumed by the system, as when the gripper diameter matches that of the object, it significantly reduces the air required compared to when the gripper is smaller (Mansor et al., [2017](#)).

The figure 2.5 illustrates a generic bernoulli type gripper, inspired by the descriptions and images in Sam and Nefti, 2011; Sam and Buniyamin, 2012; Mansor et al., 2017.

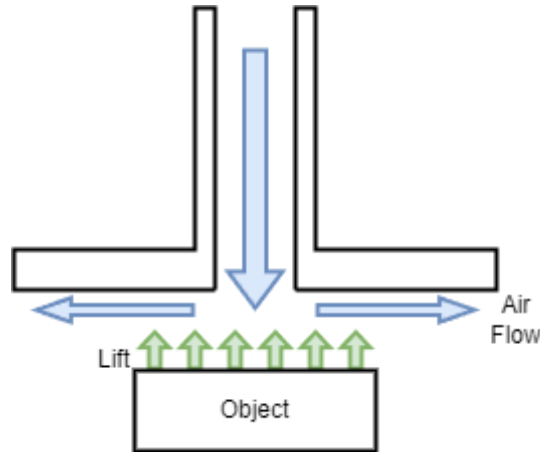


Figure 2.5: Generic diagram of a bernoulli type gripper

2.2.1.7 Vacuum grippers

Another type of gripper that uses airflow to grasp objects is the vacuum gripper or suction cup, which generates negative pressure to lift and hold an object (Rachkov and Bebenin, 2018; Bryan et al., 2019; Papadakis et al., 2020). The gripping system presented in Rachkov and Bebenin, 2018 consists of two vacuum grippers mounted on the end effector through a flexible spring bracket. Each gripper performs a different function, one for sealing and the other to provide gripping force. This configuration allows the system to establish effective contact with products that have rough or curved surfaces. In addition, the module that provides the gripping force includes a vacuum level sensor and features two concentric vacuum zones. An outer zone intended for sealing on porous or uneven surfaces and a central zone to generate gripping force.

The gripper described in the study by Bryan et al., 2019 takes influence from the grasping technique used by octopuses. It consists of three deformable fingers, each equipped with suction cups to improve grip stability on the object. The combination of deformable fingers and multiple suction cups enables the gripper to adhere effectively to objects with various geometries. To achieve the required deformation for the fingers, the authors employ a two-cable actuation system, which ensures that the suction cups make proper contact with the surface of the target object. Additionally, the gripper is capable of rotating its fingers, allowing it to adopt different grasping poses.

In the study by Papadakis et al., 2020, the authors test multiple suction cups and later develop a vacuum gripper. A shock absorber was developed to permit the grippers to press down on the object's surface and seal it with the suction cup. The suction cups tested were the VASB-40-1/4-SI-B by Festo and the BG125 and the BX15P, both by Piab.

The test results concluded that the three grippers had some difficulties when gripping deformable objects with rough surfaces. To resolve this issue, the authors developed a gripper inspired by the suction cup manufactured by Festo, using two-component silicone and 3D-printed molds. Additionally, a pressure sensor was installed on the grippers. When the sensor reads pressures, it indicates that the object was grasped. Low pressures indicate the object was not.

The figure 2.6 illustrates a generic vacuum / suction cup gripper, inspired by the descriptions and images in Rachkov and Bebenin, 2018; Papadakis et al., 2020.

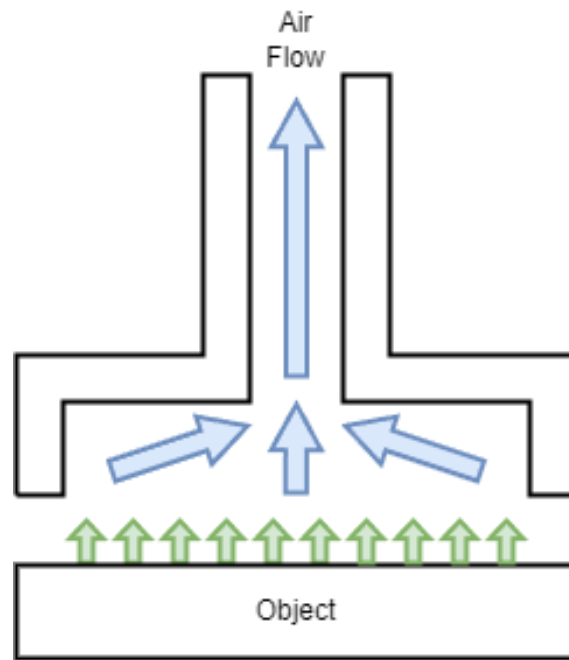


Figure 2.6: Generic diagram of a vacuum / suction cup gripper

From the previously described grippers, the fin ray effect gripper stood out as one of the most adequate for manipulating delicate objects. Being an electric actuated gripper brings the advantage of cheaper actuation cost and not requiring compressed air tanks or loud compressors to operate. Additionally, as this gripper type is able to deform around the object, it limits the potential damage caused to delicate and fragile items. The fin ray effect gripper still presents some problems, as the material used for the manufacturing of the gripper must be food safe to avoid contamination. The soft-fingers gripper has similar advantages to the fin ray gripper when manipulating delicate items. However, because this type of gripper is pneumatically actuated, it requires either a loud compressor or compressed air tanks. This creates a louder working environment, and the air used is required to meet quality level specifications in order to avoid food contamination, as highlighted by the ISO 8573-1 standard. Bernoulli's effect and vacuum type grippers work well for objects with flat surfaces. However, most food products do not have such

geometries. Additionally, these types of grippers require compressed air to work, and as previously mentioned, this air must meet higher quality levels. Vacuum grippers also present problems when dealing with rough surfaces and might leave permanent marks on products, as shown in the article by Raja et al., [2024](#). The soft fingertip grippers are more complex and more adequate when the food being grasped is very fragile. The gripper that makes use of flexible fingers and is actuated by a tendon system is more complex than the fin ray types. Being both electrically and pneumatically actuated, it presents the same problems as the compressed air of the soft-finger grippers. The hook type grippers are more adequate to an surgical environment than a food manipulation one.

2.3 Computer vision

Computer vision is a field that uses methods and algorithms to analyze and extract information from images or videos. Allowing computers, machines, and autonomous robotic systems to perceive and analyze images to obtain detailed information about their surrounding environment (Cherapanamjeri and Rao, [2022](#)). Tasks in computer vision involve object detection, image classification, object dimension measurement, pose estimation, object tracking, etc. The task of object detection consists of localizing in the image different types of objects, like cars, people, animals, plants, etc. For image classification, the task involves labeling the entire image in different classes. Object tracking is a task performed on video where it detects objects and then tracks them through each frame of the video. For the task of pose estimation, the objective is to determine the position and orientation of an object in an image or video, typically in relation to the camera or other reference.

In the task of object detection exist two main approaches, one using traditional image processing algorithms and another using neural networks and deep learning algorithms. In the traditional approach the algorithm extracts features from the image, including shape and texture, using algorithms that do not require training. As such, this approach has the advantage of not requiring training data and can be used from the get-go. However, this type of approach has the downside of requiring a manual fitting of the parameters (Cherapanamjeri and Rao, [2022](#)). The deep learning approach can be subdivided into two methods: single-stage and double-stage methods. Single-stage methods predict the bounding box and do the classification at the same time. You Only Look Once (YOLO) and Single Shot Detector (SSD) are examples of this method. Double-stage methods initially extract a region of interest and later perform the classification of the region. Convolutional Neural Network (CNN), Region Based Convolutional Neural Network (R-CNN), and Fast R-CNN are examples of this method (Cherapanamjeri and Rao, [2022](#)).

Nayeem et al., [2022](#) describe an approach for the detection and classification of objects in images. It's mentioned Recurrent Neural Network (RNN), which, according to the study,

"is a class of neural network where the connections of the units form a directed cycle". The paper also talks about residual learning, which allows for deeper neural networks that are more accurate and other neural networks, and maintains ease of training.

The paper by OTHMAN et al., [2018](#) presents an approach to object detection using a more traditional method and algorithms, where its used functions from the OpenCV libraries in order to detect and also measure the size of objects in images. To obtain the real size measurements of the detected objects, that being centimeters or millimeters instead of pixels, a reference object is used to calibrate a pixel per meter variable. The object detection process described in the paper starts with the application of a Gaussian filter to reduce noise, next, the image is converted to grayscale, and then the Canny edge algorithm is applied. Because sometimes the Canny edge does create closed objects, a closing operation is performed, a dilation and a erosion filters are applied, and the final step is to detect the contours to identify the objects in the image. According to the paper, the accuracy on the measurements for the size of the objects was between 95.45% and 98.23%, and when the camera was above the objects, the accuracy of the measurements improved. One problem of using a traditional approach for the object detection is that some parameters, in this case the Canny edge hysteresis thresholds, were determined in an empirical manner and would likely need to be changed if the environment changed.

The paper by Shreyas et al., [2021](#) presents some models and frameworks for 3D object detection. The first one is using plain Red Green Blue (RGB) images for the detection algorithm. RGB images have sufficient information to be used for 3D detection of objects. Other method is using Red Green Blue Depth (RGB-D) cameras instead of plain RGB cameras. That means that RGB-D cameras also have the capability to measure the distance between the sensor and the object in the image. Another method presented by the paper is to make use of point clouds. There are two variations for this method. Either using all points of the clouds for the object detection, thus becoming a more costly method, or reducing the number of points to make the method less costly, computationally speaking. Other methods mentioned by the paper are using a combination of RGB images and Light Detection And Ranging (LiDAR) point clouds. The RGB images would be used in a simple 2D object detection algorithm to reduce the region of interest, and then the point cloud given by the LiDAR would be used for the 3D part.

Another common task in computer vision is 6D object pose estimation, calculating the 3D position (usually Cartesian coordinates) and orientation (Euler angles or quaternions). This task is extremely important for robotic manipulators to determine a proper and stable grasping pose for a target object (Tran and Lin, [2020](#); Gupta et al., [2019](#); Wong et al., [2023](#)). Some difficult challenges come associated with object pose estimation, usually in the form of background clutter, object occlusion, objects with no or little texture, changes in rotation, translations, and scale changes (Gupta et al., [2019](#); Yun et al., [2017](#)).

Approaches for 6D object pose estimation include:

- RGB only methods (Gupta et al., 2019)
- RGB-D methods (Yun et al., 2017; Höfer et al., 2021)
- Point cloud based methods (Wong et al., 2023; Chen et al., 2020; Wu et al., 2022)
- Structured light techniques (Tran and Lin, 2020)
- Stereo vision systems (Taryudi and Wang, 2017)

2.3.0.1 RGB only method

The study by Gupta et al., 2019 focuses on resolving the previously presented challenges regarding object pose estimation. The method described by the authors outputs the pose of the detected object in 6 Degrees of Freedom (DoF) in relation to the camera and a score associated with the estimated pose. The authors used an RGB camera and reference 3D mesh models. The test images consist of cluttered environments with only one instance of the selected textureless object. The approach used is divided into a pipeline with two stages. The first stage uses a YOLOv3-inspired model that outputs three 3D tensors from an input of an RGB image. Each tensor has different spatial sizes of 13x13, 26x26, and 52x52 cell grids, each cell being a vector with a $2n + 1 + c$ dimension, where $2n$ are xy coordinates of the 2D keypoints, 1 is for a confidence proposal, and c is the class scores. The second stage uses 3D keypoints and estimates poses using an E-PnP algorithm. Following that, the original RGB image and the various pose-rendered masks are cropped together and passed to the CullNet that outputs calibrated object confidences. These confidences are then used to pick the pose estimate (Gupta et al., 2019). The authors use a different network to calculate the confidence score of the estimated pose because the keypoint proposal network does not have an estimate of the accuracy of the resulting pose (Gupta et al., 2019). The figure 2.7 represents the flowchart of the proposed method by Gupta et al., 2019.

2.3.0.2 RGB-D methods

The authors of Höfer et al., 2021 present a framework for pose estimation for small objects in a highly cluttered environment. The objects are first detected using a mask R-CNN, capable of predicting both the bounding box of the object and their segmentation mask. For the pose estimation, the pixels visible in the segmentation mask are used in an augmented autoencoder. This augmented autoencoder uses a dimensionality reduction technique that is trained to extract 3D features from image crops. To select the best pose estimations, a method using the depth values of the original image is employed. The predicted 6D pose of an object has its depth rendered and then compared with the real value of the original image. If the absolute error is smaller than a defined threshold,

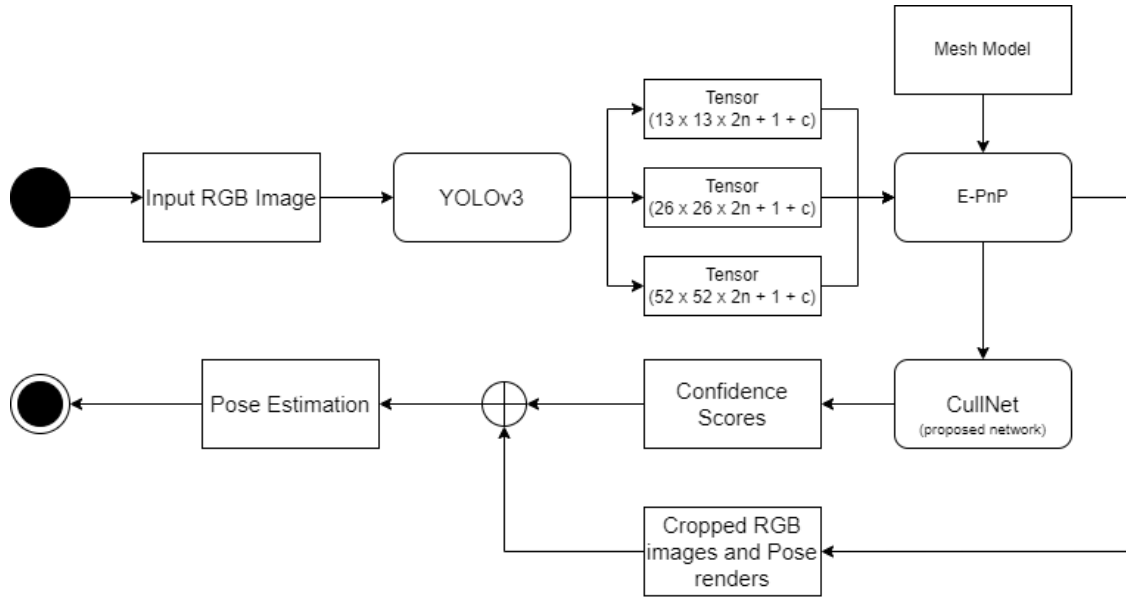


Figure 2.7: Flowchart of the previously described algorithm [adapted from Gupta et al., 2019]

the pose is selected. For training the neural network, the authors of the study created a synthetic dataset of 2D images, generated using the Computer Aided Design (CAD) models of the objects. However, a model trained only using a synthetic dataset does not generalize very well to real-life scenarios. For that reason some techniques were applied to make the dataset more closely resemble a realistic scenario (Höfer et al., 2021).

The study also makes a comparison between the method developed to estimate object poses, using only RGB information and using RGB and depth information, and the Iterative Closest Point (ICP) algorithm, an algorithm used to discover the alignment between two cloud points by minimizing the distance between points (Bai, 2023). The authors arrived at the conclusion that while the ICP has better performance on the pose estimation, the developed method is faster and therefore more practical for real-time scenarios (Höfer et al., 2021). The figure 2.8 represents the flowchart of the proposed method by Höfer et al., 2021.

The study by Yun et al., 2017 focuses on the problem of object recognition and pose estimation for a manipulation system. The information necessary for this system is the object ID, location, and pose information. To obtain this information, the authors chose a local feature-based approach as a starting point due to its simplicity of implementation and small dataset required. To generate the dataset, the authors created ten cuboid objects by scanning the surfaces of objects. 2D synthetic images are generated from these 3D models using the Blender software (Yun et al., 2017).

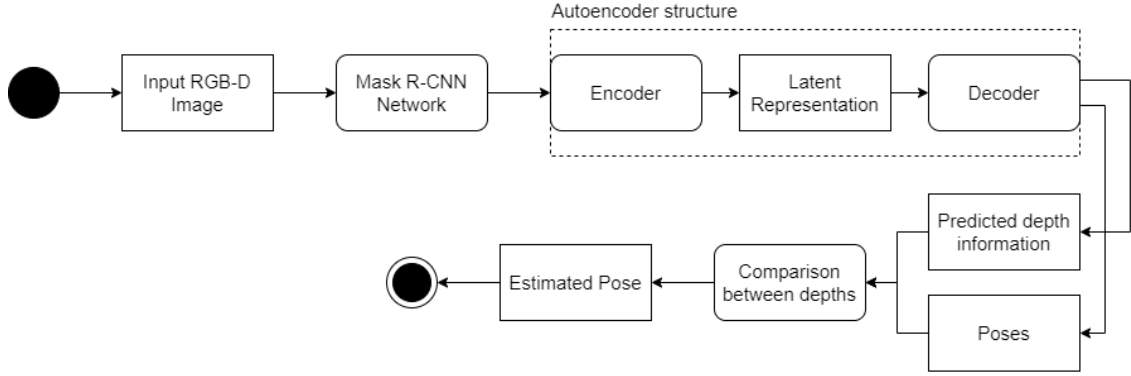


Figure 2.8: Flowchart of the previously described algorithm [adapted from Höfer et al., 2021]

The local feature matching method is easily affected by background noise, to reduce this problem, the authors utilize a background subtraction algorithm. This algorithm uses depth information to extract the foreground (Yun et al., 2017). In order to find correspondences between the features stored in the database and the features on the images, the distances between the features on the image and every feature stored in the database are calculated. The smallest distance is the one selected. The process is repeated in reverse, from the database to the image. The pair that has the same feature points in both correspondences is selected. From the selected correspondences, the distinctive one is chosen if at least one of the following conditions is true. The ratio between the two shortest distances is less than a selected threshold, or the shortest distance is less than other defined thresholds (Yun et al., 2017). Finally, to get the pose estimation and object location, a perspective-n-point problem has to be solved, to which the authors employed the Random Sample Consensus (RANSAC) algorithm (Yun et al., 2017). Methods that use perspective-n-point are less effective on scenes with object occlusion, clutter, and illumination changes, which makes them less desirable for complex environments (Chen et al., 2020; Wu et al., 2022). The figure 2.9 represents the flowchart of the proposed method by Yun et al., 2017.

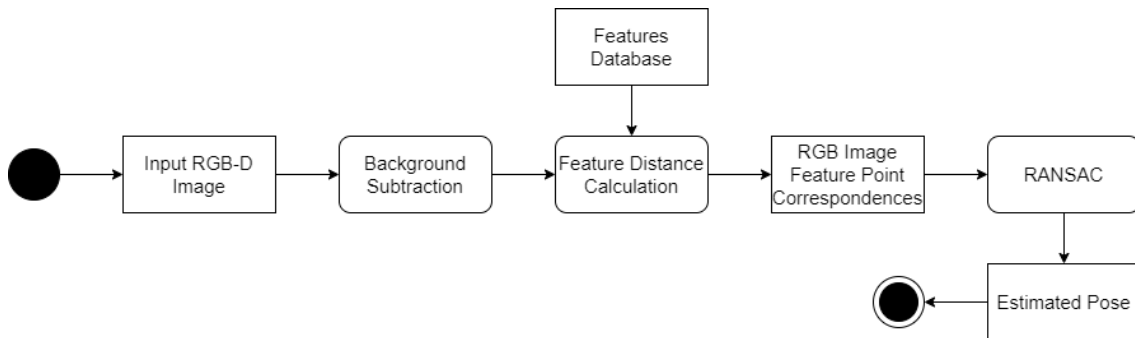


Figure 2.9: Flowchart of the previously described algorithm [adapted from the descriptions in Yun et al., 2017]

2.3.0.3 Point cloud based methods

The paper by Chen et al., 2020 proposes a multi-stage approach to estimate the 6D pose of objects using an RGB-D camera. The first stage locates the object in two dimensions using a CNN. The corresponding depth region is then transformed into a point cloud and sent to a network for object segmentation and pose estimation. The depth region delimited by the bounding box predicted by the CNN in the first stage contains both points of the background and of the object. To eliminate the background points, the authors employed two tasks. First, segmenting each point of the object with a binary classification. Secondly, have the developed network predict the directions of vectors that point to keypoints for each point (Chen et al., 2020). The keypoints defined for the network to predict are the 8 corners for the 3D bounding boxes for the object. From the 3D segmentation and the vector prediction from the previous steps, multiple keypoint hypotheses are generated. From these hypotheses, the best one is chosen using a selection algorithm that compares the points from the test scene with the mesh model (Chen et al., 2020). The figure 2.10 represents the flowchart of the proposed method by Chen et al., 2020.

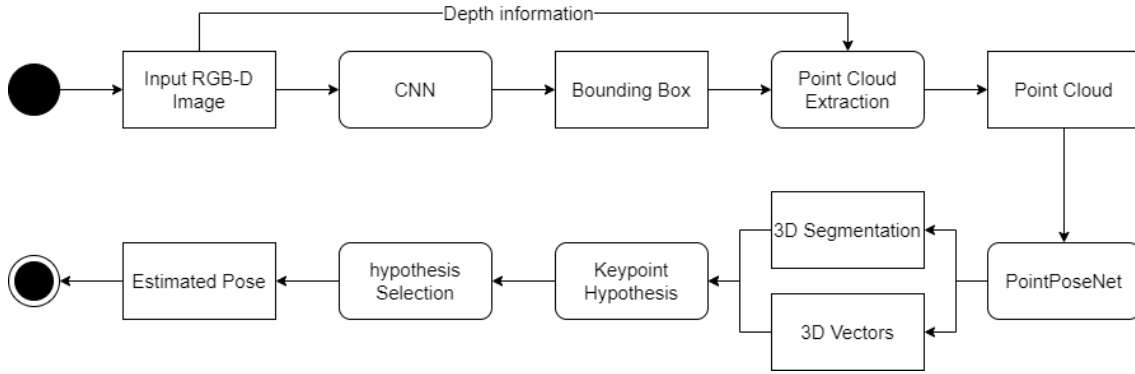


Figure 2.10: Flowchart of the previously described algorithm [adapted from Chen et al., 2020]

The paper by Wu et al., 2022 presents a 6D pose estimation framework using 3D point clouds that uses local feature matching to recognize objects and predict their poses. The proposed framework consists of two major parts. The first part uses the high compatibility correspondence grouping technique to extract various feature groups in heavily cluttered scenes. The second part uses the groups extracted by the previous part and generates the 6D pose hypotheses and verifies those pose hypotheses.

The high compatibility correspondence grouping technique proposed by the paper consists of 4 stages. First, the initial correspondences of the feature points. The second stage ranks these points. The points are clustered in the third stage. On the fourth stage are expanded (Wu et al., 2022). The pose estimation stage receives the high compatibility correspondence groups from the previous step. Each group could be considered for the object pose hypotheses. However, because of the expansion stage, the number of outlier

points increases, which negatively affects the estimation and verification stages. To address this issue, the authors employ point-fair feature constraints to eliminate the outliers. The groups are used to estimate the poses of the objects in the scene, using a local reference frame-based approach. The object poses are verified using the transformed model key points in comparison with the scene points (Wu et al., 2022). The figure 2.11 represents the flowchart of the proposed method by Wu et al., 2022.

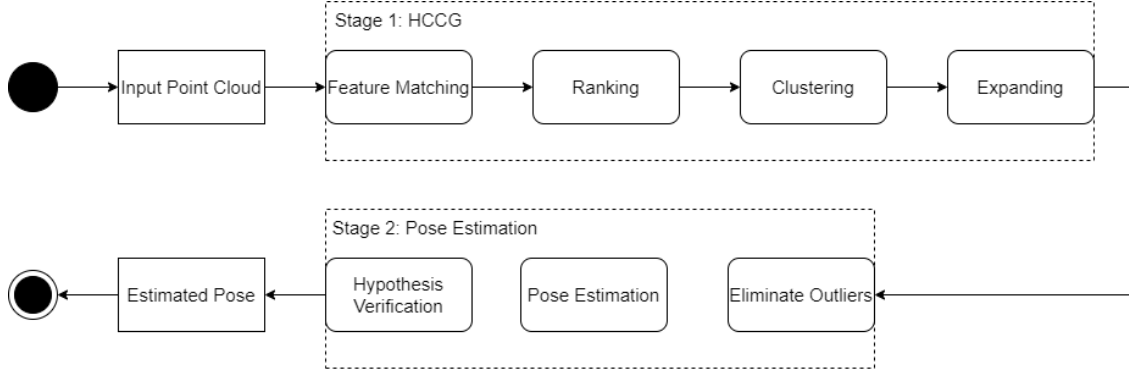


Figure 2.11: Flowchart of the previously described algorithm [adapted from the descriptions in Wu et al., 2022]

The pose estimation and robotic grasping pose estimation method proposed in Wong et al., 2023 is based on an affordance detection and segmentation network. The proposed robotic grasping pose estimation system is composed of four parts. The first one is the previously mentioned affordance detection and segmentation network. The second is the extraction of the object affordance point cloud. The third is the estimation of the object pose. The last stage is the generation of the grasp pose for the robotic manipulator.

The affordance detection and segmentation network receives as an input the RGB image from a RGB-D camera and outputs the center pixel of the detected object, the centroid, the object bounding box, and lastly the segment object affordance mask (Wong et al., 2023). The module that performs the extraction of the affordance point cloud of the object receives as an input the point cloud that is provided from the RGB-D camera and the mask generated by the affordance detection and segmentation network module (Wong et al., 2023). The next module, the pose estimation module, takes the bounding box, center pixel, and center point from the first module, and the affordance point cloud calculated from the previous module and outputs the estimated pose in respect to the camera (Wong et al., 2023). The grasp pose estimation module uses the 3D object pose estimated in the previous module and the gripper information from the robotic manipulator and outputs a 2D planar grasp pose in the coordinates of the camera system (Wong et al., 2023). The figure 2.12 represents the flowchart of the proposed method by Wong et al., 2023.

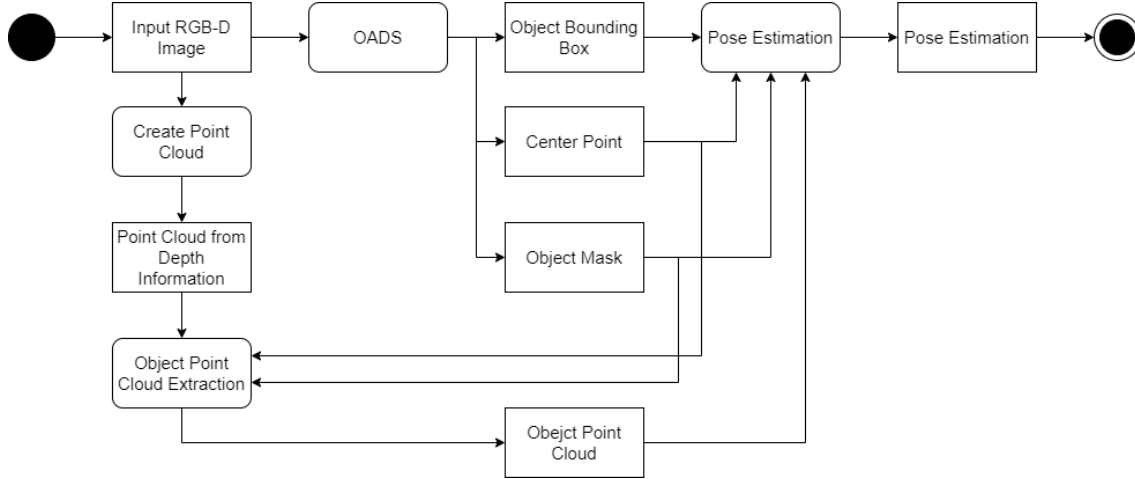


Figure 2.12: Flowchart of the previously described algorithm [adapted from Wong et al., 2023]

2.3.0.4 Structured light method

The paper by Tran and Lin, 2020 proposes a technique for 6D pose estimation using mask R-CNN, a neural network for instance segmentation problems, with RGB-D images. The approach developed uses as an input RGB images and depth information from a structured light system, 3D models from the target objects, and the 6D pose of the camera system, then outputs the rotation matrix and translation vector that represent the relationship between the target object and the 3D model.

The structured light system acquisition system works by projecting encoding patterns, patterns made of black and white lines, into the scene using a data projector. The pattern-coded images captured by the camera are processed in order to find points associated with the data projector and RGB-D camera (Tran and Lin, 2020).

The neural network used for object detection and segmentation in this paper is a mask R-CNN with two stages. The first stage scans the image and creates regions of interest where there could be an object. The second stage predicts the class of the object, improves the bounding box, and creates a mask of the object based on the previous stage proposal (Tran and Lin, 2020). The ICP algorithm was used to estimate the 6D pose of the objects. This algorithm calculates the transformation between the objects and the model. A major problem is the determination of the correct data associations. Past that, the transformations can be obtained using Singular Value Decomposition (SVD). The developed method achieved an accuracy of 87.4% (Tran and Lin, 2020). However, this method requires a dark environment for the projection of the encoding patterns. The figure 2.13 represents the flowchart of the proposed method by Tran and Lin, 2020.

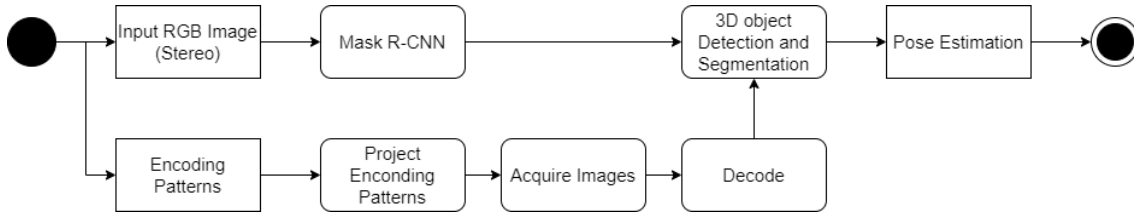


Figure 2.13: Flowchart of the previously described algorithm [adapted from Tran and Lin, 2020]

2.3.0.5 Stereo vision method

The study by Taryudi and Wang, 2017 presents a method for 3D object pose estimation using a calibrated stereo vision camera system. The object position is estimated by a triangulation algorithm on the center of the object. The orientation is obtained using a Matlab function that extracts the image region properties. An Adaptive Neuro Fuzzy Inference System (ANFIS) is used to then obtain the pose of the object in relation to the arm frame of the robot. The object is detected using a Hue Saturation Brightness (HSV) color threshold and a filter algorithm. Then, the blob analysis function from Matlab is used to get the position and orientation of the object in the 2D plane. Following that, using the centroid of the object, calculated in the previous step, the 3D position of the object is estimated. The orientation of the object is extracted as the angle between the x-axis and the major axis of an ellipse that has the same second moments as the region (Taryudi and Wang, 2017). The ANFIS structure consists of 4 inputs and 4 outputs; the inputs are from the position and orientation of the object in relation with the stereo camera, and the outputs are the position and orientation of the object in relation with the base frame (Taryudi and Wang, 2017). This method has the problem of only giving one orientation angle for the object, which means that it extracts less information for the manipulation of the object. The figure 2.14 represents the flowchart of the proposed method by Taryudi and Wang, 2017.

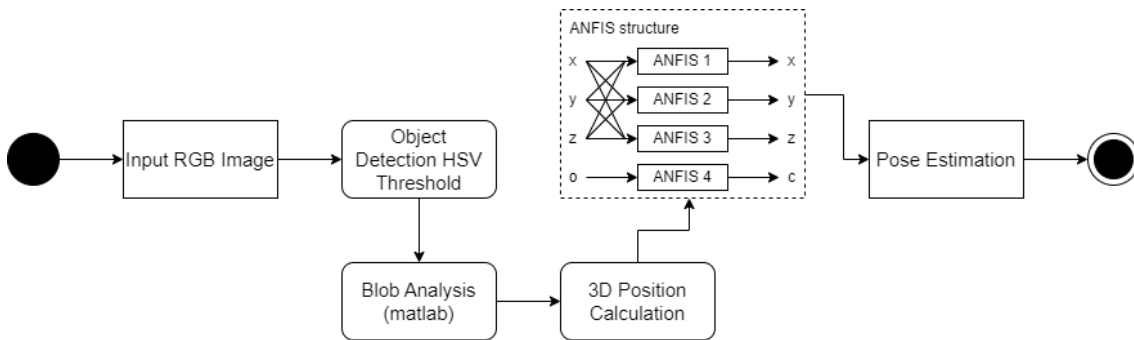


Figure 2.14: Flowchart of the previously described algorithm [adapted from the descriptions in Taryudi and Wang, 2017]

2.3.1 Computer vision applied in industry

Computer vision is an important field of artificial intelligence for the industry, especially for robotics and automation. Various tasks associated with computer vision are used in industrial operations, for example, object detection, image classification, and pose estimation.

The article by Okarma, [2020](#) gives various examples of applications of computer vision, specifically in the field of automation and robotics. The examples go from a calibration technique for augmented reality see-through headsets using cameras, detection of pedestrians at nighttime by processing infrared images, and quality evaluation of surfaces from 3D-printed objects to a machine prototype with the task of sorting different plastic granulations.

One of the tasks for which computer vision is used is for quality control. The paper by Villalba-Diez et al., [2019](#) describes a quality control system that makes use of computer vision with a Deep Neural Network (DNN) on printing cylinders. The authors claim that the proposed system can achieve an accuracy of 98.4%. The system developed by the authors is composed of three stages. The first stage is responsible for the data acquisition and the preprocessing of that data in order to be sent to the next stage. The second stage has the objective of extracting features from the input images. The final stage receives the extracted features, classifies those features, and outputs the quality of the printing cylinder, if it passed the quality inspection or if it failed.

Another task that may use computer vision algorithms is the counting process in inventories. The authors of the article Ji et al., [2023](#) created a system for warehouses to detect and record metal sheet picks using computer vision. The authors claimed that the developed system can count the metal sheet picks with a precision of 97.83% and a recall of 100%.

Computer vision is also used in the agri-food sector. The article by Kakani et al., [2020](#) talks about the evolution of the food industry and artificial intelligence, and in particular the application of artificial intelligence in the food industry. The article highlights multiple use cases for computer vision, which is typically paired with other technologies like robotics, sensors, etc. Such use cases are crop health monitoring, management of the use of herbicides, measuring the irrigation of certain lands, and others.

The studies Yaguchi et al., [2016](#) and Hu et al., [2021](#), previously mentioned in the section [2.1](#), both make use of image processing that enters into the area of computer vision to detect the crops the robot is supposed to harvest, giving it autonomous behavior.

In Yaguchi et al., 2016, the authors used a stereo camera to obtain a colored point cloud and used the HSV color space to detect the possible tomatoes, taking advantage of the tomato's red color. Next the tomato clusters are determined using the Euclidean distance clustering, and the closest tomato is chosen. The authors of the study by Hu et al., 2021 used an RGB-D camera to identify the citrus on trees to be harvested.

The authors of the study by Fernandes et al., 2022 present a solution for the classification of cocoa beans using a computer vision system to distinguish the various types and create a more objective form of quality control in comparison to human verification. This type of quality control also has the advantage of not causing any harm to the beans being verified. The presented system makes use of deep learning in order for the system to learn the different varieties of cocoa beans from a dataset.

2.4 State of the art summary

From this review of the state of the art, we can find that various types of grippers have already been developed specifically designed with fragile or delicate products in mind, composed of either soft tissues or other types of materials. These grippers need to take into consideration several factors for the manipulation of agri-food. Namely the hygiene and safety requirements. Because of these factors, as well as the general shape of the agri-food products chosen for this work, the fin ray effect type of gripper gives a great promise of success when compared with the other types of presented gripper. However, there are no papers studying the capabilities of this type of gripper when it comes to the grasping of sausage-type products.

This review also explored various types of algorithms, approaches, and methods for the detection and calculation of the position and orientation of objects, object pose estimation. Some of these presented approaches utilize plain RGB images. Others use these color images in combination with depth information to calculate the object's poses. Some use the depth information directly. Other first transform the depth information into point clouds to then use them for object pose estimation. All of these algorithms presented by the reviewed studies utilize complex reference data, such as point clouds or 3D mesh files. For that reason these algorithms will have higher difficulty accurately calculating the pose of objects that have poorly defined shapes, as for agri-food products.

PROPOSED APPROACH

This chapter describes the case study scenario, as well as the proposed solution and approach to accomplish the objectives of the scenario. Both the approaches for achieving the goals related to the soft-tissue manipulation gripper and vision system are going to be described, as well as the methods for their evaluation and validation tests.

3.1 Case Study scenario description

This work is derived from handling tasks of the charcuterie industry, with a focus on the packaging part of various traditional smoked sausages. The scenario corresponds to the SEL factory located in Portugal's Alentejo region. In the current state of the industrial process, human operators are responsible for handling these various smoked sausages, transferring them from a feeder tray, where they are displaced, and placing them on the packaging line. The figure 3.1 presents the workspace of these operators.

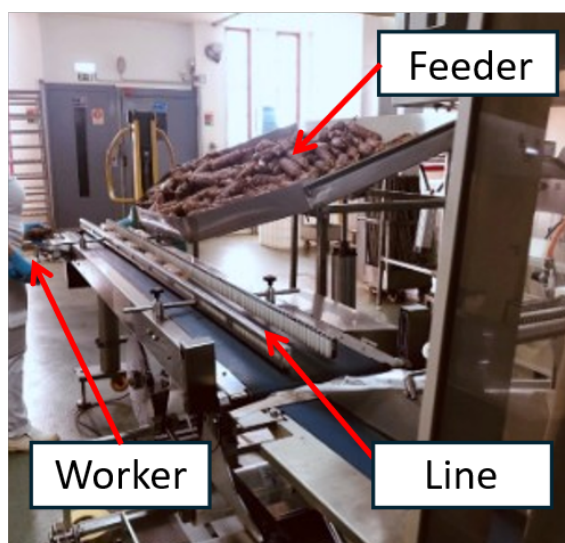


Figure 3.1: Packaging line

This task is physically demanding, repetitive, and time-consuming, often leading to operator fatigue and increasing the risk of work-related musculoskeletal injuries. Besides this factor, another motivation for the automation of this task is the growing labor shortage in the interior region of Portugal. Adding to the complexity of the packaging step of the industrial process of the products, the packaging stage is the final process on which all the products converge. As a consequence, the workers are required to handle a wide range of products, each with different shapes, firmness, texture, and other physical attributes. This requires the workers to have a high level of adaptability and a significant level of training. This creates more difficulties for the limited existing workforce.

Maintaining consistency and efficiency under these difficult conditions is challenging. The manual nature of the task, combined with the unpredictable presentation of the various products displayed on the tray, makes it difficult to establish standardized movements. The requirement of handling the various products with care, necessary to preserve the physical integrity of each item, is especially important for this industry, as appearance and texture are key quality indicators. The products handled in this industrial operation exhibit characteristics that pose challenges not only for robotic manipulation but also for vision systems. This is due to their texture properties, object softness, deformable surfaces, and sensitivity to pressure-induced damages of these products. These characteristics create difficult conditions for the robotic manipulation system. Additionally, the irregular shapes and surfaces of these products, combined with the high variability of the products, create difficulties for the vision system associated with the robotic system. All these factors hinder accurate object pose estimation and complicate the development of reliable and efficient grasping techniques.

3.1.1 Case study objective

The objective of this case study is to develop a robotic system that is capable of accurately detecting, identifying, and calculating the product position and orientation (object pose estimation task) and later manipulating it from the feeder tray to the packaging line (pick and place task). While preserving the products' structural integrity and avoiding damage. The experimental focus of this work is on the packaging operation. However, this work can also be applied to other stages of the complete industrial process. The ultimate aim of this work is to reduce the physical strains and demands on the industrial workers, improve the working conditions, and at the same time enhance the process efficiency without compromising on the product quality and value.

3.2 Proposed solution

The proposed solution for this case study is a robotic system that can be subdivided into two distinct parts with different tasks. First, the robot itself, comprising the manipulator

and the end effector, in this case a gripper. The second, the vision system, which is responsible for detecting, locating, and estimating the position and orientation of the objects to be picked up by the robot. Because of this division, the proposed approach can be subdivided into two subparts: one focused on the soft gripper development and testing and the other focused on the development of the computer vision system for estimating poses of objects.

3.2.1 Gripper subsystem

Regarding the soft manipulation gripper part of the system, the tasks will involve the development and testing of a fin ray effect gripper, as it seems the most promising gripper type to handle various types of agri-food products. The chosen agri-food products for the testing of the gripper were *chouriço*, *farinheira*, and *morcela*. As demonstrated in the state of the art chapter (chapter 2, section 2.2.1), the fin ray type grippers can be constructed using 3D-printing technology. For this, one of the tasks will be the construction of a fin ray type gripper and testing it to verify its performance in grasping food types with cylindrical shapes. The reasoning behind the choice of the fin ray effect gripper is because of several factors associated with this type of gripper. The gripper can be actuated using electric motors, instead of being pneumatically driven like many of the gripper types presented in the state of the art (chapter 2, section 2.2.1). This type of gripper can be easily constructed via 3D-printing technology, which allows for faster iterative development. The fin ray effect gripper is well suited for the grasping of objects with cylindrical/rounded shapes. Because the three chosen products for this work have rounded features, this gripper seems to be the best one.

3.2.2 Vision subsystem

Regarding the computer vision part of the system, the objective mentioned in the introduction chapter (chapter 1) is to study and develop a system capable of detecting and locating different objects, in this case multiple *chouriço*. As discussed in the state of the art chapter (chapter 2, section 2.3), the two main methods of pose estimation rely on either depth images from Red Green Blue Depth (RGB-D) cameras or point clouds, usually created from RGB-D cameras. The reason for using an approach based on depth cameras instead of plain Red Green Blue (RGB) cameras or structured light systems is because of several factors. The approaches using RGB cameras require more complex algorithms for object pose estimation to make up for the lack of depth perception of those camera systems. Because the robot is already equipped with a camera system that can provide depth information, using just the RGB sensor would be wasting the capabilities of the installed camera system. The structured light approach has the problem of requiring very specific light conditions to be able to estimate the various objects' poses. However, these light conditions cannot easily be replicated on an industrial floor. This reason makes this approach inviable for the object pose estimation algorithm. For the detection

part of the object pose estimation algorithm, an You Only Look Once (YOLO) model is going to be used. This is because this neural network model can be easily trained with images containing the three chosen products to be able to identify and locate them in future images. This approach is better when comparing with more traditional approaches because training a YOLO model is substantially easier than developing an object detection algorithm using methods such as Hue Saturation Brightness (HSV) color threshold and offers better end results.

3.2.3 Overall robotic system

The robotic system that's going to be used for this project is composed of an IRB 120 robot from ABB, a 6 Degrees of Freedom (DoF) robot, a depth camera, an Intel RealSense D415 RGB-D, and a gripper, which can be changed to test various gripper systems. The grippers and the object pose estimation algorithms will be tested with this robotic system. This robot will be programmed using the Robot Operating System (ROS) framework, specifically the Kinetic Kame version of ROS 1, as it is the one compatible with the ABB robot. Figure 3.2 is a simplified diagram visually describing the robotic system. The red arrows represent rotations around the x-axis in rotational joints, the green arrows for rotations around the y-axis, and the blue arrows for rotations around the z-axis.

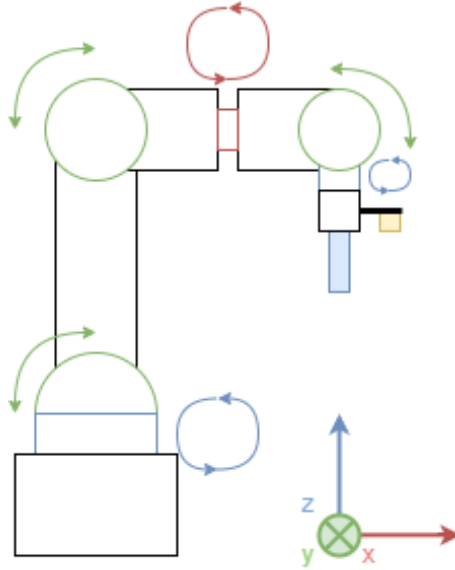


Figure 3.2: Diagram of the robot system that's going to be used. Red for rotations on x-axis, green for y-axis, blue for z-axis.

The robotic system pipeline would work as follows. Firstly, the camera attached to the robot's end effector captures two images of the scene where the products are displayed, one image containing color information (RGB), and another containing depth information. These images are then used by the developed object pose estimation algorithm to calculate the various detected objects' poses, which in turn are sent to the robot controller. Finally,

the robot ROS controller, using the previously calculated poses as targets, generates a grasp plan to perform a pick and place operation with each product.

3.3 Evaluation methodology

To evaluate and validate the developed robotic system and its subsystems, namely the gripper system and the vision, along with the object pose estimation algorithm, various tests need to be conducted. First, the two subsystems of the developed robotic system, the gripper and the computer vision algorithm, must be tested independently. After each subsystem has been evaluated separately, the final assessment is performed using the complete integrated system.

Initially the first task in the validation process is to test the developed gripper system. The developed gripper would be tested independently from the developed vision algorithm. This is done to isolate the results of the gripper from the vision system and verify if the gripper is actually capable of picking the various agri-food products. For this test, one of the three chosen products is placed at a designated position on the workspace table, then the robot is sent to that position, on which the gripper is closed around the object to grasp it. If the robot is successful in picking up the product without it falling, the test is considered a success. This test is performed for all three products.

The second task is to individually test each developed object pose estimation algorithm's method. This test is done by placing one of the chosen agri-food products on the workspace table and running the algorithm to calculate its pose. The object is placed in various poses, some flat to the table and others with an angle, to determine the resilience of the developed methods. The main objective of this first test is not to test the accuracy of the methods but to test the resilience and stability of each method. If the developed method has problems with noise or cannot handle products that are not presented flat to the table or to the camera system, it is discarded. This test is to discover the limitations, problems, and advantages of each method and to select the best method for this scenario.

The final task is to test the fully integrated robotic system. For this task various tests will be conducted, with similar methodology. The methodology for the tests is as follows: First, a product is sent to a designated position with a determined pose (x , y , z , roll, pitch, and yaw). Next, the robot at its home position runs the developed object pose estimation algorithm to determine the place product pose. With the calculated pose, the robotic system computes a plan to pick and place the product. If the product is picked and placed without falling off the robot's fin ray gripper, the test is considered a success. On these tests the coordinates and Euler angles obtained by the object pose estimation are recorded to obtain more information on the system's performance. The test will be performed using the products placed flat on the workspace table, with an angle against the table,

and with products on top of each other occluding the bottom product. These tests should encompass all the predicted scenarios that may occur on the real production line this robotic system is developed for.

SOFT-TISSUES GRIPPER DEVELOPMENT

This chapter presents the development process of the fin ray effect gripper, designed for the handling and manipulation of agri-food products composed of soft tissues. Firstly are presented the various gripper designs gathered for inspiration for the final design. Next, the development process of the gripper and its various iterations is described. Finally, the developed digital interface is presented.

4.1 Fin ray effect grippers gathering

As already presented in the state of the art chapter (section 2.1.1), when working with food products, special requirements designed for food safety and hygiene must be followed to avoid contamination and damage. This ensures the quality of the products for the final consumer.

Multiple gripper designs were already described in the section 2.2.1 specifically developed for handling delicate and fragile items. Presented designs included electrically and pneumatically actuated grippers. As already referred to in the proposed approach chapter (section 3.2), the chosen gripper design was the fin ray effect type. Due to the advantage of being electrically actuated, it is simple in its design and is well adapted to handle delicate products.

The previously presented fin ray effect gripper in the state of the art chapter (chapter 2) by Yang et al., 2021 is a three-finger configuration gripper, which has the advantage of having a more stable grasping position for the products having a larger contact surface. However, for longer objects, the third finger can become a shortcoming, as it can disrupt the target grasping pose. The design of the fingers for this gripper has a length of 50mm and 10 struts (Yang et al., 2021). These fingers were mounted onto a plate that can be moved using a servo motor with a cable, creating an angular type of gripper. The fin ray effect fingers were 3D printed with "Formlabs elastic resin material". A force sensor was also included on the gripper made using a pressure film sensor. The gripper is controlled via an Arduino Uno (Yang et al., 2021).

Another fin ray effect gripper previously presented by Liu and Adelson, 2022 consists of a two-finger configuration that is electrically actuated through a linear gripper. Each finger incorporates a tactile sensor based on a camera, and its overall construction is significantly more complex compared with the gripper described in Yang et al., 2021. In this design, the fingers are hollow to accommodate the tactile sensor, which required the use of more rigid materials. The struts are composed of 1.75 mm Thermoplastic polyurethane (TPU) from Overture. The back section of the fingers is composed of Onyx from Markforged. The tactile sensor itself consists of a silicone pad marked with a matrix of black dots, which allows the camera to detect deformation patterns and determine whether an object has been grasped Liu and Adelson, 2022.

The article by Gidugu et al., 2024 presents multiple patterns for the fin ray structure for soft robotic grippers. All the proposed grippers have the same external dimensions, having a length of 100mm, a base width of 60mm, and a height of 8mm. The presented patterns for the fin ray struts include simple rectangular shapes, trapezoidal shapes, struts with ends that taper down, and designs with the struts inclined by 45 and -45 degrees. The struts have ranges from 2 to 4 millimeters in thickness. The authors chose to 3D print these gripper designs using TPU 95A due to its softness and flexibility (Gidugu et al., 2024).

4.2 Fin ray effect gripper requirements

As stated in the state of the art chapter (chapter 2), grippers designed to handle food products must comply with specific standards to prevent contamination and avoid damaging the items being manipulated. Among the referenced standards are ISO 14159, ISO 22000, ISO 10218, and ISO 8573. The European Union also establishes regulations for materials intended to come into contact with food. Namely regulation (EC) No 1935/2004 (European Parliament and Council, 2004), which defines general requirements for such materials. These materials must also be manufactured in accordance with regulation (EC) No 2031/2003 (European Commission, 2003). For plastic materials specifically, regulation (EU) No 10/2011 (European Commission, 2011), which sets migration limits and manufacturing requirements.

Besides hygiene and safety requirements, the gripper must also be capable of creating secure grasps on the chosen agri-food products (*chouriço, farinheira, morcela*), and for that, the fin ray gripper must be able to bend when in contact with the food products.

The material chosen for the fin ray gripper was TPU 95A, also used by the authors of the paper Gidugu et al., 2024, due to its elastic and soft properties, as well as meeting the requirements for food safety. TPU also had the added bonus of being able to be used in 3D printers. This will facilitate the construction of the gripper, as well as prototyping.

4.3 Fin ray effect gripper development

To develop the fin ray fingers for the gripper, the Onshape website was used. The first design was inspired by the designs presented by [Gidugu et al., 2024](#). The base of the finger consists of a rectangular cuboid with a width of 20mm, a length of 60mm, and a height of 8mm. The base also has six holes 3.4mm in diameter for attachment of the gripper, spaced 10mm apart and 5mm from the ends of the base. The finger itself consists of a right triangle that is 100mm in length and 2mm in thickness. The finger has nine struts, spaced evenly by 8mm, that are 2mm in thickness for the fin ray effect.

The decision to make the finger format in a right triangle, rather than an isosceles triangle, was motivated by the intended use of a linear gripper. This geometry allows one side of the finger to remain parallel to the opposing side and, in theory, enables the gripper to deform uniformly around the object during grasping. The figure 4.1 illustrates the dimensions of the first version of the developed fin ray gripper, presented in millimeters.

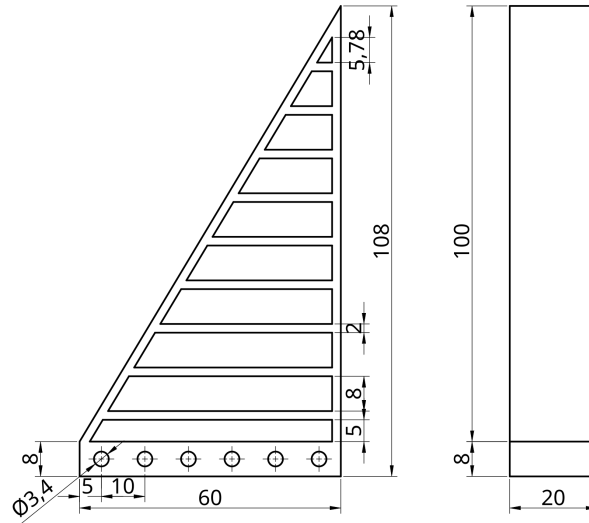


Figure 4.1: Dimensions of the first developed fin ray effect gripper finger (in millimeters)

To enable the attachment of the developed fin ray effect finger to the gripper assembly, a custom-designed holder was created to interface between the fingers and the main gripper body. The holder has a U-shaped geometry and shares the same length (60mm) and internal width (20mm) as the fingers. The two lateral walls are 8mm high and 2mm thick each and incorporate six holes with a diameter of 3.4mm that align with the corresponding holes located on the base of the finger, allowing it to be securely mounted to the holder.

For the attachment of with the main gripper structure, a small mounting extension with 10 mm in length was added. This extension has the same total width as the holder (24mm)

and a thickness of 5 mm, and it contains two 3.4mm holes spaced 16mm apart. These holes allow the holder to be fixed to the main gripper assembly using two screws and nuts. The dimensions of the first version of the developed holder for the fin ray finger are presented in Figure 4.2.

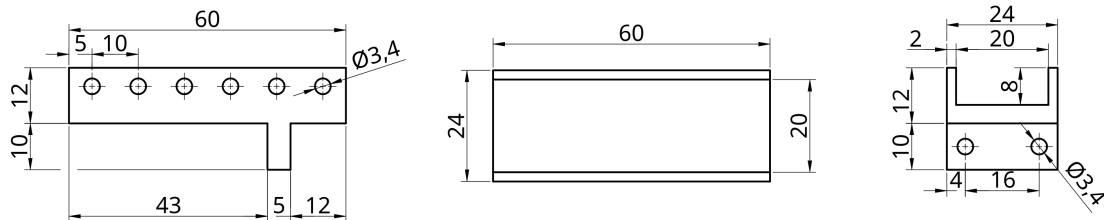


Figure 4.2: Dimensions of the first developed holder for gripper finger (in millimeters)

The following figure (figure 4.3) is a photograph the complete gripper, with the developed fin ray fingers attached via the developed holder. The gripper its at closed position.



Figure 4.3: Image of the complete assembly of the first version of fin ray gripper assembly

After testing the first gripper design, it was observed that the 2mm wall thickness of the fin ray fingers made the structure excessively stiff. This rigidity prevented the fingers from bending and deforming around the grasped object, reducing the overall effectiveness of the gripper and creating a risk of damaging delicate products. It was also found that the finger was noticeably stiffer on the right side when compared with the hypotenuse and that the stiffness increased toward the fingertip. As a result, the finger did not deform uniformly. When pressed against a rigid surface, such as the workspace table, the fingers also did not deform. To reduce stiffness, the ends of the fingers were trimmed. Although this modification improved the bending, the fingers still remained too rigid to properly deform around the grasped objects, and the asymmetry in flexibility between the two sides of the finger persisted.

Another issue identified during testing was the difficulty in attaching the fingers to the holder piece. The original finger dimensions did not account for 3D-printing tolerances, which resulted in a mismatch between the finger width and the holder's internal dimensions. To address this problem, a tolerance of 0.1mm on each side was introduced in the subsequent iterations of the fin ray fingers. Another problem concerned the extension used to attach the holder to the main gripper body. Due to the 90 degree angle, this part proved to be a structural weakness point. Due to its fragility, this section could break after extended use. The figure 4.4 shows the cut made to the fingers in an attempt to reduce stiffness at the tips, along with the original, unmodified fingers. The image on the left side corresponds to the original gripper fingers, while the right side image corresponds to the result of modification.

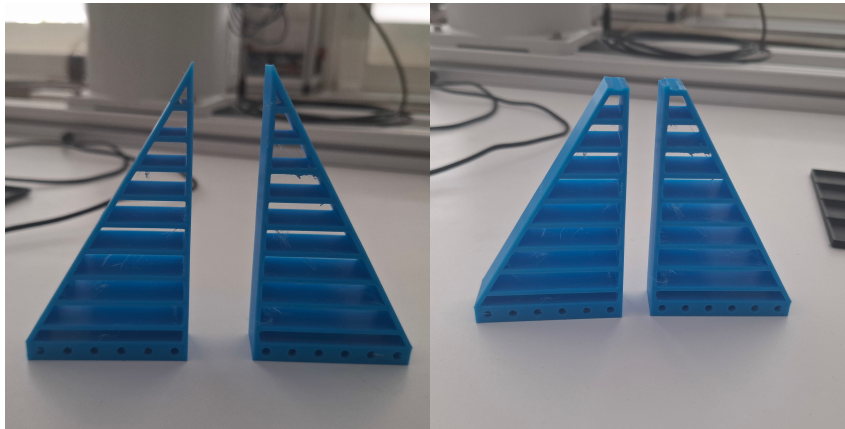


Figure 4.4: Solution attempt to the stiffness problem on the ends of the finger. left - before, right - after

In the second iteration of the fin ray finger design, the base still consists of the same rectangular cuboid format, although its dimensions were adjusted to incorporate the previously discussed tolerances. As a result, the base has now a width of 19.8mm and a height of 7mm (this height is the result of an error, it should have been 7.9 millimeters). The length remains unchanged, maintaining the 60mm. The number and dimensions of the mounting holes also remain the same, keeping the 3.4 mm diameter and the 10 mm spacing between holes. The finger shape changes from a right triangle to an isosceles triangle, while maintaining the 100mm in length. The overall thickness of the finger was reduced from 2mm to 1mm. Additionally, the internal structure was updated by increasing the number of struts from 9 to 10, now spaced 7.8mm apart. The updated dimensions for this second version of the fin ray gripper are shown in Figure 4.5.

Because of the changes introduced in the second fin ray finger design and the structural integrity issues observed, the holder piece also required a redesign. With the new finger format, the holder needed to incorporate an angle that would allow the two fin ray fingers to remain parallel to each other. Since the angle between the triangle legs and the base is

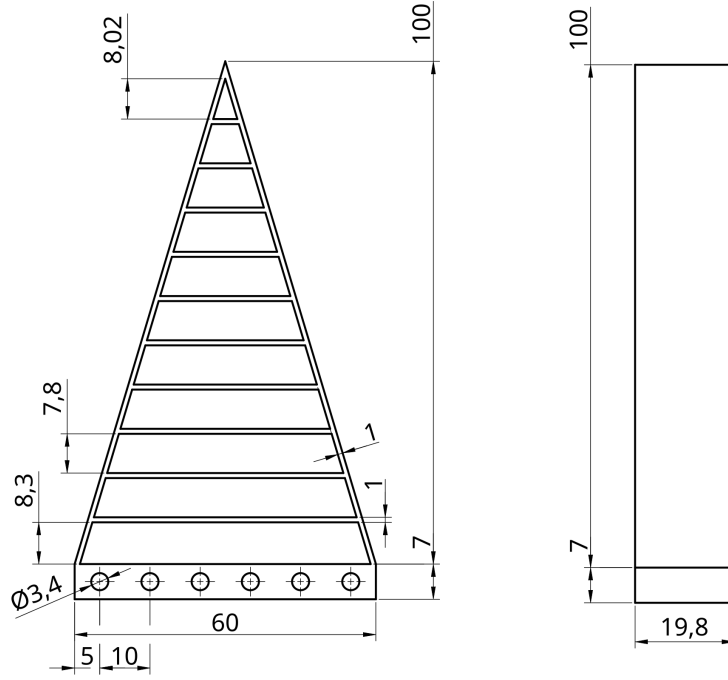


Figure 4.5: Dimensions of the second developed fin ray effect gripper finger (in millimeters)

73.301 degrees (measured in Onshape), the corresponding holder angle is calculated as:

$$\alpha = 90^\circ - 73.301^\circ = 16.699^\circ \approx 17^\circ$$

To adjust the holder to this angle, the extension section of the holder needed to be redesigned, as the gripper was unable to fully close with the original geometry. The extension part was rotated to the calculated angle and, to ensure total closure, extended to the front by a length of 3.669mm, calculated as:

$$\tan(\alpha) = \frac{\text{opposite}}{\text{adjacent}} \Leftrightarrow \text{opposite} = \text{base} = \tan(\alpha) \cdot \text{adjacent}$$

The remaining portion of the extension was kept identical to the previous version. The figure 4.6 illustrates the redesigned extension section of the holder piece.

To solve the issue with the structural integrity of the holder piece, a fillet with a radius of 7.3mm was added between the extension part. This permits the extension to resist against forces perpendicular to its direction. The figure 4.7 presents the dimensions of the second version of the developed holder part. The figure 4.8 shows a photograph of the second version of the full gripper assembly on the robot end effector.

When testing the second iteration of the Fin Ray gripper, it was observed that the fingers were significantly more malleable and bent more easily, allowing them to deform around the grasped product. This deformation around the product created a very secure grip. The gripper fingers were also able to bend when pressed against flat, hard surfaces, such

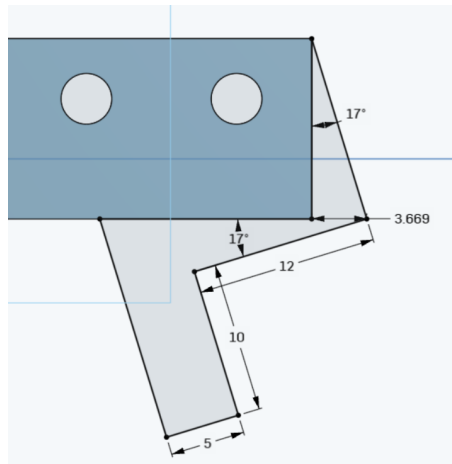


Figure 4.6: Dimensions of the redesign extension part of the holder (in millimeters)

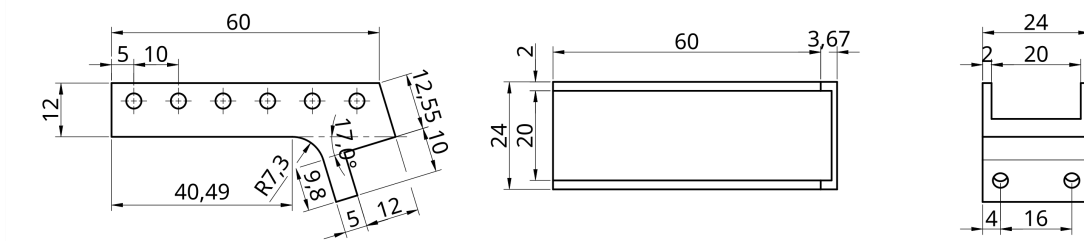


Figure 4.7: Dimensions of the second version of the developed holder for gripper finger (in millimeters)



Figure 4.8: Image of the complete assembly of the second version of fin ray gripper assembly

as the workspace table. This characteristic permitted grasping the products from below, allowing for an even more secure grip. The improvements on the holder piece were also visible. The holder part was much less fragile when compared with the previous design iteration. The introduction of the 3D printing tolerances allowed for an easier mounting of the gripper fingers on the holder. This allows changing the finger to be a much easier task. However, it was observed that even at full extension the gripper could barely clear the width of the *farinheira* product. In a case of small coordinate errors, this could lead to

the gripper not being able to properly grasp the product or causing damage.

To address the limitation previously identified in the gripper, a new holder design was developed. The extension part that attaches to the main gripper body was repositioned further toward the extremity with the objective to increased the maximum gripper extension distance. An extension of 10.5 was added to the design in order to create a separation of approximately 20 mm between the two Fin Ray gripper fingers, as calculated:

$$\cos(\alpha) = \frac{\text{adjacent}}{\text{hypotenuse}} \Leftrightarrow \text{hypotenuse} = \frac{\text{adjacent}}{\cos(\alpha)} = \frac{\text{distance}}{\cos(\alpha)} = \frac{10}{\cos(17^\circ)} = 10.4569 \approx 10.5$$

The figure 4.9 presents the dimensions of the final version of the holder part. The figure 4.10 shows a photograph of the final version of the full gripper assembly on the robot end effector.

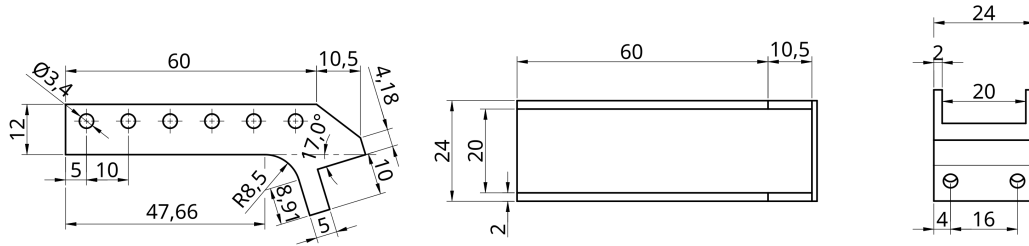


Figure 4.9: Dimensions of the third version of the developed holder for gripper finger (in millimeters)

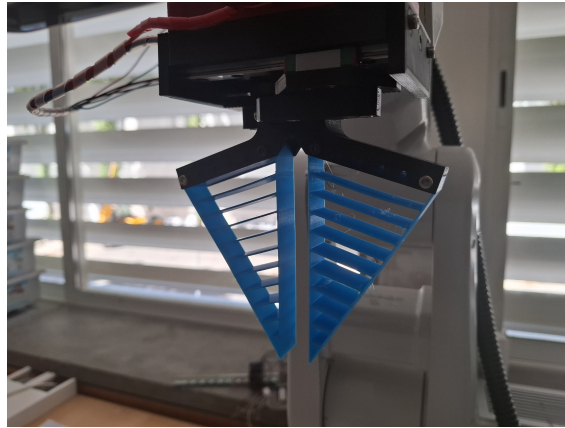


Figure 4.10: Image of the complete assembly of the third version of fin ray gripper assembly

With this modification to the holder part, the gripper was able to reliably pick up all products reliably without the risk of damaging them. Additionally, it gives the advantage of the manipulation system not requiring extremely precise positions from the vision algorithm, turning the robotic system more robust against potential noise and errors. The only downside to this solution is the inability to completely close the gripper. However,

this ability is not relevant for this project. The gripper's max width distance is around 100 millimeters.

4.4 Gripper module

The chosen actuator for the gripper module was the same as the one used in Lupi et al., 2024. This linear actuator has the advantage of not changing the gripper's reach as the fingers are closing or opening, unlike angular actuators. The gripper is made of 3D-printed parts made from Polylactic acid (PLA). The fingers are actuated via a stepper motor, specifically a Nema 17 bipolar motor with a step angle of 1.8 degrees and with a holding torque of 13 Ncm. The max gripping force is 40 Newtons. The gripper controller is based on an ESP-32 microcontroller (Lupi et al., 2024). The model of this gripper assembly actuator is available in the GitHub repository: <https://github.com/NOVA-RICS-Open-Lab/Linear-Gripper-3D>. Commands are sent to the gripper via HTTP requests, allowing the distance between the fingers to be adjusted. The gripper module also allows for easy replacement of the fingers, which is advantageous during the development of the Fin Ray effect fingers, as it enables rapid testing of new designs. This gripper module has an extension to mount the depth camera for the computer vision part of the system.

The figure 4.11 shows a rendered version of the complete assembly of the gripper system. The item highlighted by the number 1 is the camera attached to the gripper assembly. The camera selected for this work is an Intel RealSense D415 depth camera, as previously stated in the chapter 3. The item highlighted by the number 2 is a light source utilizing an Light-Emitting Diode (LED). This feature of the gripper was not used for this work. The item highlighted by the number 3 is the main body of the gripper assembly. It houses the stepper motor that makes the fingers open and close. The items highlighted by the number 4 are the adapters for the attachment of the fin ray effect gripper fingers to the main body of the gripper assembly, described in the previous section 4.3. The items highlighted by the number 5 are the fin ray effect gripper fingers themselves, described in the previous section 4.3.

4.5 Gripper's digital interface

As mentioned in the previous section, the control of the gripper is based on an ESP-32 microcontroller. It controls the stepper motor and the distance between the two Fin Ray fingers. The communication to this controller is done through HTTP requests, using the POST method. The controller has two commands, one to open the gripper in its totality and the other to close the gripper to a determined distance. The first command can be called by sending an HTTP request with the POST method with the path `open`. The second command can be called with the path `close`, setting the desired distance on

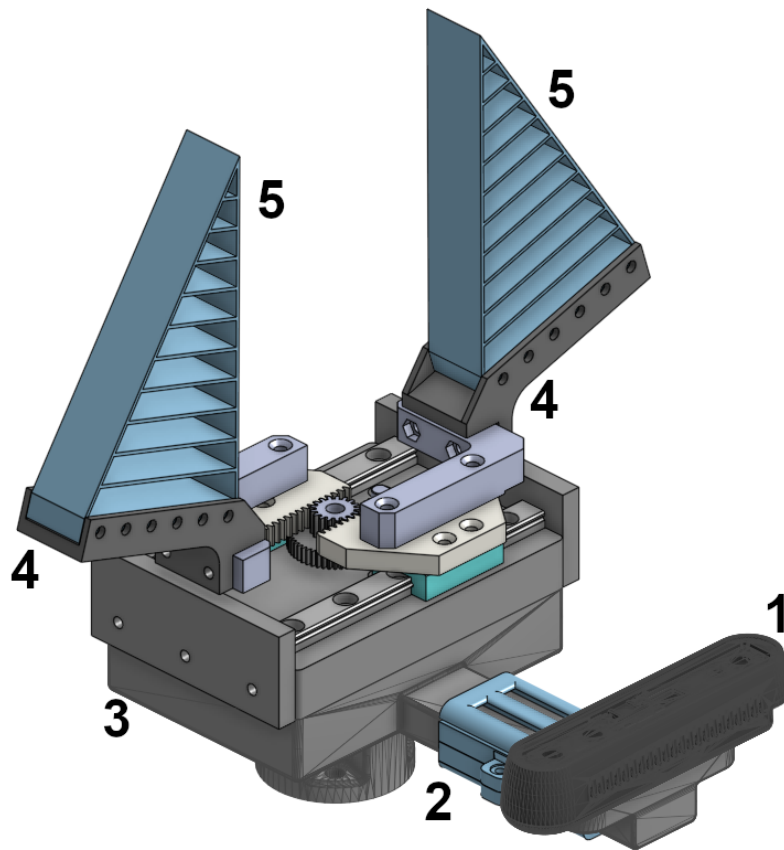


Figure 4.11: Image of the complete 3D module of the gripper

the parameter dimension. The IP address of the gripper controller is 192.168.1.2. This method of interfacing with the gripper module does pose some security issues, as it does not guarantee that the gripper is safe from attacks. The focus of this work is not on the communication between the gripper and the robot controller, and as such this issue will not be resolved. Future work should, however, improve this method of communication for the gripper.

Initially, to test these commands, a Python script was developed. The script launched a small window application that allows the user to input a desired value for the distance and call the two previously described commands. The figure 4.12 presents the graphic interface of the developed Python test script for testing these commands.

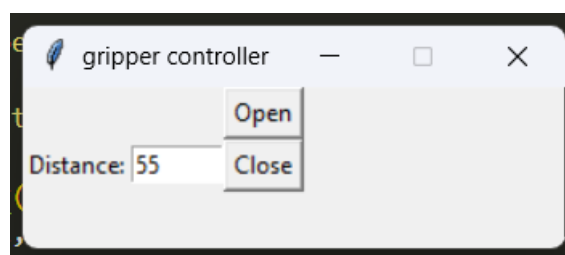


Figure 4.12: Image python script for testing the gripper HTTP request services

To integrate the gripper controller with the rest of the robotic system, a Robot Operating System (ROS) node was developed. This node implements two ROS services, a service to open the gripper to its full extension and another to close or open the gripper to a given distance. With this node, the main controller node for the robot can call the services and operate the gripper during its run time without the need to handle the HTTP requests on the main code. This approach also provides a higher level of versatility and customizability to the robotic system. In a case where the robot gripper assembly might change, the main controller code won't need any change. The user just needs to update the gripper's controller ROS node or use another ROS node as a substitute. This node is further discussed in the [chapter 6](#).

POSE ESTIMATION ALGORITHM DEVELOPMENT

This chapter describes the development process of the 6D object pose estimation algorithm, created to detect and calculate the position and orientation of the various agri-food products. The first section presents the overall pipeline of the computer vision algorithm. The second section describes the hardware used for the vision system. The third section presents the image preprocessing steps. The fourth section describes the development of the algorithm, with subsections dedicated to each stage: object detection and segmentation, position calculation, and orientation calculation. The final section explains the decisions behind the chosen approaches for the object pose estimation algorithm, as well as some final considerations including limitations and shortcomings of the algorithm.

5.1 Algorithm pipeline

The developed object pose estimation algorithm took inspiration from various approaches and methods discussed in the state of the art chapter (section 2.3). The detection and segmentation stage was inspired by the approaches from the works by Tran and Lin, 2020; Wong et al., 2023; Höfer et al., 2021; Chen et al., 2020. In addition, it incorporates the idea of using a keypoint approach from the works by Gupta et al., 2019; Chen et al., 2020.

The developed algorithm can be divided into three stages. The first stage is responsible for the object detection and segmentation. It obtains the list of correspondent pixels of the detected object present on the Red Green Blue (RGB) image given by the camera. The second stage is tasked with calculating the position (in Cartesian coordinates) of the various detected objects. The third stage is responsible for calculating the object orientation (in Euler angles) in relation to the camera. The figure 5.1 presents the overall pipeline of the developed pose estimation algorithm.

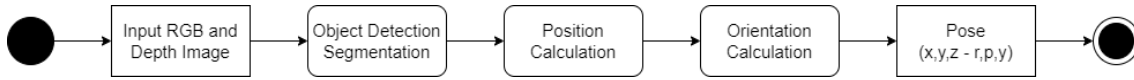


Figure 5.1: High level flowchart of the developed algorithm

5.2 Camera system hardware

The approach used for the algorithm for 6D pose estimation requires the use of a vision system capable of measuring the distance between the environment and objects to the camera. For that, either a stereo vision camera system or an Red Green Blue Depth (RGB-D) camera can be used. The chosen camera for this work was the Intel RealSense D415, a depth camera that utilizes stereoscopic technology to calculate the depth for each pixel. The camera has two imagers, with an Infrared (IR) projector between them that projects non-visible light patterns to help with the depth calculation on low-texture surfaces. The camera also includes an RGB camera for color images (Intel Corporation, 2019).

The camera is connected to a computer using a Universal Serial Bus (USB) cable. It requires the cable to be capable of transmitting in USB2.0 or USB3.1 Gen 1. However, when using a cable capable of only USB 2.0, the maximum resolution can only reach 640 by 480 pixels. This greatly diminishes the quantity of data and, consequently, the precision of the depth measurement when compared with the maximum resolution of 1280 by 720 pixels (Intel Corporation, 2019). Initially the development of the computer vision system was being made with a cable only capable of USB 2.0 transmission. Later this cable was replaced by one capable of USB 3.1 transmission. The camera is mounted on the gripper assembly described in the gripper development chapter (section 4.3), from an extension on the gripper body.

5.3 Depth and color images preprocessing

Before the images provided by the depth camera can be used in the pose estimation algorithm, they have to be preprocessed to reduce the noise on the images. Especially the depth image, which is relatively vulnerable to noise caused by light sources. Additionally, the frames provided by the camera have an offset between them and must be aligned before use.

The first step of the preprocessing is the alignment of the frames. This is done using the `pyrealsense2` library provided by Intel using the `align` object. For the filtering step of preprocessing, various filters provided by the same library were tested. The tested filters provided by the Intel library are decimation filters, threshold filters, temporal filters, and spatial filters. These filters were tested using the Depth Quality Tool for Intel RealSense Cameras provided by Intel (shown in the figure 5.2).

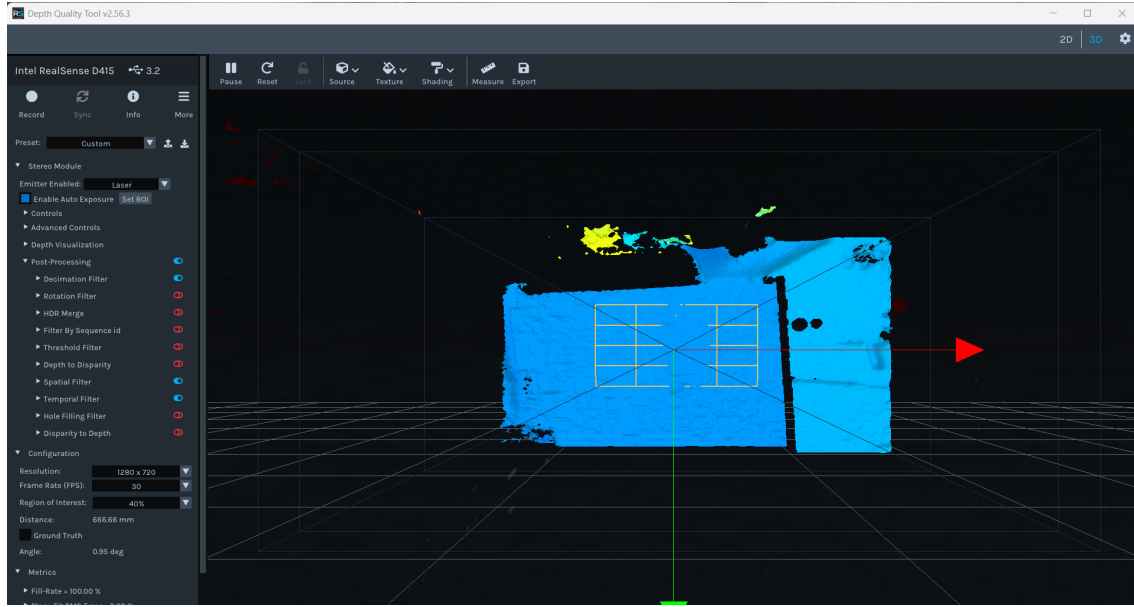


Figure 5.2: Intel - Depth Quality Tool for Intel RealSense Cameras

Experimental results showed that the best preprocessing procedure involved a combination of a decimation filter with a temporal filter. The decimation filter was applied to the depth frame with a magnitude of 2, downsampling the resolution of the image in half. The temporal filter was configured with an alpha value of 0.3 for the exponential moving average (Intel RealSense SDK Team, 2025). A delta factor of 20 for the step boundary to help preserve edges (Intel RealSense SDK Team, 2025). Additionally, it employed a hole-filling mode of "Valid in 2/last 4", meaning that a pixel's depth value would be preserved if the pixel had a valid depth value in 2 frames of the last 4 frames (Intel RealSense SDK Team, 2025).

Because of the decimation filter, the RGB and depth images' resolution becomes different (the resolution of the depth images is half of the color image); for the vision algorithm to make the correspondence between the depth pixels and color pixel, the resolution of the depth image needs to be upscaled. This is done by resizing the image using the NumPy library with the function `np.repeat`. This repeats the pixels two times, effectively doubling the size of the image back to its original resolution.

5.4 Algorithm development

As previously stated, the developed algorithm can be divided into three stages: object detection and segmentation, position calculation, and orientation calculation. Each stage builds from the previous one. The main objective of this algorithm is to calculate 6D pose, a combination of position (Cartesian coordinates of the object's center) with the orientation of the object in relation to the camera (in Euler angles, x-y-z order). With this

pose, the robotic system will be able to plan an approach to pick and place the detected product.

5.4.1 First stage - Object detection and segmentation

The first stage of the algorithm's pipeline is to detect the object and obtain its corresponding pixels on the color (RGB) image. For that, a You Only Look Once (YOLO) segmentation model was chosen because it could at the same time do the object detection operation as well as the segmentation task and for its operation speed (Ultralytics, 2023a). Version 11 of the model, YOLO11-seg, was chosen, as it was the latest version.

The use of a neural network, such as YOLO neural networks, requires it to be previously trained before its deployment. For that reason, the first task for the development of the first stage is to obtain a dataset for training the model. The first version of the dataset had 154 images in total: 133 for training, 14 for validation, and 7 images for testing (86%, 9%, 5%). The test images are used for the verification of the performance of the trained model. The images were augmented with rotations of 90 degrees, clockwise and counterclockwise, and inversions of the image (upside down) and had a resolution of 640 by 480 pixels. With this dataset, the first model was trained using a pre-trained YOLOv11 nano segmentation model with 50 epochs.

The dataset is composed of images with just one of the chosen agri-food products, images with multiple products, and images with one product on top of another. All the products were labeled using the Roboflow website in the instance segmentation mode. In the case of images with products on top of each other, just the product that is on top is labeled. This is done to avoid detecting objects that are occluded by others and just focus on the top products that can be picked first. These labeling rules are followed for all the following datasets.

During the testing of the trained model, it was found that it lacked generalization. If a product was placed in a location that wasn't represented in the dataset, the model would either assign a very small confidence score for that product or would fail to detect it. This was more noticeable when the product was placed at angles of 45 and -45 degrees. The results of the first model training are presented in the figure 5.3. The plots illustrate the evolution of two standard performance metrics, precision and recall. The labels **B** and **M** refer to the bounding box and segmentation mask outputs, respectively (Ultralytics, 2023b).

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

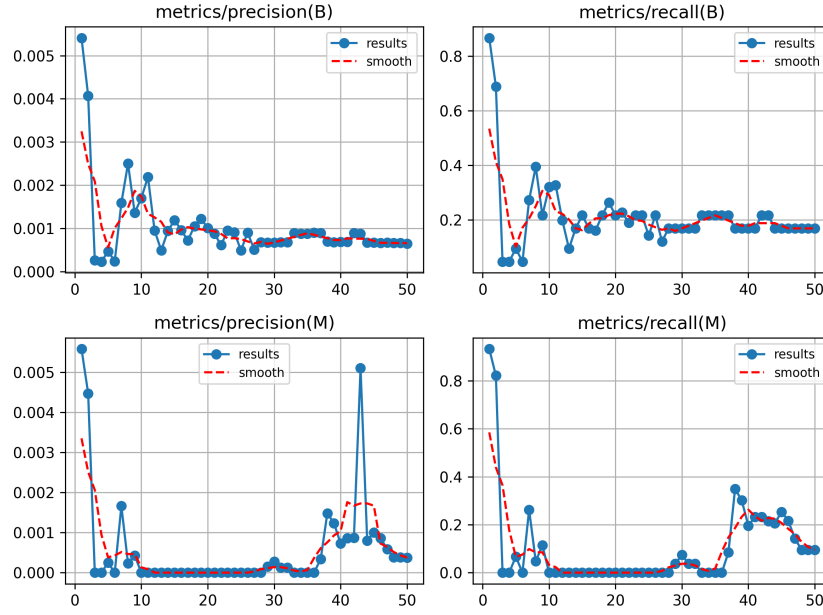


Figure 5.3: Results of the training of the first model

To solve the issue of lack of generalization, a new dataset was created from the same images as the previous one, adding rotations between 45 and -45 degrees. This created a dataset with 209 images in total, 183 images dedicated for training, 17 images for validation, and 9 images for testing (88%, 8%, 4%). The model was trained using, again, a pre-trained YOLOv11 nano segmentation model with 50 epochs.

During the testing of the second trained model, it was observed that with the rotations applied to augment the dataset, the trained model showed better generalization. No longer having difficulty with positions and object angles that were not represented on the labeled images. The figure 5.4 presents the results of the training of the second model.

In an attempt to improve the model, new images were added to the dataset, as well as new augmentation post-processing steps. The third version of the dataset is composed of 573 images in total, with 546 images dedicated to the training of the model, 18 images for validation, and 9 images for testing. The augmentation processes used were the vertical and horizontal flipping of the image, the rotation by 90 degrees of the image in clockwise and counterclockwise directions, the rotation of the image by 180 degrees (upside down), rotation by -45 and 45 degrees, and finally the addition of noise (up to 1.49% of the pixels). The size of the model used for training was also changed from nano to small in an attempt to get even better results for the segmentation predictions.

When testing the newly trained model, it was found that it gave better results for the instance segmentation mask as well as better confidence scores for the detected objects. However, the processing time of the model was increased dramatically. This is due to

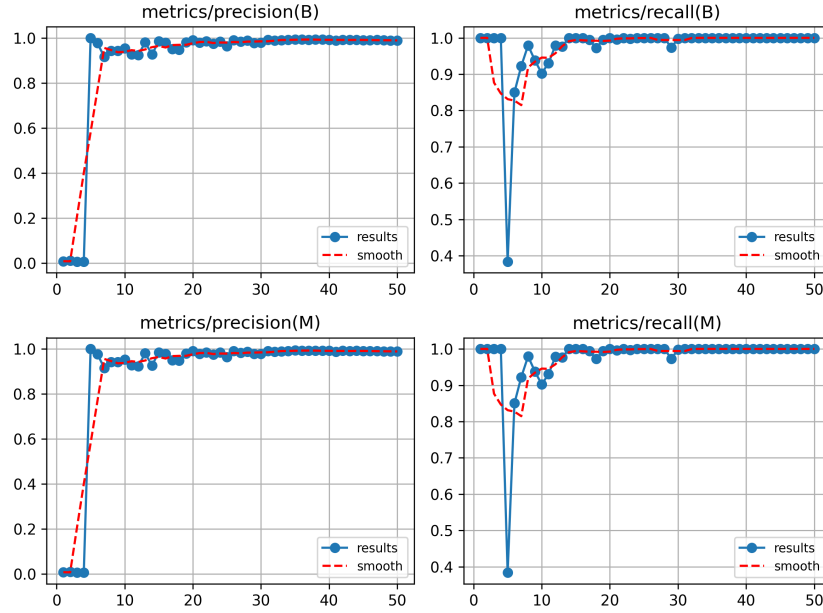


Figure 5.4: Results of the training of the second model model

the change of model size from nano to small. The figures 5.5 presents the results of the training of the third model. The figure 5.6 presents a chart with a speed test done with 120 images. The first trained model corresponds to the blue line, the second trained model corresponds to the orange line, and the third trained model corresponds to the green line. The chart shows that the processing time for the model with the "small" size takes around double the processing time when compared with the models of "nano" size. The "nano" models take close to 100 milliseconds for each inference, while the "small" model takes close to 200 milliseconds.

When the USB cable was replaced with a cable capable of USB 3.0 transmission, it consequently allowed the capability of using images of 1280 by 720 pixels in resolution. This allows for better depth precision, as this resolution is the optimal setting for the camera (Intel Corporation, 2019). As a downside, it also requires the use of a new model capable of handling images with the new resolution.

For the new model, a new dataset was created. Composed of 230 images, 201 for training, 19 images for validation, and 10 images for testing (87%, 8%, 4%). The augmentation processes used were horizontal and vertical flips, rotations between -45 and 45 degrees, brightness changes between -15% and 15%, and the addition of noise.

When testing the model, it was found that it lacked generalization on some positions, more specifically closer to the center of the image. This is probably caused by the lower quantity of images when compared with the previous dataset. Another problem found with the newly trained model is that the confidence scores for the detected and segmented

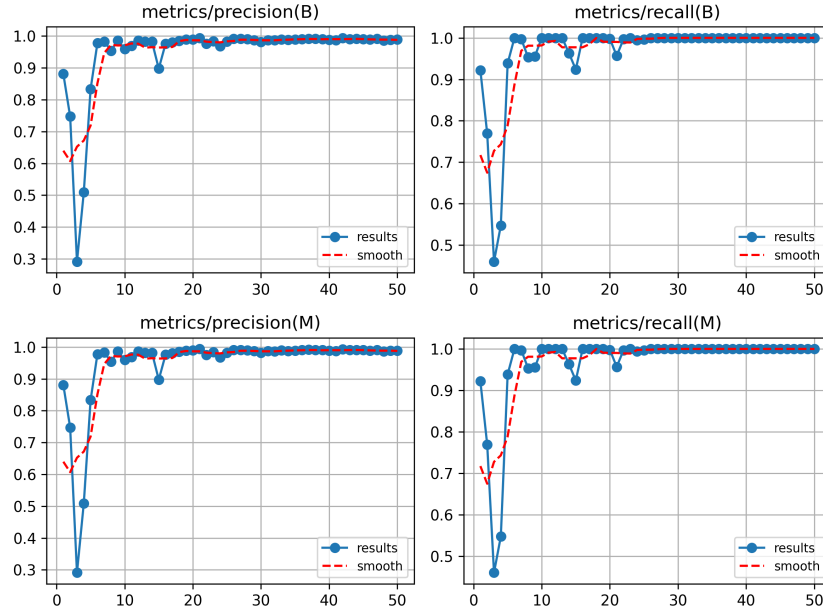


Figure 5.5: Results of the training of the third model model

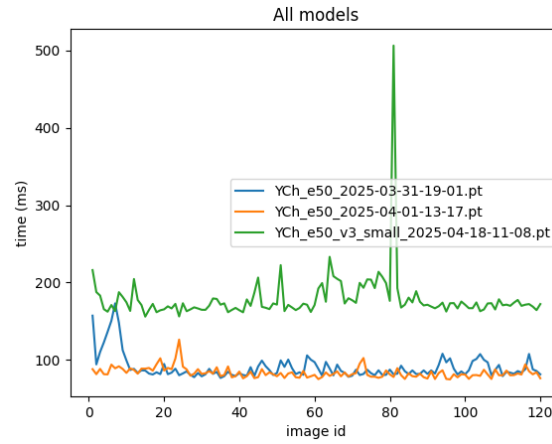


Figure 5.6: Speed test for the first three trained models (resolution 640x480)

objects were lower. The previous model would give scores starting from 90%, while this model would give scores starting from 80%. The figures 5.7 presents the results of the training of the fourth model (the first using the resolution of 1280 by 720 pixels).

To solve the issue of lack of generalization in the center of the image, new images were added to the dataset with representation of the positions where the model had difficulty detecting objects. Additionally, more images were added to the validation group. The new dataset has 229 images, with 195 images for training, 24 images dedicated to validation, and 10 images for the testing group (85%, 10%, 4%).

With the new dataset, the issue of lack of generalization in the center of the image was solved. The newly trained model was now capable of detecting and segmenting objects in

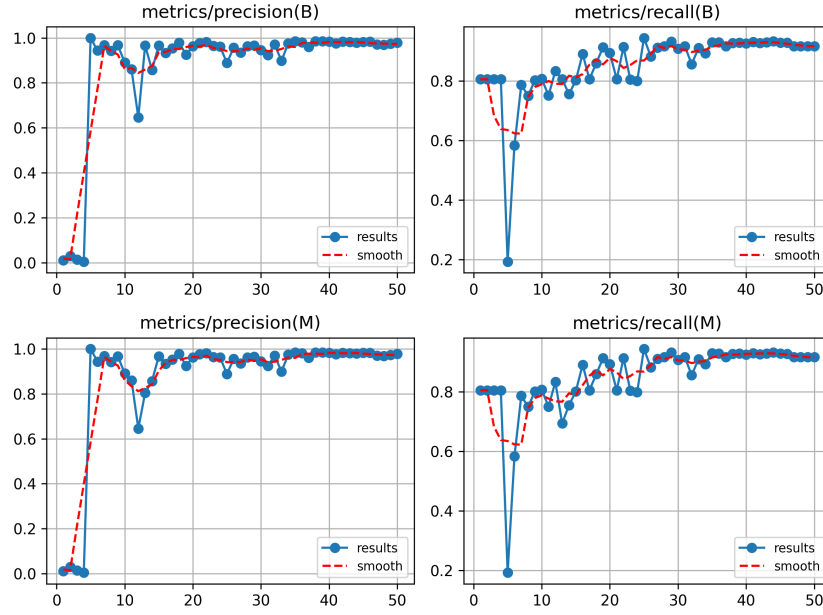


Figure 5.7: Results of the training of the fourth model model

the center of the image. However, the model still had the issue of relatively low confidence scores. This is resolved by lowering the confidence score filter value. The figure 5.8 presents the results of the training of the final yolo segmentation model.

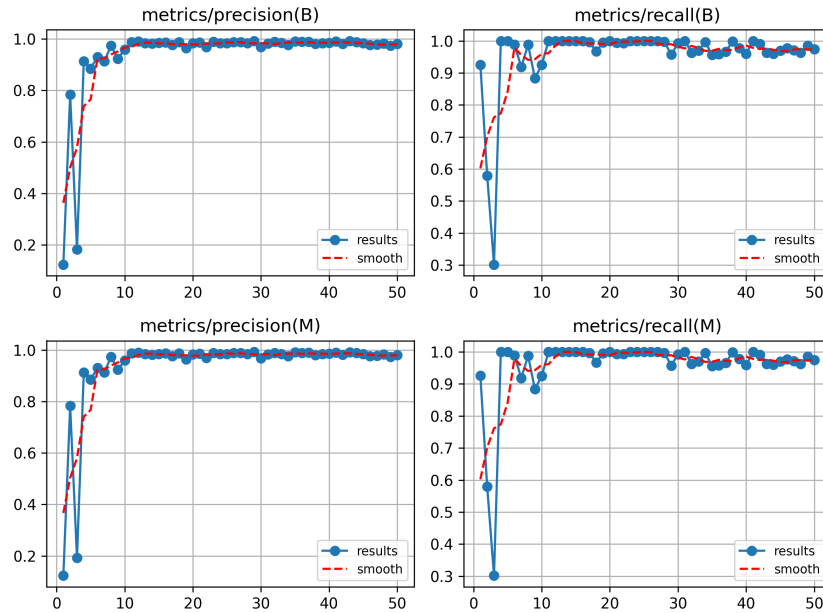


Figure 5.8: Results of the training of the final model model

The architecture of the used YOLO model requires the input image size to be a multiple of 32. Consequently, the images with the resolution of 1280 by 720 pixels are fed to the model. The resulting segmentation mask, returned as a binary image, has a resolution

of 1280 by 736 pixels, since 8 pixels are added to both the top and bottom of the image to ensure that its vertical dimension is a multiple of 32. This requirement comes from the initial layers of the YOLOv11 architecture, which downsample the image. Therefore, once the model generates the segmentation mask, 8 pixels are removed from the top and another 8 from the bottom of the resulting output.

The output values from this first stage of the algorithm are the detected object's confidence score, class name, bounding box, and the instance segmentation mask in the format of a binary image. The figure 5.9 presents the results of the YOLO segmentation model. Note that in the right image, only the top product is detected and segmented. The product on the bottom is ignored by the model. This behavior is intended, as the product on the bottom could not be picked before the top product is moved away.

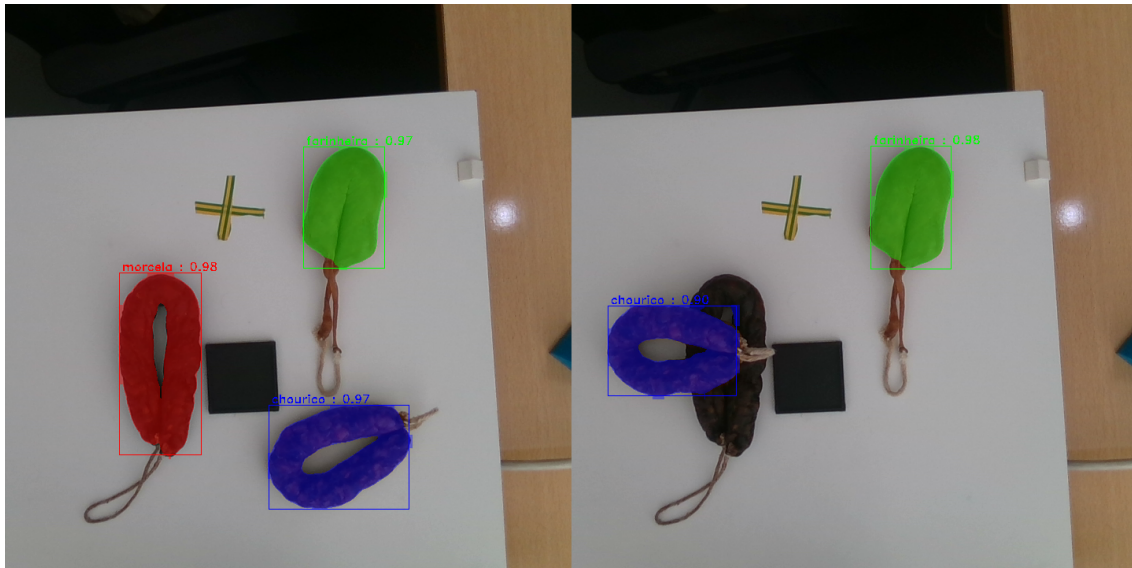


Figure 5.9: Result of the model for a images with all products. The right image shows an example with one of the products being occluded by another

5.4.2 Second stage - Position calculation

The second stage involves the calculation of the position of the center of the object in relation to the camera system. Two methods were developed for this task. The first one uses the average pixel depth of the object. The second uses the individual depth of each pixel of the object to create a point cloud of the object. Both of these approaches require the use of the object's segmentation mask provided by the previous stage model. The output of this stage is the coordinates center point of the object (3D point) as well as the centroid (2D center of the object).

5.4.2.1 Average depth method

The first developed method to calculate the position of the object directly used the depth values given by the camera. Initially all the corresponding pixels of the segmentation mask have their depth values all summed to obtain the average depth. Special caution needs to be taken due to invalid depth pixels (pixels with a depth value equal to 0). Before calculating the average, the invalid pixels are filtered out.

The next step is to obtain the position of the centroid of the object. This is done by calculating the image moments of the object. Moments in image processing are a form of weighted average of the pixel intensities.

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Where $I(x, y)$ is the intensity of the pixel with the coordinates x, y . To calculate these moments, the OpenCV library was used. Using the M_{00} , M_{01} , and M_{10} moments, it's possible to calculate the x and y coordinates for the centroid of the object.

$$\text{Centroid}_x = \frac{M_{10}}{M_{00}}, \text{Centroid}_y = \frac{M_{01}}{M_{00}}$$

The final step of this method is to do the 3D point "deprojection". This is done using the Python wrapper of the Intel RealSense SDK 2.0, `pyrealsense2`. Using the function `rs2_deproject_pixel_to_point` with the calculated average depth value and the object's centroid, it is possible to obtain the Cartesian coordinates of the center of the object. This function additionally requires the video stream intrinsics of the depth camera. The figure 5.10 presents a simplified pipeline of the presented method for the center point of the object that corresponds to the Cartesian position of said object.

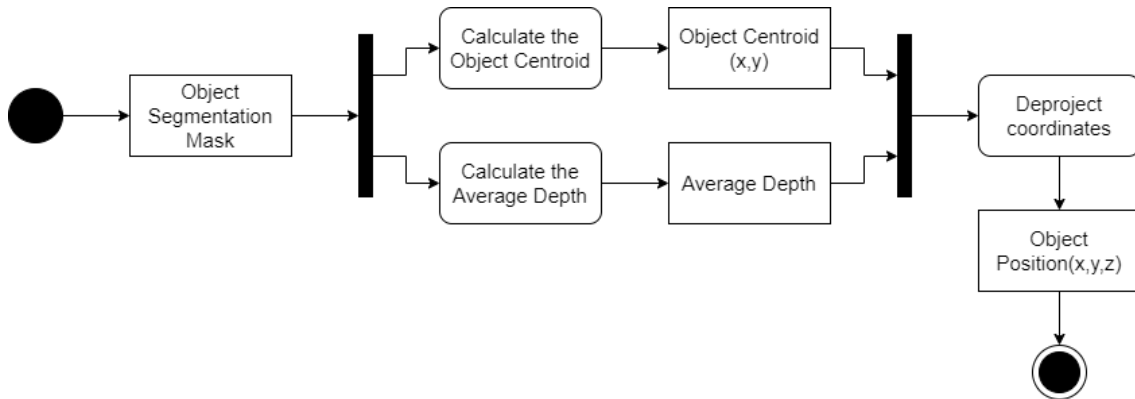


Figure 5.10: Simplified pipeline for the average depth method for the calculation of the center point of the object

5.4.2.2 Point Cloud method

The second developed method to calculate the position of the object used the depth values of the corresponding pixels of the segmentation mask to first create a point cloud and then use it for the calculations of the position of the object.

The first step of this method is the creation of a point cloud for the detected object. This is done by "deprojecting" each individual pixel of the segmentation mask provided by the previous stage. Like with the calculation of the average depth on the previously described method, the invalid pixels are also filtered. This first step outputs a list of 3D points, which corresponds to the object's point cloud.

The second step is calculation of the average on the 3 axes. This step obtains the object's 3D center point, which corresponds to the Cartesian position of the object in relation to the camera. Because the centroid is also needed for the third stage of the algorithm, its center point is "projected" onto the 2D space using the function `rs2_project_point_to_pixel` from the `pyrealsense2` library. The figure 5.11 presents a simplified pipeline of the presented method for the center point of the object.



Figure 5.11: Simplified pipeline for the point cloud method for the calculation of the center point of the object

The two developed methods were tested back-to-back for comparison in the same experimental conditions. The two methods showed very close results. However, the method that used the average depth for the calculations was faster and much less computationally expensive. This difference in speed results from the creation of the point cloud, which is slower than calculating the average depth of all pixels corresponding to the segmentation mask. The processing speed between the two methods on the images with the resolution of 1280 by 720 pixels was significantly different, due to the size of the images. For that reason, unless the third stage requires the use of a point cloud, the first method will be preferred. The figure 5.12 presents a comparison between the two developed methods. The left image is the method that uses the average depth. The right image is the point cloud method.

5.4.3 Third stage - Orientation calculation

The final stage of the algorithm is responsible for the calculation of the orientation of the detected object in relation to the camera system. Various methods were developed and tested for this stage. These include using approaches such as Principal Components

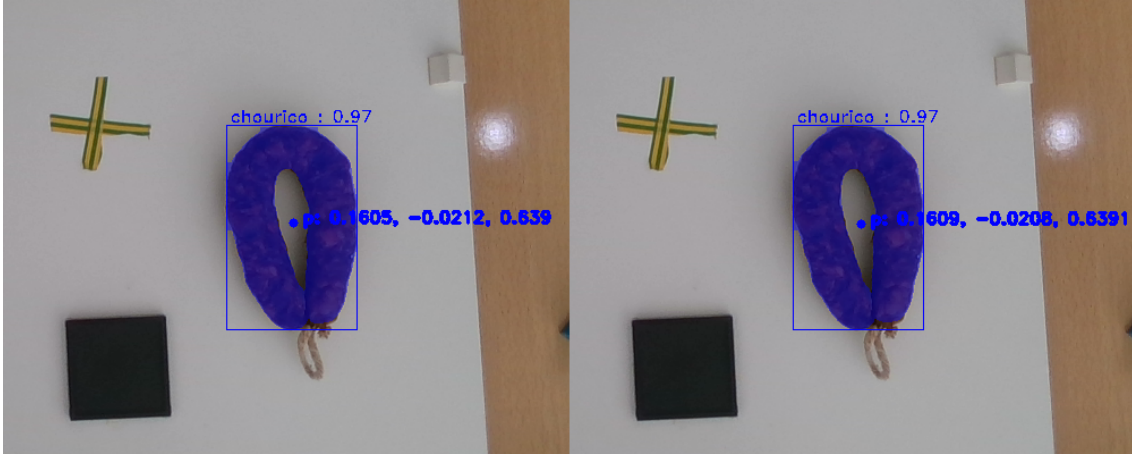


Figure 5.12: Comparison between the depth values method and the point cloud method

Analysis (PCA), Ordinary Least Squares (OLS) regression, Singular Value Decomposition (SVD), and keypoints.

5.4.3.1 PCA 2D method

The first developed method used PCA on the segmentation mask points to obtain the rotation of the object in the 2D plane. PCA is a statistical method for data reduction, in which n axes are rotated to align with the directions with the most possible data dispersion. Each n th axis follows the n th direction with the most data dispersion Gewers et al., 2021. For example, given a set of points in a 2D space with an elliptical format, the first axis calculated by the PCA will follow the semi-major axis, while the second axis will follow the semi-minor axis. Figure 5.13 is a graphic representation of this example.

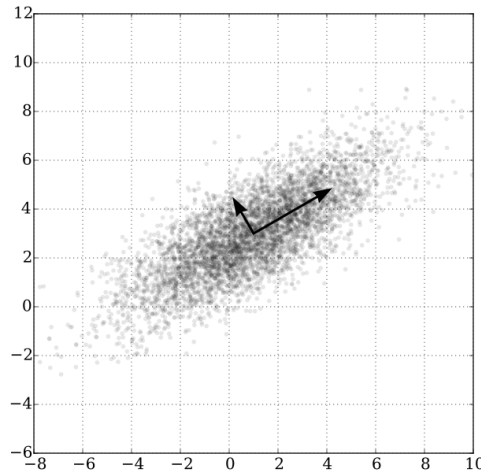


Figure 5.13: Visual representation of the PCA method (source: Nicoguardo, 2016)

The first developed method uses this technique to obtain the principal component axes of the detected object. These axes will correspond to the x and y axes in the 2D space (x corresponding to the first PCA axis and y to the second). Due to the shape of the object in the image, the two axes will always be orthonormal. Consequently, only the angle between the image x-axis and the first PCA axis needs to be considered. This angle represents the rotation of the object in the 2D plane.

The PCA operation is done using the function `PCACompute2` from the OpenCV library. Before using the segmentation mask points for the PCA, their data structure needs to be modified. The YOLO outputs the segmentation mask as a NumPy multi-dimension array composed of 32-bit integers, while OpenCV requires the use of a multi-dimension array composed of 8-bit integers. To modify the YOLO's segmentation mask data structure, each value of the multi-dimension array is multiplied by 255 and turned into an 8-bit integer.

One limitation of this method for orientation is that it can only calculate the angle of the object in the 2D space. This is fine if the object is rested on the table. However, if the object is at an angle with the workspace table, the robotic system will have some difficulty creating a valid grasping pose to pick and place the object. The figure 5.14 presents the result of the application of the method for object orientation calculation. Note that the object only has rotation on the z-axis (yaw), while the other angles always have a value of 0.

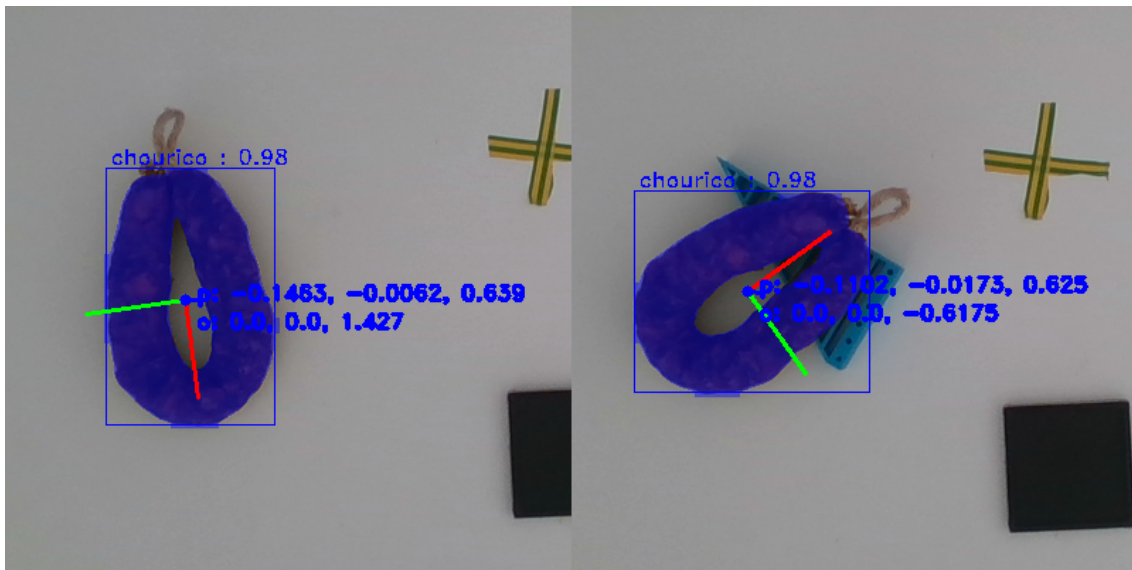


Figure 5.14: Results of orientation calculation method that uses PCA on the segmentation mask. On the right image the product has some inclination.

5.4.3.2 PCA 3D method

The second developed method also used PCA to obtain the object orientation. However, instead of being used on the segmentation mask point, it is used on the point cloud of the object. This enables the analysis to be performed with three axes. The library scikit-learn (sklearn) is used for this task.

The first principal component axis represents the x-axis of the object, the second principal component axis the y-axis, and the third the z-axis. These axes are vertically stacked as a rotation matrix.

$$R = \begin{bmatrix} x_{axis} \\ y_{axis} \\ z_{axis} \end{bmatrix}$$

The Euler angles that represent the orientation of the object are extracted from this matrix. The order of extraction of the Euler angles affects their values. The use of intrinsic rotations, rotations around the coordinate system of the object, or extrinsic rotations around the fixed coordinate system will also alter the angle values. Consequently, the extraction of the Euler angles must follow the conventions of Robot Operating System (ROS), which are extrinsic rotations in the other x-y-z (roll, pitch, yaw). To extract these rotations, two functions from the SciPy Python library are used. The function `from_matrix` is first used to create the rotation matrix object. Next, the function `as_euler` with the argument 'xyz' is used to extract the Euler angles in the correct order.

During the testing of this approach, it was observed that it had minor problems when the product was rested parallel to the workspace table. In the case of the result shown in the figure 5.15, having a deviation of around 7.45 degrees on the roll angle. However, when the product was rested at an angle against the table, this method of orientation calculation completely failed. The figure 5.15 presents the results of applying this method for orientation calculation.

5.4.3.3 OLS linear regression method

Another method tested for orientation calculation is the OLS linear regression technique. This method obtains a plane that traverses the points from the object's point cloud, and using the plane normal vector, it is possible to calculate the object's orientation by comparing it to the normal vector of the xy plane.

OLS is a technique for the estimation of the coefficients of a multiple regression model. The objective of this method of estimation is the minimization of the value of the least

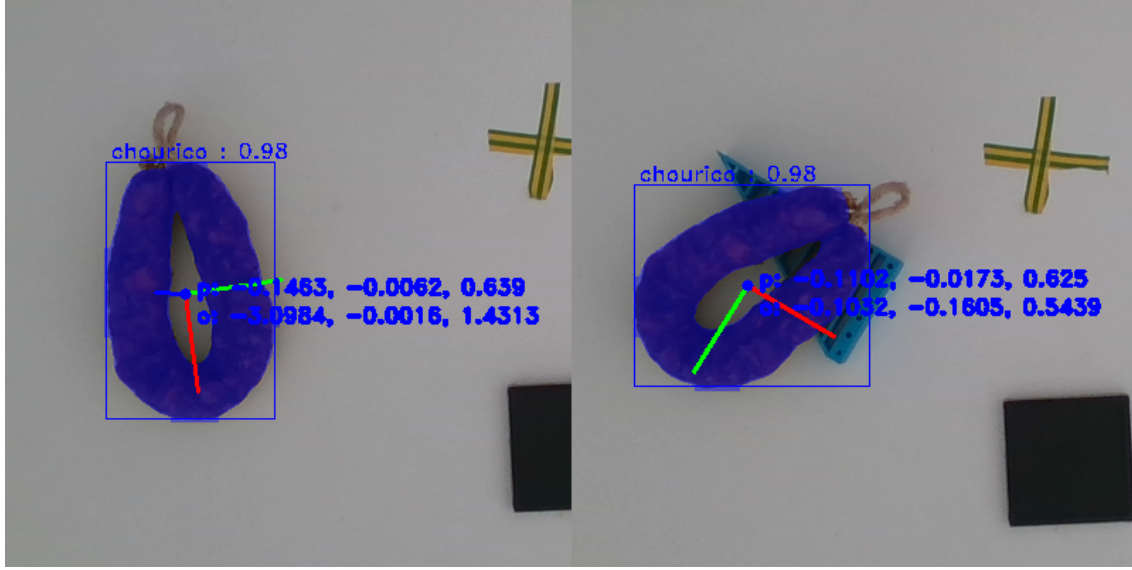


Figure 5.15: Results of orientation calculation method that uses PCA on the object's point cloud. On the right image the product has some inclination.

squares function, where $\beta_0, \beta_1, \beta_k$ are the coefficients of the regression.

$$L = \sum_{i=1}^n \epsilon_i^2 = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^k \beta_j x_{ij} \right)^2$$

Using the point cloud of the object, the scikit-learn library applies the OLS technique. The coefficients were estimated using the object `LinearRegression` with the function `fit`. The plane is created from the coefficients obtained from the OLS regression, as:

$$\beta_1 x + \beta_2 y + \beta_0 = z$$

The next step of this approach is to compare the OLS plane's normal vector to that of the xy plane. A rotated axis is obtained by the cross product between the normalized OLS plane's normal vector and the normal vector of the xy plane. The rotation angle is calculated from the *arccosine* of the dot product between the normalized object's normal vector and the normal vector of the xy plane.

$$\vec{Rot}_{xyz} = \vec{N}_{object} \times \vec{N}_{xy}$$

$$\theta_{Rot} = \arccos(\vec{N}_{object} \cdot \vec{N}_{xy})$$

Using the rotation axis vector and the rotation angle, a quaternion is constructed to represent the orientation of the object in relation to the camera system.

$$Q = \begin{cases} w = \cos(\frac{\theta_{Rot}}{2}) \\ x = \vec{Rot}_x \sin(\frac{\theta_{Rot}}{2}) \\ y = \vec{Rot}_y \sin(\frac{\theta_{Rot}}{2}) \\ z = \vec{Rot}_z \sin(\frac{\theta_{Rot}}{2}) \end{cases}$$

From the quaternion, it is possible to extract the orientation as Euler angles by first converting it to a rotation object using the SciPy function `from_quat` and then using the `as_euler` with the argument 'xyz'.

This method has the limitation of not being capable of determining the rotation around the z-axis (yaw). This is a consequence of just comparing the object's estimated plane normal vector to the normal of the xy plane. To be able to determine all the angles, the object's normal vector would have needed to be compared with at least two planes' normal vectors. The figure 5.16 presents the results of applying the described orientation calculation method. The method showed promising results when applied to a detected product with no angle with the workspace table. However, for products with some inclination, the method was shown to have worked poorly.

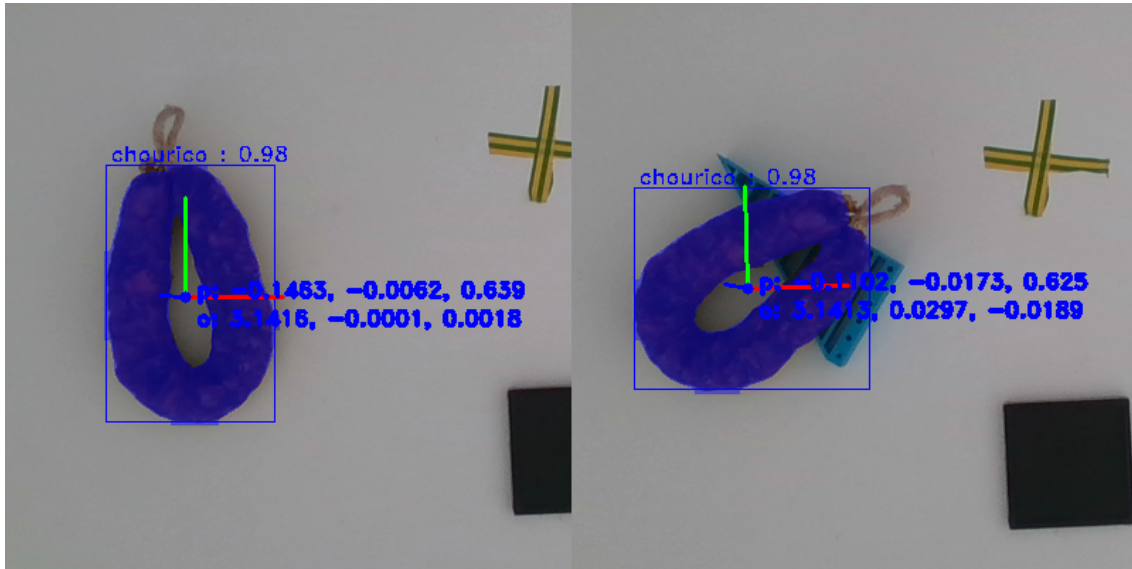


Figure 5.16: Results of orientation calculation method that uses OLS on the object's point cloud. On the right image the product has some inclination.

5.4.3.4 PCA 2D combined with OLS linear regression method

As discussed in the previous methods of calculating the orientation of the detected object, the method that uses PCA on the segmentation mask has the disadvantage of only

being capable of determining the angle of the object around the z-axis, while the method that uses the OLS linear regression can only determine the angles of rotation around the x and y axes.

The next developed method combines these methods (OLS and 2D PCA) in order to try to work around the limitations of both. First, the same steps that explain the OLS linear regression method are applied, determining the roll and pitch angles. Finally, the PCA technique is applied to the segmentation mask pixels to obtain the missing yaw angle. The figure 5.17 presents the results of applying the combined methods. From the discussed methods so far, this is the most reliable so far. However, because of the independent nature of how the rotation around the z-axis is calculated from the angles of the x and y axes, this method has problems when calculating poses with high pitch and roll angles.

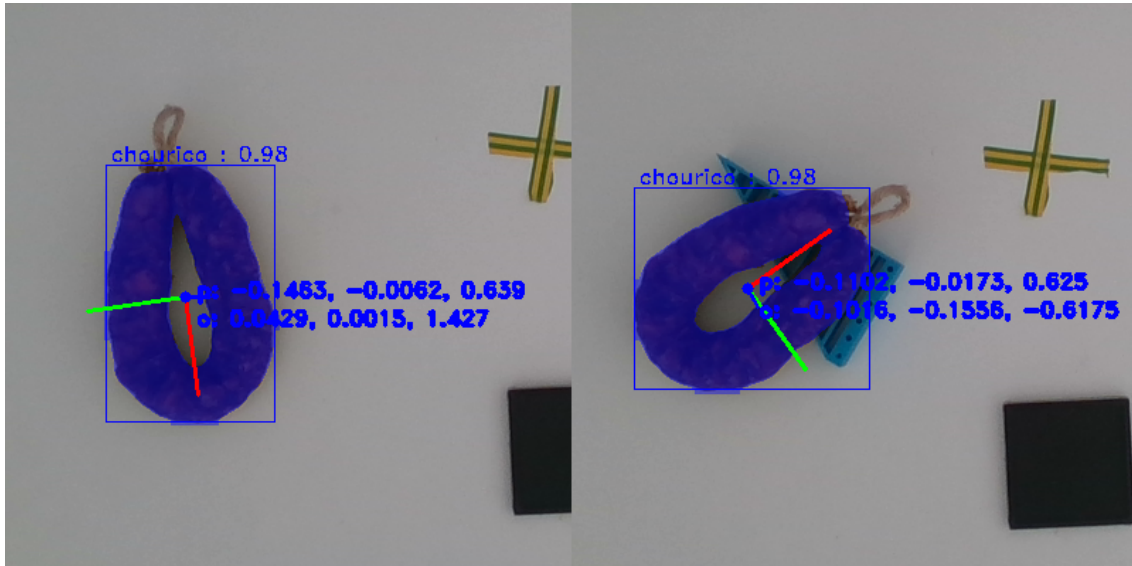


Figure 5.17: Results of orientation calculation method that uses the combined OLS on the object's point cloud with PCA on the segmentation mask pixels. On the right image the product has some inclination.

5.4.3.5 Keypoints method

The last developed method uses a keypoint approach to calculate the object orientation. The basic premise of this approach is to find the coordinates of a series of points in certain positions of the object and use them to calculate the orientation of the object. This approach begins by first doing a PCA on the pixels of the segmentation mask of the object to obtain the rotation of the object in the 2D plane. With this angle, the object is rotated on the image, and its width and height are extracted by calculating the bounding box of the object, using the function `boundingRect` from the OpenCV library.

Four keypoints are necessary to calculate the orientation of the object. From these four keypoints, two vectors are calculated, one vector to represent the x-axis and another to represent the y-axis of the object. The z-axis is obtained by the cross product between the x and y axes vectors. The figure 5.18 illustrates the position of the keypoints on the object that are used to calculate these vectors.

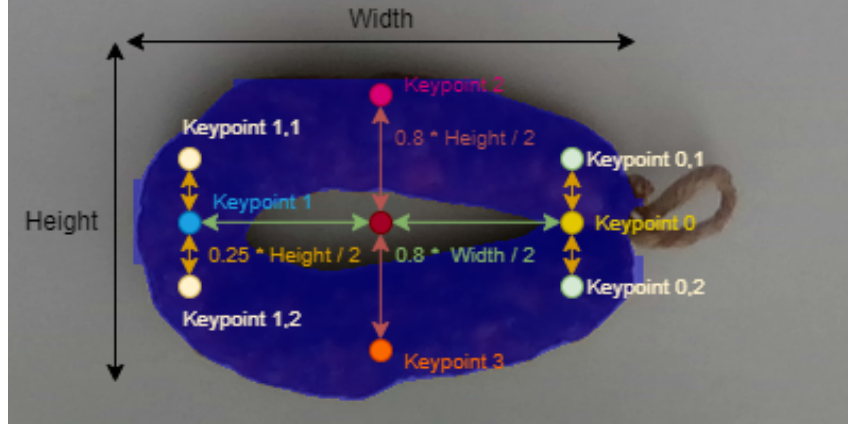


Figure 5.18: Location of the keypoints on a product.

Keypoints 0 and 1 are used to compute the x-axis vector of the object, while keypoints 2 and 3 are used to compute the y-axis vector. Due to the shape of the products, keypoints 0 or 1 may occasionally be located in a region of the depth image where the number of invalid depth values is too high to allow for precise calculation. For that reason, secondary keypoints are employed to assist with the calculation of the depth value of the x-axis keypoints.

The first step to calculate the positions of these keypoints is to determine the distance between them and the centroid of the object. For the x-axis keypoints, this is calculated as $distance_x = 0.8 \frac{width}{2}$. For the y-axis keypoints, this is calculated as $distance_y = 0.8 \frac{height}{2}$. Next, using the angle calculated by the PCA step and the centroid, the position of the x and y keypoints is obtained by:

$$keypoints_x = \begin{cases} x = centroid_x \pm distance_x \cos(\theta) \\ y = centroid_y \pm distance_y \sin(\theta) \end{cases}$$

$$keypoints_y = \begin{cases} x = centroid_x \pm distance_x \cos(\theta + 90^\circ) \\ y = centroid_y \pm distance_y \sin(\theta + 90^\circ) \end{cases}$$

The distance between the secondary keypoints is calculated in a similar way for the distance between the centroid and the y-axis keypoints, $distance_{x,small} = 0.25 \frac{height}{2}$. The position of the secondary keypoints is calculated in the same way as the y-axis keypoints,

with the difference of using the main keypoints of the x-axis as a starting point instead of the centroid:

$$\text{keypoints}_{x,\text{secondary}} = \begin{cases} x = \text{keypoint}_x \pm \text{distance}_{x,\text{small}} \cos(\theta + 90^\circ) \\ y = \text{keypoint}_y \pm \text{distance}_{x,\text{small}} \sin(\theta + 90^\circ) \end{cases}$$

The next step of this approach is to calculate the average depth value of the region around the coordinates of the calculated keypoints. The chosen size for the neighborhood was a radius of 8 pixels. This was empirically determined to be the best size. For the case of the x-axis keypoints, the average between the depth values of the associated secondary keypoints is used as the depth value for the main keypoint:

$$\overline{\text{keypoint}_i} = \frac{\overline{\text{keypoint}_{i,0}} + \overline{\text{keypoint}_{i,1}}}{2}$$

To obtain the neighborhood pixels, first a binary image with a circle with a center with the coordinates of the keypoint and with radius 8 is created using the function `circle` from OpenCV. Next, to obtain just the pixels that are inside the segmentation mask, a bitwise AND is done between the circle pixels and the mask using `bitwise_and` from OpenCV. The reason to use an average of the depth values of the surrounding pixels instead of just the depth value of the exact pixel of the keypoint is to give this approach resilience to noise and other disturbance factors.

With the depth values calculated for each keypoint, the next step is to "deproject" the keypoints into 3D coordinates, using the function `rs2_deproject_pixel_to_point` from the `pyrealsense2` library. With these 3D keypoints, the vectors that represent the x and y axes are calculated, using the keypoints 0 and 1 to calculate the x-axis vector and the keypoints 2 and 3 for the y-axis vector. The vector that represents the z-axis is calculated from the cross product between the x-axis vector and the y-axis vector.

$$\begin{cases} \vec{v}_x = \text{keypoint}_1 - \text{keypoint}_0 \\ \vec{v}_y = \text{keypoint}_3 - \text{keypoint}_2 \\ \vec{v}_z = \vec{v}_x \times \vec{v}_y \end{cases}$$

These vectors are normalized and vertically stacked to create a rotation matrix that represents the orientation of the object. To obtain the orientation from this rotation matrix, the same process as the previously described methods is used. A rotation object is created using the function `from_matrix` from SciPy, then the Euler angles are extracted with the function `as_euler`.

$$R = \begin{bmatrix} \vec{v}_x \\ \vec{v}_y \\ \vec{v}_z \end{bmatrix}$$

This method is also capable of calculating an approximation of the required grasp width to assist the pick and place task performed by the robot. This is done by calculating the Euclidean distance between the keypoints 2 and 3. The calculated distance is multiplied by 1000 to change the units from meters to millimeters, as required by the gripper system (section 4.5).

$$width_{grasp}(mm) = \sqrt{(k_{3,x}^2 - k_{2,x}^2) + (k_{3,y}^2 - k_{2,y}^2) + (k_{3,z}^2 - k_{2,z}^2)} \cdot 1000$$

The figure 5.19 shows the results of applying the keypoints methods on a detected product. This method proves to be the most reliable and resilient to noise and other disturbance factors. The orientations calculated for objects parallel to the workspace table or with inclination were all valid. One limitation of this approach is that it requires the object's top face to have a great portion visible. At extreme angles, the approach will not be capable of determining the object's orientation.

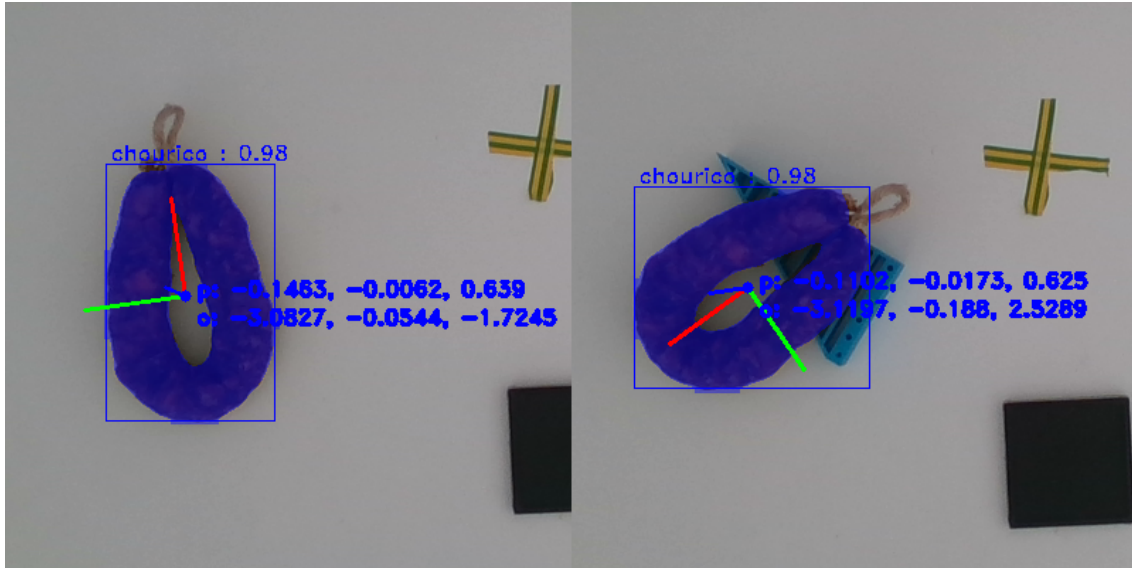


Figure 5.19: Results of orientation calculation method previously discussed (keypoints method). On the right image the product has some inclination.

5.5 Chosen approach and limitations

After evaluating the strengths and limitations of the various orientation calculation methods described in the previous section, the keypoints approach was selected as the most suitable method. It demonstrated superior performance compared with the other developed methods. Some were unable to calculate all the orientation angles (such as PCA on the segmentation mask pixel and OLS regression methods), while others encountered difficulties when the object exhibited inclination relative to the workspace table (PCA applied to the point cloud method).

The keypoints approach was preferred to the combination of the PCA + OLS approach for two factors. The first factor is that, in the PCA + OLS method, the yaw angle is calculated independently. This poses a significant limitation, as the Euler angles are interdependent, and as such, there is no guarantee of the consistency of the calculated yaw angle. The second factor is that the keypoints approach provides the additional ability of calculating the product's grasp width. Giving an advantage to the robotic system, as it can calculate the required distance to close to properly pick up the various products in a dynamic manner.

As the keypoints based approach for the orientation part of the pose calculation does not require the use of a point, the best method for the position calculation is using the depth values directly provided by the camera. Due to the faster and less computationally expensive nature of this method when compared with the point cloud based approach.

The complete pipeline of the algorithm is as follows. First, the depth and color images pass through a preprocessing block, where they are aligned, and depth image is filtered using decimation and temporal filters. Next the color image is fed to a YOLOv12 segmentation model, which generates the segmentation masks, classes, bounding boxes, and confidence score for each detected product. The results have a threshold of 0.75, any result below is discarded. Following this, the segmentation mask and depth image are used to calculate the object's position, as well as its centroid. Finally, using the centroid, segmentation mask, and depth image the orientation of the object is obtained using the keypoints approach described earlier. The figure 5.20 presents a high level flowchart of the algorithm.

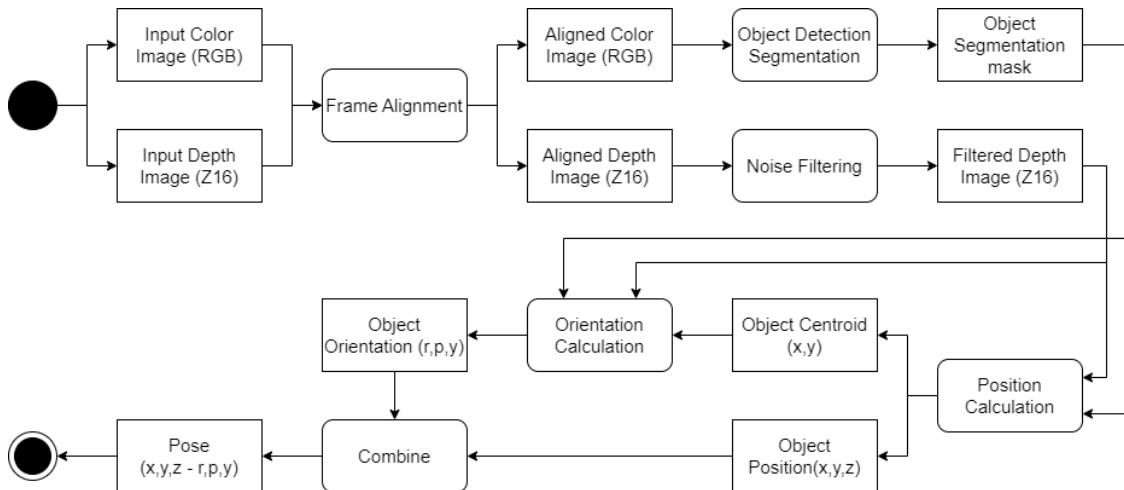


Figure 5.20: High level flowchart of the complete vision algorithm

The chosen approaches for the pose estimation algorithm have some limitations and shortcomings. Firstly, as already explained in the subsection 5.4.3, the keypoints approach

to calculate the object's orientation requires the top surface of the object to be visible. This poses a problem when the object is at extreme angles, close to 90 degrees on the roll and pitch. For the scenario, the robotic system is inserted. Finding product at these extreme angles is very unlikely. A solution for this problem is to move the robot end effector and change the angle at which the product is detected.

Another problem is that the pose estimation algorithm cannot distinguish on which side of the object is located the string between the two "legs" of the sausages. This is because the PCA performed in the beginning of the method can only give angles between -90 and 90 degrees. For pick and place tasks, such as this work, this does not create problems. However, for other tasks, there might be a need to know on which side the string is located. One solution for this problem is to use a detection algorithm after the pose estimation to detect on which side the string is.

In the case of the *farinheira* product, as it was more malleable when compared with the *chouriço* and *morcela*, the "legs" could be more open in the way that the secondary keypoints for the x-axis would also be located in invalid positions. This would lead to a failure of the pose estimation algorithm. A solution for this shortcoming is using an iterative process to calculate the position of the secondary keypoints.

In the case of the *farinheira* product, as it was more malleable when compared with the *chouriço* and *morcela*, the "legs" could be more open in the way that the secondary keypoints for the x-axis would also be located in invalid positions. This would lead to a failure of the pose estimation algorithm. A solution for this shortcoming is using an iterative process to calculate the position of the secondary keypoints.

ROBOTIC SYSTEM DEVELOPMENT

This chapter describes the developed robotic system and the various Robot Operating System (ROS) nodes that compose the system and their function. The first section highlights the robotic manipulator used, as well as the virtual machine used to implement the robotic system controller. The second section presents the configurations of the robot. The third section describes the complete robotic system from a high-level point of view. The fourth section discusses the development of the various "peripheral" nodes of the overall ROS implementation. The fifth section describes the development of the two main controller nodes for the robotic system. The final section presents the limitations of the developed robot system.

6.1 Robotic system hardware and software

As already briefly mentioned in the proposed approach chapter (chapter 3), the robotic manipulator employed in this work is an ABB IRB 120 with 6 Degrees of Freedom (DoF). This ABB robot has a capacity to handle up to 30 kg, with a working range of 580 millimeters around 165 degrees, and a position repeatability of 0.01 millimeters (ABB Robotics, 2019). The robot was mounted with an end effector composed of the previously presented fin ray effect gripper (chapter 4), as well as the depth camera for the vision system, an Intel RealSense D415 (chapter 5).

The robot controller was created using the ROS framework in conjunction with MoveIt, a motion planning, manipulation, and kinematics calculation framework. The IRB 120 manipulator is compatible with just the MoveIt 1 version of the framework (Edwards, 2018). For this reason a virtual machine running the Ubuntu 16.04 LTS operating system, installed with the ROS Kinetic Kame distribution, is used, as it is the compatible version for the ABB manipulator. The virtual machine has the advantages of faster deployment and easier backup creation, as well as simpler reconfiguration for the system.

The virtual machine is configured with Network Address Translation (NAT) networking. The communication with the robot and the gripper controller is established via the host's

Wi-Fi interface. The host is connected to a dedicated private network to control the robot (with an IP address of 192.168.1.1) and the gripper (with an IP address of 192.168.1.2).

The camera is connected to the computer via a Universal Serial Bus (USB) cable. The object pose estimation algorithm, presented in the chapter 5, runs directly on the host computer, as the algorithm requires higher processing power due to the use of the You Only Look Once (YOLO) segmentation model. The communication between the vision algorithm and the developed ROS controller is established via Transmission Control Protocol (TCP) messages and is explained further in the section 6.4. The following figure (figure 6.1) illustrates communication between the various components of the robotic system.

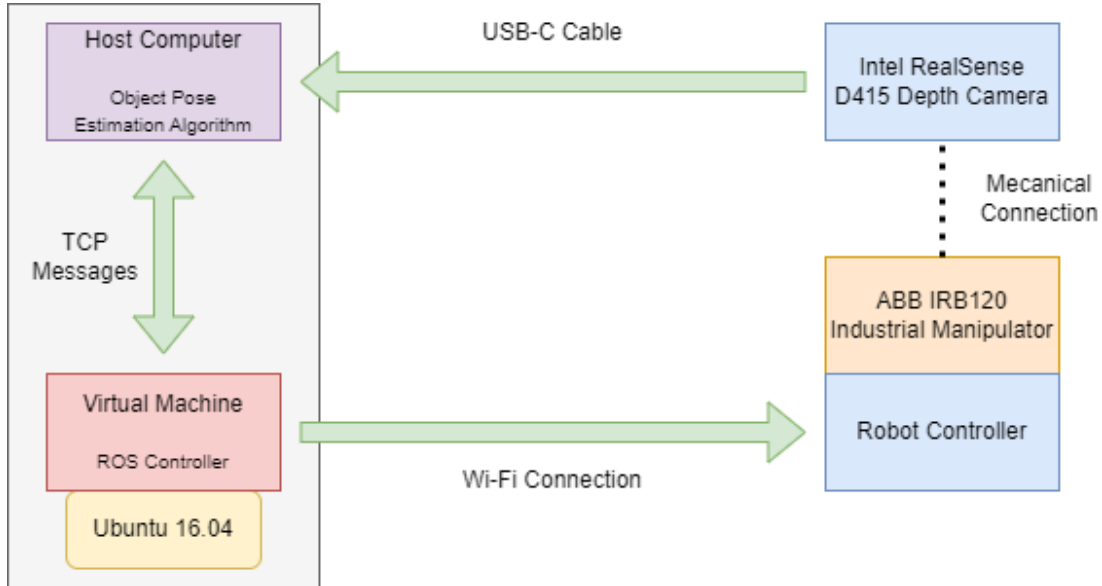


Figure 6.1: Robotic system connections

The robot is mounted on supports 4 centimeters high on a table with dimensions of 90 centimeters in length by 70 centimeters in width. However, because the robot is mounted on one corner, the workspace is restricted to a workspace of 90 by 50 centimeters. In this area, various agri-food products are placed for the robot to manipulate. The robot's controller unit is placed underneath the workspace table and is connected to a private network using an Ethernet cable.

6.2 Robot's URDF, SRDF and manipulator configurations

To accommodate the modifications to the robot's end effector, specifically the addition of the fin ray effect gripper and the D415 depth camera, updates were made to the Unified Robot Description Format (URDF) file that describes the robot model. Two links and joint pairs were added: one for the gripper and another for the depth camera. The

joint origins were defined to represent the physical length of the gripper assembly and, in the case of the camera, aligned with the position of its left imager. The position of the fin ray gripper joint was fixed at $x = 0, y = 0, z = 0.2$, as the overall gripper assembly length is around 20 centimeters. The position of the depth camera was fixed at $x = -0.116, y = 0.02, z = 0.068731$. The y coordinate was dictated by the offset of the left imager to the camera's center. The x and z coordinates are the physical offset of the camera to the robot's final link.

The Semantic Robot Description Format (SRDF) file was also updated to accommodate the added links and joints. The added group has the name "manipulator4_finray" and has a base starting from the base link and tip at the newly added fin ray gripper link. The group is added to the `kinematics.yaml` file as well, with the default parameters:

- KDL solver
- 5mm search resolution
- 5ms timeout
- Maximum of 3 attempts

To the file `ompl_planning.yaml`, an instance is added for the newly created group to incorporate the various planners provided by ROS MoveIt framework.

6.3 Overall ROS controller architecture

The ROS implementation of the robotic system controller follows an architecture that is based on a single main controller node and multiple peripheral nodes offering services to the main controller. This architecture creates a modular robotic system and allows for more scalability and easier debugging. The peripheral node has the function of abstracting more complex tasks for the main controller, while the main controller only needs to implement the logic of the robotic system's procedures and operations. The figure 6.2 illustrates the architecture of the implemented robotic system, its various nodes, the services the peripheral nodes implement, and the two possible controllers implemented for the main node.

The communication between the peripheral nodes and the main controller node is established via services. The main controller calls the peripheral node's services to accomplish various tasks, such as opening or closing the gripper, obtaining the poses of the detected products, moving the robot to a different pose, etc. This modular architecture allows for the testing and debugging of each node independently, allowing the testing of each individual function without requiring a fully operational system. A modular approach also permits replacing individual parts of the system without changing the main logic of the robotic system.

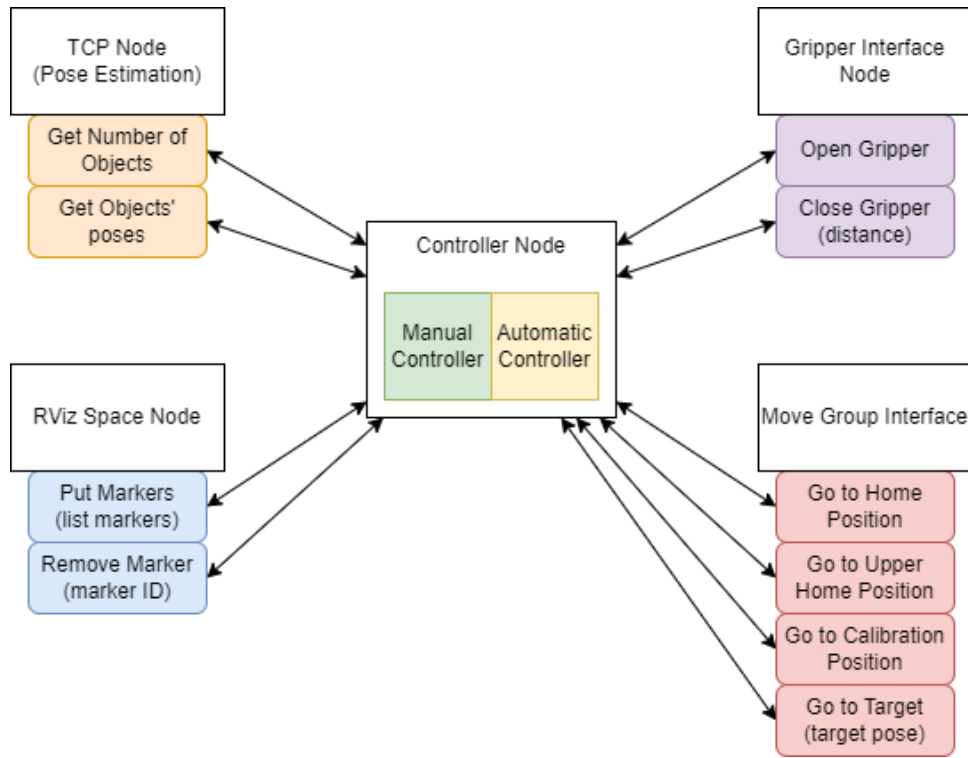


Figure 6.2: Robotic system architecture

6.4 Robotic system ROS peripheral nodes

Four ROS nodes were developed to abstract complex operations such as path planning and communication with the object pose estimation algorithm. Developed peripheral nodes include a node dedicated to handling the communication between the computer vision algorithm, a node dedicated to handling the path planning of the robot, a node to control the gripper, and a node to put markers on the RViz software to symbolize the various detected objects. These nodes are implemented using Python in the package `irb120_control`, and the services used are defined in the package `obj_dect_msgs`.

6.4.1 Vision system communication node

To handle the communications between the ROS main controller node running on the virtual machine and the object pose estimation algorithm running directly on the computer, a node was specifically developed to abstract this task to the main controller node. The node, named `tcp_node`, establishes the communication via TCP messages. The developed node sends a TCP message to the object pose estimation algorithm and receives as a response a list of the detected objects, their confidence value, their pose in relation to the camera frame, and the grasp width in millimeters. The node is also responsible for transforming the poses of the detected objects from the camera frame to the robot base frame.

The node implements two services, one that obtains the number of the detected objects (`get_number_objects`) and another that obtains lists of the confidence values, the transformed poses, and the widths of the objects (`get_object_position`).

The node initialization process begins by storing the IP address of the host computer, then creating a ROS TF2 buffer to store the transformations for the various robot frames, and later creating the two services. When the service `get_number_objects` is requested, the node creates a socket object and connects to the host computer. Then, the node sends a request to the object pose estimation algorithm for the detected objects. Finally, it sends the length of the obtained as a response to the main node. When the service `get_object_position` is called, the exact same procedure is followed, except instead of sending the length of the list, it first transforms the poses contained in the list to the coordinate system of the base link, then sends the list to the main node.

6.4.2 Gripper controller interface node

To handle the communications between the ROS main controller and the gripper controller, a node was developed to abstract these procedures, allowing more flexibility and modularity to the overall robotic system. This node was named `gripper_interface_node` and is responsible for sending commands to the gripper microcontroller to open or close the gripper's fingers.

The node implements two services, one that opens the gripper to its maximum extension (`open_gripper`) and another that closes the gripper's fingers to a given distance (`close_gripper`).

The node on the initialization process creates the two services. When the service `open_gripper` is called, the node sends an HTTP POST request to the gripper microcontroller to the path `/open`, using the requests Python library. If it receives a successful response code (200), the node sends a message to the main controller node with a boolean value of `true`. Otherwise, it sends a message with a boolean value of `false`. In case of failing to establish a connection, the node returns a value of `None`, signifying the failure of the service call. When the service (`close_gripper`) is called, the node sends an HTTP POST request with the path `/close` with the parameter `dimension` set to the desired distance specified in the service call message. The same procedure as the service `open_gripper` is followed for the rest of the service handling.

6.4.3 Path planning and move group interface node

To abstract the path planning and kinematics calculations for the main controller, the node `move_group_interface` was developed. This node implements four services. The services `go_to_home`, `go_to_home_up`, and `go_to_calib` all go to hardcoded positions. The

service `go_to_home` sends the robot to the "home" position, all joints at 0 degrees except for the fifth, which is at 90 degrees. This makes the camera point towards the workspace table at 90 degrees. The service `go_to_home_up` is similar to the `go_to_home` service. However, the end effector of the robot is at a higher position. The service `go_to_calib` sends the robot to the position where all joints are at 0 degrees. This is to point the camera towards a wall to help its calibration. The fourth service is `go_to_target`. This service sends the robot to the target pose designed on the service request sent by the main controller.

When the node is run, it begins by initializing the MoveIt commander and obtaining the move group commander and other robot interfaces, as well as information regarding the current status of the robot. The move group commander object is initialized with the end effector kinematics configuration "manipulator4_finray". The planner is modified to have a planning timeout of 5 seconds and 10 maximum attempts. A constraint on the sixth joint is also added to prevent it from rotating more than 180 degrees clockwise and counterclockwise.

The services `go_to_home`, `go_to_home_up`, and `go_to_calib` are implemented in a similar manner. When the main controller calls one of these services, the `move_group_interface` node begins by obtaining the current joint angle values from the move group commander, then modifies these values to create a "joint goals" object. The node sends these joint goals to the move group commander to make the robot move to the designated pose. Finally, the node sends a stop command to prevent residual movement on the robot. The service `go_to_target` is implemented in a different manner. When the main controller sends a service request with a designated target pose, the node begins by first creating a "pose goal" object from the target pose values. Next, send this pose goal object to the move group commander and await the completion of the plan. Finally sends the stop command to the robot and clears any pose targets to prevent residual movements.

The return value of all of these services is determined by an auxiliary function that measures the error between the real position of the robot and the desired position and compares it to a tolerance value. If the robot is outside of the tolerance, the function returns a boolean value of false. Otherwise, it sends a boolean value of true. In the case of the services `go_to_home`, `go_to_home_up`, and `go_to_calib`, the comparison is made on the joint value. In the case of the service `go_to_target`, the comparison is done on the pose values (position x-y-z, orientation x-y-z-w). The following figure (figure 6.3) presents the various robot's hard coded positions, from left to right the positions are: home position, upper home position, and calibration position.

6.4.4 RViz object visualizer node

To assist in the debugging of the vision system as well as provide feedback to the user on the detected objects' positions, a node responsible for handling the addition and removal



Figure 6.3: Three robot's hard coded positions (from left to right: home position, upper home position, calibration position)

of markers was created with the name of `rviz_space_node`.

This node creates two services: the service `put_cylinder` adds markers in the format of cylinders to the RViz visualizer, and the service `rem_cylinder` removes markers with a given ID number.

When the node is run, it first creates the two services and later creates a topic publisher to publish messages with the marker data to the RViz visualizer. When the main controller sends a `put_cylinder` service request with a list of poses for representing each detected object's pose, the node, for each pose on the list, first gets a valid ID number for the marker and stores the number. Then creates a marker message of the type "cylinder", with the action "ADD", with the ID number, a random color, and with the pose given on the service request list. After creating a message for the marker, the node adds it to the list. When all marker messages are created, the node publishes the message, and RViz places these markers on the visualization screen. As a response, the node returns to the main controller a list with the ID numbers of the markers. When the main controller calls the service `rem_cylinder`, the node creates a message with the action "DELETE" for the given maker ID and removes the ID number from the stored IDs list. As a response returns a boolean value of true.

6.5 Controller nodes

Two main controller nodes were developed to manage the robotic system's operations: a manual controller and an automatic controller. The manual controller allows the user to have full control of the robotic system, including the robot's movement and position and calling services provided by the peripheral nodes. On the other hand, the automatic

controller is designed to perform the complete pick and place routine autonomously. It coordinates the robotic system sequentially by calling the required services from the peripheral nodes and handling the logic for the task without user intervention.

6.5.1 Manual controller

The first developed main ROS controller for the robotic system was the manual controller. This controller permits the user to have full control of the robotic system operations and call services from the other nodes on demand. The controller features a Graphical User Interface (GUI) that permits the user to operate the system. The GUI features a zone, on the left of the application window, for the user to input the desired pose for the robot (position as x-y-z in meters, orientation as roll-pitch-yaw in radians). By default, the orientation fields already have values: roll and yaw as 0 radians and pitch as 3.14 radians. An entry for the user to input the desired distance for the gripper fingers and another entry to input the marker ID. On the right half of the GUI window are displayed various buttons, some that call the various services implemented by the peripheral nodes, and other buttons that display the stored object poses or allow for some automatic actions, specifically functions for the later tests of the robotic system.

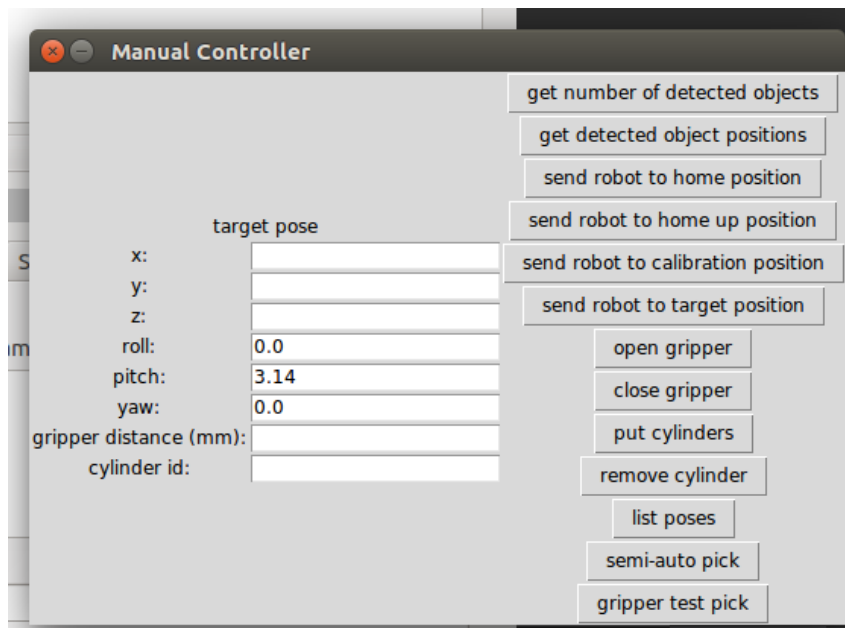


Figure 6.4: Manual controller GUI

When the node is run, it first waits for the required services from the peripheral nodes to be available. After all services become available, the node proceeds to create the GUI using the Tkinter framework from Python, creates a list object to store the detected objects' poses given by the `tcp_node`, and creates a dictionary object to store the marker IDs with the corresponding pose of the detected object, given by the service `put_cylinder` from

the `rviz_space_node`. When the user presses one of the buttons placed on the right of the GUI's window, the node proceeds to send service requests to the peripheral nodes:

- "get number of detected objects" - calls the service `get_number_objects` from `tcp_node` (6.4.1) and displays on the console the number of the detected objects by the vision system.
- "get detected object positions" - calls the service `get_object_position` from `tcp_node` (6.4.1), stores the obtained poses by the vision system of the detected objects, and displays them on the console for the user.
- "send robot to home position" - calls the service `go_to_home` from `move_group_interface` (6.4.3) to move the robot to the home position and displays if the operation was successful on the console.
- "send robot to home up position" - calls the service `go_to_home_up` from `move_group_interface` (6.4.3) to move the robot to the upper home position and displays if the operation was successful on the console.
- "send robot to calibration position" - calls the service `go_to_calib` from `move_group_interface` (6.4.3) to move the robot to the camera's calibration position and displays if the operation was successful on the console.
- "send robot to target position" - calls the service `go_to_target` from `move_group_interface` (6.4.3) reads the values input by the user on the GUI, sends them on the service request to move the robot to the designated target pose, and displays if the operation was successful on the console.
- "open gripper" - calls the service `open_gripper` from `gripper_interface_node` (6.4.2) to open the gripper to its full extension, and displays if the operation was successful on the console.
- "close gripper" - calls the service `close_gripper` from `gripper_interface_node` (6.4.2) to close the gripper's finger to the designated distance by the user on the GUI and displays if the operation was successful on the console.
- "put cylinders" - calls the service `put_cylinder` from `rviz_space_node` (6.4.4), sends the poses stored on the list of poses on the service request, obtained by calling `get_object_position`, and stores the IDs given as response with their respective poses.
- "remove cylinder" - calls the service `rem_cylinder` from `rviz_space_node` (6.4.4), reads the ID designated by the user on the GUI, sends a service request with that number, and deletes the entry of the pose dictionary with that number.

The buttons "list poses", "semi-auto pick", and "gripper test pick" perform complex actions to aid the testing of the robotic system. The button "list poses" lists all the object poses stored in the markers dictionary. It iterates through the dictionary and displays all the marker IDs, positions and orientations (in Euler angles), and the objects' width.

The button "gripper test pick" was created to assist in the fin ray effect gripper testing. It sends the robot to a designated hard-coded position and operates the gripper to pick an object. The procedure performed by this button is the following: The first action is to open the gripper by calling the service `open_gripper`. Secondly, it sends the robot to the hard-coded pose, $x = 0.5, y = 0, z = -0.04, r = 0, p = \pi, y = 0$. After arriving at the hard-coded pose, the main controller calls the service `close_gripper` with a distance of 40 millimeters. To finalize, the main controller calls the service `go_to_home` to send the robot back to the home position.

The button "semi-auto pick" was created to assist in the overall testing of the robotic system, testing both the object pose estimation algorithm and the gripper. The first step of the function called by this button is to read the marker ID and retrieve its corresponding object's pose stored in the dictionary. Then it calculates the opposing orientation. This orientation is going to be used for the robot pose. This is done by obtaining the rotation matrix from the quaternion that represents the object's orientation, then inverting the z-axis (and the x-axis to maintain the right-hand rule). The y-axis is calculated from the cross product of the z-axis and x-axis.

$$\begin{cases} \vec{z}_{\text{gripper}} = -\vec{z}_{\text{object}} \\ \vec{x}_{\text{gripper}} = -\vec{x}_{\text{object}} \\ \vec{y}_{\text{gripper}} = \vec{z}_{\text{gripper}} \times \vec{x}_{\text{gripper}} \end{cases}$$

This rotation matrix is then transformed into a quaternion to be used as the orientation of the robot pose. The next step is to find a position above the object to be grasped. The above position is defined by adding 5 centimeters on the z-axis direction to the retrieved object position.

$$\begin{cases} x_{\text{abv}} = x_{\text{object}} + 0.05 \cdot \vec{z}_x \\ y_{\text{abv}} = y_{\text{object}} + 0.05 \cdot \vec{z}_y \\ z_{\text{abv}} = z_{\text{object}} + 0.05 \cdot \vec{z}_z \end{cases}$$

Next, the robot moves to the calculated position using the `go_to_target` service. After reaching the target pose, the controller calls the `open_gripper` service and waits for the gripper to open to its full extension. In the following step, the controller computes the "exact" position of the object. This computation is performed in the same manner as the previous position calculation, except with a subtraction of 2 centimeters. This offset allows

the gripper to grasp the object along a more central region, resulting in a more secure grip.

$$\begin{cases} x_{\text{exact}} = x_{\text{object}} - 0.02 \cdot \vec{z}_x \\ y_{\text{exact}} = y_{\text{object}} - 0.02 \cdot \vec{z}_y \\ z_{\text{exact}} = z_{\text{object}} - 0.02 \cdot \vec{z}_z \end{cases}$$

As with the previous position, the robot calls the `go_to_target` service to move to this location. After reaching the designated target position, the controller calls the `close_gripper`. The distance provided in the service request corresponds to the detected object's grasp width minus 30 millimeters, allowing the gripper to deform around the object:

$$\text{distance}_{\text{request}} = \text{width}_{\text{object}} - 30$$

Next, the robot moves to the home position. The controller then calls the service `go_to_target` with the pose $x = 0.25, y = 0.45, z = -0.02, \text{roll} = 0, \text{pitch} = \pi, \text{yaw} = \pi$ (with the orientation in a quaternion format). Once the robot reaches this pose, the main controller calls the service `open_gripper` to release the grasped product. To complete the pick-and-place routine, the controller finally calls the `go_to_home_up` service, returning the robot to its upper home position.

6.5.2 Automatic controller

The second developed main controller was an automatic controller. This controller was implemented as a ROS node that automatically detects the various displayed agri-food products on the workspace table and performs the pick and place task for each one. This node automatically calls the required services for the detection of products, robot movement, and gripper control. This controller requires no input from the user.

Same as the manual controller, when the automatic controller node is run, the first step is to await for the required services from the peripheral nodes to be available. Next, create a list to store the detected objects from the object pose estimation algorithm. For efficiency the computation of the drop site pose is done before entering the main loop of the node. The chosen drop site pose is $x = 0.25, y = 0.45, z = 0.02, \text{roll} = 0, \text{pitch} = \pi, \text{yaw} = \frac{\pi}{2}$. The last step before entering the main loop of the controller is to send the robot to its upper home position. This is done to guarantee that the robot is located at the optimal position for the object pose estimation algorithm.

When the node enters its main logic loop, the first operation it does is call the service `get_object_position` to obtain the list of detected objects' poses, along with their probabilities and widths. This temporary list is then sorted by the probability of the

corresponding detected objects. In case of an empty list, the node waits for 5 seconds and starts from the beginning.

The next operation of the node is to add the markers to the RViz visualizer by calling the service `put_cylinder` and to request the various poses of the temporary list of detected objects. After obtaining the return ID numbers for each marker, the list initially created at the beginning of the execution is populated by the combination of the objects' pose, probabilities, and widths, with their corresponding markers' ID. To give feedback to the user, this list is printed on the console.

For each detected object the controller runs a similar pick and place routine as the manual controller (6.5.1). First the controller sends the robot to an above position of the product, with `go_to_target` from `move_group_interface`. Next opens the gripper to its full extension using the service `open_gripper`. Then sends the robot to the "exact" position of the product and closes the gripper using the service `close_gripper`, the distance on the request being the width of the detected object subtracted by 30 millimeters. The next step differs from the manual controller. Instead of sending the object to the home position, the automatic controller sends the robot to an intermediary position, calculated by adding 10 centimeters to the z coordinate of the original agri-food product.

$$\begin{cases} x_{\text{exact}} = x_{\text{object}} \\ y_{\text{exact}} = y_{\text{object}} \\ z_{\text{exact}} = z_{\text{object}} + 0.1 \end{cases}$$

After the robot reaches the intermediary position, the controller uses the service `go_to_target` to send the robot to the drop site pose, calculated at the beginning of the execution. When the robot reaches the drop site pose, the controller calls `open_gripper` to drop the grasped product and then `go_to_home`. After all the steps are completed, the controller removes the object from the list of poses and calls the service `rem_cylinder` from `rviz_space_node` to remove the marker from the RViz visualizer. The controller continues this procedure until all the detected agri-food products are picked and placed at the drop site. When the list of objects becomes empty, the controller sends the robot back to its upper home position by calling the service `go_to_home_up` and starts from the beginning of the main loop. The following figure (figure 6.5) presents a flowchart of the logic of automatic controller.

6.6 Robotic system limitations

Despite the successful implementation of the robotic system and its integration with the developed fin ray effect gripper (chapter 4) and the developed object pose estimation

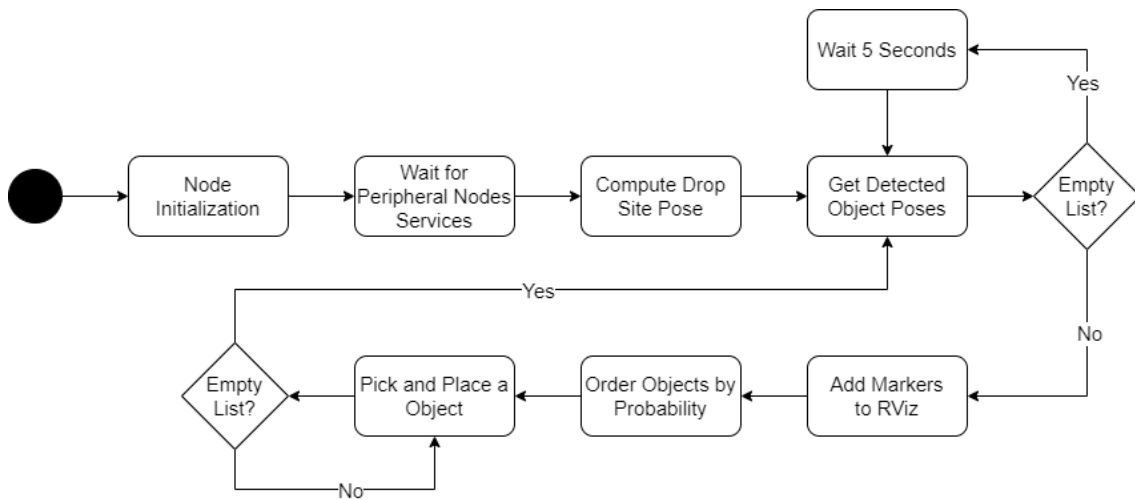


Figure 6.5: Automatic controller flowchart

algorithm (chapter 5), several limitations and constraints occurred during its development and testing.

One of the main limitations found on the robotic system is related to the motion planning behavior. On several instances, the planner would compute unnecessary joint movements instead of a smoother approach to the target pose.

Another issue, specifically related to the robot's sixth joint, concerned its extended rotational range. Because this joint can rotate beyond 360 degrees (with a lower limit of -400 degrees and an upper limit of 400 degrees by default ABB Robotics, 2019), the planner would occasionally generate trajectories in which the joint performed unnecessary rotations exceeding one full turn, rather than selecting a shorter equivalent angle within the -180 to 180 degree interval. This behavior resulted in inefficient paths and, in some cases, led to failures in grasping the detected product. To address this issue, two solutions were considered: applying a joint constraint within the `move_group_interface` node or restricting the joint's limits to -180 and 180 degrees directly in the URDF file.

Another limitation is that sometimes when the robot is performing the pick and place routine, moving from the "above" position to the "exact" position of the product, the planner would generate a path with unnecessary joint movement, usually resulting in a failure to pick the product. This could be resolved using Cartesian path planning, forcing the robot to move in a linear manner to the "exact" position of the product.

VALIDATION AND TESTS

The chapter presents the various tests done to validate the developed gripper, the object pose estimation algorithm, and the robotic system. The first section presents the test done solely on the developed gripper. The second section will present the various tests done on the object pose estimation algorithm in conjunction with the robotic system, giving more focus on the former.

7.1 Fin ray effect gripper validation test

To validate the developed fin ray effect gripper, a test was conducted using the function "gripper test pick" from the developed manual controller for the robotic system. This test was conducted using all three chosen agri-food products for this work: *chouriço*, *farinheira*, and *morcela*. Both the second and third iterations of the fin ray effect gripper were tested. This test was not conducted on the first iteration of the gripper, as it proved to be too stiff at first impression, and for that reason, this test could damage the products.

7.1.1 Test description

As said previously, this test was conducted using the "gripper test pick" function of the robotic system's manual controller. When calling this function, the robotic system sends the robot to a hard-coded position, $x = 0.5, y = 0, z = -0.04, r = 0, p = \pi, y = 0$, picks the product, and goes back to the home position. For a more in-depth explanation of the function, see the subsection 6.5.1 of the robotic system development chapter. This procedure is conducted 20 times for each product, in a total of 60 tests for each iteration of the gripper. This test will allow to evaluate the performance of the gripper in grasping the various chosen agri-food products and verify if any improvements should be made for the fin ray effect gripper.

7.1.2 Test results

The test results indicate that both developed grippers are capable of handling the selected products. However, for the *farinheira*, the second iteration did not perform particularly well,

especially when compared with its performance on the other two products, achieving only an 80% success rate. In contrast, the third iteration successfully handled all test products with very high success rates ($\geq 95\%$). The lower success rate of the second iteration when handling the *farinheira* was attributed to the product's width being nearly equal to the gripper's maximum extension. This limitation motivated the modifications implemented in the gripper system for the third iteration, as described in Section 4.3.

Among the three test products, the *farinheira* was found to be the most difficult to manipulate, as it is significantly more malleable than the *chouriço* and *morcela*. Consequently, achieving a secure grasp without damaging the product becomes more challenging. The table 7.1 presents the results of the conducted test. The figure 7.1 shows an example of a successful pick.

Table 7.1: Fin ray effect gripper test results

Product	Second iteration success rate	Third iteration success rate
<i>Chouriço</i>	95% (19/20)	95% (19/20)
<i>Farinheira</i>	80% (16/20)	95% (19/20)
<i>Morcela</i>	95% (19/20)	100% (20/20)



Figure 7.1: Successful fin ray effect gripper agri-food grasp

7.2 Object pose estimation algorithm validation tests

Following the tests performed on the developed gripper, the focus shifted to the testing and validation of the object pose estimation algorithm. During its development, several tests were already done to verify and test the various methods explored. In these preliminary tests, a product was placed on the workspace table, detected, and had its pose

estimated by the vision algorithm. These experiments helped determine which approaches were most suitable for each stage of the object pose estimation pipeline, as discussed in Section 5.5. To better analyze and understand the limitations of the chosen approach and make improvements on the pose estimation algorithm as well as the developed robotic system, more tests were conducted.

7.2.1 First object pose estimation algorithm test

To fully validate the chosen approach for object pose estimation as well as the developed robotic system, a test was conducted where the three products were placed in randomly chosen positions. In a total of 15 poses, derived from 5 randomly selected positions combined with 3 rotations of 0, 45, and 90 degrees on the z-axis. The poses for the products for this test are described in the following table (table 7.2). These poses were measured physically and later verified by commanding the robot to each pose using the developed manual controller. The coordinates of the poses are expressed in meters, while the orientation angles are expressed in radians.

Table 7.2: List of the various poses chosen for the first test

Pose Number	x	y	z	roll	pitch	yaw
1	0.2025	-0.1425	-0.01	0	0	0
2	0.2555	0.0025	-0.01	0	0	0
3	0.3575	-0.1315	-0.01	0	0	0
4	0.3405	0.1225	-0.01	0	0	0
5	0.4695	0.1255	-0.01	0	0	0
6	0.2025	-0.1425	-0.01	0	0	1.5708
7	0.2555	0.0025	-0.01	0	0	1.5708
8	0.3575	-0.1315	-0.01	0	0	1.5708
9	0.3405	0.1225	-0.01	0	0	1.5708
10	0.4695	0.1255	-0.01	0	0	1.5708
11	0.2025	-0.1425	-0.01	0	0	0.7854
12	0.2555	0.0025	-0.01	0	0	0.7854
13	0.3575	-0.1315	-0.01	0	0	0.7854
14	0.3405	0.1225	-0.01	0	0	0.7854
15	0.4695	0.1255	-0.01	0	0	0.7854

The test procedure is the following. First, the robot is sent to the test pose with the gripper at its full extension. Next, one of the products is placed with its center point where the robot gripper is located. This is done to avoid misplacing the product. After that, the robot is sent back to the upper home position, and its gripper is closed. When the robot reaches its home position, the object detecting service is called and retrieves the pose of the product. Then, the detected object is added as a marker to the RViz visualizer, and the pose is visualized using the list poses command. After that, the pick and place command is run ("semi-auto pick") to make the robot perform the pick and place operation.

The pose calculated by the object pose estimation algorithm, as well as the success of the pick and place operation, are written down. This test was conducted using the manual controller, as it offers full control of the robotic system. The manual controller offers the possibility of sending the robot to specific poses through the Graphical User Interface (GUI), as well as the execution of the pick and place routine described in the previous chapter (section 6.5.1). The speed of the robot was set at 25% due to safety reasons.

7.2.2 Test results

The following table (table 7.3) presents the success or failure of the pick and place operation performed at the previously presented poses, with three chosen agri-food products. "Yes*" represents a successful pick and place operation. However, the grasped pose was visibly unstable. "Yes**" represents a pick and place operation where the product to be detected needed to be adjusted at this pose. This was a result of the product not being clearly visible on the camera's image.

Table 7.3: First test pick and place operation results. Yes* - represents a successful pick but with an unstable grasp. Yes** - represents a successful pick but the product was heavily adjusted

Pose Number	<i>chouriço</i>	<i>morcela</i>	<i>farinheira</i>
1	Yes*	Yes**	Yes*
2	Yes	Yes	Yes
3	Yes	Yes	Yes
4	No	No	Yes*
5	No	No	No
6	Yes	Yes	Yes
7	Yes	Yes	Yes
8	Yes	Yes	Yes
9	No	Yes	No
10	No	No	No
11	Yes	Yes	Yes
12	Yes	Yes	Yes
13	Yes	Yes	Yes
14	Yes	No	No
15	No	No	No

The results of this test demonstrate that the object pose estimation algorithm is able to calculate a valid pose for the majority of the randomly selected configurations. However, it consistently failed to estimate a valid pose for all three products in poses 5, 10, and 15, which correspond to the coordinates $x = 0.4695, y = 0.1255, z = -0.01$. Similarly, for poses 4, 9, and 14, corresponding to the coordinates $x = 0.3405, y = 0.1225, z = -0.01$, the algorithm also exhibited difficulties, succeeding in calculating a valid pose for only one of the three products.

Considering all poses, the success rate of the algorithm was relatively low, at approximately 66.67%. However, if the positions where the algorithm consistently failed are excluded, namely poses 5, 10, and 15, the success rate increases to 83.33%. This test highlighted a potential limitation of the algorithm, as it appears capable of computing valid poses primarily for objects located closer to the center of the camera's field of view.

7.2.3 Second object pose estimation algorithm test

To further investigate this potential limitation, a new test was conducted. In this case, rather than using randomly selected positions, the test positions were arranged on a grid covering the workspace area. The grid started at coordinates $x = 0.2\text{m}$ and $y = -0.15\text{m}$ with the x coordinate incremented by 0.1m up to $x = 0.5\text{m}$, and the y coordinate incremented by 0.15m up to $y = 0.15\text{m}$. The objects were also rotated about the z -axis by 0 , 45 , and 90 degrees, as in the previous test, resulting in a total of 36 poses. These poses are summarized in Table 7.4.

This test follows the same procedure as the previous test. Likewise, the speed of the robotic manipulator is set at 25% due to safety reasons.

7.2.4 Test results

The following table (table 7.5) presents the success or failure of the pick and place operation performed at the previously presented poses, with three chosen products.

The results presented in Table 7.5 confirm the presence of a limitation in the effective operational area of the object pose estimation algorithm, as previously suggested by the outcomes of the previous test. By grouping poses that share the same position coordinates for all three products, the resulting success rates can be calculated, as shown in Table 7.6. Figure 7.2 presents a heatmap illustrating the success rates reported in Table 7.6.

The results shown in the table 7.6 indicate that at different positions of the workspace, the object pose estimation algorithm had more success. It was observed that the algorithm had difficulty with positions with $y = -0.15$, and $x = 0.2$ and $x = 0.5$. To better understand this phenomenon, the average absolute errors for each axis were calculated. The average absolute errors were calculated for the combined poses with the same positions. The graphs presented in the figure 7.3 represent the values for the average absolute errors for each axis.

As seen on the x -axis graph, the absolute error rises as the y coordinate rises, meaning that for poses closer to the left side of the camera's image, the object pose estimation algorithm gives worse estimations. By analyzing the y -axis graph, it's possible to verify that for poses that have a position of $x = 0.5$, the absolute error of the y coordinate is

Table 7.4: List of the various poses chosen for the second test

Pose Number	x	y	z	roll	pitch	yaw
1	0.2	-0.15	-0.02	0	0	0
2	0.2	0	-0.02	0	0	0
3	0.2	0.15	-0.02	0	0	0
4	0.3	-0.15	-0.02	0	0	0
5	0.3	0	-0.02	0	0	0
6	0.3	0.15	-0.02	0	0	0
7	0.4	-0.15	-0.02	0	0	0
8	0.4	0	-0.02	0	0	0
9	0.4	0.15	-0.02	0	0	0
10	0.5	-0.15	-0.02	0	0	0
11	0.5	0	-0.02	0	0	0
12	0.5	0.15	-0.02	0	0	0
13	0.2	-0.15	-0.02	0	0	0.7854
14	0.2	0	-0.02	0	0	0.7854
15	0.2	0.15	-0.02	0	0	0.7854
16	0.3	-0.15	-0.02	0	0	0.7854
17	0.3	0	-0.02	0	0	0.7854
18	0.3	0.15	-0.02	0	0	0.7854
19	0.4	-0.15	-0.02	0	0	0.7854
20	0.4	0	-0.02	0	0	0.7854
21	0.4	0.15	-0.02	0	0	0.7854
22	0.5	-0.15	-0.02	0	0	0.7854
23	0.5	0	-0.02	0	0	0.7854
24	0.5	0.15	-0.02	0	0	0.7854
25	0.2	-0.15	-0.02	0	0	1.5708
26	0.2	0	-0.02	0	0	1.5708
27	0.2	0.15	-0.02	0	0	1.5708
28	0.3	-0.15	-0.02	0	0	1.5708
29	0.3	0	-0.02	0	0	1.5708
30	0.3	0.15	-0.02	0	0	1.5708
31	0.4	-0.15	-0.02	0	0	1.5708
32	0.4	0	-0.02	0	0	1.5708
33	0.4	0.15	-0.02	0	0	1.5708
34	0.5	-0.15	-0.02	0	0	1.5708
35	0.5	0	-0.02	0	0	1.5708
36	0.5	0.15	-0.02	0	0	1.5708

higher. The z-axis graph shows a much more constant z value. This indicates that the developed object pose estimation algorithm is more effective at positions closer to the center of the camera's image. From the heatmap presented in the figure 7.2, it can be observed that there is an area of higher success for the camera's image. This area seems to be located between $-0.15 < y < 0$ and $0.2 < x < 0.4$. However, for $x = 0.2$, sometimes the product is not completely visible.

Table 7.5: Second test pick and place operation results

Pose Number	<i>chouriço</i>	morcela	farinheira
1	Yes	No	No
2	Yes	Yes	No
3	Yes	Yes	No
4	Yes	Yes	Yes
5	Yes	Yes	Yes
6	Yes	No	Yes
7	Yes	Yes	Yes
8	Yes	No	Yes
9	No	No	Yes
10	No	No	No
11	Yes	No	No
12	No	No	Yes
13	Yes	Yes	Yes
14	Yes	Yes	Yes
15	No	No	No
16	Yes	Yes	Yes
17	Yes	Yes	Yes
18	No	No	No
19	Yes	Yes	Yes
20	Yes	No	Yes
21	No	No	No
22	Yes	Yes	Yes
23	No	No	No
24	No	No	No
25	Yes	No	Yes
26	Yes	No	Yes
27	No	No	Yes
28	Yes	Yes	Yes
29	Yes	Yes	Yes
30	No	No	Yes
31	Yes	Yes	Yes
32	Yes	Yes	Yes
33	No	No	No
34	Yes	Yes	Yes
35	Yes	Yes	Yes
36	No	No	No

7.2.5 Third object pose estimation algorithm test

The two previous tests only considered poses without any inclination with the workspace table. To test the performance of the object pose estimation algorithm with objects that have angles against the table, twelve more poses were added to the previous list. The poses were created by using the positions that had better success on the previous test and adding an inclination of 10 degrees with the workspace table to place products. The

Table 7.6: Second test pick and place operation success rates for the combined poses

Combined pose numbers	Success rate
1-13-25	75.00%
2-14-26	66.67%
3-15-27	50.00%
4-16-28	100.00%
5-17-29	83.33%
6-18-30	50.00%
7-19-31	100.00%
8-20-32	58.33%
9-21-33	25.00%
10-22-34	66.67%
11-23-35	33.33%
12-24-36	11.11%

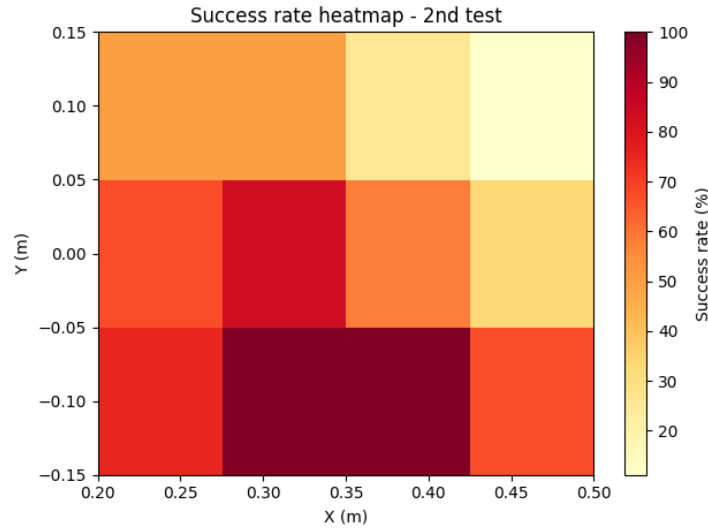


Figure 7.2: Graphs containing the average absolute errors for each axis

resulting twelve poses are described in the following table (table 7.7). This test follows the exact same procedure as the two previous tests.

7.2.6 Test results

The following table (Table 7.8) presents the success or failure of the pick and place operation performed at the previously defined poses using the three selected products. For the majority of these poses, the robotic manipulator successfully completed the pick and place procedure, achieving an overall success rate of 81.48% in this test. The results indicate that, within the region of higher success probability, the object pose estimation algorithm is capable of computing valid poses even when the objects exhibit inclination relative to the workspace table.

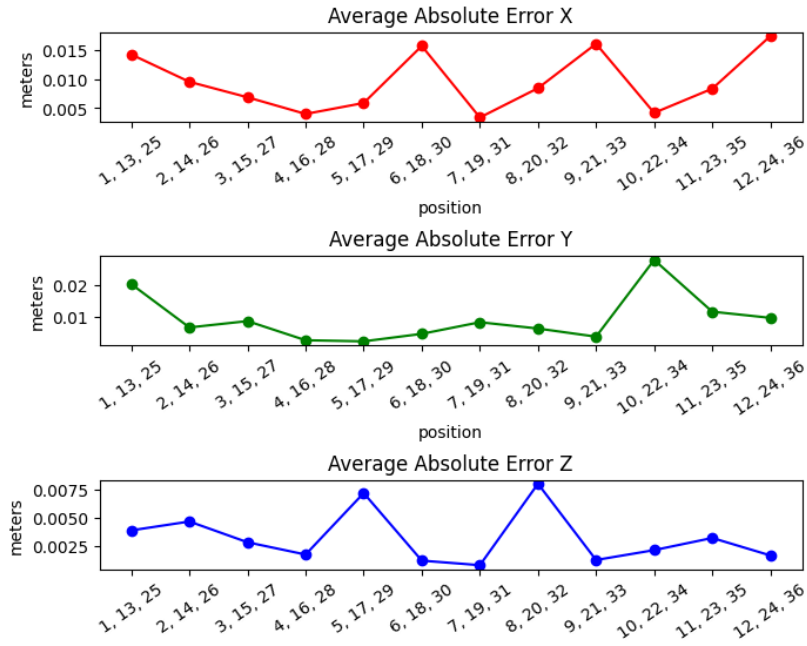


Figure 7.3: Graphs containing the average absolute errors for each axis

Table 7.7: List of the various poses chosen for the third test

Pose Number	x	y	z	roll	pitch	yaw
37	0.3	-0.15	0	0	-0.1745	0
38	0.3	0	0	0	-0.1745	0
39	0.3	0.15	0	0	-0.1745	0
40	0.3	-0.15	0	0	0.1745	0
41	0.3	0	0	0	0.1745	0
42	0.3	0.15	0	0	0.1745	0
43	0.3	-0.15	0	-0.1745	0	1.5708
44	0.3	0	0	-0.1745	0	1.5708
45	0.3	0.15	0	-0.1745	0	1.5708
46	0.3	-0.15	0	0.1745	0	1.5708
47	0.3	0	0	0.1745	0	1.5708
48	0.3	0.15	0	0.1745	0	1.5708

7.2.7 Fourth object pose estimation algorithm test

A final test was conducted to evaluate the performance of the object pose estimation algorithm when dealing with products on top of others, occluding the bottom products. For this test were used certain poses from the second test (7.2.3).

The procedure used for this test is similar to the previous ones, with an exception on how the products are placed on the workspace table. Two products are placed, one on top of the other, instead of a single one. The product on top is aligned to the robotic gripper, as this product is the one in testing. The figure 7.4 shows a representation of how the

Table 7.8: Second test pick and place operation results

Pose Number	<i>chouriço</i>	<i>morcela</i>	<i>farinheira</i>
37	Yes	Yes	Yes
38	Yes	Yes	Yes
39	Yes	No	Yes
40	Yes	Yes	Yes
41	Yes	Yes	Yes
42	No	No	Yes
43	Yes	Yes	Yes
44	Yes	Yes	Yes
45	Yes	Yes	Yes
46	Yes	Yes	Yes
47	Yes	No	Yes
48	No	No	Yes

products are placed on the workspace table.

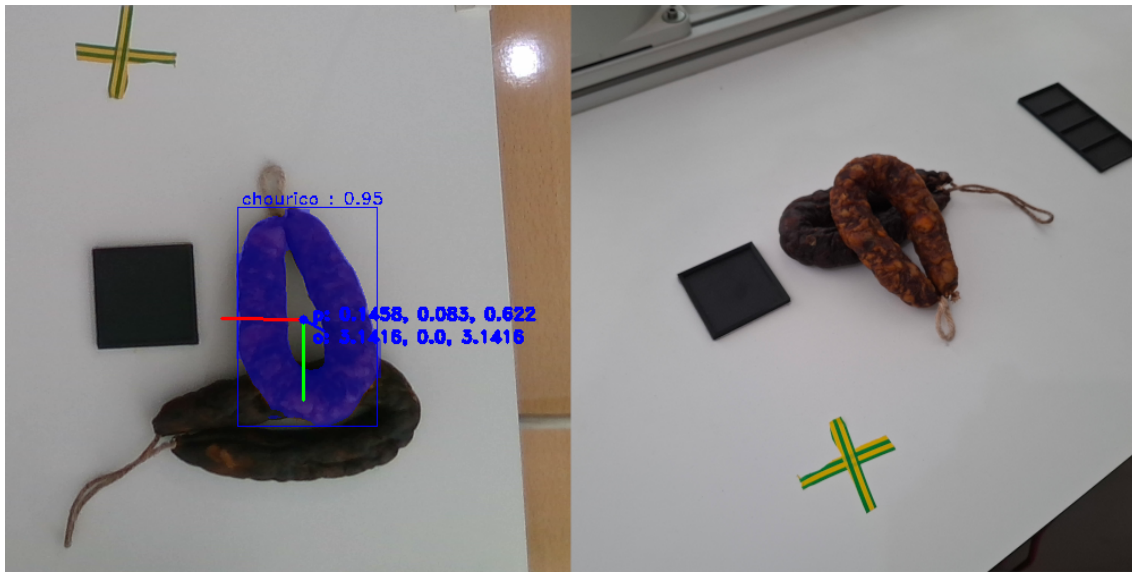


Figure 7.4: Example of placed products for the fourth test

The main objective of this test is to verify how the object pose estimation algorithm handles scenarios where some products occlude others. The expected result is for the algorithm to ignore the bottom product and only calculate the pose for the top product, as the first stage's You Only Look Once (YOLO) model was trained with that output in mind. As with the previous tests, all chosen agri-food products were used in this test. Table 7.9 shows the test poses. Only the poses 4, 5, 7, 8, 16, 17, 19, 20, 28, 29, 31, and 32, described in the table 7.4 from the second test, were used.

Table 7.9: List of the various poses chosen for the fourth test

Pose Number	x	y	z	roll	pitch	yaw
4	0.3	-0.15	-0.02	0	0	0
5	0.3	0	-0.02	0	0	0
7	0.4	-0.15	-0.02	0	0	0
8	0.4	0	-0.02	0	0	0
16	0.3	-0.15	-0.02	0	0	0.7854
17	0.3	0	-0.02	0	0	0.7854
19	0.4	-0.15	-0.02	0	0	0.7854
20	0.4	0	-0.02	0	0	0.7854
28	0.3	-0.15	-0.02	0	0	1.5708
29	0.3	0	-0.02	0	0	1.5708
31	0.4	-0.15	-0.02	0	0	1.5708
32	0.4	0	-0.02	0	0	1.5708

7.2.8 Test results

The following table (table 7.10) shows the results of the pick and place operation of the fourth test. The product on the table is the top product. The bottom product was not considered for the test, as after picking the top product, picking the bottom product is just the same test as the second test (section 7.2.3).

Table 7.10: Fourth test pick and place operation results

Pose Number	<i>chouriço</i>	<i>morcela</i>	<i>farinheira</i>
4	No	Yes	No
5	Yes	Yes	Yes
7	No	Yes	No
8	Yes	No	Yes
16	Yes	Yes	Yes
17	Yes	Yes	Yes
19	Yes	No	Yes
20	Yes	Yes	Yes
28	Yes	No	Yes
29	Yes	Yes	No
31	Yes	Yes	Yes
32	No	Yes	No

From the results shown in the table, it is evident that the object pose estimation algorithm and the robotic system encounter greater difficulty when the products partially occlude one another. Nevertheless, for most poses, the robotic manipulator successfully executed the pick and place operation, indicating that the algorithm was able to compute a valid pose. Some of the final failures resulted from the gripper not closing sufficiently around the product during the pick phase. This issue was unlikely to be related to the object pose estimation algorithm nor to the robotic system Robot Operating System (ROS)

implementation. The most probable cause was the gripper itself, which occasionally exhibited finger overtravel and undertravel during the test. The overall success of the robotic system on the pick and place routine of this test was 72.22%. In comparison, the success of this same routine on the second test on the same poses had a success rate of 85.42%, representing a reduction of 13.19%.

7.3 Validation tests conclusion

The tests presented in this chapter prove that the developed fin ray effect gripper, the object pose estimation algorithm, and the robotic system, developed in ROS, are capable of handling the various chosen agri-food products (*chouriço*, *farinheira*, and *morcela*).

The tests conducted on the fin ray effect gripper demonstrated that it was capable of handling all the selected agri-food products, although it showed greater difficulty with the *farinheira* due to this product's higher malleability compared with the other two. However, during the subsequent tests, it was observed that the gripper system, specifically its moving components, began to exhibit issues related to overtravel and undertravel. These problems were likely caused by damage incurred throughout the various tests, particularly in situations where the gripper attempted to close to a distance shorter than the actual width of the product. This typically occurred either because too large of a subtraction was applied to the grasp width provided by the object pose estimation algorithm or because the algorithm itself estimated a smaller than expected grasp width.

The tests conducted on both the object pose estimation algorithm and the robotic system demonstrated that the algorithm was capable of generating valid poses for the various products and that the robotic system was able to execute pick-and-place operations on the detected items. However, several shortcomings and limitations of the algorithm became evident.

In the first test, the object pose estimation algorithm achieved a success rate of approximately 66.67%. However, most errors were concentrated in a small set of specific poses. Pose number 5 consistently resulted in an invalid pose estimation, and excluding this pose from the calculation increased the success rate to 83.33%. This test further highlighted the possibility of the presence of a restricted area within which the algorithm performed optimally, outside this area, the frequency of invalid pose estimations increased significantly.

To better understand this phenomenon, a second test was performed using grid-like pose distributions rather than randomly selected ones. The results confirmed the behavior observed in the first test. As shown by the heatmap in Figure 7.2 and the absolute error charts for the combined positions in Figure 7.3, the algorithm operates at its best

performance within a specific region of the workspace, while beyond this region it produces too many invalid poses to be considered reliable. The overall success rate of the algorithm in this second test was 59.99%. However, when excluding the poses located at $y = 0.15$ and $x = 0.5$, which are outside the algorithm's area, the success rate increases to approximately 80.56%.

The third test was done to verify if the object pose estimation algorithm could handle products that are presented at an angle against the workspace table and do not have their top surface completely flat in relation to the camera system. On this test only the poses that had better results on the previous test were used, discarding the poses that had $x = 0.2$ and $x = 0.5$. An angle of 10 degrees was applied to the remaining poses, and the product rotated by 90 degrees on the z-axis. The success rate of this test was around 81.48%, very close to the previous test with the discard poses. This proves that the algorithm is capable of handling products presented at an angle to the workspace table.

The final test was conducted to determine whether the object pose estimation algorithm was capable of handling products in occluded scenarios. The test used the same poses as the previous test, ignoring the poses with $x = 0.2$, $x = 0.5$, and $y = 0.15$. In this test, the algorithm achieved a success rate of 72.22%, lower than in the previous tests. This result indicates that, although the algorithm is capable of dealing with occlusions, its performance is negatively affected in such situations. One reason for this reduced performance may have been the onset of undertravel issues in the gripper, which prevented it from closing sufficiently to properly grasp and place the products.

The overtravel and undertravel problems suggest that the gripper actuator, or some components of the main body assembly responsible for the actuation, sustained damage during the various tests. When the object pose estimation algorithm returned a grasp distance smaller than the actual width of the object, the gripper was forced backward by the object as it attempted to close beyond its mechanical limits. This situation occurred both when the vision algorithm underestimated the object width and when the value subtracted by the ROS controller was not properly validated. Over time, this repeated stress likely caused damage to the gripper actuator assembly and appears to be the root cause of the observed problems.

Regarding the errors in position calculation along the x-axis (y-axis in camera coordinates), several possible root causes exist. To better understand this phenomenon, a test was conducted using images from the training dataset of the YOLO segmentation model. In these images, a single product was positioned at similar x values and moved from left to right across the frame. When the y coordinates were recorded, it was found that for objects at similar y values, on the left of the image, they were usually more than 2 centimeters higher than when placed on the right of the image. This indicates that the

error originates from the object pose estimation algorithm itself rather than from the coordinate transformation performed by the ROS controller. The following figure 7.5 shows the comparison between the estimated product poses of products that are relatively close in y coordinates. It is possible to observe that the products on the right have relatively close y coordinates in their calculated pose. However, the two products on the right have their y coordinates with higher values, despite being relatively at the same y coordinates.

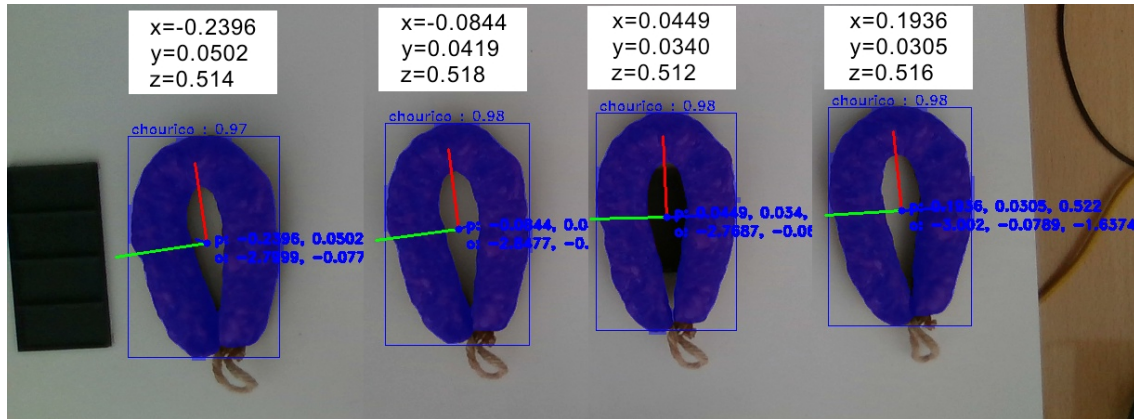


Figure 7.5: Edited image with the position error demonstrated

CONCLUSION

The work presented in this thesis involves the study and development of a robotic system capable of handling various agri-food products, giving focus specifically to meat products like smoked and cured meats. In the case of this work, three products were chosen: *chouriço*, *farinheira*, and *morcela*. For the robotic system, both a fin ray effect gripper and an object pose estimation algorithm were developed. The developed fin ray effect gripper's fingers make use of a fin ray structure 3D printed out of elastic plastics, so-called elastomers, to make the fingers deform around the grasped product and in this way avoid damaging or deforming it. The developed object pose estimation algorithm makes use of a keypoints approach for the calculation of the orientation of the detected products.

Given the current necessities and demands from the growing world population, namely the rising demand for food products, combined with the increased concern about the environment and climate change, are currently creating challenges for the agri-food industry. These challenges create a need for new sustainable solutions that combine precision, high efficiency, and wastage control that maintain or improve the current quality of the food products produced by the industry. To face these challenges, many technologies are being employed for this industry, including artificial intelligence and robotics. The automation of some processes in this industry improves their precision, and for hazardous or tedious tasks, it improves the operators' working conditions. However, the manipulation of agri-food products has some intrinsic challenges and requirements, namely hygiene and safety requirements. The products, due to their fragile and irregular natures, pose some difficulties to their manipulation. The aim of this work is to address these challenges and difficulties by developing a robotic system capable of autonomously detecting and manipulating various agri-food products.

This work's developed robotic system is composed of an ABB IRB120 robot, a 6 Degrees of Freedom (DoF) industrial manipulator, on which was added to the end effector the developed fin ray effect gripper, as well as the camera system. The camera used is an Intel RealSense D415 depth camera, which is a stereo depth camera capable of providing both

Red Green Blue (RGB) and depth images. The camera uses a color sensor to provide the RGB image and two imagers to provide depth information.

The first developed part of the overall robotic system was the gripper system. The gripper system was composed of a linear actuator used in previous works. On it were installed the developed fin ray effect gripper fingers. The fingers were developed through three iterations. The first was not tested, as it was too stiff and could not be bent around the grasped products, which was obviously a failure. This was due to the thickness of the finger as well as the shape being a rectangular triangle, which made the fingers too stiff at their points. The second iteration was far more successful. The gripper fingers were in the shape of an isosceles triangle, had a length of 100 millimeters, and had 10 struts spaced by 7.8 millimeters. To hold these fingers to the linear actuator, a holder piece was developed. The piece held the fingers at 17 degrees to make the fingers have a surface completely parallel between them. When tested, this iteration was found to be adequate to handle all three chosen agri-food products. However, for the *farinheira*, even at maximum extension, the gripper could not fully grasp it if there was a small error in the coordinates given by the object pose estimation algorithm. For that reason, on the third iteration of the design, the fingers were kept the same, only modifying the holder piece by adding an extension that allowed for a longer full extension distance. The test performed on the third iteration of the fin ray effect gripper system proved that the modifications solved the problem of the previous iteration, as the gripper was successful in grasping all three chosen agri-food products.

After finishing the gripper, the object pose estimation algorithm development started. The algorithm was composed of three stages. The first stage was responsible for detecting the various products and providing a segmentation mask for each detected object for the two following stages. This stage uses a trained You Only Look Once (YOLO) segmentation model, more specifically a YOLOv12-seg, that was trained to detect and segment the three chosen agri-food products. The model was also trained to ignore products that are below other products. This was done so the algorithm only picks the top products that can be picked by the robotic manipulator. The second stage is responsible for calculating the coordinates (x , y , and z) for each detected object's position in relation to the camera system. For this, two approaches were developed, one that first transforms the depth information provided by the camera system at a point of the object, and another that makes direct use of the depth information to calculate the product's position. The performance of the two developed approaches was very similar in accuracy, giving similar coordinates for the same products on the same images, but in the speed comparison, the point cloud approach was slower than using directly the depth information. The third stage has the task of calculating the orientation of the detected product in relation to the camera system. The orientation angles are calculated on Euler angles in the x - y - z order (roll, pitch, and yaw). Various methods were developed and tested for this stage, including

approaches using Principal Components Analysis (PCA) on the segmentation mask of the product, Ordinary Least Squares (OLS) regression on the product's point cloud, PCA on the product's point cloud, a combination of PCA on the segmentation mask and OLS, and a keypoints approach. Ultimately for the orientation stage, the keypoints approach was chosen, as it offered better performance in accuracy than the others and was more resilient. For that reason, for the second stage, the chosen method was using directly the depth information provided by the camera, as there was no need to create a point cloud for the keypoints method of the third stage.

To control the robotic system, a Robot Operating System (ROS) controller for the system was developed. The controller architecture was based on a central main node and several peripheral nodes that communicate through services. Each separate peripheral node implements either a different part of the robotics system or an auxiliary function. The developed peripheral nodes included a node to control the gripper system. A node that interfaces with the development object pose estimation algorithm. A node to control the robotic manipulator and for path planning. And a node to help debug and visualize the poses obtained by the vision system in RViz. For the main control, two separate controller nodes were implemented: a manual controller and an automatic controller. The manual controller provides full access to the robotic system and includes optional assisted operations to facilitate system testing. In contrast, the automatic controller implements the logic required to autonomously execute the pick and place routine defined in this work.

Various tests were done to validate the developed vision algorithm, gripper system, and the overall robotic system. To validate the object pose estimation algorithm, as well as the performance of the overall robotic system, four tests were conducted. On the first test, 15 poses were randomly chosen. During this test, it was found that for some of the positions, the object pose estimation algorithm could not generate valid poses for the robotic system to pick the various products.

To better understand the working area limitation phenomenon, a second test was conducted where, instead of using randomly chosen poses, the poses presented a grid-like displacement, varying between $0.2 < x < 0.5$ and $-0.15 < y < 0.15$ in a total of 36 poses. The results from this test confirmed the existence of an area, located on the left side of the camera's image, where the algorithm had difficulty generating valid poses. If these problematic poses are ignored, the success rate is around 80.56%.

A third test conducted was to verify the performance of the algorithm when dealing with products that have some angle against the workspace table. This test only used the positions that had greater success on the previous test, ignoring the ones that the

algorithm failed constantly. On this test the success rate was around 81.48%, for all intents and purposes equal to the second test.

A final fourth test was conducted to verify how the system handled products that were on top of each other. On this test the poses used were the ones with greater success, as with the previous test. During this test it was found that the algorithm suffered from reduced accuracy, having a lower success rate when compared with the two previous tests, only 72.22%. However, during this test it was verified that the gripper started developing some issues, suffering from undertravel, likely from damage caused by making the gripper close too much on some harder products.

The developed object pose estimation algorithm is not without some limitations. The algorithm requires the products to have their top surfaces exposed to the camera to be able to be detected and to calculate their poses. If the product is at an "extreme" angle in relation to the camera, for example, when the pitch or roll angles are close to 90 degrees, the algorithm will not be able to accurately calculate the product pose. A workaround for this limitation is to take advantage of the fact that the vision system is mounted to the end effector of the robot manipulator and move the camera to a different position in relation to the detected product to be able to better calculate its pose. Another limitation of the vision algorithm is the fact that it cannot distinguish which side of the chosen agri-food products contains the string. This is not a problem for a pick and place task, as it does not matter which side the string is located on, but for other tasks, it might be necessary to know.

A limitation identified in the developed fin ray effect gripper was its inability to sense the amount of pressure applied to the grasped product. Instead, it relied entirely on the vision algorithm to determine the required closing distance for a successful grasp. As such, if the vision algorithm provided an inaccurate width measurement, or if the robotic system's controller sends a closing command that was too small relative to the actual object width, the gripper could either damage the product or cause harm to its own structure. To address this limitation, a potential improvement would be the integration of sensing capabilities, for example, through the addition of a pressure sensor. Such tactile sensing would enable the gripper to regulate its closing distance more precisely, reducing the risk of damage and improving the reliability of grasping across different agri-food products.

Future work should focus on improving the robustness and reliability of the developed object pose estimation algorithm by enhancing the keypoint selection. The current vision algorithm assumes that the products have relatively flat surfaces, which may not hold true for other agri-food items. Therefore, adapting the algorithm for products that have different irregular shapes would be an improvement. Additionally, it should address the issue identified in the position calculation for products located on the left side of the depth

camera images. One possible way to bypass this restriction is to move the robot camera to a different position to observe the product from a different perspective and thus be able to properly calculate the product's pose. Regarding the gripper system, future work should focus on giving the developed fin ray effect gripper tactile sensing via pressure/force sensors, for example. This would improve grasp reliability and help prevent product or gripper damage. Further evaluation of the gripper with a wider range of sausages, meat products, and other agri-food items from the production line would provide better insights into its versatility and performance.

BIBLIOGRAPHY

- Raffik, R., Roshan, R. P., Subash, C., & Sanjeev, K. B. (2023). Role of robotics and automation in food industries: An overview. *2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, 1–5. <https://doi.org/10.1109/ICAECA56562.2023.10199301> (cit. on pp. 1, 4–7).
- Derossi, A., Di Palma, E., Moses, J., Santhoshkumar, P., Caporizzi, R., & Severini, C. (2023). Avenues for non-conventional robotics technology applications in the food industry. *Food Research International*, 173, 113265. <https://doi.org/https://doi.org/10.1016/j.foodres.2023.113265> (cit. on pp. 1, 4, 5).
- Konfo, T. R. C., Djouhou, F. M. C., Hounhouigan, M. H., Dahouenon-Ahoussi, E., Avlessi, F., & Sohounhloue, C. K. D. (2023). Recent advances in the use of digital technologies in agri-food processing: A short review. *Applied Food Research*, 3(2), 100329. <https://doi.org/https://doi.org/10.1016/j.afres.2023.100329> (cit. on pp. 1, 4, 5).
- Duong, L. N., Al-Fadhli, M., Jagtap, S., Bader, F., Martindale, W., Swainson, M., & Paoli, A. (2020). A review of robotics and autonomous systems in the food industry: From the supply chains perspective. *Trends in Food Science & Technology*, 106, 355–364. <https://doi.org/https://doi.org/10.1016/j.tifs.2020.10.028> (cit. on pp. 1, 4–6).
- de Medeiros Esper, I., From, P. J., & Mason, A. (2021). Robotisation and intelligent systems in abattoirs. *Trends in Food Science & Technology*, 108, 214–222. <https://doi.org/https://doi.org/10.1016/j.tifs.2020.11.005> (cit. on pp. 1, 5, 6).
- Aly, B. A., Low, T., Long, D., Baillie, C., & Brett, P. (2023). Robotics and sensing technologies in red meat processing: A review. *Trends in Food Science & Technology*, 137, 142–155. <https://doi.org/https://doi.org/10.1016/j.tifs.2023.05.015> (cit. on pp. 5, 7, 8).
- Iqbal, J., Khan, Z., & Khalid, A. (2017). Prospects of robotics in food industry. *Food Science and Technology (Campinas)*, 37, 159–165. <https://doi.org/10.1590/1678-457X.14616> (cit. on pp. 5–7).
- Yeshmukhametov, A., Kalimoldayev, M., Mamyrbayev, O., & Amirgaliev, Y. (2017). Design and kinematics of serial/parallel hybrid robot. *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, 162–165. <https://doi.org/10.1109/ICCAR.2017.7942679> (cit. on p. 6).

- Yaguchi, H., Nagahama, K., Hasegawa, T., & Inaba, M. (2016). Development of an autonomous tomato harvesting robot with rotational plucking gripper. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 652–657. <https://doi.org/10.1109/IROS.2016.7759122> (cit. on pp. 6, 25, 26).
- Hu, X., Yu, H., Lv, S., & Wu, J. (2021). Design and experiment of a new citrus harvesting robot. *2021 International Conference on Control Science and Electric Power Systems (CSEPS)*, 179–183. <https://doi.org/10.1109/CSEPS53726.2021.00043> (cit. on pp. 6, 25, 26).
- Özentürk, U., Chen, Z., Jamone, L., & Versace, E. (2024). Robotics for poultry farming: Challenges and opportunities. *Computers and Electronics in Agriculture*, 226, 109411. <https://doi.org/https://doi.org/10.1016/j.compag.2024.109411> (cit. on p. 7).
- Safety of machinery — hygiene requirements for the design of machinery* (Standard). (2002, April). International Organization for Standardization. Geneva, CH. <https://www.iso.org/standard/23748.html> (cit. on p. 7).
- Food safety management systems — requirements for any organization in the food chain* (Standard). (2018, June). International Organization for Standardization. Geneva, CH. <https://www.iso.org/standard/65464.html> (cit. on p. 8).
- Robots and robotic devices — safety requirements for industrial robots* (Standard). (2011, July). International Organization for Standardization. Geneva, CH. <https://www.iso.org/standard/51330.html> (cit. on p. 8).
- Ross, S., Korostynska, O., Cordova-Lopez, L., & Mason, A. (2022). A review of unilateral grippers for meat industry automation. *Trends in Food Science & Technology*, 119, 309–319. <https://doi.org/https://doi.org/10.1016/j.tifs.2021.12.017> (cit. on p. 8).
- Raja, R., Burusa, A. K., Kootstra, G., & Van Henten, E. J. (2024). Advanced robotic system for efficient pick-and-place of deformable poultry in cluttered bin: A comprehensive evaluation approach. *IEEE Transactions on AgriFood Electronics*, 2(2), 355–371. <https://doi.org/10.1109/TAFE.2024.3379190> (cit. on pp. 8, 16).
- Guo, J., Sun, Y., Liang, X., Low, J.-H., Wong, Y.-R., Tay, V. S.-C., & Yeow, C.-H. (2017). Design and fabrication of a pneumatic soft robotic gripper for delicate surgical manipulation. *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, 1069–1074. <https://doi.org/10.1109/ICMA.2017.8015965> (cit. on pp. 8, 11).
- Low, J. H., Khin, P. M., Han, Q. Q., Yao, H., Teoh, Y. S., Zeng, Y., Li, S., Liu, J., Liu, Z., Valdivia y Alvarado, P., Chen, I.-M., Tee, B. C. K., & Yeow, R. C. H. (2022). Sensorized reconfigurable soft robotic gripper system for automated food handling. *IEEE/ASME Transactions on Mechatronics*, 27(5), 3232–3243. <https://doi.org/10.1109/TMECH.2021.3110277> (cit. on p. 9).
- Salem, M. E. M., Wang, Q., Wen, R., & Xiang, M. (2018). Design and characterization of soft pneumatic actuator for universal robot gripper. *2018 International Conference on Control and Robots (ICCR)*, 6–10. <https://doi.org/10.1109/ICCR.2018.8534483> (cit. on p. 9).

- Low, J. H., Lee, W. W., Khin, P. M., Thakor, N. V., Kukreja, S. L., Ren, H. L., & Yeow, C. H. (2017). Hybrid tele-manipulation system using a sensorized 3-d-printed soft robotic gripper and a soft fabric-based haptic glove. *IEEE Robotics and Automation Letters*, 2(2), 880–887. <https://doi.org/10.1109/LRA.2017.2655559> (cit. on p. 9).
- Hashanjana, L., Senanayaka, P., Madushan, I., Himaruwan, H. D., Kulasekera, A. L., & Dassanayake, P. C. (2023). Design and development of a soft gripper system for difficult-to-handle food items. *Moratuwa Engineering Research Conference, MERCon*, 678–683. <https://doi.org/10.1109/MERCon60487.2023.10355477> (cit. on p. 9).
- Yang, Y., Jin, K., Zhu, H., Song, G., Lu, H., & Kang, L. (2021). A 3d-printed fin ray effect inspired soft robotic gripper with force feedback. *Micromachines*, 12(10). <https://doi.org/10.3390/mi12101141> (cit. on pp. 10, 11, 33, 34).
- Liu, S. Q., & Adelson, E. H. (2022). Gelsight fin ray: Incorporating tactile sensing into a soft compliant robotic gripper. <https://arxiv.org/abs/2204.07146> (cit. on pp. 10, 11, 34).
- Guo, J., Low, J.-H., Liang, X., Lee, J. S., Wong, Y.-R., & Yeow, R. C. H. (2019). A hybrid soft robotic surgical gripper system for delicate nerve manipulation in digital nerve repair surgery. *IEEE/ASME Transactions on Mechatronics*, 24(4), 1440–1451. <https://doi.org/10.1109/TMECH.2019.2924518> (cit. on p. 11).
- Maruyama, R., Watanabe, T., & Uchida, M. (2013). Delicate grasping by robotic gripper with incompressible fluid-based deformable fingertips. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5469–5474. <https://doi.org/10.1109/IROS.2013.6697148> (cit. on p. 12).
- He, L., Lu, Q., Abad, S. A., Rojas, N., & Nanayakkara, T. (2020). Soft fingertips with tactile sensing and active deformation for robust grasping of delicate objects. *IEEE Robotics and Automation Letters*, 5, 2714–2721. <https://doi.org/10.1109/LRA.2020.2972851> (cit. on p. 12).
- Godaba, H., Sajad, A., Patel, N., Althoefer, K., & Zhang, K. (2020). A two-fingered robot gripper with variable stiffness flexure hinges based on shape morphing. *IEEE International Conference on Intelligent Robots and Systems*, 8716–8721. <https://doi.org/10.1109/IROS45743.2020.9341554> (cit. on p. 13).
- Sam, R., & Nefti, S. (2011). A novel, flexible and multi-functional handling device based on bernoulli principle. *2011 IEEE International Conference on System Engineering and Technology*, 166–171. <https://doi.org/10.1109/ICSEngT.2011.5993443> (cit. on pp. 13, 14).
- Sam, R., & Buniyamin, N. (2012). A bernoulli principle based flexible handling device for automation of food manufacturing processes. *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, 214–219. <https://doi.org/10.1109/ICCAIS.2012.6466590> (cit. on pp. 13, 14).
- Mansor, M. F., Sam, R., Masrie, M., & Janin, Z. (2017). Design and development of flexible end-effector based on bernoulli principle. *2017 IEEE 13th International Colloquium*

- on *Signal Processing & its Applications (CSPA)*, 318–322. <https://doi.org/10.1109/CSPA.2017.8064973> (cit. on pp. 13, 14).
- Compressed air part 1: Contaminants and purity classes* (Standard). (2010, April). International Organization for Standardization. Geneva, CH. <https://www.iso.org/standard/46418.html> (cit. on p. 13).
- Rachkov, M., & Bebenin, V. (2018). Automatic two-stage vacuum gripper system. *2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, 1–4. <https://doi.org/10.1109/ICIEAM.2018.8728733> (cit. on pp. 14, 15).
- Bryan, P., Kumar, S., & Sahin, F. (2019). Design of a soft robotic gripper for improved grasping with suction cups. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2405–2410. <https://doi.org/10.1109/SMC.2019.8914479> (cit. on p. 14).
- Papadakis, E., Raptopoulos, F., Koskinopoulou, M., & Maniadakis, M. (2020). On the use of vacuum technology for applied robotic systems. *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, 73–77. <https://doi.org/10.1109/ICMRE49073.2020.9065189> (cit. on pp. 14, 15).
- Cherapanamjeri, J., & Rao, B. N. K. (2022). Neural networks based object detection techniques in computer vision. *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, 1092–1099. <https://doi.org/10.1109/ICIRCA54612.2022.9985581> (cit. on p. 16).
- Nayeem, T. A., Motaharuzzaman, S. M., Hoque, A. T., & Rahman, M. H. (2022). Computer vision based object detection and recognition system for image searching. *2022 12th International Conference on Electrical and Computer Engineering (ICECE)*, 148–151. <https://doi.org/10.1109/ICECE57408.2022.10089019> (cit. on p. 16).
- OTHMAN, N. A., SALUR, M. U., KARAKOSE, M., & AYDIN, I. (2018). An embedded real-time object detection and measurement of its size. *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 1–4. <https://doi.org/10.1109/IDAP.2018.8620812> (cit. on p. 17).
- Shreyas, E., Sheth, M. H., & Mohana. (2021). 3d object detection and tracking methods using deep learning for computer vision applications. *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 735–738. <https://doi.org/10.1109/RTEICT52294.2021.9573964> (cit. on p. 17).
- Tran, V. L., & Lin, H.-Y. (2020). 3d object detection and 6d pose estimation using rgb-d images and mask r-cnn. *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–6. <https://doi.org/10.1109/FUZZ48607.2020.9177601> (cit. on pp. 17, 18, 23, 24, 44).
- Gupta, K., Petersson, L., & Hartley, R. (2019). Cullnet: Calibrated and pose aware confidence scores for object pose estimation. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2758–2766. <https://doi.org/10.1109/ICCVW.2019.00337> (cit. on pp. 17–19, 44).

- Wong, S.-W., Chiu, Y.-C., & Tsai, C.-Y. (2023). A real-time affordance-based object pose estimation approach for robotic grasp pose estimation. *2023 International Conference on System Science and Engineering (ICSSE)*, 614–619. <https://doi.org/10.1109/ICSSE58758.2023.10227244> (cit. on pp. 17, 18, 22, 23, 44).
- Yun, W.-h., Lee, J., Lee, J.-H., & Kim, J. (2017). Object recognition and pose estimation for modular manipulation system: Overview and initial results. *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 198–201. <https://doi.org/10.1109/URAI.2017.7992711> (cit. on pp. 17–20).
- Höfer, T., Shamsafar, F., Benbarka, N., & Zell, A. (2021). Object detection and autoencoder-based 6d pose estimation for highly cluttered bin picking. *2021 IEEE International Conference on Image Processing (ICIP)*, 704–708. <https://doi.org/10.1109/ICIP42928.2021.9506304> (cit. on pp. 18–20, 44).
- Chen, W., Duan, J., Basevi, H., Chang, H. J., & Leonardis, A. (2020). Pointposenet: Point pose network for robust 6d object pose estimation. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2813–2822. <https://doi.org/10.1109/WACV45572.2020.9093272> (cit. on pp. 18, 20, 21, 44).
- Wu, L., Li, X., Zhong, K., Li, Z., Wang, C., & Shi, Y. (2022). Hccg: Efficient high compatibility correspondence grouping for 3d object recognition and 6d pose estimation in cluttered scenes. *Measurement*, 197, 111296. <https://doi.org/https://doi.org/10.1016/j.measurement.2022.111296> (cit. on pp. 18, 20–22).
- Taryudi & Wang, M.-S. (2017). 3d object pose estimation using stereo vision for object manipulation system. *2017 International Conference on Applied System Innovation (ICASI)*, 1532–1535. <https://doi.org/10.1109/ICASI.2017.7988217> (cit. on pp. 18, 24).
- Bai, H. (2023). Icp algorithm: Theory, practice and its slam-oriented taxonomy. *Applied and Computational Engineering*, 2(1), 10–21. <https://doi.org/10.54254/2755-2721/2/20220512> (cit. on p. 19).
- Okarma, K. (2020). Applications of computer vision in automation and robotics. *Applied Sciences*, 10(19). <https://doi.org/10.3390/app10196783> (cit. on p. 25).
- Villalba-Diez, J., Schmidt, D., Gevers, R., Ordieres-Meré, J., Buchwitz, M., & Wellbrock, W. (2019). Deep learning for industrial computer vision quality control in the printing industry 4.0. *Sensors*, 19(18). <https://doi.org/10.3390/s19183987> (cit. on p. 25).
- Ji, J., Pannakkong, W., & Buddhakulsomsiri, J. (2023). A computer vision-based system for metal sheet pick counting. *Computers, Materials and Continua*, 75(2), 3643–3656. <https://doi.org/https://doi.org/10.32604/cmc.2023.037507> (cit. on p. 25).
- Kakani, V., Nguyen, V. H., Kumar, B. P., Kim, H., & Pasupuleti, V. R. (2020). A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, 2, 100033. <https://doi.org/https://doi.org/10.1016/j.jafr.2020.100033> (cit. on p. 25).

- Fernandes, J., Turrissi da Costa, V., Barbin, D., Cruz-Tirado, J., Baeten, V., & Barbon Junior, S. (2022). Deep computer vision system for cocoa classification. *Multimedia Tools and Applications*, 1, 1. <https://doi.org/10.1007/s11042-022-13097-3> (cit. on p. 26).
- Gidugu, L., Javed, A., & Faller, L.-M. (2024). Versatile 3d-printed fin-ray effect soft robotic fingers: Lightweight optimization and performance analysis. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 46. <https://doi.org/10.1007/s40430-024-04957-0> (cit. on pp. 34, 35).
- European Parliament and Council. (2004, October). Regulation (ec) no 1935/2004 of the european parliament and of the council on materials and articles intended to come into contact with food [Official Journal of the European Union]. <https://eur-lex.europa.eu/eli/reg/2004/1935/2021-03-27> (cit. on p. 34).
- European Commission. (2006, December). Commission regulation (ec) no 2023/2006 on good manufacturing practice for materials and articles intended to come into contact with food [Official Journal of the European Union]. <https://eur-lex.europa.eu/eli/reg/2006/2023/2008-04-17> (cit. on p. 34).
- European Commission. (2011, January). Commission regulation (eu) no 10/2011 on plastic materials and articles intended to come into contact with food [Official Journal of the European Union]. <https://eur-lex.europa.eu/eli/reg/2011/10/oj> (cit. on p. 34).
- Lupi, F., Freitas, N., Arvana, M., Rocha, A. D., Maffei, A., Barata, J., & Lanzetta, M. (2024). Next-generation vision inspection systems: A pipeline from 3d model to reco file. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-024-02456-6> (cit. on p. 41).
- Intel Corporation. (2019, January). *Intel® realsense™ d400 series product family datasheet* (Revision 005) [Document Number: 337029-005]. <http://www.intel.com/design/literature.htm> (cit. on pp. 45, 49).
- Intel RealSense SDK Team. (2025). Post-processing filters [GitHub documentation, accessed 2025-07-21]. (Cit. on p. 46).
- Ultralytics. (2023a). Yolo ultralytics, instance segmentation. <https://docs.ultralytics.com/tasks/segment/> (cit. on p. 47).
- Ultralytics. (2023b). Yolo ultralytics, performance and metrics deep dive. <https://docs.ultralytics.com/guides/yolo-performance-metrics/> (cit. on p. 47).
- Gewers, F. L., Ferreira, G. R., Arruda, H. F. D., Silva, F. N., Comin, C. H., Amancio, D. R., & Costa, L. D. F. (2021). Principal component analysis: A natural approach to data exploration. *ACM Comput. Surv.*, 54(4). <https://doi.org/10.1145/3447755> (cit. on p. 55).
- Nicoguardo. (2016, February). *Gaussianscatterpca.svg* [Image licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)]. <https://commons.wikimedia.org/wiki/File:GaussianScatterPCA.svg> (cit. on p. 55).
- ABB Robotics. (2019, November). *Irb 120: Abb's 6 axis robot - for flexible and compact production* [Revision J, Document No. ROBO149EN_D]. Zurich, Switzerland.

BIBLIOGRAPHY

- <https://new.abb.com/products/robotics/industrial-robots/irb-120> (cit. on pp. 66, 78).
- Edwards, S. (2018). Abb package summary. <https://wiki.ros.org/abb> (cit. on p. 66).

OBJECT POSE ESTIMATION ALGORITHM'S TESTS VALUES

This appendix shows the values obtain during the tests done on the object pose estimation algorithm. In total were recorded values for 4 tests, described in the previous chapter 7. Yes* represents a successful pick and place operation; however, the grasped pose was visibly unstable. Yes** represents a pick and place operation where the product to be detected needed to be adjusted at this pose, this was because the product was not clearly visible on the camera's image.

A.1 Values obtained during the first test

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
chouriço	1	0.2200	-0.1486	-0.0169	-0.0132	0.0385	0.2463	Yes*
morcela	1	0.2335	-0.1445	-0.0149	-0.0159	0.0081	0.1382	Yes**
farinheira	1	0.2136	-0.1449	-0.0169	0.2723	0.3152	0.6122	Yes*
chouriço	2	0.2558	-0.0066	-0.0139	0.0432	-0.0543	0.0181	Yes
morcela	2	0.2521	-0.0029	-0.0119	0.0368	-0.0891	0.0232	Yes
farinheira	2	0.2503	-0.0059	-0.0159	0.0564	0.0590	0.1957	Yes
chouriço	3	0.3609	-0.1309	-0.0208	0.0283	0.0092	0.0388	Yes
morcela	3	0.3596	-0.1291	-0.0189	0.0437	-0.0367	0.0187	Yes
farinheira	3	0.3687	-0.1217	-0.0208	0.0599	-0.0066	0.0063	Yes
chouriço	4	0.3325	0.1218	-0.0179	0.0227	0.0048	0.0416	No
morcela	4	0.329	0.1226	-0.0178	0.077	-0.0262	0.0195	No
farinheira	4	0.3232	0.1268	-0.0178	0.0991	-0.0102	-0.0767	Yes*
chouriço	5	0.4592	0.1346	-0.0207	0.0479	-0.0257	-0.0084	No
morcela	5	0.4561	0.1328	-0.0198	0.0787	-0.0644	0.0000	No
farinheira	5	0.4507	0.1313	-0.0237	0.1668	-0.0286	0.1336	No
chouriço	6	0.2062	-0.1462	-0.0159	0.1466	-0.0225	1.5127	Yes
morcela	6	0.2069	-0.1469	-0.0159	0.2034	-0.0452	1.4837	Yes

APPENDIX A. OBJECT POSE ESTIMATION ALGORITHM'S TESTS VALUES

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
farinheira	6	0.2189	-0.1365	-0.0159	0.0653	-0.082	1.7041	Yes
chouriço	7	0.2489	-0.0002	-0.0159	0.1564	-0.0210	1.5816	Yes
morcela	7	0.2488	-0.0017	-0.0159	0.1535	-0.0678	1.5318	Yes
farinheira	7	0.2473	-0.0003	-0.0189	0.2707	-0.0915	1.6072	Yes
chouriço	8	0.3553	-0.1271	-0.0198	0.1038	-0.0394	1.5068	Yes
morcela	8	0.3630	-0.1252	-0.0208	0.1754	-0.0857	1.5393	Yes
farinheira	8	0.3602	-0.1245	-0.0209	0.2620	-0.1245	1.5571	Yes
chouriço	9	0.3302	0.1184	-0.0188	0.1453	-0.0193	1.6451	No
morcela	9	0.3258	0.1221	-0.0198	0.1995	-0.0827	1.6226	Yes
farinheira	9	0.3236	0.1213	-0.0199	0.1873	-0.1267	1.4989	No
chouriço	10	0.4512	0.1323	-0.0207	0.1253	-0.0026	1.5399	No
morcela	10	0.452	0.1282	-0.0218	0.0238	-0.0714	1.6019	No
farinheira	10	0.4521	0.1347	-0.0217	0.1418	-0.1253	1.5981	No
chouriço	11	0.2175	-0.1369	-0.0159	0.0984	0.0173	0.862	Yes
morcela	11	0.2179	-0.1386	-0.0169	0.1903	0.1084	0.8538	Yes
farinheira	11	0.2081	-0.1464	-0.0169	0.2371	0.0736	1.0031	Yes
chouriço	12	0.2591	0.0076	-0.0149	0.0196	-0.0309	0.7289	Yes
morcela	12	0.2564	0.0034	-0.0139	-0.0130	-0.0750	0.7492	Yes
farinheira	12	0.2628	0.0055	-0.0169	0.1991	-0.0052	0.8194	Yes
chouriço	13	0.3596	-0.1228	-0.0198	0.0726	-0.0151	0.7309	Yes
morcela	13	0.3575	-0.1286	-0.0198	0.0743	-0.0425	0.8088	Yes
farinheira	13	0.3539	-0.1234	-0.0188	0.0107	-0.0647	0.7323	Yes
chouriço	14	0.3265	0.1228	-0.0198	0.0753	0.0098	0.8174	Yes
morcela	14	0.3286	0.1224	-0.0218	0.0737	-0.0373	0.7521	No
farinheira	14	0.3308	0.1259	-0.0208	0.1063	0.0320	0.6429	No
chouriço	15	0.4542	0.1353	-0.0217	0.0839	-0.0067	0.809	No
morcela	15	0.4507	0.1319	-0.0227	0.0359	-0.0587	0.8207	No
farinheira	15	0.4571	0.1312	-0.0227	0.0324	-0.0489	0.7913	No

A.2 Values obtained during the second test

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
chourico	1	0.2211	-0.1576	-0.0159	0.0152	-0.0118	-0.0276	Yes
morcela	1	0.2270	-0.1545	-0.0149	0.0241	-0.0094	0.1157	No
farinheira	1	0.2123	-0.1555	-0.0189	0.1248	0.1834	0.7677	No
chourico	2	0.2164	-0.0080	-0.0149	0.1058	0.0755	0.2477	Yes
morcela	2	0.2256	-0.0079	-0.0129	0.0879	0.0610	0.2676	Yes
farinheira	2	0.2101	-0.0074	-0.0169	0.2781	0.1332	1.1448	No
chourico	3	0.2144	0.1416	-0.0169	0.1067	0.0961	0.2472	Yes
morcela	3	0.2172	0.1412	-0.0149	0.1303	-0.0165	0.1332	Yes
farinheira	3	0.2048	0.1389	-0.0180	0.3063	-0.1949	1.8196	No
chourico	4	0.3103	-0.1484	-0.0189	-0.0295	0.0530	0.0298	Yes
morcela	4	0.3025	-0.1495	-0.0179	0.0153	-0.0260	0.0186	Yes
farinheira	4	0.3039	-0.1501	-0.0229	0.0545	0.0301	0.3868	Yes
chourico	5	0.2966	-0.0022	-0.0118	0.007	0.0213	0.1623	Yes
morcela	5	0.2923	-0.0007	-0.01189	0.0383	-0.0549	0.0628	Yes
farinheira	5	0.2977	-0.0037	-0.01384	0.0739	-0.0076	0.1250	Yes
chourico	6	0.2867	0.1478	-0.0189	0.1071	0.0680	0.0415	Yes
morcela	6	0.285	0.1423	-0.0169	0.2157	-0.0061	-0.0016	No
farinheira	6	0.2895	0.1448	-0.0189	0.087	0.0411	-0.1086	Yes
chourico	7	0.4067	-0.1421	-0.0218	-0.0191	0.0365	-0.0640	Yes
morcela	7	0.4055	-0.1401	-0.0197	0.0371	-0.0363	0.0696	Yes
farinheira	7	0.4053	-0.146	-0.0208	-0.0325	-0.0377	-0.0597	Yes
chourico	8	0.3928	0.0048	-0.0118	-0.0276	0.0373	0.0915	Yes
morcela	8	0.3958	0.0049	-0.0088	0.0062	-0.0170	-0.0271	No
farinheira	8	0.3913	0.0019	-0.0118	0.0086	-0.0151	-0.0417	Yes
chourico	9	0.3904	0.153	-0.0208	0.0990	0.0211	-0.0237	No
morcela	9	0.3861	0.1521	-0.0198	0.1051	-0.0386	-0.0312	No
farinheira	9	0.3875	0.1539	-0.0218	0.0926	0.0047	-0.0015	Yes
chourico	10	0.5104	-0.1401	-0.0227	-0.0224	-0.0099	0.0762	No
morcela	10	0.5015	-0.1362	-0.0217	0.0546	-0.0832	0.0113	No
farinheira	10	0.5089	-0.1336	-0.0227	-0.0583	-0.0216	-0.0862	No
chourico	11	0.4984	0.0089	-0.0158	-0.0143	0.0079	0.0393	Yes
morcela	11	0.4898	0.0082	-0.0187	-0.0506	-0.0383	0.0263	No
farinheira	11	0.4953	0.0133	-0.0167	-0.0074	-0.0459	-0.0482	No
chourico	12	0.4873	0.1598	-0.0217	0.0835	-0.0435	0.0647	No
morcela	12	0.4835	0.1622	-0.0197	0.0950	-0.0630	0.0560	No
farinheira	12	0.4815	0.1548	-0.0217	0.1700	-0.0129	0.1661	Yes
chourico	13	0.2219	-0.1583	-0.0159	0.0645	0.01700	0.7853	Yes

APPENDIX A. OBJECT POSE ESTIMATION ALGORITHM'S TESTS VALUES

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
morcela	13	0.2167	-0.1412	-0.0159	0.2055	0.0929	0.8838	Yes
farinheira	13	0.2105	-0.1476	-0.0159	0.2275	0.0421	0.9202	Yes
chourico	14	0.2089	-0.0003	-0.0159	0.2068	0.0540	1.1452	Yes
morcela	14	0.2119	0.0083	-0.0159	0.2809	0.1128	1.0219	Yes
farinheira	14	0.2097	-0.0023	-0.0139	0.366	0.1145	0.9507	Yes
chourico	15	0.2036	0.1531	-0.0169	0.0636	-0.0507	1.3235	No
morcela	15	0.2112	0.1563	-0.016	0.3033	0.1185	1.1077	No
farinheira	15	0.2023	0.1472	-0.0169	0.5129	0.1195	1.2293	No
chourico	16	0.3082	-0.1419	-0.0189	0.0766	-0.0040	0.7957	Yes
morcela	16	0.3039	-0.1497	-0.0189	0.052	-0.0041	0.7772	Yes
farinheira	16	0.3026	-0.1452	-0.0178	0.0821	0.0070	0.9046	Yes
chourico	17	0.3053	0.0006	-0.0108	-0.0168	-0.0138	0.7094	Yes
morcela	17	0.2951	-0.0023	-0.0128	-0.0658	-0.0905	0.8048	Yes
farinheira	17	0.2944	0.0006	-0.0128	-0.0137	-0.0905	0.06608	Yes
chourico	18	0.2847	0.1477	-0.0188	0.183	-0.0069	0.7573	No
morcela	18	0.2828	0.1486	-0.0199	0.1385	-0.0368	0.7482	No
farinheira	18	0.2816	0.1470	-0.0189	0.2778	-0.0355	0.6962	No
chourico	19	0.4062	-0.1345	-0.0207	0.0546	-0.0373	0.8072	Yes
morcela	19	0.4031	-0.1373	-0.0207	0.0551	-0.0515	0.8183	Yes
farinheira	19	0.3989	-0.1424	-0.0208	0.1675	-0.0427	1.0341	Yes
chourico	20	0.3900	0.0020	-0.0128	0.0339	0.0294	0.7802	Yes
morcela	20	0.3942	0.0091	-0.0118	0.0296	-0.0098	0.7969	No
farinheira	20	0.3921	0.0105	-0.0108	0.1995	0.0588	0.6495	Yes
chourico	21	0.3846	0.1559	-0.0208	0.1390	-0.0151	0.7729	No
morcela	21	0.3830	0.1540	-0.0218	0.0754	-0.0571	0.7670	No
farinheira	21	0.3803	0.1527	-0.0188	0.1848	0.0271	0.6072	No
chourico	22	0.5031	-0.0133	-0.0217	0.0265	-0.0197	0.7601	Yes
morcela	22	0.5011	-0.1372	-0.0226	-0.0130	-0.0734	0.8281	Yes
farinheira	22	0.5015	-0.1381	-0.0207	0.0252	-0.0877	0.8815	Yes
chourico	23	0.4896	0.0112	-0.0167	0.0096	0.0278	0.7190	No
morcela	23	0.4902	0.0168	-0.0156	-0.0455	-0.0155	0.8171	No
farinheira	23	0.4943	0.0148	-0.0147	-0.0373	-0.1341	0.5091	No
chourico	24	0.4822	0.1626	-0.0217	0.0563	-0.0196	0.7774	No
morcela	24	0.4824	0.1629	-0.0227	0.0555	-0.0813	0.7859	No
farinheira	24	0.4784	0.1602	-0.0208	0.1946	-0.1292	0.7799	No
chourico	25	0.2101	-0.1551	-0.0149	0.1661	-0.0080	1.5258	Yes
morcela	25	0.2051	-0.0153	-0.0169	0.2161	-0.0306	1.5021	No
farinheira	25	0.2041	-0.1554	-0.0159	0.3632	-0.0991	1.4822	Yes
chourico	26	0.2034	-0.0095	-0.0159	0.1020	-0.0460	1.5801	Yes

A.2. VALUES OBTAINED DURING THE SECOND TEST

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
morcela	26	0.2002	-0.0074	-0.0169	0.1111	-0.0770	1.5457	No
farinheira	26	0.2000	-0.008	-0.0149	0.2422	-0.1104	1.5891	Yes
chourico	27	0.2029	0.1409	-0.0170	0.1167	-0.0505	1.5774	No
morcela	27	0.1948	0.135	-0.0199	-0.1077	-0.0748	1.5795	No
farinheira	27	0.2005	0.1374	-0.018	0.3184	-0.2153	1.7017	Yes
chourico	28	0.3018	-0.1474	-0.0179	0.1309	-0.0128	1.4789	Yes
morcela	28	0.3011	-0.1486	-0.0179	0.1683	-0.0282	1.4219	Yes
farinheira	28	0.2982	-0.1533	-0.0189	0.3166	-0.1020	1.4327	Yes
chourico	29	0.2934	-0.0017	-0.0139	0.0901	0.0043	1.5281	Yes
morcela	29	0.2927	-0.0037	-0.0139	0.0617	-0.0573	1.5348	Yes
farinheira	29	0.2899	0.0041	-0.0139	0.1614	-0.0202	1.4856	Yes
chourico	30	0.2823	0.1448	-0.0189	0.182	-0.4010	1.6054	No
morcela	30	0.2831	0.1432	-0.0179	0.2228	-0.0958	1.5717	No
farinheira	30	0.2822	0.1429	-0.0199	0.3503	-0.2161	1.7451	Yes
chourico	31	0.3995	-0.1431	-0.0208	0.0986	-0.0329	1.5146	Yes
morcela	31	0.3989	-0.1453	-0.0207	0.0816	-0.0900	1.5778	Yes
farinheira	31	0.3989	-0.1452	-0.0208	0.0953	-0.1500	1.3749	Yes
chourico	32	0.3919	0.0083	-0.0139	0.1475	-0.0111	1.5505	Yes
morcela	32	0.3906	0.0089	-0.0138	0.2260	-0.0556	1.6207	Yes
farinheira	32	0.385	0.0056	-0.0128	0.1058	-0.0884	1.5223	Yes
chourico	33	0.3809	0.1552	-0.0198	0.2061	-0.0395	1.5699	No
morcela	33	0.3801	0.1527	-0.0219	0.1870	-0.0882	1.6297	No
farinheira	33	0.3816	0.1534	-0.0228	0.3042	-0.1758	1.5418	No
chourico	34	0.5002	-0.1347	-0.0217	0.1181	-0.0116	1.516	Yes
morcela	34	0.4936	-0.1332	-0.0218	0.0245	-0.0774	1.5887	Yes
farinheira	34	0.4953	-0.1324	-0.0237	0.2214	-0.0822	1.6066	Yes
chourico	35	0.4918	0.0100	-0.0176	0.1755	0.0008	1.5448	Yes
morcela	35	0.4892	0.0098	-0.0186	0.0775	-0.0587	1.5546	Yes
farinheira	35	0.4859	0.0110	-0.0167	0.1542	-0.0691	1.5385	Yes
chourico	36	0.4814	0.1604	-0.0218	0.1449	-0.0276	1.544	No
morcela	36	0.4815	0.1610	-0.0227	0.0905	-0.0799	1.626	No
farinheira	36	0.4843	0.1525	-0.0217	0.1555	-0.1545	1.603	No

A.3 Values obtained during the third test

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
chourico	37	0.3138	-0.1467	-0.0018	0.0224	-0.2448	0.0097	yes
morcela	37	0.3127	-0.1453	0.0001	-0.0135	-0.2745	0.0601	yes
farinheira	37	0.3026	-0.1460	-0.0018	-0.1480	-0.2536	0.2403	yes
chourico	38	0.2989	-0.0031	0.0041	0.0289	-0.3090	0.0335	yes
morcela	38	0.3000	-0.0030	0.0021	-0.0516	-0.2535	0.0437	yes
farinheira	38	0.2962	-0.0005	-0.0027	-0.1438	-0.2360	0.5506	yes
chourico	39	0.2920	0.1488	-0.0068	0.0825	-0.1605	0.0519	yes
morcela	39	0.2872	0.1458	-0.0059	0.0682	-0.1777	0.1384	no
farinheira	39	0.2871	0.1466	-0.0099	0.2358	-0.1189	0.0491	yes
chourico	40	0.2986	-0.1507	-0.0089	-0.0537	0.2997	0.1963	yes
morcela	40	0.2999	-0.1479	-0.0088	0.0452	0.2934	0.1136	yes
farinheira	40	0.2990	-0.1488	-0.0099	-0.0439	0.4125	0.1669	yes
chourico	41	0.2921	-0.0041	-0.0029	-0.068	0.2352	0.0597	yes
morcela	41	0.2970	-0.0033	-0.0029	0.0822	0.2140	0.1279	yes
farinheira	41	0.3066	-0.0021	-0.0068	-0.0159	0.2094	-0.0619	yes
chourico	42	0.2867	0.1456	-0.0089	0.0014	0.2796	-0.0842	no
morcela	42	0.2848	0.1452	-0.0099	0.0634	0.1450	0.0072	no
farinheira	42	0.2784	0.1446	-0.0139	0.0872	0.3659	-0.1543	yes
chourico	43	0.2994	-0.1567	-0.008	0.1686	0.2125	1.5194	yes
morcela	43	0.3042	-0.1507	-0.0089	0.2552	0.2532	1.5253	yes
farinheira	43	0.2969	-0.1382	-0.0098	0.3267	0.2838	1.5339	yes
chourico	44	0.2901	-0.0006	-0.0059	0.1404	-0.2084	1.7017	yes
morcela	44	0.2888	-0.0059	-0.0049	0.087	-0.2569	1.6210	yes
farinheira	44	0.2906	0.0014	-0.0069	0.1065	-0.2779	1.7920	yes
chourico	45	0.2815	0.1472	-0.0099	0.2320	-0.2132	1.6783	yes
morcela	45	0.2835	0.1501	-0.0099	0.2031	-0.2505	1.6501	yes
farinheira	45	0.2840	0.1440	-0.0109	0.1180	-0.3626	1.7381	yes
chourico	46	0.2984	-0.1496	-0.0099	0.1946	-0.1809	1.5743	yes
morcela	46	0.2999	-0.1444	-0.0088	0.1297	-0.2191	1.5183	yes
farinheira	46	0.3016	-0.1515	-0.0118	0.2566	-0.2028	1.5095	yes
chourico	47	0.2901	0.0022	-0.0059	0.1947	0.1886	1.5641	yes
morcela	47	0.2917	0.0015	-0.0039	0.2019	0.1157	1.5520	no
farinheira	47	0.2938	-0.0125	-0.0049	0.1436	0.2304	1.5375	yes
chourico	48	0.282	0.1462	-0.0079	0.1231	0.1561	1.5678	no
morcela	48	0.2853	0.1434	-0.0049	0.2501	0.1321	1.5604	no
farinheira	48	0.2845	0.1437	-0.0089	0.2455	0.2702	1.488	yes

A.4 Values obtained during the fourth test

Product	Pose	X	Y	Z	Roll	Pitch	Yaw	Success
chourico	4	0.3126	-0.1454	0.0032	-0.0002	-0.3672	-0.1927	no
morcela	4	0.3127	-0.1471	0.0042	-0.0853	-0.1729	-0.1077	yes
farinheira	4	0.3059	-0.1475	0.0001	-0.3546	-0.1982	0.8926	no
chourico	5	0.3012	-0.0008	0.0082	0.0695	-0.2678	-0.1183	yes
morcela	5	0.3013	0.0013	0.0091	-0.0554	-0.3163	0.1084	yes
farinheira	5	0.2977	-0.0024	0.004	-0.0441	-0.4122	0.368	yes
chourico	7	0.4148	-0.1382	-0.0008	0.0034	-0.3048	-0.1102	no
morcela	7	0.4100	-0.1382	0.0013	-0.1221	-0.2436	0.0684	yes
farinheira	7	0.4100	-0.1407	-0.0057	-0.3726	0.3119	2.2769	no
chourico	8	0.3989	0.0038	0.0042	0.051	-0.2482	0.0139	yes
morcela	8	0.4023	0.0037	0.0025	-0.0898	-0.3202	0.1734	no
farinheira	8	0.4016	0.0045	0.0003	-0.1414	-0.4300	0.3469	yes
chourico	16	0.2966	-0.1534	-0.0009	0.0724	0.3326	0.8000	yes
morcela	16	0.2958	-0.1544	-0.0049	0.2143	0.3120	0.8214	yes
farinheira	16	0.3071	-0.1405	-0.0049	0.059	0.5448	0.8187	yes
chourico	17	0.2830	-0.0094	0.0001	-0.0467	0.1655	0.7114	yes
morcela	17	0.2866	-0.0087	0.0011	0.0461	0.2920	0.7498	yes
farinheira	17	0.2886	-0.0048	-0.0029	0.1278	0.4574	0.7480	yes
chourico	19	0.3939	-0.1458	-0.0038	-0.0883	0.3372	0.9035	yes
morcela	19	0.3939	-0.1542	-0.0038	0.4644	0.1995	0.9265	no
farinheira	19	0.4027	-0.1433	-0.0108	0.1923	0.4215	0.8956	yes
chourico	20	0.3827	-0.0019	0.0002	0.031	0.3347	0.7506	yes
morcela	20	0.3863	-0.0044	0.0012	0.1844	0.3346	0.7570	yes
farinheira	20	0.3883	0.0022	-0.0038	0.1023	0.4840	0.7921	yes
chourico	28	0.3014	-0.146	-0.0019	-0.0605	0.3090	1.4691	yes
morcela	28	0.3049	-0.1642	-0.0049	0.0021	0.2111	1.5314	no
farinheira	28	0.3085	-0.1475	-0.0048	0.358	0.4457	1.6782	yes
chourico	29	0.2925	-0.0123	-0.0009	-0.13	0.2434	1.2865	yes
morcela	29	0.2949	-0.004	-0.0028	0.2156	0.3049	1.4179	yes
farinheira	29	0.2915	-0.0035	-0.0059	0.2972	0.3520	1.4965	no
chourico	31	0.3993	-0.1451	-0.0068	-0.0643	0.2235	1.3325	yes
morcela	31	0.3994	-0.1391	-0.0048	-0.1767	0.2471	1.4389	yes
farinheira	31	0.3992	-0.1437	-0.0118	0.3047	0.3990	1.6206	yes
chourico	32	0.3883	-0.0138	-0.0028	-0.0613	0.0320	1.3597	no
morcela	32	0.3924	0.0016	-0.0018	0.0123	0.3099	1.3689	yes
farinheira	32	0.3979	0.0044	-0.0058	0.3432	0.3237	1.6288	no





2025

Robotic manipulation of soft-tissues in agri-food processing

Diego Fino

