INSTITUTO POLITÉCNICO DE BRAGANÇA ipb Escola Superior de Tecnologia e Gestão

UTFPR UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Hands Tracking and Fuzzy Speed Control to Improve Human-Robot Collaboration

Alexandre Carelli Borsoi

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering.

Work oriented by:
Prof. Dr. Paulo Leitão
Prof. Dr. Luiz Amilton Pepplow

Bragança
2021/2022

INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

Hands Tracking and Fuzzy Speed Control to Improve Human-Robot Collaboration

Alexandre Carelli Borsoi

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering.

Work oriented by:
Prof. Dr. Paulo Leitão
Prof. Dr. Luiz Amilton Pepplow

Bragança
2021/2022

Dedication

I dedicate this work to my parents, for all the support and love they have given me throughout my life. . .

Acknowledgement

I would like to thank everyone who helped me in some way, who made it possible for me to finish this thesis.

First of all I would like to thank my family, who supported me a lot in this journey and without them, it would not have been possible. I thank my mother Silvia Carelli Borsoi, my father Sergio Antonio Borsoi, and my sister Julia Carelli Borsoi. I also thank my grandparents Feliz Carelli and Flavia Chies Carelli for all their affection.

I also thank my supervisor Prof. Dr. Paulo Leitão, who gave me the opportunity to carry out this project, and helped me along the way.

I also thank Luis Fernando Piardi, for all the attention and help during the development of this thesis.

I also thank Prof. Luiz Amilton Pepplow for being my co-supervisor in this work.

To all my friends I want to express my gratitude for being part of my life, and always encourage me to continue, I want to thank especially my friends Andre Luis França, Rebeca Baron Kalbermatter, Nicoli Jacon, Larissa dos Santos Paulo, your support was essential for me to complete this project.

I also thank UTFPR and IPB for this unique opportunity.

Finally, to all of you, my sincere thanks.

Abstract

The demand of collaborative robots has been growing in the industry in general, and with it the need for new ways to improve and make this work environment between human and robot safer and efficient. The objective of this work is to improve and make this environment safer and efficient by controlling the robot's speed using a fuzzy approach and by getting track of the hand of the operator. For this purpose, the UR3 robot from Universal Robots and Leap Motion was used, which is a sensor capable of detecting the hand, as well as its movements, with the data obtained it was possible to create a system that has the robot's speed as an output through fuzzy logic, and using the distance between the hand and the gripper obtained from the Leap Motion and UR3 data respectively as input to the fuzzy logic. With this it was possible to achieve satisfactory speed control, moreover, in all the tests performed the approach proved to be able to avoid collisions, and with the testing of different defuzzification methods in the fuzzy control, it was also possible to achieve smooth speed control for some of the methods used, with this in mind the system showed promise for improving Human-Robot Collaboration.

Keywords: Human-Robot Collaboration; Human-Robot Interaction; Collaborative Robotics; Hand Tracking; Fuzzy Controller.

Resumo

A procura de robôs colaborativos tem crescido na indústria em geral, e com ela a necessidade de novas formas de melhorar e tornar este ambiente de trabalho entre o ser humano e o robô mais seguro e eficiente. O objetivo deste trabalho é melhorar e tornar este ambiente mais seguro e eficiente, controlando a velocidade do robô através de uma abordagem fuzzy e da localização da mão do operador. Para o efeito, foi utilizado o robô UR3 dos Universal Robots e do Leap Motion, o qual é um sensor capaz de detectar a mão, bem como os seus movimentos. Com os dados obtidos foi possível criar um sistema com a lógica fuzzy, tendo como saída a velocidade do robô e a entrada a distância entre a mão e a garra, obtida pelos dados do Leap Motion e do UR3, respectivamente. Com isto foi possível obter um controlo de velocidade satisfatório, além disso, em todos os testes realizados a abordagem provou conseguir evitar colisões, e com o teste de diferentes métodos de defuzzificação no controle fuzzy, também foi possível alcançar um controle suave da velocidade para alguns dos métodos utilizados, com isto em mente o sistema mostrou-se promissor para melhorar a Colaboração Humano-Robot.

Palavras-chave: Colaboração Humano-Robo; Intereção Humano-Robo; Robotica Colaborativa; Rastreio das mãos; Controlador Fuzzy.

Contents

List of Tables

List of Figures

Acronyms

API Application Programmer Interface.

BICom Back-Input Compensation.

CeDRI Research Center in Digitalization and Intelligent Robotics.

EMG Electromyography.

GMO Generalized Momentum Observer.

HPF High-Pass Filter.

HRC Human-Robot Collaboration.

HRI Human-Robot Interaction.

IR Infrared.

LoM Largest of Maxima.

MFs Membership Functions.

MoM Mean of Maxima.

R-CNN Region-based Convolutional Neural Network.

SDK Software Development Kit.

SoM Smallest of Maxima.

STC Safety Through Control.

STP Safety Through Prediction.

Chapter 1

Introduction

Throughout history, humanity has undergone several transformations in its economic structure due to major technological changes in the production of goods, such as the industrial revolutions. The first of these was marked by the emergence of the steam engine and the mechanization of work, while the second brought electricity, production lines, and large-scale manufacturing. The third revolution was marked by the emergence of computers and the internet, and the beginning of automation [1]. Currently, the industrial revolution is in its fourth phase, also called industry 4.0 [2]. This is a German term that emerged in 2011, which is defined by the use of highly technological strategies to create highly automated industries through Human-Robot Interaction (HRI), that is, the purpose of industry 4.0 is to try to bring a smarter, more efficient and more accurate industrial technology [3].

Among the technologies related to Industry 4.0, this thesis has a major focus on collaborative robotics. To understand it first it is necessary to understand that robotics in general can be divided into three directions. The first is industrial, where the more traditional robots are found. These are reprogrammable, multipurpose industrial manipulators. The second direction is service robotics. The third is the collaborative robotics [4]. An example from the first direction are the robots shown in Figure 1.1, where they are being used in the automotive industry, in order to assemble and weld car body parts.

A service robot is defined by ISO/TC 299 as a robot that performs useful tasks for

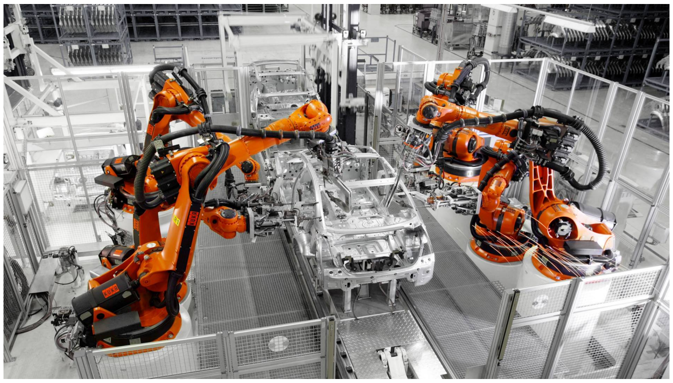


Figure 1.1: Industrial Robot in Automotive Industry [5].

humans or equipment, excluding industrial automation-related applications. In Figure 1.2 it is possible to observe an example of this direction, being this cleaning robot Cleanfix - RA 660 from ©BlueBotics.

Figure 1.2: Service Robot Cleanfix - RA 660 of ©BlueBotics [6].

In the Figure 1.3 it is possible to see a collaborative robot that was implemented at Scott Fetzer Electrical Group, this robot being an example of the second direction of robotics.

The collaborative robots or cobots objective is not to replace the human being on the production line, but to work side by side [8], [9], thus uniting the ability of a robot to perform tasks with greater speed, precision, repeatability and strength with cognitive skills, flexibility, intelligence, decision-making and the ability to act in unexpected situations provided by a human [8], [10]. Most applications of collaborative robots are in the automotive manufacturing industries, where cobots perform a variety of tasks ranging from

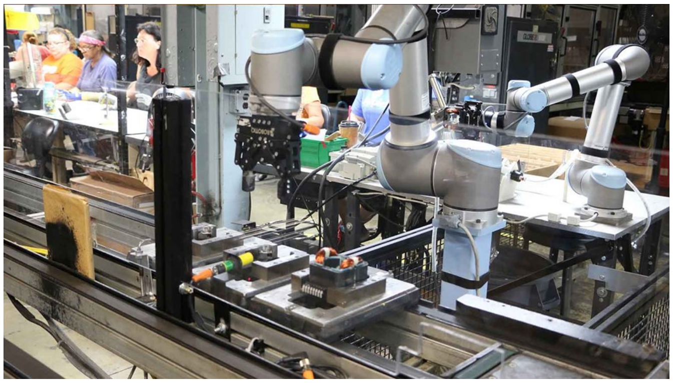


Figure 1.3: Collaborative Robot [7].

picking, packing and palletizing, welding, assembling items, handling materials, product inspection [8].

Cobots are also lighter, offer great mobility, can be used to perform a wide variety of tasks, thus allowing them to be used in a wide range of industries and are also easier to program compared to large industrial robots [8]. However, due to this great interaction between human and robot, several challenges in how to guarantee safety and efficiency in this shared work environment arise [4]. One of these challenges is the unpredictability and the difficulty of accurately and at runtime obtaining the exact position of the operator, which can collide, creating a dangerous situation around the robot [11].

1.1 Objectives

The objective of this thesis is to develop a system capable of identifying the position of the operator's hands in real time, in order to avoid collisions and improve the Human-Robot Interaction (HRI), improving the safety and efficiency between a collaborative robot and collaborators. To do this, image processing systems will be used to detect hand position and a fuzzy control algorithm will be developed to control the speed of a collaborative robot.

To achieve this goal, the work must guarantee the following conditions:

- Develop fuzzy control algorithm with low response time, if the delay of the algorithm response is too high it may not be able to avoid the collision.

- A robust hand tracking, being able to detect the hand at every angle on the bench. Blind spots in this collaborative work environment can lead to accidents, since the sensor will not be able to track the operator in these areas.
- Ensure safety in industrial environments, guaranteeing the safety of the collaborator, as well as for other possible workers that may be in the vicinity of this environment, not being able to harm them in any way.
- Identify all the hands present at the collaborative workbench, both for safety and efficiency, failure to detect all the hands present can lead to accidents, as well as hinder the collaboration between human and robot.
- Identify which hand is closer to the robot, in case of more than one hand present on the robot bench, since this hand is in a more dangerous situation (higher chance of collision) compared to the other more distant hands.

The major contribution of this thesis is an algorithm development that can increase the efficiency and safety of the system by improving the HRI, through fuzzy speed control and hand position detection.

1.2 Document Structure

This document is organized in 6 chapters. After this introductory chapter, Chapter 2 consists of a review of the literature regarding collaborative robotics, as well as the related works and the study tools used. Chapter 3 presents the description of the case study, which shows the environment where the system proposed in this thesis will be developed. Chapter 4 contains the methodology and development of the project. While chapter 5 contains the experiments performed to validate this solution as well as the results found. Finally, the last chapter presents the conclusions and suggestions for future work.

Chapter 2

State of the Art and Study of Tools

This chapter brings the concepts of collaborative robotics and fuzzy logic and the works related to these areas. In the first part there is a review of the literature on collaborative robotics, followed by related work, and in the second part a review of fuzzy logic, and finally related work.

2.1 Collaborative Robotics

More and more industrial robots are being deployed in industry to replace or assist humans in hazardous and repetitive activities [9], having an increase of approximately 31% in new installations in 2021 compared to 2020 [12]. With this also grows the need for humans not only to share the same work environment with robots, but also to work with them as collaborators in the industrial environment. Robots used for this purpose are called industrial collaborative robots or simply cobots [9].

The cobots are designed for easy programming [8], [13], because they have a friendly interface and simple and intuitive programming language, facilitating their use by people with no prior knowledge in programming [14]. Compared to industrial robots, collaborative robots offer greater productivity, flexibility, versatility, and safety [8]. In addition to a lower installation and maintenance cost compared to traditional robots [13], [14], but these advantages over industrial robots come at the cost of reduced load capacity and speed, in order to avoid crushing and drilling, as well as to avoid collisions [13], [15].

Other advantages of cobots are related to Human-Robot Collaboration (HRC), as it unites the precision, repeatability and speed of a robot, with the intelligence, flexibility and ability to act in unexpected situations of a human being [16]. Sherwani, Asad, and Ibrahim [8] bring in a table, which can be seen in Figure 2.1, a comparison between human and robot capabilities, to show that the robot is designed to complement human weaknesses in certain things.

Humans		**Robots**	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
Dexterity	Weakness	Strength	No Process Knowledge
Flexibility	Fatigue	Endurance	Lack of Experience
Creativity	Imprecision	Precision	Lack of Creativity
Decision Making	Low Productivity	High Productivity	No Decision Power

Table 2.1: Comparison Between Humans and Robots. Adapted from [8].

To ensure a good HRC, it is necessary that the HRI is safe and efficient [9]. In [9], Abdelfetah et al. brings challenges and constraints that need to be studied and developed in order to ensure better HRI, these being:

- *Obtain robust detection of human motion in order to build good predictive systems.*
- *Achieving robust detection that can identify when contact between robot and human occurs.*
- *Develop responsive controllers capable of rapidly replanning the robot's trajectory in a complex and unpredictable environment in real time.*

Next, some works related to the challenges presented by Hentout, Aouache, Maoudj, *et al.* [9] will be briefly presented, in order to verify what has already been developed on these issues.

Regarding the route planning, Dumonteil, Manfredi, Devy, *et al.* [17] uses in its work reactive planning and a Kinect sensor to avoid collision trajectories and detect obstacles on the way, respectively. In simplified form, the operation happens as follows: if the robot is executing a path and encounters a new obstacle, it will request a new route, thus avoiding the collision. In [18] the researchers develop a method to plan paths for robotic arms based on computer vision, Q-learning and neural networks, where the system works as follows, first it was trained to detect three 3D shapes, a sphere, a pyramid and a cube, where they would be the starting point, obstacle and target for the robotic arm, respectively. Having the coordinates of these objects, the Q-learning algorithm determines the sequence of actions that the arm must take, and finally a trained neural network converts this sequence of actions into the respective joint angles, quickly replanning the robot's trajectory in a 3D environment and in real time.

As for collision detection, Heo, Kim, Lee, *et al.* [19] propose a new collision detection framework called CollisionNet, which is based on deep learning. In this project, the researchers trained the neural network with 160,055 samples from 210 collisions, which contain high-dimensional signals from the robot joints. Thus, it was possible to achieve high collision sensitivity and robustness to false-positive detection induced by noisy signals and model uncertainties. In [20], a Back-Input Compensation (BICom) system is proposed in countermeasure to Generalized Momentum Observer (GMO) collision detection techniques that are widely used in cobots and High-Pass Filter (HPF) based detection techniques. Simply put, GMO are techniques that detect impact using information from robot joints [21]. Because GMO is somewhat of a low-pass filter, it does not have much delay to detect sharp collisions. HPF-based methods are able to detect sharp collisions almost instantaneously, but are not sensitive to quasi-static contacts. BICom proved to be able to efficiently detect both soft and hard collisions, hard collisions are when a sharp impact occurs, while soft collisions are linked to pull, push and grab movements. In addition to also being able to monitor quasi-static contact, and also to decrease the number of false positives compared to the methods [20].

On the subject of human detection, Kang, Kim, and Kim [22] have developed a collaborative robot workplace safety monitoring system, to this end the researchers use depth sensors to acquire images of the workspace, and use a neural network model to identify the worker and the robot. From the detected information, the system estimates the distance between the worker and the robot, and the classic in three levels:

- Worker safety area.
- Worker attention area.
- Dangerous worker area.

In the first level nothing happens to the robot, in the second level it is slowed down, and finally in the hazardous area it is stopped. In this way, the safety of the operators is increased by controlling the robot in dangerous situations. In [23], meanwhile, the researchers present a control strategy that allows a cobot UR3 to be triggered to perform the task of object transfer between an operator's hand and the robot's tool. To this end, the researchers use multiple Microsoft Kinects sensors, to acquire a dynamic skeleton of the hand. After detection, the skeleton data is sent to an algorithm that plans a route for the robot tool to stop at an opportune position to collect the object, ensuring safety, improving efficiency, and making the HRC more dynamic. Different from the paper Scimmi, Melchiorre, Mauro, *et al.* [23], Aleotti, Micelli, and Caselli [24] aiming to improve also the HRC are not only concerned with transferring the object to the user, but transferring the object with comfort. To accomplish this 3 processes are performed, where in the first one an eye-in-hand laser scanner sensor is used, which first detects what is the shape of the object to be handed to the operator, and then the object image is processed in order to verify what is the part of the object (known a priori) that will be pointed to the operator in order to guarantee the best comfort when picking up the object. In the second part a fixed depth sensor (kinect) is used, which detects when a human approaches the table where the robot is fixed and also consists in the route planning part, which consists in finding the claw's closing force and planning the trajectory so that the object is delivered in a comfortable configuration for the user. The last step consists in verifying that the user has caught the object after the robot finishes the trajectory to open the gripper. This is done using an algorithm that uses the kinect data to detect the collision.

In work [25], a workbench composed of a collaborative robot was used, with a force sensor coupled to the cobot grip and camera to perform quality control in small electronic devices. The collaborative robot presses buttons on the device and, through a machine learning algorithm, identifies whether the buttons are working or not. In addition, through visual inspection based on camera images, the system detects failures in the displays of electronic devices. As a result of the work, the authors identified that the robot's collaboration in the quality control task reduces errors caused by repetition or fatigue of the operator.

In the work Kaplanoglu, Nasab, Erdemir, *et al.* [26], Kaplanoglu, Nasab, Erdemir, *et al.* [26] uses detection to develop a way to control a robotic arm through hand movements, in order to improve HRC other than through predictive systems. The hand gestures are detected through the MYO bracelet, which is a bracelet that uses Electromyography (EMG) to identify the gestures. EMG is the process of measuring the electrical activity caused by muscles during their contraction. The gestures are sent to a Raspberry Pi, where data processing occurs, resulting in a movement that the robotic arm must perform In order to verify the effectiveness of this control method, the researchers simulated an assembly line. First, the process was done manually, and three different people were used for the test, who did everything individually. After this, the same simulation was performed, but with the help of the robot, and finally the assembly was carried out in a fully automated way, in order to verify the assembly time for each situation. The subjects working together with the robot resulted in a 40% decrease in assembly time compared to the subjects without the help of the robot, and compared to the robot alone there was only an approximate 8% decrease [26]. However, bearing in mind that there are some tasks that may be too complex to be fully achieved by traditional robots or too expensive to be fully automated [9]. In this way, the authors have shown the efficiency of the proposed solution. In [27], the authors complement a Faster Region-based Region-based Convolutional Neural Network (R-CNN) gesture recognition system to improve performance over the previous project [28]. R-CNN is a deep convolutional network used for object detection. For this, the researchers trained the convolutional neural network to detect four simple and different gestures in order to test the capabilities of the Faster R-CNN method. The new solution implemented by the researchers proved promising with a low inference time, which is suitable for real-time applications [27].

In addition, there are also projects being done to improve safety and productivity through speed control. In [29]. The researchers develop a method and speed control that satisfies both safety and productivity in the collaborative work environment by applying the maximum safe speed allowed. In order to find the maximum safe allowed speed, the researchers use a layered collision model to calculate the impact pressure/force using the pressure and force limit specified in ISO/TS 15066 as a basis. In this way, the researchers are able to keep the robot's speed at the maximum possible within the control zone that does not exceed the force and pressure limits in an impact with a human.

2.2 Fuzzy Control Systems

Fuzzy logic is a technique was first proposed in 1965 by Zadeh [30] in his publication . According to Lanzillotti, Lanzillotti, and SINTZ [31], unlike the Boolean logic, where 0 indicates false and 1 true, fuzzy logic can simply assume any real value between 0 and 1, in other words, it is a logic that was extended to deal with the concept of partial truth, seeking to imitate human reasoning Ahlawat, Gautam, Sharma, *et al.* [32]. In this way, this concept can be used to develop a controller through human experience. Where the main advantage is the possibility to implement through experience, intuition and risk, and the fact that it is not necessary to have a model of the process [33]. That is, fuzzy control logic is nothing more than a form of control to control complex process systems through human experience. Unlike conventional (nonfuzzy) where the control system is designed with the help of physical models of the process [34].

In 1975 the first fuzzy controller applied successfully by Mamdani and Assilian in a laboratory environment. Then occurred its first industrial application of fuzzy controllers by Holmblad and Østergaard, two civil engineers, around 1980, the company F.L. Schmidt for the control of cement kilns [35]. Since its first publication, fuzzy logic has stood out among researchers due to its versatility and expressiveness.

2.3 Fuzzy Solutions in Collaborative Robotics

With the insertion of collaborative robotics in the industrial environment, creates situations of risk to humans and the need to mitigate these risks, identifying the hazards in their nature and severity among other parameters to apply risk assessment algorithms based on fuzzy logic. This being a tool capable of dealing with subjectivity and complex issues through mathematical methods and analysis and probabilistic [36] [37].

For such estimates it is necessary to obtain information about the environment, work space, task being executed as well as the behavior of the users obtained through cameras and sensors that are arranged by the work environment of this robot. The application of this method, or even together with artificial neural network algorithms as produced by Beltran, Diwa, Gales, *et al.* [38], proved to be capable of controlling and managing the speed and the movement vectors of the robotic manipulators, obtaining satisfactory results and providing risk reduction associated to the operation of these robots [38] [37] [39].

In the paper [40] the fuzzy approach ensured greater efficiency in dealing with the randomness of the system, where fuzzy logic was used in the schedule of tasks performed between an operator and a cobot, which due to the unpredictable nature of the operator and its variability in time to complete the same task, in addition to the possible interactions that it could do with the robot, which makes the cycle time non-deterministic, but with the use of fuzzy logic to model this system, there is a decrease in the prediction of the temple of cycle and compared to linear modeling methods [40]. The work also presents the proposal of a new fuzzy approach to reduce and quantify the risk in human-robot interaction.

In the work [41] the author combines two methods, the Safety Through Prediction (STP), which consists of using a depth camera and thermals for a reliable detection of a human operator, together with the Safety Through Control (STC) that proposes a fuzzy inference system capable of dynamically computing the shortest protective distance according to a risk analysis based on perceptual data. The combination of these two methods allowed the system to be more productive, minimizing robot downtime, while still ensuring safety.

2.4 Python

Python is a high-level programming language [42], which was designed with emphasis on the work of the developer. For it has a simpler, cleaner and more readable code writing, making it easier to write code for smaller applications as well as more complex programs [43].

Python is used in a variety of applications, ranging from website development, scientific computation, software development and games [44]. Currently, there are two series of versions of Python, being 2.x and 3.x, where version 2.x is older, its support and maintenance ended on January 1, 2020 [45], Version 3.x consists of a redesign of version 2.x, which is actively supported [44].

2.4.1 PyCharm

PyCharm is a Python Integrated Development Environment (IDE), that is software for building applications that combines common developer tools into a single graphical user interface (GUI). The objective is to simplify software development and facilitate the identification and minimize coding mistakes and typos [46].

Segundo Gayratovich [47]: PyCharm is an indispensable modern environment for Python programming language.

.

Chapter 3

Case Study Description

With the ever-increasing advancement of technology involving collaborative robotics, these types of robots are being used more and more [9]. Thus, there is an increasing amount of workspace sharing between humans and robots. An example of this is the bench in Figure 3.1.

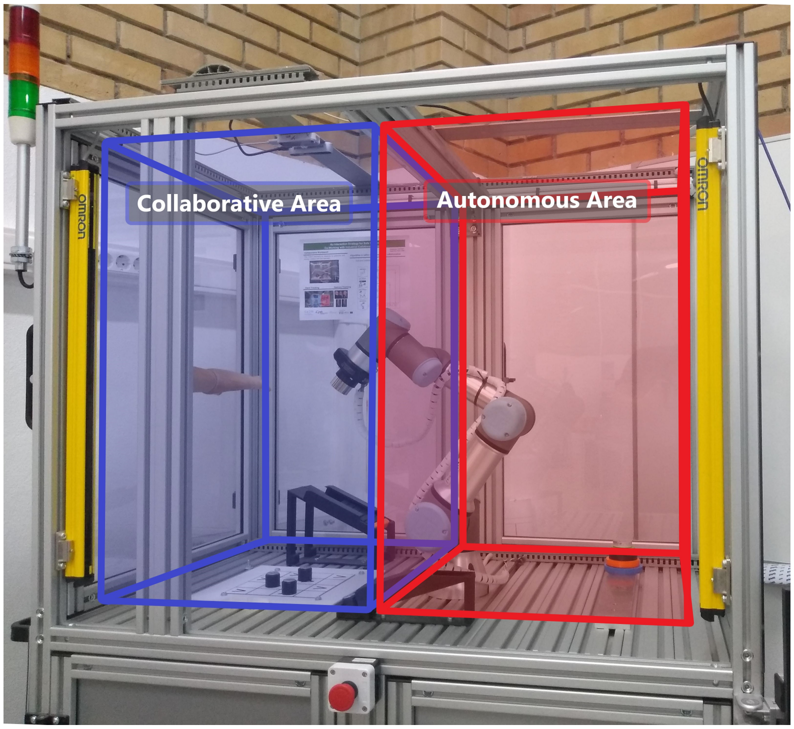


Figure 3.1: Workbench Division.

The workbench in Figure 3.2 can be used for applications that require Pick and Place of objects, as well as for part assembly and quality inspection, for example. Currently, the bench is located at IPB in the laboratory of the Research Center in Digitalization and Intelligent Robotics (CeDRI). In which it is used with the intention of similar collaborative working environment between man and cobot.

As observed in Figure 3.1, this bench is divided into two spaces, one for collaborative work, and one autonomous space. In the collaborative space the human and the robotic arm share the space, being able to perform activities simultaneously, and the other space is dedicated to the autonomous activities of the robot.

The workbench consists of a UR3, which is a collaborative robotic arm (cobot) from Universal Robots, the Leap Motion which is an optical hand tracking sensor, and a light curtain safety sensor. All the components can be seen in Figure 3.2.

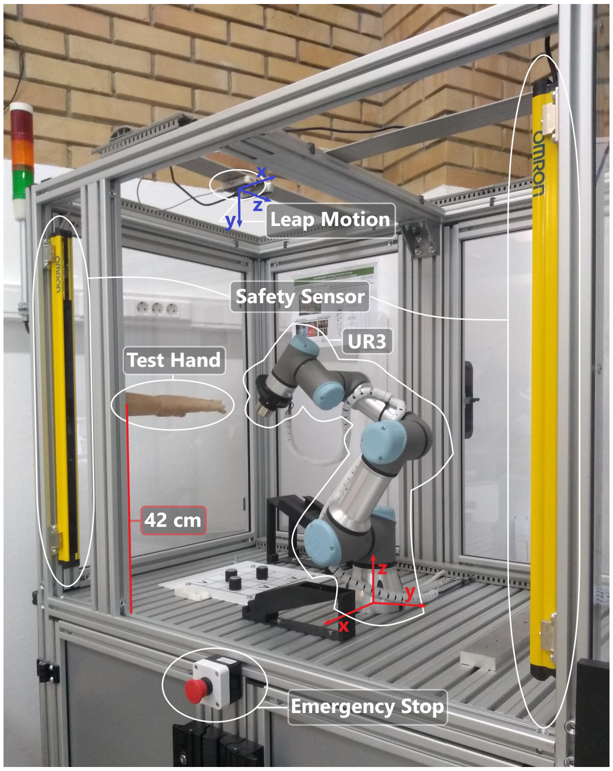


Figure 3.2: Arrangement of the Components.

3.1 Collaborative Robot Arm UR3

Unlike a traditional robotic arm, the objective of cobot is not to replace the human being in the industry, but to work side by side with them, thus uniting the precision, repeatability and speed of a robot, with the intelligence, flexibility and ability to act in unexpected situations of a human being, in order to obtain better productivity and industrial quality [8], [9], [16].

The UR3 is a collaborative robot arm from Universal Robots, with the ability to lift up to 3 kg, 500 mm reach and 6 rotating joints, it is a robot to perform light tasks, such as assembly, and also for jobs that require a lot of precision [48]. The cobot is equipped with a magnetic gripper and a Wrist camera model Q-00104, where the gripper with camera assistance can be used for activities such as Pick and Place with metal objects.

3.2 Leap Motion Controller

Leap Motion sensor is an optical and infrared light hand tracking sensor, which can capture hand and finger movement, that was developed by Leap Motion, Inc.

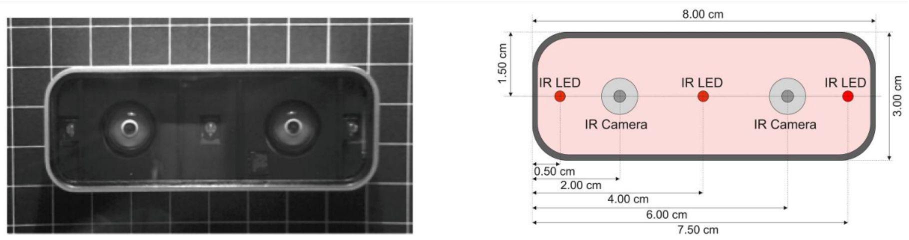


Figure 3.3: Leap Motion Controller Schematic [49].

The sensor in conjunction with its Application Programmer Interface (API) can provide the position of hands and fingers with good accuracy in a Cartesian plane relative to the center point of the Leap Motion Controller, which is located on the second Infrared (IR) LED, where it is centered, as illustrated in Figure 3.3. It has two IR cameras, with 640 x 240 pixels. It can track hands from a depth of 10 cm, up to a maximum of 80 cm, and with a 140×120° field of view.

3.3 OMRON Safety Light Curtain F3SG

A light curtain sensor consists of two parts, a receiver and a transmitter, where the transmitter contains multiple LEDs that send IR pulses to the receiver, while the receiver waits to receive these pulses at the right frequency and time. If an object, such as a hand, passes between the transmitter and receiver, it will cause interference to be generated in the signal, causing the receiver to stop receiving pulses, and send a trigger signal to a machine or device connected to it. An example of this can be seen in Figure 3.4.

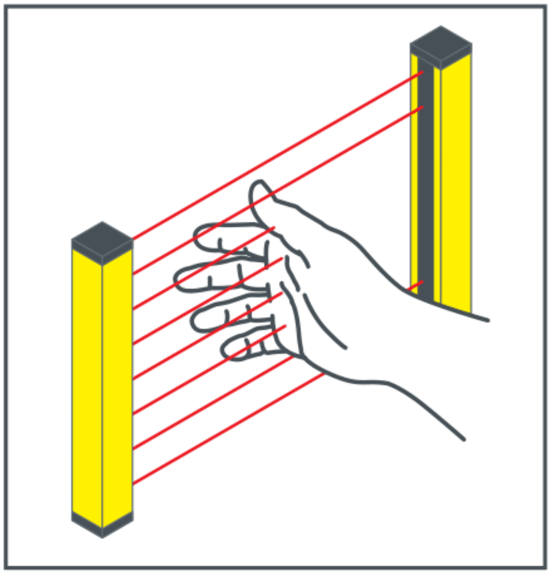


Figure 3.4: Example of Interference with the Light Curtain Sensor [50].

On the workbench, the sensor used is the OMRON Safety Light Curtain F3SG-2RE0750P30 model, which is specific for hand protection. Some of its specifications are that the safety field reaches a height of up to 750 mm and can cover up to a maximum distance of 20 meters between the receiver and the transmitter. This sensor also has a response time of only ms, which is the time it takes for the receiver to trigger when it detects an object between the receiver and the transmitter.

Chapter 4

Methodology and Development

There is a growing need for systems that ensure the safety of the human being, who works side by side with the machine, that is, a system capable of dealing in real time with human unpredictability in order to ensure safety and efficiency in the workplace.

Therefore, the approach adopted in this study in order to improve the HRI, consists of implementing a fuzzy control of the robotic arm speed. Where this in turn has as input the Euclidean distance between the operator's hand and the robot's claw, the robot's speed, and whether the claw is approaching or moving away from the hand.

The use of fuzzy logic happens, because it is a powerful tool to deal with the unpredictability characteristic of the real world [36], because it allows to represent models that contain a certain degree of uncertainty or imprecision. Thus, fuzzy control was used to deal with the unpredictability of human-machine interaction.

The speed control, on the other hand, is done to avoid possible collisions and thus increase the operator's safety. The better the speed control, the better the efficiency of the system.

Python was used as the programming language for Leap Motion, UR3 and also to develop the fuzzy controller. The Python version used in this project is 2.7.3 because versions 2.x and 3.x are not fully compatible, and the latest Leap Motion Software Development Kit for Python is developed for version 2.7.3. To obtain statistical data from the acquired data set and to make box diagrams.

4.1 System Architecture

In this section the architecture of the system will be presented, using the block diagram contained in Figure 4.1 and each block will be explained.

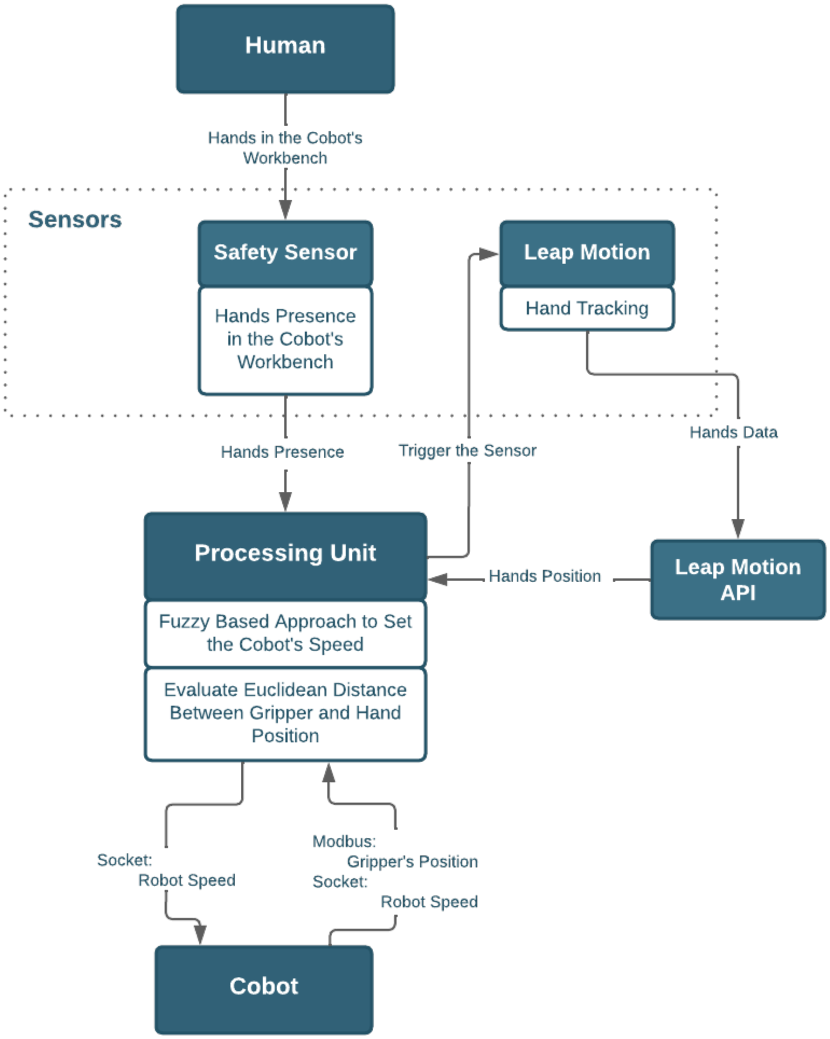

Human
Hands in the Cobot's Workbench
Sensors
Safety Sensor
Hands Presence in the Cobot's Workbench
Leap Motion
Hand Tracking
Hands Presence
Trigger the Sensor
Hands Data
Processing Unit
Hands Position
Leap Motion API
Fuzzy Based Approach to Set the Cobot's Speed
Evaluate Euclidean Distance Between Gripper and Hand Position
Socket: Robot Speed
Modbus: Gripper's Position
Socket: Robot Speed
Cobot

Figure 4.1: System Architecture.

The system starts when the security sensor detects the presence of one or more hands between the receiver and the transmitter. The sensor provides this information to the processing unit. From this the processing unit will trigger Leap Motion, where from its API it can extract the position of the hands, which will be sent to the processing unit.

The processing unit will also extract from the UR3's Modbus server the position of the robot's claw, as well as its speed via socket. Modbus is a serial communication protocol developed by Modicon in 1979. Modbus is a method used to transmit information over serial lines between electronic devices. The device requesting or writing information can be called the Modbus client, and the device providing or receiving this information is referred to as the Modbus server [51]. With the data acquisition, the processing unit can make the necessary calculations to finally provide the robot arm with the new speed. In the flowchart present in Figure 4.2, it is possible to observe in more detail how the system works.

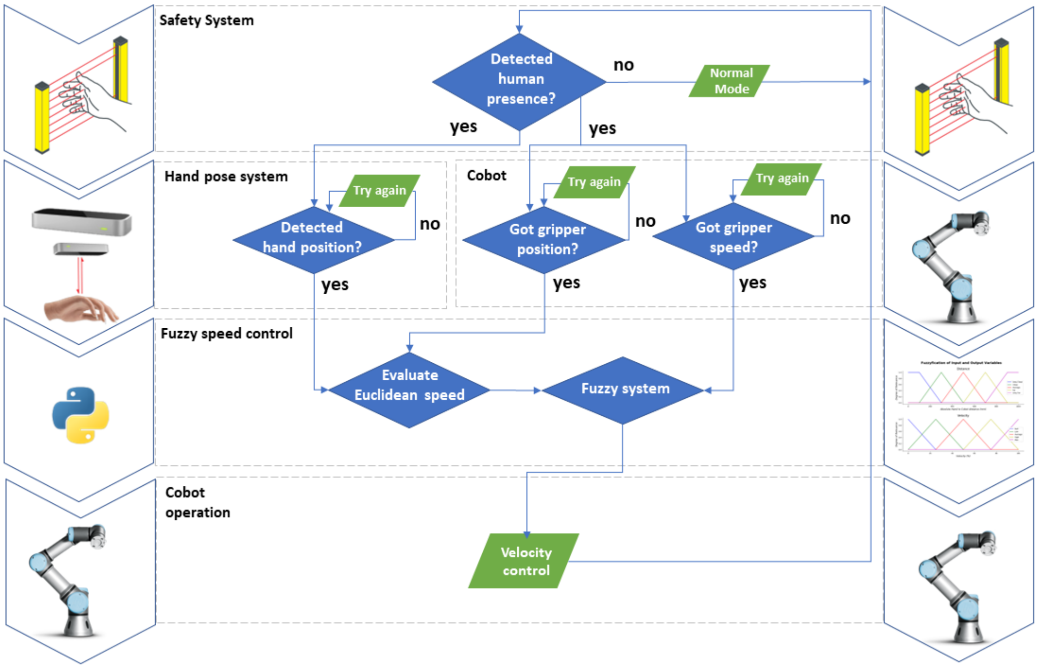


Figure 4.2: Flowchart Showing the Simplified Operation of the System.

4.2 Safety and Hand Tracking Based on OMRON and Leap Motion

Two sensors were used, one being the OMRON Safety Light Curtain F3SG which is called the "Safety Sensor" in the diagram in Figure 4.1 and the Leap Motion Controller. The safety sensor as mentioned earlier is responsible for detecting the presence of the hand(s) on the robot workbench. As soon as the hand is detected by the safety sensor, it sends a signal to the processing unit. For this, it was necessary to develop a routine in Python that uses ModBus communication. The functioning of the code can be observed by the flowchart in Figure 4.3.

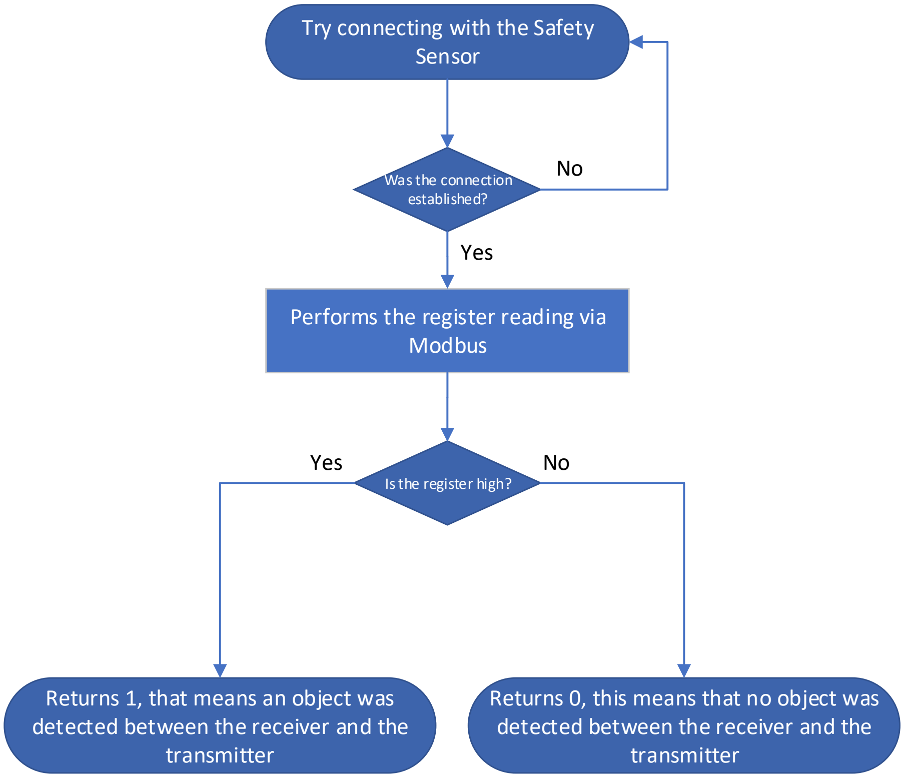

Try connecting with the Safety Sensor
Was the connection established?
No
Yes
Performs the register reading via Modbus
Yes
No
Is the register high?
Returns 1, that means an object was detected between the receiver and the transmitter
Returns 0, this means that no object was detected between the receiver and the transmitter

Figure 4.3: Flowchart of the Safety Sensor Routine.

The Leap Montion controller together with its API is responsible for tracking the hand in real time and providing the respective data to the processing unit. For this, a Python code was developed using the Software Development Kit (SDK) provided by Leap Motion, Inc. Using this SDK made it possible to develop a class in Python that contains a function with the ability to collect in real time the position of the operator's hand. You can see how the function works by looking at the flowchart in Figure 4.4.

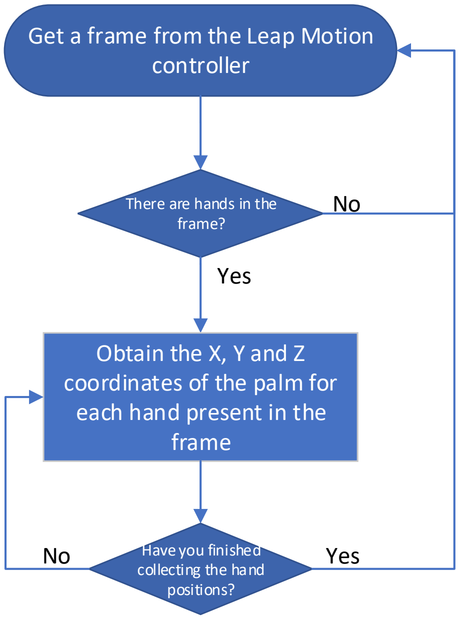
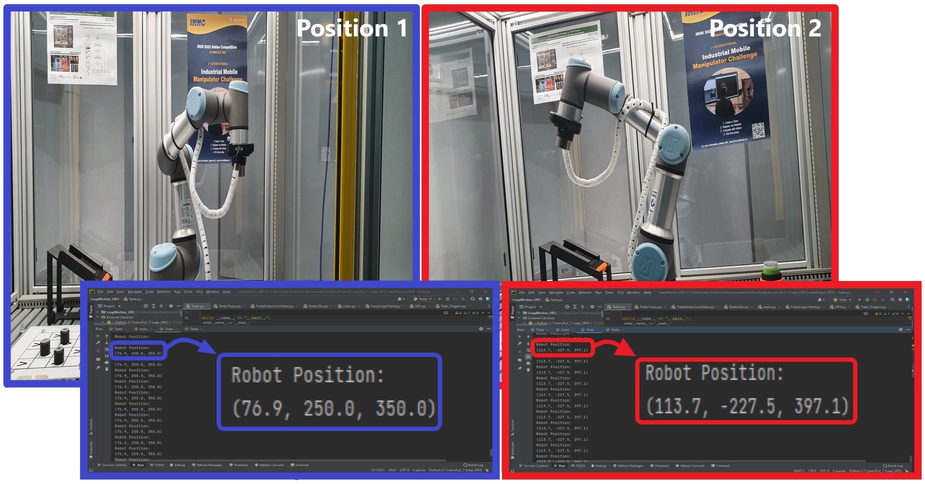

Get a frame from the Leap Motion controller
There are hands in the frame?
No
Yes
Obtain the X, Y and Z coordinates of the palm for each hand present in the frame
Have you finished collecting the hand positions?
No
Yes

Figure 4.4: Flowchart of the Hand's Palm Position Function.

4.3 Speed Control in the UR3 Controller

The UR3 controller was called “Cobot” in Figure 4.1, it acts as a Modbus server (port 502), which allows connections to be made to it in order to send Modbus requests, either read or write. The UR3 controller was used to obtain the UR3 gripper position in runtime, as well as to control the speed of the robot arm. To access the gripper position in real time, a Python code was developed using the pyModbusTCP library version 0.1.10. Figure 4.5 shows the gripper position obtained remotely with this code.

The speed control of the UR3 occurs by remotely changing the speed slider of the teach pendant. To perform this control remotely, a socket connection was established with port 30002, which is responsible for connecting to the UR3 dashboard. To make this connection, Python's socket library was used. The resulting function works as follows: first it connects via socket to port 30002, when connected it is able to receive a value between 0 and 1, where 0 means 0% of the maximum cobot speed and 1 indicates 100% of it, with the speed value set the function sends it to the dashboard. The operation of



```
Robot Position:
(76.9, 250.0, 350.0)
```

```
Robot Position:
(113.7, -227.5, 397.1)
```

Figure 4.5: Gripper Position Data in Real-time.

this function is shown in Figure 4.6.

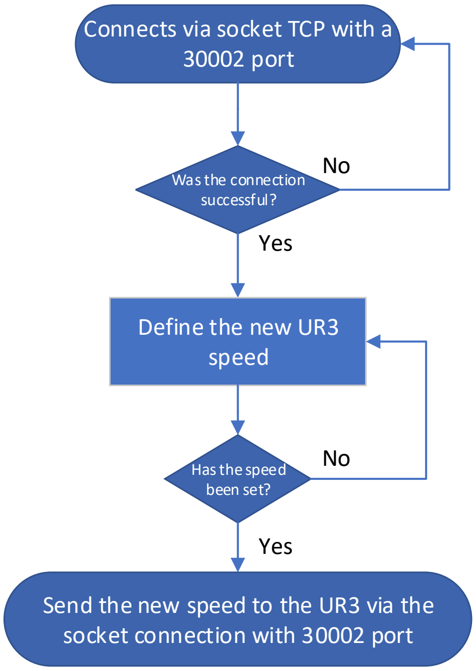

Connects via socket TCP with a 30002 port
Was the connection successful?
No
Yes
Define the new UR3 speed
Has the speed been set?
No
Yes
Send the new speed to the UR3 via the socket connection with 30002 port

Figure 4.6: Flowchart of the Set Speed Function.

4.4 Algorithm for Fuzzy Speed Control Based on Euclidean Distance

In the processing unit is where the calculation of the new UR3 speed through the fuzzy controller, and the calculation of the Euclidean distance between the robot and the hand occurs in real time.

4.4.1 Fuzzy Controller for the Robot Speed

The fuzzy controller was developed in Python using the library skfuzzy V0.4.2. The fuzzy inference method used was the Mamdani, because it is very intuitive, and well suited for a human input and has a more interpretable rule base. Three inputs and one output were defined, and they are respectively the Euclidean distance between the robot gripper and the hand measured in millimeters (mm), the current speed of the robot in percentage (%), and whether the gripper is approaching or moving away from the adimensional robot, and this is a Boolean variable, and finally the output speed in percentage (%).

To use the Mamdani method you must fuzzify the input and output variables, then define a fuzzy rule set, activate that rule set for the input crisp values, and finally defuzzify.

Fuzzification

First the input and output variables were fuzzified, that is, the linguistic variables with their Membership Functions (MFs) for each input variable are defined. In Figure 4.7 it is possible to observe the linguistic variables with their respective membership functions for each of the input variables. The fuzzy set of the Euclidean distance, which is an input variable, was defined according to the following linguistic terms: *very close (vC), Close (C), Average (A), Far (F), Very Far (vF).* Whereas for the input speed variable, the linguistic terms are: *very slow (vS), slow (S), average (A), high (H), very high (vH)*, and finally the Boolean approximation variable, which was represented through two linguistic variables: *separating, approaching.*

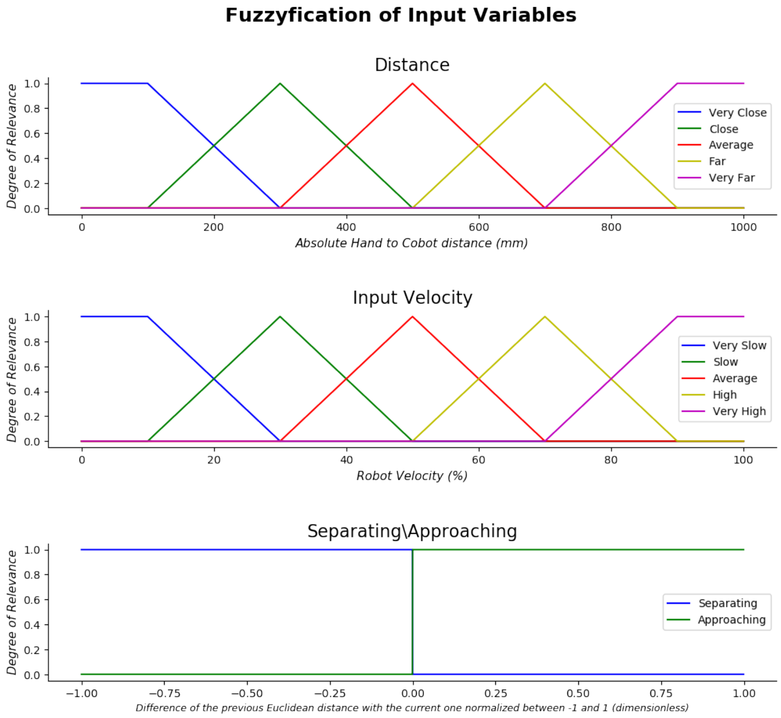


Figure 4.7: Fuzzification of Input Variables.

The fuzzy sets of the input variables (Euclidean distance and velocity) were defined using trapezoidal (at the edges) and triangular (in the middle) MFs.

For the variable output speed were used the same linguistic variables that the input speed, but the fuzzy sets were defined using only triangular MFs, in order to bring the results of defuzzification closer to 0 and 100. The MFs of the output speed can be seen in Figure 4.8.

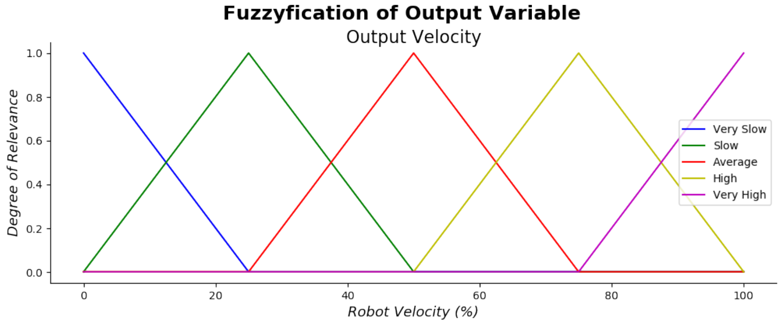
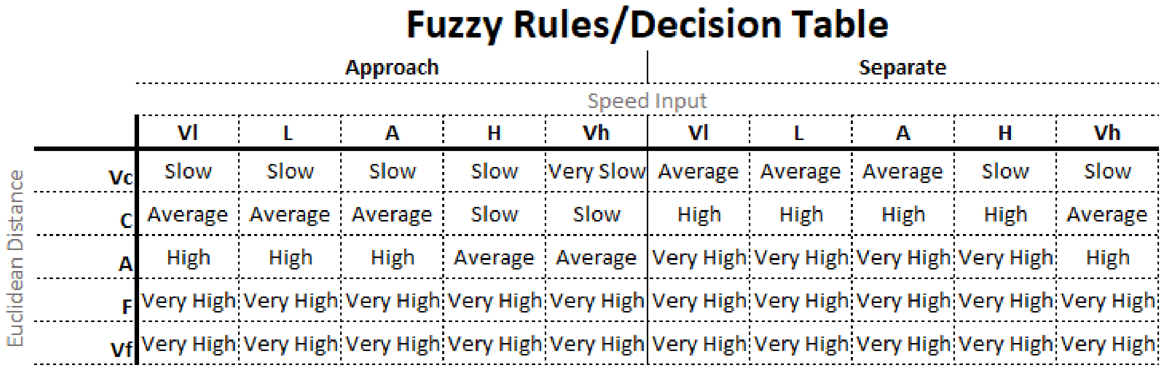


Figure 4.8: Fuzzification of Output Variables.

Fuzzy Rules

The Fuzzy inference system uses a set of IF-THEN rules that maps the MFs of the input variables to the MFs of the output variables. The decision table is shown in Figure 4.9, the table presents the rules defined for the Fuzzy system. Each cell represents the result of a logical operation AND between the input variables (rows and columns).



Fuzzy Rules/Decision Table

	Approach					Separate				
	Speed Input									
Euclidean Distance	Vl	L	A	H	Vh	Vl	L	A	H	Vh
Vc	Slow	Slow	Slow	Slow	Very Slow	Average	Average	Average	Slow	Slow
C	Average	Average	Average	Slow	Slow	High	High	High	High	Average
A	High	High	High	Average	Average	Very High	Very High	Very High	Very High	High
F	Very High	Very High	Very High	Very High	Very High	Very High	Very High	Very High	Very High	Very High
Vf	Very High	Very High	Very High	Very High	Very High	Very High	Very High	Very High	Very High	Very High

Figure 4.9: Fuzzy Rules/Decision Table.

Rules Activation

With the rules and membership functions defined, it is possible to calculate the activation of the rules in the inference step for a given Euclidean distance in mm, a speed in percent, and an approximation value. In a given scenario, a rule can be activated as follows: "IF Euclidean distance = vC AND input velocity = vH AND approach\separate = approaching THEN velocity output is vS". The result of this inference is an output fuzzy set, which is used in the defuzzification step.

Defuzzification

Defuzzification consists of transforming a fuzzy set into a crisp output value. In order to find the best defuzzification method for this system, different defuzzification methods were tested, consisting of Centroid, Bisector, Mean of Maxima (MoM), Largest of Maxima

(LoM), Smallest of Maxima (SoM), in order to verify if there are significant improvements in speed control in one method over another.

The centroid method consists of returning a crisp value based in the center of gravity of the fuzzy set. That is, the total area of the membership function distribution used to represent the output is divided into several subareas. The area and centroid of each subarea are calculated, and then the sum of all these subareas is taken to find the defuzzified value for a fuzzy set. In Figure 4.10 you can see the crisp values of the output velocity as a function of the 3 inputs for this defuzzification method.

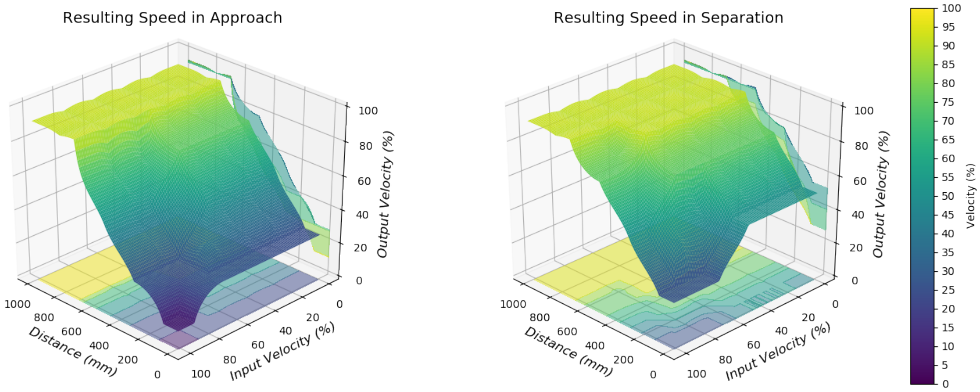


Figure 4.10: Centroid Defuzzication.

The bisector method consists of calculating the position under the curve where the areas of both sides are equal. The crisp output for this method can be seen in Figure 4.11.

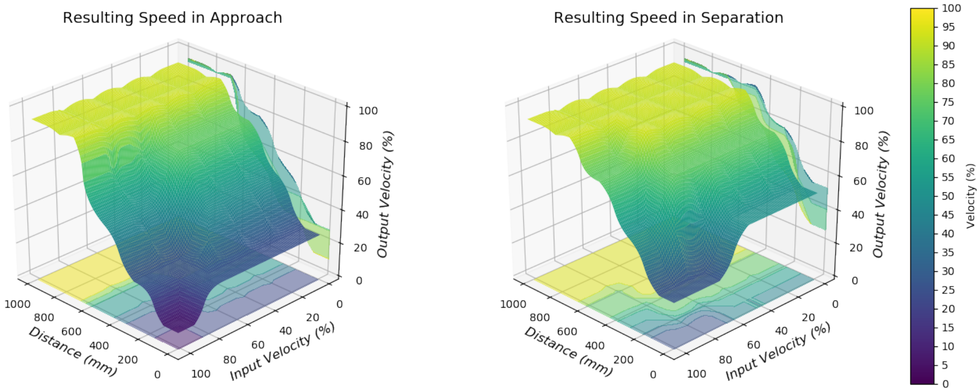
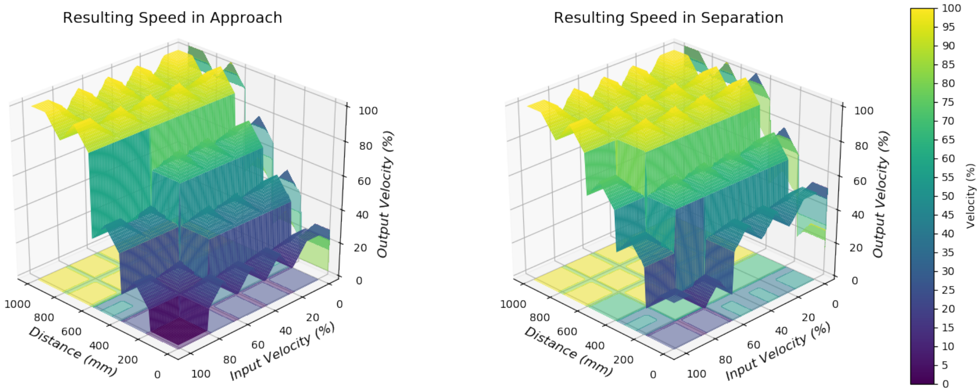


Figure 4.11: Bisector Defuzzication.

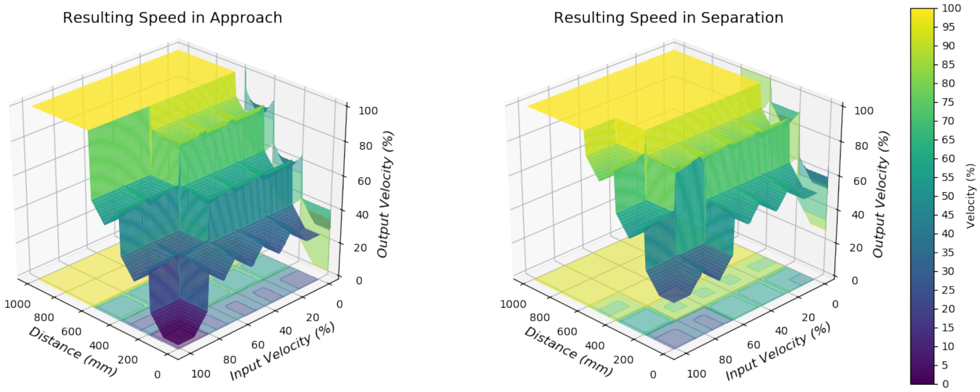
The SoM is the method that determines the smallest value of the domain with maximum membership value. The crisp output for this method can be seen in Figure 4.12.


Resulting Speed in Approach

Resulting Speed in Separation

Figure 4.12: Smallest of Maxima (SoM) Defuzzication.

The LoM method is the opposite of SoM, in this method the highest value of the domain with maximum association value is determined. The crisp output for this method can be seen in Figure 4.13.


Resulting Speed in Approach

Resulting Speed in Separation

Figure 4.13: Largest of Maxima (LoM) Defuzzication.

Finally, MoM is the method where the defuzzified value is taken as the element with the highest membership values. When there is more than one element with maximum membership values, the average value of the maximums is taken. The crisp output for this method can be seen in Figure 4.14.

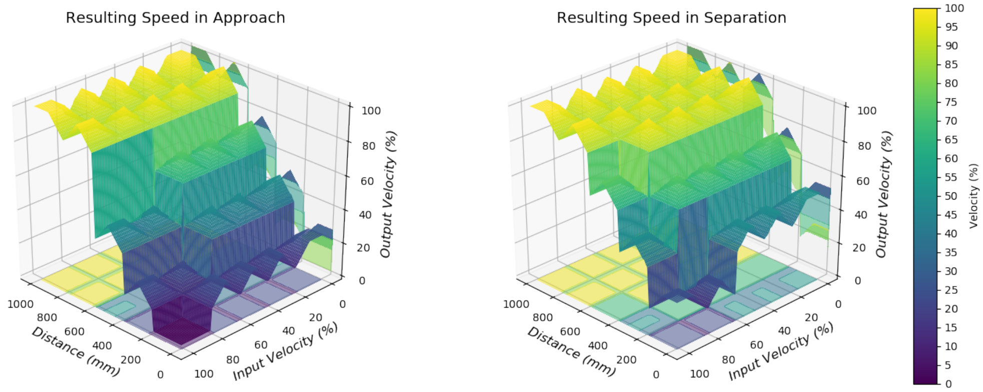


Figure 4.14: Meam of Maxima (MoM) Defuzzication.

Fuzzy Speed Controller Function

With all the steps to perform the fuzzy control defined, it was possible to develop the function in Python, the function has as input the current speed of the robot, the current Euclidean distance, the previous Euclidean distance to the current one and the deffuzification method to be used. With the previous and current Euclidean distance it's possible to calculate if the gripper is approaching or moving away from the hand, for this the previous distance is subtracted from the current distance, the result of this operation is normalized to a value between -1 and 1, where a negative result indicates that the hand is moving away, a result equal to zero indicates that there is no variation between the distance of the robot and the gripper, and a positive result indicates that the gripper and the hand are moving closer. After performing this operation all the input variables necessary for the inference method defined were obtained, so the next step consisted in calculating the pertinence of the input variables for each linguistic variable, with the pertinence values obtained it was possible to perform the activation of the rules, the next step consisted in calculating the pertinence of the input variables for each linguistic variable, with the obtained pertinence values it was possible to perform the activation of the rules, thus obtaining the activity of the MFs of the output velocity, finally the activity of the MFs were aggregated, allowing to calculate the velocity based on the chosen defuzzication method, which consists of a value between 0% and 100%, the resulting velocity was divided by 100 in order to normalize it to a value between 0 and 1. In Figure 4.15 you can see a flowchart containing the step-by-step of this routine.

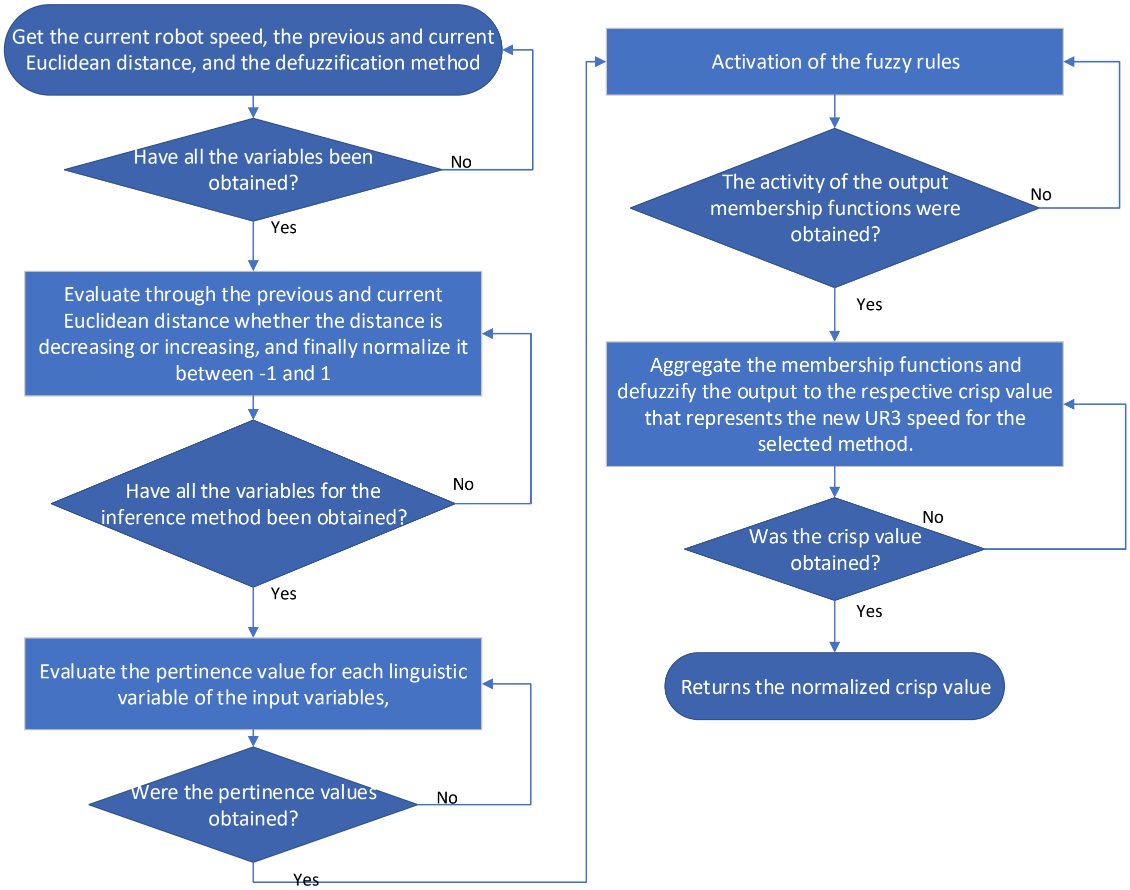

Get the current robot speed, the previous and current Euclidean distance, and the defuzzification method
Have all the variables been obtained?
No
Yes
Evaluate through the previous and current Euclidean distance whether the distance is decreasing or increasing, and finally normalize it between -1 and 1
Have all the variables for the inference method been obtained?
No
Yes
Evaluate the pertinence value for each linguistic variable of the input variables,
Were the pertinence values obtained?
No
Yes
Activation of the fuzzy rules
The activity of the output membership functions were obtained?
No
Yes
Aggregate the membership functions and defuzzify the output to the respective crisp value that represents the new UR3 speed for the selected method.
Was the crisp value obtained?
No
Yes
Returns the normalized crisp value

Figure 4.15: Flowchart of the Fuzzy Speed Control Function.

4.4.2 Euclidean Distance Evaluation

To obtain the Euclidean distance between the robot and the hand it was necessary two steps, where the first step consists of the code shown in Figure 4.4, which obtains the position of the hand palm in real time relative to the Leap Motion's Cartesian plane.

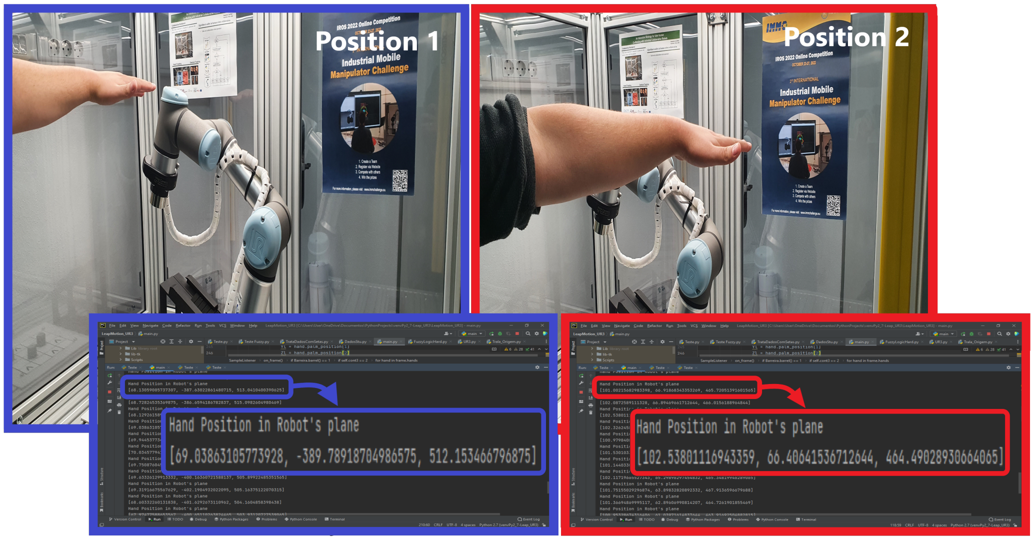
The second step is to convert this position from the Leap Motion Cartesian plane to the robot's plane, which is located at the base of the robot. The position and orientation of the planes can be seen in Figure 3.2. This is necessary, because to be able to calculate the Euclidean distance between them, which is one of the input variables needed for the fuzzy controller, both positions must be in the same plane.

From the properties of rotation and translation, it was concluded that it was necessary to rotate once with respect to the x-axis, once with respect to the z-axis, and finally to translate to the origin of the robot plane. The final equation is 4.1:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 0 & 0 \\ sin(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_2) & -sin(\theta_2) & 0 \\ 0 & sin(\theta_2) & cos(\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.1}$$

Being *x', y'* and *z'* the Cartesian coordinates of the center of the hand in the UR3 plane in mm. The *dx, dy* and *dz* are the physical distance between the origin of the sensor and the origin of the robotic arm, where *dx, dy* and *dz* are respectively worth approximately 55 mm, -310 mm and 815 mm. These values were found from the physical measurement of the distance between the origin of the sensor and the UR3. The angle of rotation in z, θ_1 is equal to that of rotation in x, θ_2 which is worth -90º. Whereas *x, y* and *z* consist of the position of the hand relative to the plane in mm of Leap Motion.

Thus, using Python, it was possible to make a routine that converts in runtime the position of the hand from one plane to the other. In Figure 4.16 it is possible to observe the position of the hand in real time in relation to the origin of the collaborative robot. If more than one hand is present on the robot workbench, the routine will also compare the Euclidean distance of each hand to the gripper, checking which is the smallest, in order to return the Euclidean distance of the hand closest to the gripper to be used by the fuzzy speed control function.



```
Hand Position in Robot's plane
[68.13059005737307, -387.63022861480715, 513.0410400390625]
```

```
Hand Position in Robot's plane
[69.03863105773928, -389.78918704986575, 512.153466796875]
```

```
Hand Position in Robot's plane
[101.00215682983398, 66.91868343353269, 465.72051391601565]
```

```
Hand Position in Robot's plane
[102.53801116943359, 66.40641536712644, 464.49028930664065]
```

Figure 4.16: Hand Position Data in Robot Plane.

4.5 Systems Integration

The Python programming language was used to integrate all the functions and routines used in this project, the main routine gathers what was developed for each block of the system architecture (Figure 4.1), in order to perform speed control and avoid collisions in the system. The main routine was developed on top of the class shown in Figure 4.4. In order to facilitate the understanding of the routine, a flowchart with its operation is shown in Figure 4.17. As not all defuzzification methods can reach the 0% output speed value, and because the minimum speed that can be sent to the UR3 controller is 2% (controller limitation), it was necessary to add to the main routine a condition based on the Euclidean distance between the gripper and the hand, where if it is smaller than 75 mm the robot is paused, if the distance is greater than 75 mm and less than 550 mm the robot movement is resumed and speed control happens by the fuzzy controller, if the distance is greater than 550 mm the robot speed is kept at maximum, since the gripper is physically far enough from the operator, and any speed control in this situation would only result in loss of efficiency for the system. All the codes, functions and routines used and developed in this project are available in the link present in annex A.

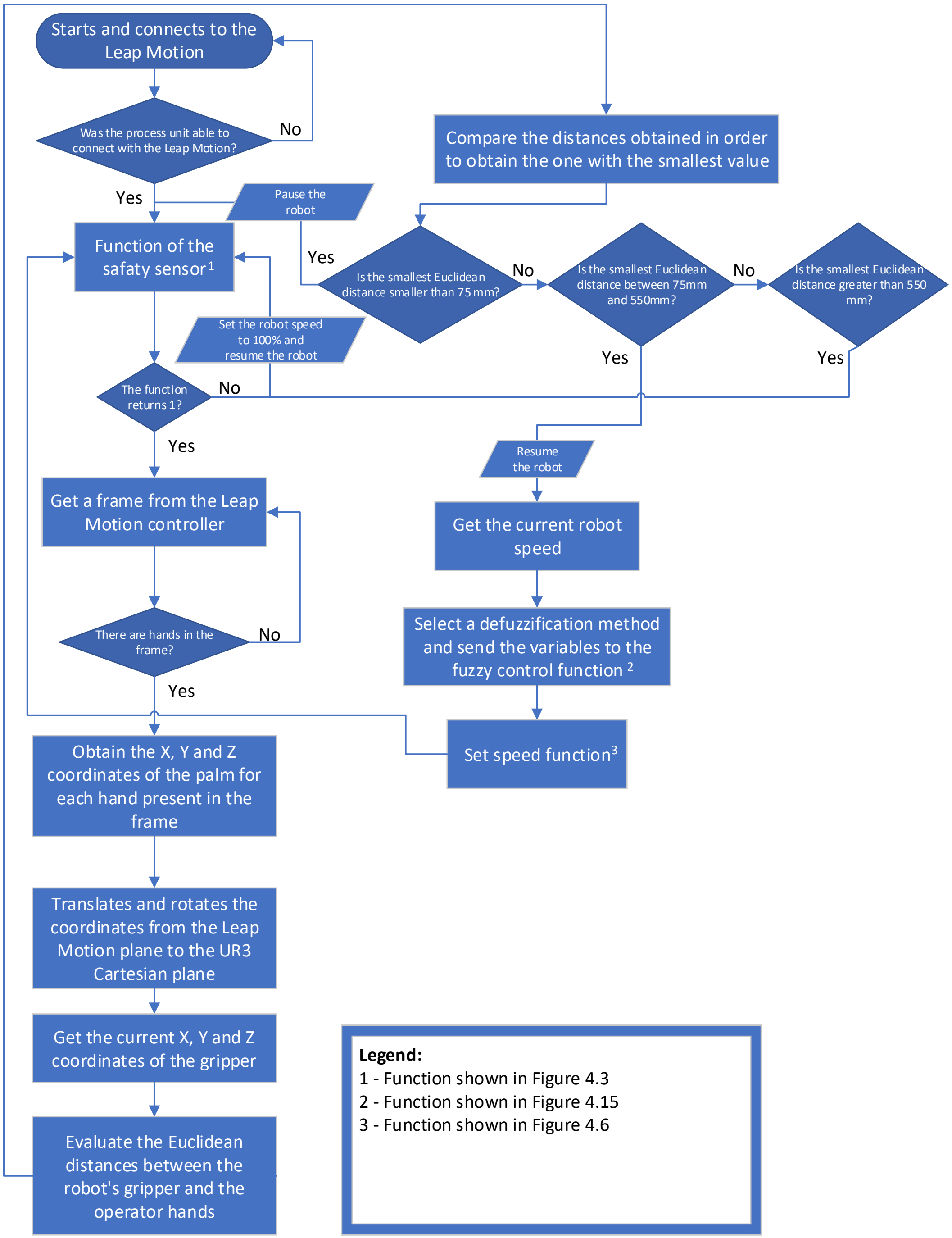

Starts and connects to the Leap Motion
Was the process unit able to connect with the Leap Motion?
No
Yes
Pause the robot
Function of the safaty sensor [1]
Set the robot speed to 100% and resume the robot
The function returns 1?
No
Yes
Get a frame from the Leap Motion controller
There are hands in the frame?
No
Yes
Obtain the X, Y and Z coordinates of the palm for each hand present in the frame
Translates and rotates the coordinates from the Leap Motion plane to the UR3 Cartesian plane
Get the current X, Y and Z coordinates of the gripper
Evaluate the Euclidean distances between the robot's gripper and the operator hands
Compare the distances obtained in order to obtain the one with the smallest value
Yes
Is the smallest Euclidean distance smaller than 75 mm?
No
Is the smallest Euclidean distance between 75mm and 550mm?
No
Is the smallest Euclidean distance greater than 550 mm?
Yes
Yes
Resume the robot
Get the current robot speed
Select a defuzzification method and send the variables to the fuzzy control function [2]
Set speed function [3]
Legend:
1 - Function shown in Figure 4.3
2 - Function shown in Figure 4.15
3 - Function shown in Figure 4.6

Figure 4.17: Main Routine Developed.

Chapter 5

Experiments and Results

5.1 Experiments Methodology

In order to validate this method of fuzzy speed control, a routine was developed in Python to collect and export to real time .csv files containing the robot's gripper position, hand position, Euclidean distance, gripper speed, and timestamp, the routine is described in the chapter 5. The data was collected from four different routes that the robot traveled. The routes were called Square, Round, Collision, 3-Dimensional.

To ensure data reliability, 10 tests were performed for each defuzzification method on each of the routes, for a total of 200 tests. All tests had the same running time, consisting of 300 samples. The wooden hand was fixed in the same position for all routes.

To verify the results, graphs were made showing the speed behavior for each defuzzification method in each route, allowing a visual analysis of them. A routine in Python was also used to calculate the average velocity for each of the 200 tests, in order to generate box diagrams, allowing a visual analysis of the difference between the average velocity for each method and each route. And finally, Excel was used to perform the two-tailed t-test for two samples assuming different variances with an α of 0.05, in order to verify whether the variation in speed between the different methods is significant for each of the routes.

5.2 Experiments

As mentioned in the previous chapter, the experiment to test the fuzzy speed control consisted in attaching a wooden hand to the bench, simulating a human hand, and checking how the speed of the robotic arm varied in 4 different routes, called Square, Round, Collision and 3-Dimensional.

In the Figure 5.1 you can see how the routes of the UR3 gripper were defined, for all routes, the gripper will start its movement from the starting point, indicated in the Figure 5.1, and the direction of its movement is indicated by the red arrows, the four routes are closed loops, which will be repeated until the desired number of samples is reached, finally, in all routes the position of the center hand is indicated by a red diamond. Where in the Square route the gripper performed linear movements only in the XY plane, without varying the height, in order to facilitate the observation of the velocity behavior. The Round route also remained in the XY plane, but with circular movements. The Collision route is a path where the gripper must pass through where the wooden hand is attached, and then return to the origin point. The objective of this route is to observe if the speed control method is able to prevent a collision with the hand. The analysis to verify that contact occurred between the gripper and the hand was done empirically through a visual analysis in each test performed using this route. In the 3-Dimensional route, points were defined with different coordinates in the x, y and z axis. For this, approaches and moves away from the hand position were performed, with varied angles and directions, in order to verify the behavior of the gripper velocity for the largest number of cases.

To perform the data collection, it was necessary to develop an auxiliary routine in Python. In order to maintain a pattern in the collection, the routine kept the UR3 in the starting position of the route until it started collecting data. The routine kept the UR3 at the starting point of the route or started it, through a signal via Modbus, it also collected, the robot's claw position, the hand position, the Euclidean distance, the claw speed and the timestamp until it reached a total of 300 samples, exporting the data to a .csv file, and finally it also started the main program, responsible for performing the

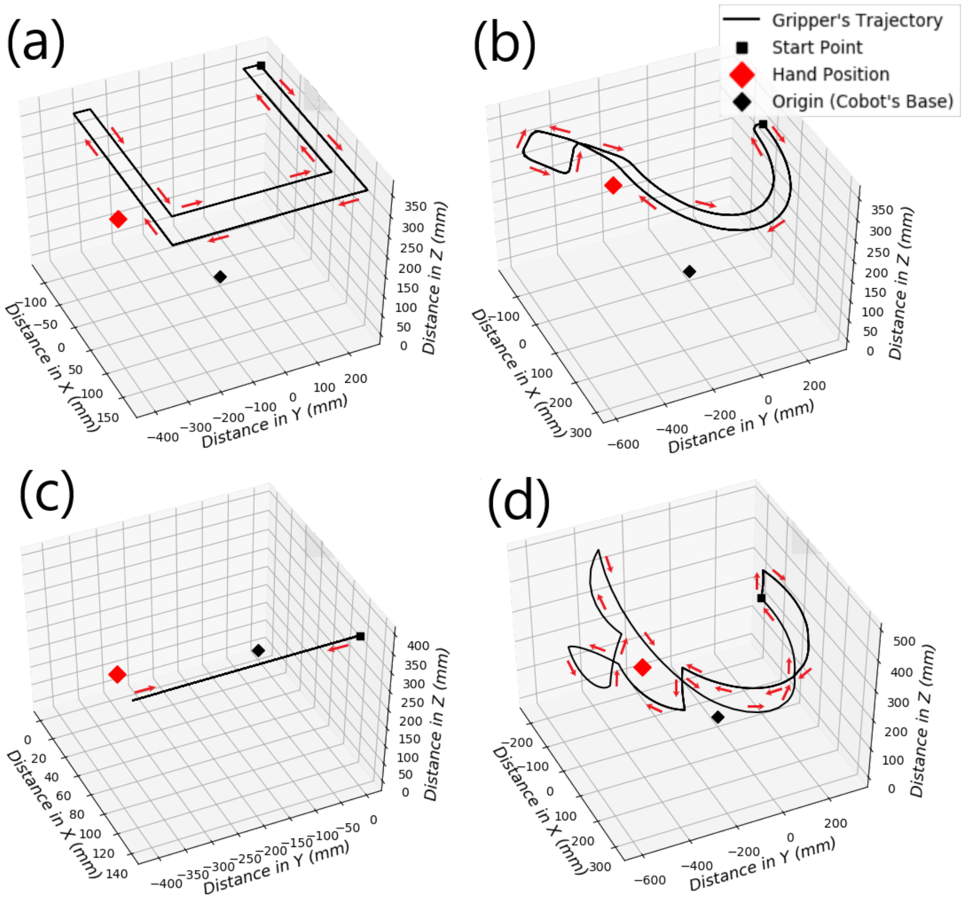


Figure 5.1: (a) Square's Trajectory, (b) Round's Trajectory, (c) Collision's Trajectory and (d) 3-Dimension's Trajectory.

speed control via fuzzy and start the hand tracking.

With this, the experiment proceeded as follows: first, the defuzzification method and the route were defined. Then the UR3 was manually positioned in the origin position of the chosen route, with the robot in the origin the auxiliary routine was manually started, and as soon as it exported the *.cvs* file it was manually interrupted. This process was repeated for all defuzzification methods and routes 10 times, totaling 200 tests. A test process was recorded [1] and in Figure 5.2 it is possible to see the stages of the test process step by step through a flowchart.

A simple experiment was also performed in order to verify if there are any hand

[1] Available in `https://youtu.be/52834KI6CzU`

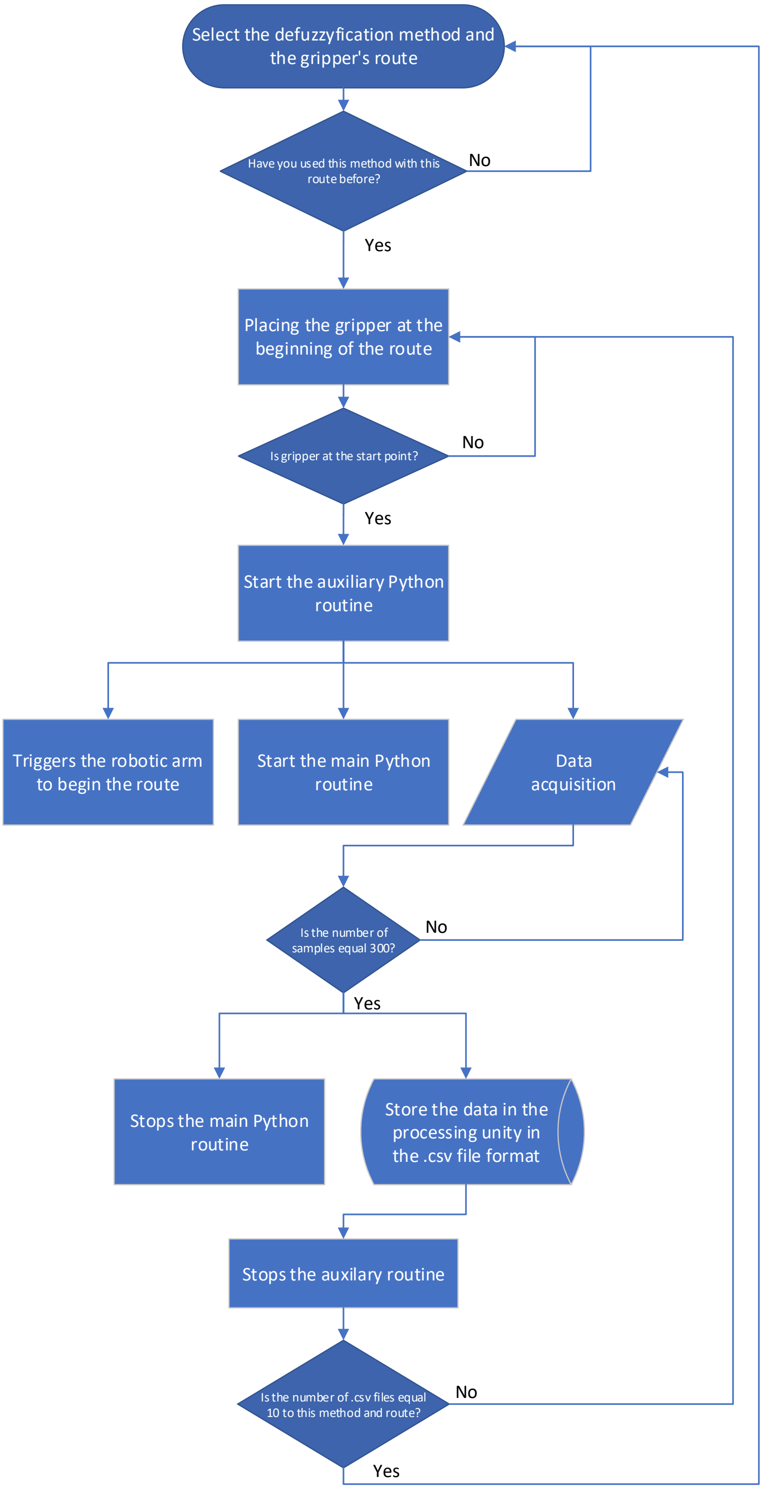

Select the defuzzyfication method and the gripper's route
Have you used this method with this route before?
No
Yes
Placing the gripper at the beginning of the route
Is gripper at the start point?
No
Yes
Start the auxiliary Python routine
Triggers the robotic arm to begin the route
Start the main Python routine
Data acquisition
Is the number of samples equal 300?
No
Yes
Stops the main Python routine
Store the data in the processing unity in the .csv file format
Stops the auxilary routine
Is the number of .csv files equal 10 to this method and route?
No
Yes

Figure 5.2: Flowchart of the Testing Process.

tracking limitations, such as possible blind spots, due to the use of only one Leap Motion sensor. The experiment consisted in positioning the UR3's gripper just below the Leap Motion sensor and passing the hand under the gripper, while a Python routine displays in real time the position of the hand. An interruption in the hand position display will indicate at any point in the experiment that Leap Motion could not keep track of the hand to that position. The results were obtained empirically through visual analysis.

5.3 Results

This section is dedicated to presenting and discussing the results found.

By performing the second experiment, it was possible to notice that one sensor is not enough to guarantee that no blind spots occur in the hand tracking, because it was visually perceived an interruption in the hand tracking at the moment it passed under the robot's gripper. One possible solution to solve this problem would be to insert more sensors to cover the blind spots in the system.

5.3.1 Speed Behavior For Each Route and Defuzzification Method

From the data set, graphs were generated for each method and route that show the robot's speed at each instant of the route. The velocities of the robot at each time point were superimposed over their respective trajectories, the absolute value of the velocity at each point was mapped using a color scale, as can be seen in the Figures 5.3, 5.4, 5.5, 5.6 and 5.7, this process was performed in one experiment for each method and route, allowing the visualization of the speed control behavior for each method. The annex B contains the average speed for each test performed on the determined routes for each defuzzification method.

Centroid

The velocity dataset for the Square, Round and Collision route was plotted on a two-dimensional Cartesian plane (Figure 5.3, having the position of the robot's hand and origin mirrored to the same plane where the gripper's trajectory occurs, in order to facilitate visualization of the result, this same process is performed for all methods.

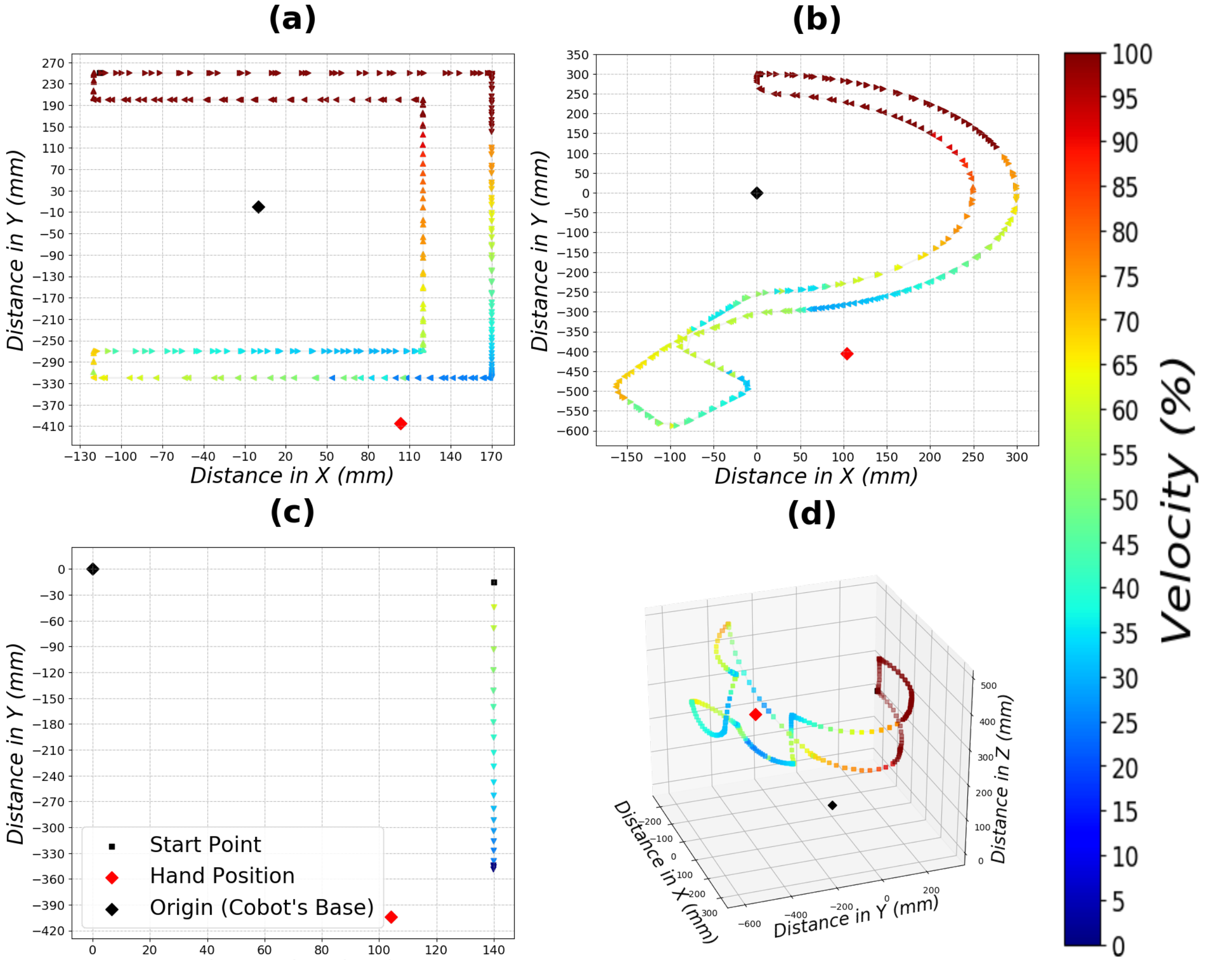


Figure 5.3: Centroid: (a) Square's Velocity Data, (b) Round's Velocity Data, (c) Collision's Velocity Data and (d) 3-Dimension's Velocity Data.

Through visual analysis of Square and Round routes showed in Figure 5.3, it is possible to observe that the Centroid method varied the robot's speed with a certain linearity and smoothness, where there was a gradual reduction in speed as the gripper approached the center of the hand and a gradual increase as it moved away.

In the Collision route, it was visually verified that in none of the tests occurred collision with the hand, i.e., the speed of the gripper reached zero, as can be observed in Collision data in Figure 5.3. In addition, as the hand was fixed, the gripper remained stationary until the end of the collection of 300 samples, when it was manually returned to its original position. A recording[2] was made using a real hand with the Centroid method in order to show the system's collision avoidance capability.

In the 3-Dimensional velocity data showed in 5.3 the repetition of the velocity behaviors is observed in the same way as in the velocity data of Square and Round routes, but for the gripper approaching and moving away from different angles and heights.

The Table B.1 shows the average speed for each test on each route using the Centroid method, In the column for the Collision route, it is possible to observe a significantly lower value than for the other trajectories. This is because the average takes into account the velocity during the 300 samples, and in the Collision route the velocity is zero most of the time. From the Table B.1 it was possible to obtain the mean average velocity as well as its standard deviation for each route, being respectively 67.3914 (%) and 0.8052 (%) for the Square route, 63.9927 (%) and 1.0773 (%) for the Round route, 2.7366 (%) and 0.5383 (%) for the Collision route and 65.4578 (%) and 0.6059 (%) for the 3-Dimensional route.

Bisector

Similarly to the Centroid method, it can be seen in Figure 5.4 that the velocity variation is smooth in all the routes. Moreover, the colors vary gradually as well as the velocity, which increases as the grip moves away from the hand and decreases as the grip approaches.

Just as in the Centroid method no collision occurred between the hand and the gripper, it can be seen in the Collision data showed in Figure 5.4 that the velocity gradually decreases to zero, where it remains until the end of the test. The Table B.2 shows the average speed for each test and route with the Bisector method. From the Table B.2 it was possible to obtain the mean average velocity as well as its standard deviation for each

[2] Available in `https://youtu.be/6PWfm4QsfzM`

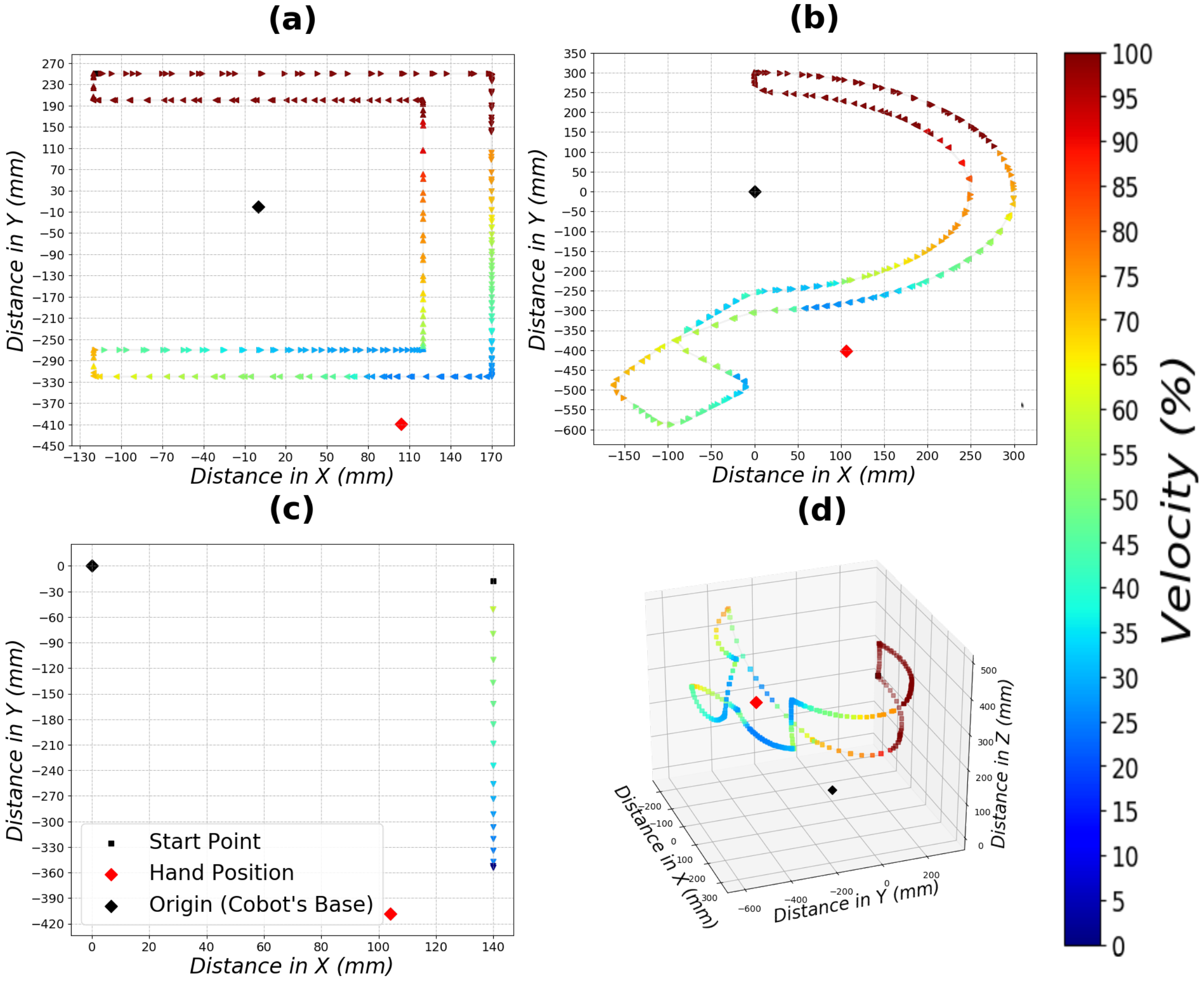


Figure 5.4: Bisector: (a) Square's Velocity Data, (b) Round's Velocity Data, (c) Collision's Velocity Data and (d) 3-Dimension's Velocity Data.

route, being respectively 66.6369 (%) and 1.0243 (%) for the Square route, 63.4883 (%) and 0.7241 (%) for the Round route, 2.5766 (%) and 0.2389 (%) for the Collision route and 62.6914 (%) and 1.2732 (%) for the 3-Dimensional route.

SoM

In the SoM method the differences in speed control are clearer compared to the previous methods, after viewing the Figure 5.5. It can be seen that the velocity varies sharply at times. Also, as the claw moves closer to the hand, at certain times there is an increase in velocity and then a decrease again.

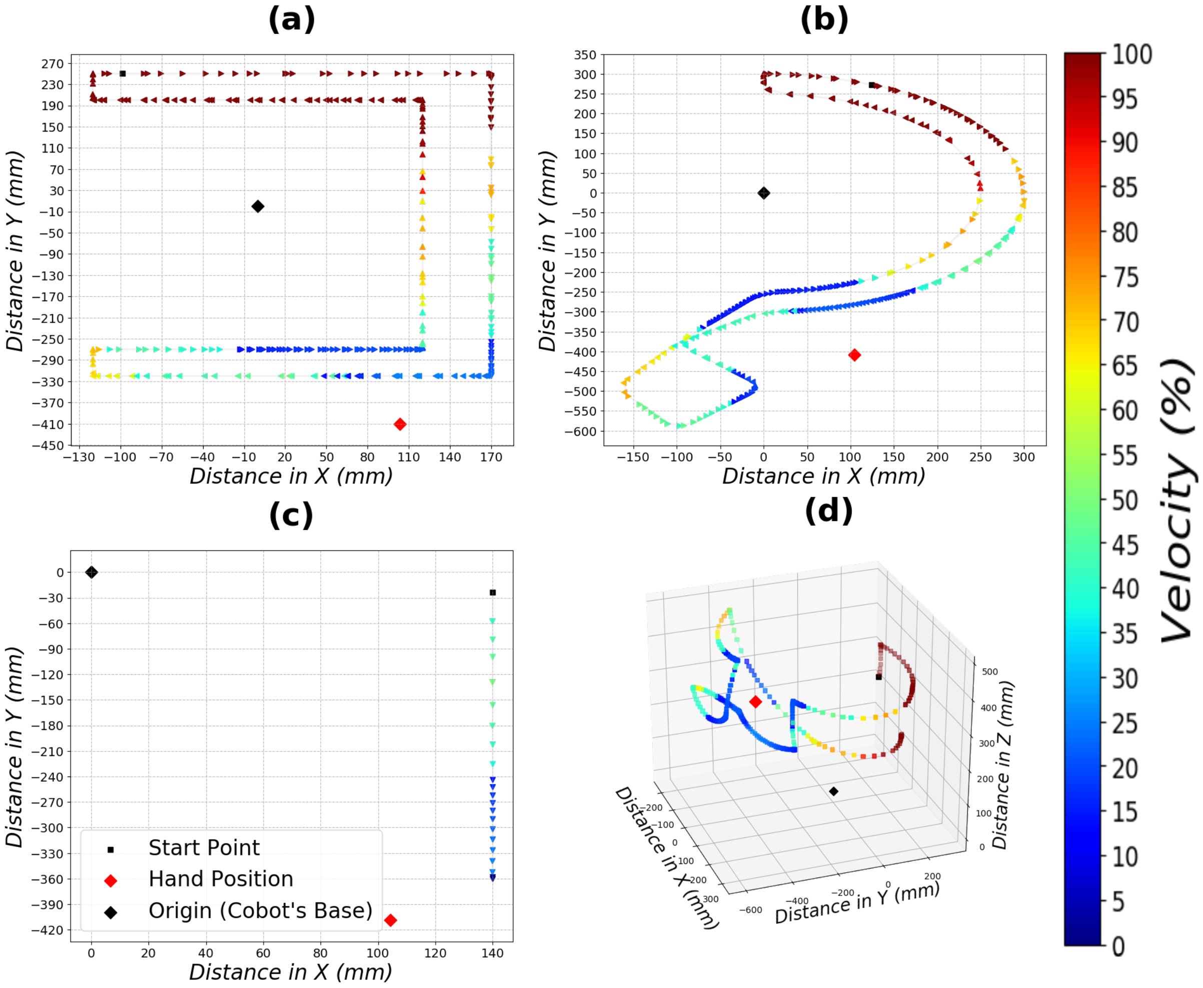


Figure 5.5: SoM: (a) Square's Velocity Data, (b) Round's Velocity Data, (c) Collision's Velocity Data and (d) 3-Dimension's Velocity Data.

As with the previous methods, no collision occurred between the Cobot and the hand. It can be seen in the velocity data showed in the Figure 5.5, that the velocity eventually reaches zero. In Table B.3 as possible to see the average speed for each test and route with the SoM method, where can be seen that, it can be seen that the SoM method generated the lowest average velocities when comparing the absolute value to the other methods. From the Table B.3 it was possible to obtain the mean average velocity as well as its standard deviation for each route, being respectively 55.8296 (%) and 1.9578 (%) for the Square route, 49.0103 (%) and 0.3779 (%) for the Round route, 1.9341 (%) and 0.3611 (%) for the Collision route and 46.0166 (%) and 3.1719 (%) for the 3-Dimensional route.

LoM

Similarly to the SoM method, sharp variations in velocity occur, which can be seen in Figure 5.6. The difference between these methods is that in LoM the velocity first decreases and then increases, in contrast to what occurs in SoM.

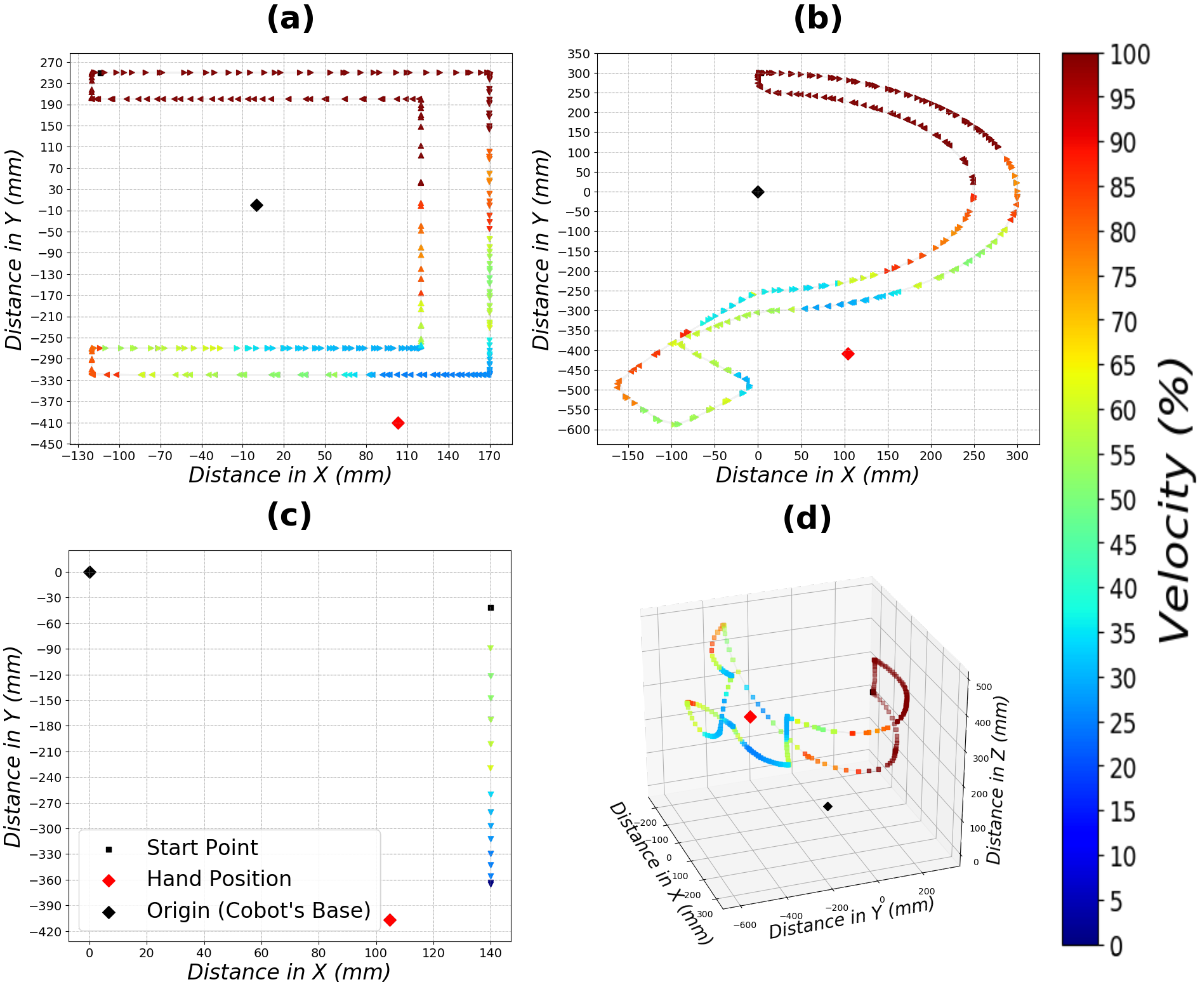


Figure 5.6: LoM: (a) Square's Velocity Data, (b) Round's Velocity Data, (c) Collision's Velocity Data and (d) 3-Dimension's Velocity Data.

In the LoM method there were also no collisions, and in the same way it is possible to observe in the Collision velocity data the velocity reducing until it reaches zero. In Table B.4 as possible to see the average speed for each test and route with the LoM method, and also it was possible to obtain the mean average velocity as well as its standard deviation for each route, being respectively 68.8178 (%) and 1.3002 (%) for the Square route, 69.0716 (%) and 1.4994 (%) for the Round route, 2.2192 (%) and 0.4418 (%) for the Collision route and 63.7896 (%) and 0.6359 (%) for the 3-Dimensional route.

MoM

The MoM method, can be described as the average between the SoM and LoM methods, and for this reason its results are similar, where sharp variations in velocity also occur, as can be seen in the Figure 5.7, with the only difference being that in the MoM method the velocity behaves a little more linearly than in the other two, with no increase in velocity on approach and no decrease in speed on separation. And as in the other method, no collisions occur in the Collision route experiments.

The Table B.5 presents the average speed for each test and route with the MoM method and from it was also possible to obtain the mean average velocity as well as its standard deviation for each route, being respectively 63.5977 (%) and 1.0286 (%) for the Square route, 59.8869 (%) and 1.1628 (%) for the Round route, 2.0416 (%) and 0.1189 (%) for the Collision route and 57.6890 (%) and 0.7106 (%) for the 3-Dimensional route.

5.3.2 Student's T Test and Box Plots

From the set of average speed data contained in annex B it was possible to perform the t-tests. In Tables 5.1, 5.2, 5.3 and 5.4 the t-test results were gathered. Results where the P value is greater than alpha are indicated with "Not different" and where the P value is less as "Different". Results labeled "Different" indicate that the difference between the mean velocities has a high probability of not being random, which implies that there is a significant difference between the mean velocities of the two groups. Results indicated as "Not different" imply that the probability of the results not being random is higher, that is, there is a greater error involved, and it is not possible to say that there is a difference between the two groups.

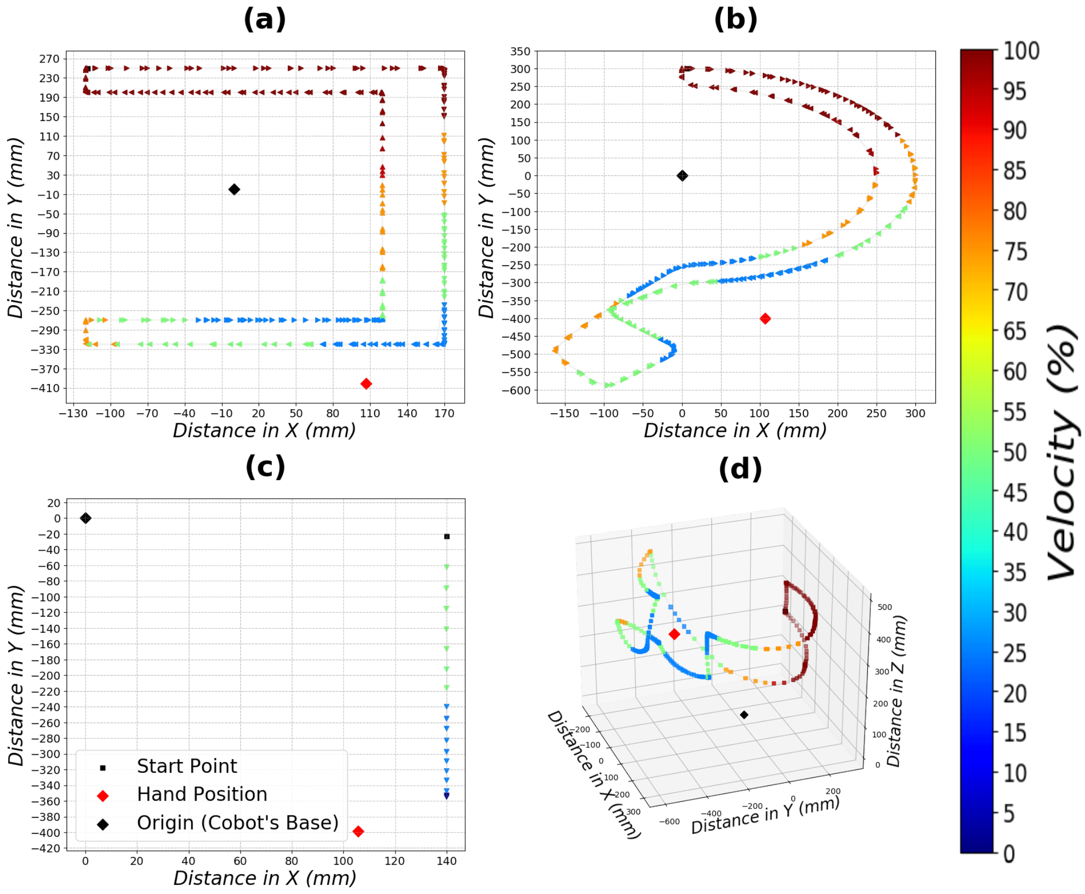


Figure 5.7: MoM: (a) Square's Velocity Data, (b) Round's Velocity Data, (c) Collision's Velocity Data and (d) 3-Dimension's Velocity Data.

Square Route					
	Centroid	**Bisector**	**LoM**	**SoM**	**MoM**
Centroid	-	Not different	Different	Different	Different
Bisector	Not different	-	Different	Different	Different
LoM	Different	Different	-	Different	Different
SoM	Different	Different	Different	-	Different
MoM	Different	Different	Different	Different	-

Table 5.1: T Test for Square Route.

In addition to the t-test, box diagrams were also made as a visual way to observe the differences between the two groups. The diagrams are shown in the Figure 5.8.

The analysis of the t-test results and box plots, as well as the visual analysis of the

Round Route					
	Centroid	**Bisector**	**LoM**	**SoM**	**MoM**
Centroid	-	Not different	Different	Different	Different
Bisector	Not different	-	Different	Different	Different
LoM	Different	Different	-	Different	Different
SoM	Different	Different	Different	-	Different
MoM	Different	Different	Different	Different	-

Table 5.2: T Test for Round Route.

Collision Route					
	Centroid	**Bisector**	**LoM**	**SoM**	**MoM**
Centroid	-	Not different	Different	Different	Different
Bisector	Not different	-	Not different	Different	Different
LoM	Different	Not different	-	Different	Not different
SoM	Different	Different	Different	-	Not different
MoM	Different	Different	Not different	Not different	-

Table 5.3: T Test for Collision Route.

3-Dimension Route					
	Centroid	**Bisector**	**LoM**	**SoM**	**MoM**
Centroid	-	Different	Different	Different	Different
Bisector	Diferente	-	Different	Different	Different
LoM	Diferente	Different	-	Different	Different
SoM	Diferente	Different	Different	-	Different
MoM	Diferente	Different	Different	Different	-

Table 5.4: T Test for 3-Dimension Route.

speed behavior for each method on each route, made it possible to compare each defuzzification method. The methods were compared both in terms of average velocity (the higher the average velocity of one method relative to the other, the better the efficiency, since a higher average velocity indicates that the method can execute more movements in the same period of time). In table 5.5 the comparison between each defuzzification method is displayed.

Through the comparisons it is possible to see that the Centroid method was the most promising for the 3-Dimensional route, since it has the highest average speed, besides ensuring a smooth control of the speed. In the Collision route, as mentioned before, during

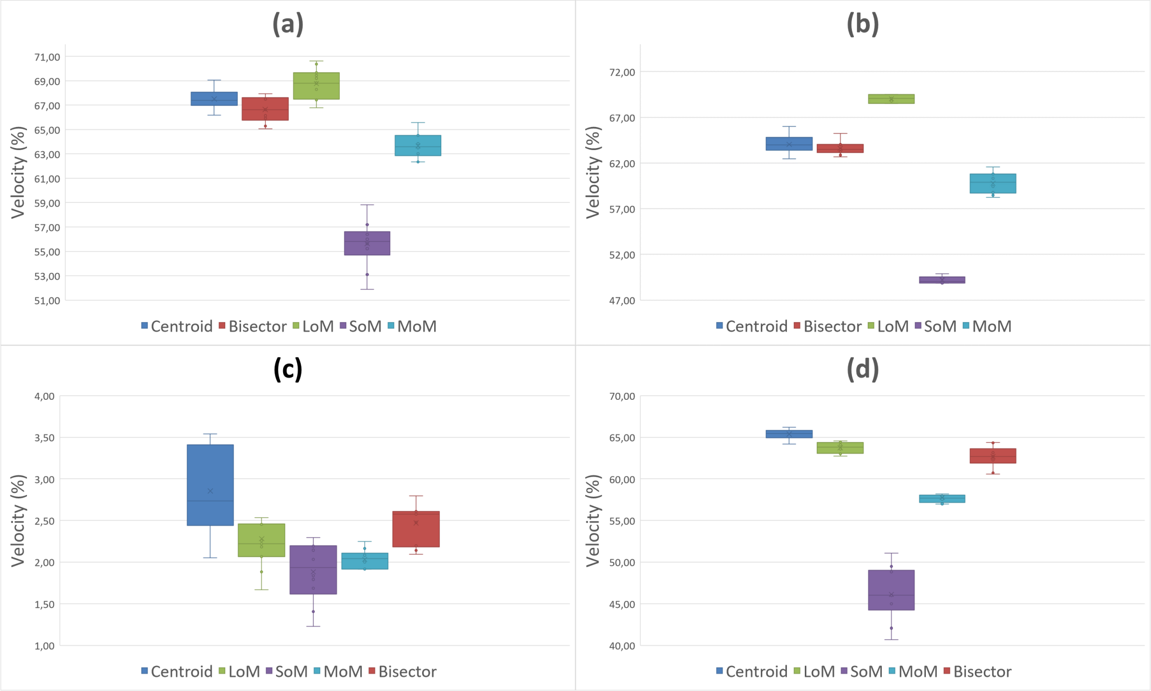


Figure 5.8: Box Plot of Average Velocity: (a) Square Route, (b) Round Route, (c) Collision Route and (d) 3-Dimensional Route.

the experiments no collision occurred, i.e., all methods are sufficient in terms of safety, as for the highest average speed and smoothest speed control the methods that stood out the most were the Centroid and Bisector, where both have a high linearity in speed control and the average speed of both is very similar, not being possible to differentiate precisely through the t-test. As for the Square and Round routes, the method that guaranteed the highest efficiency was LoM, being the one with the highest average speed, which is significantly higher than the other methods, as for smoothness, both the Bisector and the Centroid methods showed promise, besides both having the second-highest average speed in these two routes. The SoM and MoM methods proved to be significantly inferior when compared to the other three methods presented, and did not bring linearity to the speed control.

		Centroid		**Bisector**		**LoM**		**MoM**		**SoM**	
Centroid	**Advantages**	—		Faster in the 3-Dimensional route		Smoother	Faster in the 3-Dimensional and Collision routes	Smoother	Faster in all routes	Smoother	Faster in all routes
	Disadvantages			None		Slower in the Square and Round routes		None		None	
Bisector	**Advantages**	None		—		Smoother		Smoother	Faster in all routes	Smoother	Faster in all routes
	Disadvantages	Slower in the 3-Dimensional route				Slower in the Square, Round and 3-Dimensional routes		None		None	
LoM	**Advantages**	Faster in the Square and Round routes		Faster in the Square, Round and 3-Dimensional routes		—		Faster in Square, Round and 3-Dimensional routes		Faster in all routes	
	Disadvantages	Abrupter	Slower in the 3-Dimensional and Collision routes	Abrupter				None		None	
MoM	**Advantages**	None		None		None		—		Faster in the Square, Round and 3-Dimensional routes	
	Disadvantages	Abrupter	Slower in all routes	Abrupter	Slower in all routes	Slower in the Square, Round and 3-Dimensional routes				None	
SoM	**Advantages**	None		None		None		None		—	
	Disadvantages	Abrupter	Slower in all routes	Abrupter	Slower in all routes	Slower in all routes		Slower in the Square, Round and 3-Dimensional routes			

Table 5.5: Comparison Between Methods of Defuzification.

Chapter 6

Conclusion and Future Work

This work presents a speed control method for collaborative robotic arms through fuzzy logic and hand position, in order to improve HRC. For this, an algorithm was implemented, which collects data from the cobot and the position of the operator's hand on the workbench. The experiments in order to validate the method were performed, due to the low response time the system managed to avoid collisions in all the experiment's tests. However, there were limitations due to the use of only one Leap Motion sensor, since the UR3 can get between the hand and the sensor, interrupting the tracking, thus not guaranteeing the operator's total safety. When performing experiments with more than one hand on the bench, the system was able to identify and track both, and the algorithm was able to consider only the hand closest to the cobot to perform the fuzzy speed control, avoiding collisions. Despite the limitations, the system showed promise for dealing with safety issues.

For the project to keep improving, the following steps can be done:

- Insert more sensors capable of detecting and tracking hands, in order to limit as much as possible blind spots in the system.

- Compare the control method with other methods in order to verify its efficiency.

- Perform tests with other cobot models in order to verify the flexibility of the system.

- Use the sensors not only to track the hand position, but also to use them for gesture detection, in order to improve HRC.
- Identify the direction of the hand, to perform preventive speed reduction.

Bibliography

[1] C. J. Loureiro, “Cobots na indústria 4.0,” Ph.D. dissertation.

[2] B. Marr, “What everyone must know about industry 4.0,” *Forbes, June*, vol. 20, p. 2016, 2016.

[3] G. Erboz, “How to define industry 4.0: Main pillars of industry 4.0,” *Managerial trends in the development of enterprises in globalization era*, vol. 761, p. 767, 2017.

[4] R. Galin and R. Meshcheryakov, “Automation and robotics in the context of industry 4.0: The shift to collaborative robots,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 537, 2019, p. 032 073.

[5] KUKA, *Automação na indústria automotiva*, `https://www.kuka.com/pt-br/ramos-de-atividade/indÃžstria-automotiva`, Oct. 2022.

[6] I. F. of Robotics, *Service robots*, `https://ifr.org/service-robots/`, Oct. 2022.

[7] U. Robots, *Service robots*, `https://www.universal-robots.com/pt/industrias/eletrÃşnica-e-tecnologias/`, Oct. 2022.

[8] F. Sherwani, M. M. Asad, and B. Ibrahim, “Collaborative robots and industrial revolution 4.0 (ir 4.0),” in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, 2020, pp. 1–5. DOI: `10.1109/ICETST49965.2020.9080724`.

[9] A. Hentout, M. Aouache, A. Maoudj, and I. Akli, "Human–robot interaction in industrial collaborative robotics: A literature review of the decade 2008–2017," *Advanced Robotics*, vol. 33, no. 15-16, pp. 764–799, 2019. [Online]. Available: `https://doi.org/10.1080/01691864.2019.1636714`.

[10] S. Bragança, E. Costa, I. Castellucci, and P. M. Arezes, "A brief overview of the use of collaborative robots in industry 4.0: Human role and safety," *Occupational and environmental safety and health*, pp. 641–650, 2019.

[11] D. Rodríguez-Guerra, G. Sorrosal, I. Cabanes, and C. Calleja, "Human-robot interaction review: Challenges and solutions for modern industrial environments," *IEEE Access*, vol. 9, pp. 108 557–108 578, 2021.

[12] I. F. of Robotics, *World Robotics 2022 Industrial Robots*. 2022.

[13] R. Soares and A. V. R. Lucato, "Robótica colaborativa na indústria 4.0, sua importância e desafio," *Revista Interface Tecnológica*, vol. 18, no. 2, pp. 747–759, 2021.

[14] F. Santoni and A. V. R. Lucato, "Robótica colaborativa: A utilização de robôs nos processos produtivos," *RECIMA21-Revista Cientıfica Multidisciplinar-ISSN 2675-6218*, vol. 1, no. 1, e210914–e210914, 2021.

[15] M. J. Rosenstrauch and J. Krüger, "Safe human-robot-collaboration-introduction and experiment using iso/ts 15066," *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pp. 740–744, 2017.

[16] H. Modares, I. Ranatunga, F. L. Lewis, and D. O. Popa, "Optimized assistive human–robot interaction using reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 46, pp. 655–667, 2016.

[17] G. Dumonteil, G. Manfredi, M. Devy, A. Confetti, and D. Sidobre, "Reactive planning on a collaborative robot for industrial applications," in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 02, 2015, pp. 450–457.

[18] A. Abdi, M. H. Ranjbar, and J. H. Park, "Computer vision-based path planning for robot arms in three-dimensional workspaces using q-learning and neural networks," *Sensors*, vol. 22, no. 5, 2022. DOI: 10.3390/s22051697. [Online]. Available: https://www.mdpi.com/1424-8220/22/5/1697.

[19] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: A deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019. DOI: 10.1109/LRA.2019.2893400.

[20] S. Huang, M. Gao, L. Liu, J. Chen, and J. Zhang, "Collision detection for cobots: A back-input compensation approach," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2022. DOI: 10.1109/TMECH.2022.3169084.

[21] N. Briquet-Kerestedjian, M. Makarov, M. Grossard, and P. Rodriguez-Ayerbe, "Generalized momentum based-observer for robot impact detection — insights and guidelines under characterized uncertainties," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 1282–1287. DOI: 10.1109/CCTA.2017.8062635.

[22] S. Kang, M. Kim, and K. Kim, "Safety monitoring for human robot collaborative workspaces," in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, 2019, pp. 1192–1194. DOI: 10.23919/ICCAS47443.2019.8971756.

[23] L. S. Scimmi, M. Melchiorre, S. Mauro, and S. Pastorelli, "Experimental real-time setup for vision driven hand-over with a collaborative robot," in *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2019, pp. 1–5. DOI: 10.1109/ICCAD46983.2019.9037961.

[24] J. Aleotti, V. Micelli, and S. Caselli, "Comfortable robot to human object handover," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, 2012, pp. 771–776. DOI: 10.1109/ROMAN.2012.6343845.

[25] L. Variz, L. Piardi, P. J. Rodrigues, and P. Leitão, "Machine learning applied to an intelligent and adaptive robotic inspection station," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, IEEE, vol. 1, 2019, pp. 290–295.

[26] E. Kaplanoglu, A. Nasab, E. Erdemir, M. Young, and N. Dayton, "Hand gesture based motion control of collaborative robot in assembly line," in *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, 2021, pp. 1–4. DOI: 10.1109/ICEET53442.2021.9659795.

[27] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, and G. Sansoni, "Deep learning-based hand gesture recognition for collaborative robots," *IEEE Instrumentation & Measurement Magazine*, vol. 22, no. 2, pp. 44–51, 2019. DOI: 10.1109/MIM.2019.8674634.

[28] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, and G. Sansoni, "Deep learning based machine vision: First steps towards a hand gesture recognition set up for collaborative robots," in *2018 Workshop on Metrology for Industry 4.0 and IoT*, 2018, pp. 28–33. DOI: 10.1109/METROI4.2018.8439044.

[29] H. Shin, K. Seo, and S. Rhim, "Allowable maximum safe velocity control based on human-robot distance for collaborative robot," in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 401–405. DOI: 10.1109/URAI.2018.8441887.

[30] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[31] R. Lanzillotti, R. Lanzillotti, and C. SINTZ, "Logica fuzzy: Uma abordagem para reconhecimento de padrão," *PACO EDITORIAL*, 2014.

[32] N. Ahlawat, A. Gautam, N. Sharma, *et al.*, "Use of logic gates to make edge avoider robot," *International Journal of Information & Computation Technology*, vol. 4, no. 6, p. 630, 2014.

[33] W. Kickert and E. Mamdani, "Analysis of a fuzzy logic controller," English, *Fuzzy Sets and Systems*, vol. 1, no. 1, pp. 29–44, 1978, ISSN: 0165-0114.

[34] H.-J. Zimmermann, "Fuzzy control," in *Fuzzy Set Theory—and Its Applications.* Dordrecht: Springer Netherlands, 2001, pp. 223–264, ISBN: 978-94-010-0646-0. [Online]. Available: `https://doi.org/10.1007/978-94-010-0646-0-11%22`.

[35] V. Kumar, B. Nakra, and A. Mittal, "A review on classical and fuzzy pid controllers," *International Journal of Intelligent Control and Systems*, vol. 16, no. 3, pp. 170–181, 2011.

[36] O. Ogorodnikova, "A fuzzy theory in the risk assessment and reduction algorithms for a human centered robotics," in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, 2009, pp. 340–345. DOI: `10.1109/ROMAN.2009.5326293`.

[37] M. N. Hidayati, D. Adzkiya, and H. Nurhadi, "Motion control design and analysis of ur5 collaborative robots using fuzzy logic control (flc) method," in *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, 2021, pp. 162–167. DOI: `10.1109/ICAMIMIA54022.2021.9807732`.

[38] E. P. Beltran, A. A. S. Diwa, B. T. B. Gales, C. E. Perez, C. A. A. Saguisag, and K. K. D. Serrano, "Fuzzy logic-based risk estimation for safe collaborative robots," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2018, pp. 1–5. DOI: `10.1109/HNICEM.2018.8666421`.

[39] L. Roveda, S. Haghshenas, M. Caimmi, N. Pedrocchi, and L. Molinari Tosatti, "Assisting operators in heavy industrial tasks: On the design of an optimized cooperative impedance fuzzy-controller with embedded safety rules," *Frontiers in Robotics and AI*, vol. 6, p. 75, 2019.

[40] E. Bozkuş, İ. Kaya, and M. Yakut, "A fuzzy based model proposal on risk analysis for human-robot interactive systems," in *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, 2022, pp. 1–6.

[41] M. Costanzo, G. De Maria, G. Lettera, and C. Natale, “A multimodal approach to human safety in collaborative robotic workcells,” *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1202–1216, 2021.

[42] B. Venners, “The making of python,” *Artima Developer*, vol. 1, 2003.

[43] L. Tulchak and. rchuk, “History of python,” Ph.D. dissertation, BHTy, 2016.

[44] J. Hao and T. K. Ho, “Machine learning made easy: A review of scikit-learn package in python programming language,” *Journal of Educational and Behavioral Statistics*, vol. 44, no. 3, pp. 348–361, 2019.

[45] A. Goldani Rodrigues Peixoto, “Python2 to 3: migration and the improvement of the WMAgentScripts repository for CMS Computing Operations,” 2021. [Online]. Available: `https://cds.cern.ch/record/2780408`.

[46] Q. Nguyen, *Hands-on application development with PyCharm: Accelerate your python applications using practical coding techniques in PyCharm*. Packt Publishing Ltd, 2019.

[47] E. N. Gayratovich, “Using visual program technology methods in engineering education,” *European Journal of Research and Reflection in Educational Sciences Vol*, vol. 7, no. 10, 2019.

[48] U. Robots, *Universal robot UR3*, `https://www.universal-robots.com/pt/produtos/ur3-robot/`, Oct. 2022.

[49] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, “Analysis of the accuracy and robustness of the leap motion controller,” *Sensors*, vol. 13, 2013. [Online]. Available: `https://www.mdpi.com/1424-8220/13/5/6380`.

[50] I. Efector, *Technology overview - safety light curtains*, `https://www.ifm.com/us/en/us/overview/light-curtains/technology/light-curtains-technology`, Oct. 2022.

[51] B. Drury, *Control techniques drives and controls handbook*. IET, 2001.

Appendix A

Developed Python Algorithms and Routines

https://github.com/Aleborsa/Thesis-Alexandre-Developed-Python-Algorithm

Appendix B

Average Velocity Data of All Routes and Defuzzification Methods

Test number	Square (%)	Round (%)	Collision (%)	3-Dimension (%)
0	69,0536	64,7718	2,7294	65,6360
1	68,2764	66,0086	3,3903	64,1774
2	68,0079	64,0312	3,4694	64,9283
3	67,4510	64,4854	2,7091	66,0280
4	66,7962	63,6790	3,5391	65,2812
5	66,1658	63,9541	3,2472	64,9346
6	67,3318	63,6811	2,0523	66,2136
7	67,3122	62,4494	2,7439	65,6344
8	67,5969	64,8678	2,5433	65,1666
9	67,0415	62,5059	2,1261	65,7740

Table B.1: Mean Velocity Centroid Method.

Test number	Square (%)	Round (%)	Collision (%)	3-Dimension (%)
0	65,2934	63,6188	2,1402	60,6004
1	65,0672	65,2459	2,0940	63,1371
2	65,9162	64,0214	2,5784	60,7183
3	66,1299	64,0732	2,6143	62,2680
4	67,6194	63,3454	2,5749	63,3866
5	67,6579	63,2186	2,6090	62,7139
6	66,6508	62,6853	2,7976	64,3535
7	67,9323	63,4578	2,1973	62,4234
8	66,6231	63,5187	2,4786	64,3157
9	67,4806	62,8688	2,6075	62,6690

Table B.2: Mean Velocity Bisector Method.

Test number	Square (%)	Round (%)	Collision (%)	3-Dimension (%)
0	58,8172	49,5852	2,1431	40,6770
1	57,1813	48,8688	2,2139	46,0585
2	55,6195	49,4074	1,8357	46,0242
3	55,6668	49,5260	2,2943	51,0634
4	55,2245	48,8807	2,0325	46,0090
5	51,8830	49,0116	2,1925	44,9903
6	53,1003	48,8499	1,6876	45,9984
7	56,4232	49,9023	1,7910	42,0585
8	55,9925	49,0090	1,2253	48,8331
9	56,3831	48,9031	1,4056	49,4835

Table B.3: Mean Velocity SoM Method.

Test number	Square (%)	Round (%)	Collision (%)	3-Dimension (%)
0	68,2904	71,4201	2,0655	64,5374
1	70,3614	66,6169	2,0887	63,5978
2	69,6617	67,0168	2,4551	63,8217
3	70,6266	69,2523	1,6673	64,4127
4	69,2054	71,0216	2,4599	63,0094
5	68,4303	69,2958	1,8836	64,3718
6	69,4525	68,8908	2,5336	64,0909
7	66,7756	69,4936	2,2565	63,7576
8	67,4883	68,7693	3,2825	62,7365
9	67,4217	68,5240	2,1818	63,0611

Table B.4: Mean Velocity LoM Method.

Test number	**Square (%)**	**Round (%)**	**Collision (%)**	**3-Dimension (%)**
0	64,5899	58,4732	2,1635	59,4009
1	63,1647	60,3414	1,9179	57,5439
2	62,3412	60,8207	1,9167	57,8895
3	62,3287	60,8150	2,0818	57,8235
4	64,5079	60,4232	2,0015	56,9914
5	63,6911	58,7701	2,0859	57,5545
6	65,5874	58,2534	2,0834	58,2130
7	63,0220	59,4325	1,9155	57,2173
8	63,5043	58,7875	2,2499	57,9698
9	63,8467	61,5474	1,9151	56,9806

Table B.5: Mean Velocity MoM Method.