CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

Autonomous Obstacle-Avoiding Mobile Robot for Human-Robot Collaborative Warehouse Environments

Lucas Costa de Sousa

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientadores:
Milena Faria Pinto
Vinicius Barbosa Schettino

Rio de Janeiro,
Julho, 2025

Autonomous Obstacle-Avoiding Mobile Robot for Human-Robot Collaborative Warehouse Environments

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Lucas Costa de Sousa

Aprovada por:

Presidente, Milena Faria Pinto (Orientadora)
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET/RJ

Vinicius Barbosa Schettino, D.Sc. (Coorientador)
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG

Leonardo Correia Resende, D.Sc.
Instituto Federal do Rio de Janeiro - Campus Paracambi

Wouter Caarls, D.Sc.
Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio

Rio de Janeiro,
Julho, 2025

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

S725 Sousa, Lucas Costa de
Autonomous obstacle-avoiding mobile robot for human-robot collaborative warehouse environments / Lucas Costa de Sousa. — 2025.
xii, 79f. : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2025.
Bibliografia : f. 73-79
Orientador: Milena Faria Pinto
Coorientador: Vinicius Barbosa Schettino

1. Robôs móveis. 2. Objetos - Detecção. 3. Robôs - Sistemas de controle. 4. Redes neurais (Computação). 5. Robótica. I. Pinto, Milena Faria. (Orient.). II. Schettino, Vinicius Barbosa. (Coorient.). III. Título.

CDD 305.896

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

ACKNOWLEDGMENTS

To my advisor Milena, for believing in me like a true friend

To my friends, for the fun good times

To my love Laura, for helping me through difficult times

To my family, for their kind care

To my parents Valdir and Fhabiany, for everything!

Lucas Costa Sousa

A strong person doesn't need to read the future. He makes his own.

Solid Snake

RESUMO

Autonomous Obstacle-Avoiding Mobile Robot for Human-Robot Collaborative Warehouse Environments

Lucas Costa Sousa

Orientadores:
Profa. Milena Faria Pinto, Dsc.
Prof. Vinicius Barbosa Schettino, Dsc.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre em Engenharia Elétrica.

A pesquisa proposta aborda o desenvolvimento de um robô móvel autônomo com capacidade de evitar obstáculos, projetado para operar em ambientes de armazéns colaborativos entre humanos e robôs (do inglês, HRC - Human-Robot Collaborative). O sistema integra lógica fuzzy com uma estrutura de detecção de objetos baseada em Redes Neurais Convolucionais (do inglês, CNN), visando aprimorar a navegação do robô e garantir segurança em cenários dinâmicos. Por meio dessa abordagem, o robô é capaz de ajustar dinamicamente sua velocidade e trajetória para evitar colisões com obstáculos, incluindo humanos e outras entidades móveis, assegurando uma operação segura e eficiente. O objetivo principal é melhorar a segurança operacional sem comprometer a produtividade, superando desafios críticos na robótica colaborativa. Para validação, o sistema foi testado em um ambiente simulado utilizando o Sistema Operacional Robótico (do inglês, ROS) e o software Gazebo, demonstrando eficácia na navegação e no desvio de obstáculos em tempo real. Os resultados obtidos indicam que a proposta permite uma adaptação contínua e interações seguras em ambientes industriais complexos e sujeitos a mudanças, contribuindo para a evolução de soluções robóticas em cenários de colaboração humano-robô.

Fuzzy, CNN, Detecção de objetos, Robô Youbot, Colaboração Segura, Ambiente Compartilhado, Controlador de Velocidade, Navegação autônoma.

ABSTRACT

Autonomous Obstacle-Avoiding Mobile Robot for Human-Robot Collaborative Warehouse Environments

Lucas Costa Sousa

Advisor:
Profa. Milena Faria Pinto, Dsc.
Prof. Vinicius Barbosa Schettino, Dsc.

Master Thesis submitted to Electrical Engineering Graduate Program, Federal Center of Tecnologic Education Celso Suckow da Fonseca, CEFET/RJ, as partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

The proposed research addresses the development of an autonomous obstacle-avoiding mobile robot designed to operate in human-robot collaborative (HRC) warehouse environments. The system integrates fuzzy logic with a Convolutional Neural Network (CNN)-based object detection framework, aiming to enhance robot navigation and ensure safety in dynamic scenarios. Through this approach, the robot can dynamically adjust its speed and trajectory to avoid collisions with obstacles, including humans and other mobile entities, ensuring safe and efficient operation. The primary goal is to improve operational safety without compromising productivity, overcoming critical challenges in collaborative robotics. For validation, the system was tested in a simulated environment using the Robot Operating System (ROS) and Gazebo software, demonstrating effectiveness in real-time navigation and obstacle avoidance. The results indicate that the proposed solution enables continuous adaptation and safe interactions in complex and ever-changing industrial environments, contributing to the advancement of robotic solutions in human-robot collaboration scenarios.

Manipulator, Youbot Robot, Safe Collaboration, Sharable Workspace, Speed Controller, Autonomous Navigation.

Contents

I Introduction **13**

I.1 Initial Considerations and Work Motivation 13

I.2 Main Contributions 16

I.3 Published Works 17

I.4 Organization 18

II Background and Related Works **19**

II.1 Mobile Robot 19

II.2 ROS and Gazebo 21

II.3 Related Works 24

II.3.1 Computer Vision 24

II.3.2 Control Approach 26

II.3.3 Safety Aspects 27

II.3.4 Safety and Navigation Strategies in Collaborative Robotics 28

III Proposed Methodology **32**

III.1 Overall Idea 32

III.2 Safety Threshold Methodology 36

III.3 Simulation with ROS and Gazebo 37

III.3.1 Autonomous Navigation 40

III.3.2 Object Detection 44

III.3.3 Camera Intrinsics Fundamentals 49

III.3.4 Hybrid-Fuzzy Speed Approach 52

III.3.5 Scripts and programming 56

IV Results and Discussion **62**

IV.1 Neural Network Performance 62

IV.2 Autonomous collaborative results 64

IV.3 Safety Conduct Analysis 66

V Conclusions and Future Work **71**

V.1 Concluding Remarks 71

V.2 Future Ideas 72

Bibliographic References 73

List of Tables

II.1 Robot Joints and Characteristics 21
II.2 Related Works. 30

III.1 DWA Local Planner Parameters. 44
III.2 Costmap Parameters. 44
III.3 Hyperparameters used in YOLO's training. 46
III.4 Kinect v1 Intrinsic Parameters (BURRUS, 2011) 50
III.5 Fuzzy rules for angular velocity. 54
III.6 Speed rules for the robot. 54
III.7 Speed multipliers. 55

IV.1 Test Results for Static, Sudden, and Dynamic Environments. 66

List of Figures

II.1 KUKA Youbot's hardware with implementations. Adapted from (SHARMA et al., 2012). 20
II.2 Robot Joints Image (STORE, 2015). 21
II.3 ROS workflow 22
II.4 ROS structured nodes, topics, services and messages process. The image can be found at (ROBOTICS, 2024). 23

III.1 Overview of the proposed method. The target application scenario is a warehouse with a mobile platform equipped with a camera (**A**). This setup enables the implementation of trajectory planning methods that utilize goal point coordinates. The system enables RGB-D odometry-based path planning by leveraging the Kinect Camera to create a costmap from the depth camera's point cloud (**B**). In parallel, the method employs YOLOv3 for object detection to identify specific obstacles in the path. The solution uses the depth camera to measure the distance to the centroid of detected objects while also gathering speed data from the robot's sensors (**C.1**). This information is input into the fuzzy logic module (**C.2**), where it is processed. Based on the object class, robot speed, and distance, the module adjusts the robot's reference velocity while maintaining its trajectory (**D**). 33
III.2 Scheme of fuzzy addition to the speed controller. 34
III.3 Safety threshold circle methodology showing protected zones around (A) human workers, (B) static obstacles, and (C) other robots. The concentric circles represent varying danger thresholds that trigger different response behaviors. 36
III.4 Robot Youbot used in the proposed scenario with an eye-in-hand camera 38
III.5 Proposed test scenario 38
III.6 Active ROS nodes and topics in the simulation. 39
III.7 Standard use of the ROS navigation stack, as described in Granosik et al. (GRANOSIK et al., 2016). The architecture highlights the integration of sensors, planners, and controllers for autonomous navigation. 40
III.8 A suspended toolbox (red circle) placed on a table in front of the robot's path during the navigation phase. Note that while the table itself and the object above it are detected, they are not marked as obstacles in the costmap. However, the table's legs are correctly identified and registered as obstacles (green circles). 41

III.9 Example of a costmap generated by the system, showing static and dynamic obstacles with inflation layers. The inflation radius defines a safety buffer around detected obstacles, ensuring safe navigation. Adapted from West et al. (WEST et al., 2022). 42
III.10 Architecture based on the ROS navigation stack and adapted to use object identification in order to provide safer autonomous movement. 43
III.11 YOLOv3 simplified structure. (**a**) Input image. (**b**) Image resizing. (**c**) Convolutional layers. (**d**) Object detection. 45
III.12 Kinect V1 depth processing method simplified structure. 48
III.13 Fuzzy flowchart implementation in this study. The diagram illustrates the process where precise inputs, such as speed (m/s) and distance (ΔS), are fuzzified into degrees of membership. Fuzzy rules are then applied to determine velocity, and defuzzification converts the fuzzy output into a precise speed value (m/s). 53
III.14 Fuzzification process surface plot. 55
III.15 Membership function of the distance. 56
III.16 Complete pseudocode describing the algorithm's logic pipeline. 58
III.17 (**A**) Initial test phase showing detection zones and safety margins. (**B**) Human movement trajectory during testing. (**C**) Costmap visualization of the moving person, showing real-time obstacle tracking and inflation layers. 60
III.18 Obstacle spawning mechanism: (**Left**) Normal operation, (**Right**) Toolbox appearing at the trigger point, testing emergency response capabilities. 61

IV.1 The mAP values corresponding to the trained Yolov3 Custom Model. 62
IV.2 Evolution of the validation loss during the YOLOv3 training. The steady decrease and stabilization of the curve indicate effective learning and convergence. 63
IV.3 A Real-time costmap generated by the robots Kinect RGB-D camera with objects, highlighting objects, the start of the test, and the average path. 64
IV.4 Comparison of navigation paths in a dynamic warehouse environment. The blue trajectory corresponds to the Default configuration (standard ROS Navigation Stack without semantic perception), while the red trajectory represents the proposed Solution, which integrates object detection (YOLOv4), depth sensing (RGB-D), and fuzzy logic-based adaptive control. 65
IV.5 Minimum distances from the robot to the respective objects. Data was acquired during static toolbox tests. 67
IV.6 Average minimal distances from the robot to the objects. Data was acquired during sudden toolbox appearance tests. 68

IV.7 Estimated minimum distances from the robot to the moving human obstacle. Data was acquired during dynamic tests by analyzing the point of maximum directional change. . 69
IV.8 Path of the robot during static tests (blue). The highest brake intensity (marked in red) happens when a large gap between the standard velocity value and the object detection solution with fuzzy logic output is detected. 69
IV.9 Second braking intensity plot now applied to the sudden obstacle results, adding the sudden toolbox location. 70
IV.10 Braking intensity plot results on a dynamic obstacle environment, in which the person moves in the negative Y-axis position, creating a notably difficult test that highlights the robustness of the solution in the simulation. 70

List of Abbreviations

CNN Convolutional Neural Network . . . 44
CPPS Cyber Physical Production Systems . . . 24
DOF Degrees Of Freedom . . . 19
LIDAR Light Detection And Ranging . . . 24
MF Membership Functions . . . 54
MIMO Multi-Input Multi-Output . . . 14
ROI Region Of Interest . . . 49
ROS Robot Operating System . . . 21, 22, 23, 52
SSD Single Shot Multi-box Detector . . . 25
UAV Unmanned Aerial Vehicle . . . 25
YOLOV3 You Only Look Once Version 3 . . . 44

Chapter I Introduction

I.1 Initial Considerations and Work Motivation

The embrace of Industry 4.0 brought about a paradigm shift in many domains, leading multiple companies to bet on automation and digitization of their manufacturing processes through robots (ELGUEA-AGUINACO et al., 2023). Robots have been used in many fields of industry for inspection (BIUNDINI et al., 2020), monitoring (GAGO et al., 2015), surveillance (PINTO et al., 2020a), home-care (KODATE et al., 2022)and industrial applications (YU; ZHANG, 2022). This wide usability is explained by the fact that robots can reach places that human beings can't or are too risky for them. High/low temperatures or pressure, insalubrious/tiny spaces like swearers and mines, and higher altitudes are examples of these areas (CASTRO et al., 2023). Despite all this danger, humanity needs to explore these locals to develop new technology, to fasten processes, and to maintain systems properly working (PINTO et al., 2020b).

Robots have achieved remarkable progress, such as the ability to replicate human behavior intricately, demonstrating unprecedented adaptability and functionality in operating effectively within hazardous environments where human intervention might pose significant risks (KUMAR et al., 2023; HENTOUT; MAOUDJ; AOUACHE, 2023). The widespread application of robotics is expected in various industries, using robots for diverse operations, ranging from component assembly and welding to painting and packaging. These automated systems execute tasks precisely and efficiently, amplifying productivity and concurrently diminishing operational costs (HAN et al., 2023).

Over the past few decades, various robots have been designed to perform tasks (KUMAR; KUMAR, 2017; SPONG; VIDYASAGAR, 2008). These robots have a specific degree of freedom, allowing them to emulate human-like actions with increasing fidelity. However, as the degree of freedom in robotic manipulators rises, the intricacies of maneuvering them become more pronounced. This poses a formidable challenge for control engineers, demanding the development of robust and effective control designs to ensure precise and reliable manipulation of these advanced robotic systems. Navigating the complexities of controlling high-degree-of-freedom manipulators is essential to advancing the robotics field (KUMAR et al., 2023).

The integration of robots in shared tasks significantly contributes to society (JIN; PAGILLA, 2020). By automating industrial processes, these robots increase efficiency and productivity while reducing risks associated with hazardous tasks. Thus, ensuring safety and implementing robust control strategies are essential for seamless collaboration. Collaborative robotic applications must prioritize the physical safety of human operators and optimize interaction to enhance overall productivity and efficiency. Moreover, integrating sophisticated algorithms and sensors is crucial in augmenting safety measures (SCALERA et

al., 2022). These systems enable robots to anticipate and respond to dynamic environments, ensuring that human-robot interactions remain secure and effective.

The increasing adoption of collaborative robots (cobots) in manufacturing environments has significantly transformed industrial production dynamics, enabling closer interaction between humans and robots. The primary challenge in this kind of application is ensuring worker safety without compromising productivity and efficiency (NTAKOLIA; MOUSTAKIDIS; SIOURAS, 2023). Moreover, integrating sophisticated algorithms and sensors is crucial in augmenting safety measures (SCALERA et al., 2022). In traditional industrial settings, robots and humans are kept apart to maintain safety, achieved through physical barriers and distinct task sequences. However, the scenario of collaborative robotic systems is specifically designed to work alongside human operators and other robots. In this sense, control strategies and sensors are implemented to avoid possible collisions and to ensure safety (BERX et al., 2022). Unlike non-collaborative robots, which traditionally operate based on pre-programmed instructions, cobots play an active role in Human-Robot Collaboration (HRC) and Robot-Robot Collaboration (RRC) (BI et al., 2021).

In many circumstances, it is difficult for robotic systems to execute tasks autonomously due to environmental uncertainties and sensing limitations (JIN; PAGILLA, 2020). In this sense, robust and effective control design is a significant challenge. In such situations, heuristic-based methods emerge as a practical alternative. The literature has already suggested that heuristic-based approaches in robotics have proven their superiority over classical techniques and graph-based/grid-based search approaches (HENTOUT; MAOUDJ; AOUACHE, 2023; CHETTY; PONNAMBALAM, 2012).

A robotic speed control system is a Multi-Input Multi-Output (MIMO) (Multiple-Input Multiple-Output), highly nonlinear, and coupled system. In this sense, the design of an efficient controller is not trivial. Conventional speed controllers often fail to effectively control nonlinear, uncertain, and coupled systems despite their advantages (ELSHEIKH; ABDO, 2022), especially in crowded dynamic environments with obstacles. Therefore, the development of fuzzy controllers represents a paradigm shift in control engineering due to their nonlinear linguistic mapping between inputs and outputs. Fuzzy logic is a noteworthy heuristic approach that devises control strategies using an "IF-THEN" rule base. This flexible approach has been successfully applied to control and maneuver various robot-like systems (CHEN et al., 2021; ANDRADE et al., 2021). Inspired by human decision-making and intuitive reasoning, fuzzy logic can handle uncertainties in the model, inaccurate information, and mechanical inconsistencies in the robot's structure, all without requiring the solving of differential equations (HENTOUT; MAOUDJ; AOUACHE, 2023).

When manipulating objects in a busy environment, it is necessary to adjust the stability of the robots' movement while avoiding obstacles to prevent accidents. Although speed drivers have a simple structure, changing their settings requires careful attention from designers, especially when non-linearities

are present in the model. This topic has received significant attention in the literature (HENTOUT; MAOUDJ; AOUACHE, 2023; LOCHAN; ROY, 2014). Fuzzy logic (FL) theory is particularly suited for systems that are difficult to model due to their non-linearities. Many researchers have applied fuzzy logic to achieve better performance and robustness compared to classical control algorithms (DONG; HE, 2019; LIU et al., 2022). Another critical aspect to consider is the need for sensors, such as cameras, to provide the robot with information about nearby objects, persons, or blockades. The integration of cameras is essential to enhance the robot's functionality. These cameras serve as sensory inputs, providing crucial visual information to the manipulator for tasks such as object positioning and identification (GIUFFRIDA; MEONI; FANUCCI, 2019). By capturing images of the surrounding environment, the cameras enable the system to calculate the precise position of objects, facilitating informed decision-making in real time.

Developing an effective navigation approach for autonomous mobile robots or assistive systems while ensuring their safety and security remains a critical challenge in autonomous mobility (LI et al., 2024).Computer vision plays a crucial role in these systems, allowing robots to process and understand visual data from their surroundings for intelligent decision-making (SILVA et al., 2022). Cameras may be fixed, attached to the robots base, or mounted on the end-effector of a robotic arm, commonly known as an eye-in-hand configuration. This configuration provides advantages such as delivering high-resolution point clouds of nearby obstacles and simplifying algorithms by avoiding the need for coordinate transformations, which our research will implement to ensure the highest levels of safety and efficiency in HRC (WANG et al., 2017).

The camera is also used for mapping, enabling the robot to create detailed, real-time maps crucial for navigating complex environments with varying levels of uncertainty and ensuring that the navigation path is always optimal and safe (LABBÉ; MICHAUD, 2019). Through the visual Self-Locating and Mapping (SLAM) approach with the eye-in-hand camera configuration, our system continuously refines its map as the robot moves through its environment. This dynamic mapping capability ensures that the robot can adjust its path on the fly, avoiding newly detected obstacles and optimizing its route to maintain both safety and efficiency. The high-resolution maps generated through this process are critical for effective HRC, where the safety of human workers depends on the robots ability to perceive and respond to its surroundings accurately.

The adoption of cobots in manufacturing environments to assure the safety of human workers without compromising productivity is an increased research topic (BI et al., 2021). As stated in Liu et al. (LIU et al., 2024), there is no completely safe robot, only secure applications. Hence, many scholars' studies are based on the capabilities of cobots developed and the safety of their application in manufacturing, not related to the robot design itself. Traditional safety measures, such as physical barriers, are inadequate in scenarios where robots and humans must work closely together.

In this sense, this work combines two safety mechanisms, particularly Speed and Separation Monitoring, to allow for dynamic mapping and real-time adaptation to ensure safe navigation and obstacle avoidance in complex environments. Besides, this work proposes using the Convolutional Neural Network (CNN) You Only Look Once Version 3 (YOLOv3) algorithm to identify the objects in the surroundings. The Tiny version of YOLOv3 was chosen for its lightweight architecture and reduced computational requirements, making it suitable for real-time applications on resource-constrained platforms. Through object detection, the system created a virtual protective boundary with a specified radius around the robot, establishing a safe zone. Incorporating CNN algorithms further augments the Autonomous Guided Vehicle (AGV)'s capabilities by allowing it to discern and interpret the visual data, contributing to the efficiency and adaptability of the AGV's decision-making process due to its capacity for learning complex patterns and features from visual data (WONG et al., 2022).

The vision approach employed to support obstacle avoidance, YOLOv3, is based on the speed detection of a single-stage detector like this one. Unlike traditional models like R-CNN and its variants, which use a two-stage process (region proposal followed by classification), YOLO simplifies the task into a single-stage network that simultaneously predicts bounding boxes and class probabilities, making it significantly faster, enhancing detection feedback and achieving better results compared to other models specially Region-based CNN (R-CNN) (DU, 2018), Fast-RCNN (COELHO et al., 2024), and Faster-R-CNN (DIWAN; ANIRUDH; TEMBHURNE, 2023). Besides, the YOLOv3 model requires fewer computational resources compared to the newer versions of YOLO, which makes it particularly suitable for environments with hardware constraints, and it ensures real-time object detection without the need for extensive hardware upgrades.

I.2 Main Contributions

The integration of autonomous mobile robots in collaborative environments represents a significant advancement in industrial automation and human-robot interaction. This research addresses the challenges of safe and efficient navigation in dynamic settings by proposing a novel approach that combines computer vision, depth sensing, and a fuzzy logic module to assist navigation. The system uses the You Only Look Once Version 3 (YOLOv3) algorithm to detect and classify objects in real time. Simultaneously, a Kinect RGB-D camera is used to measure the distance from the camera to the nearest point in the generated point cloud corresponding to the centroid of the detected object. This sensory input is integrated with a costmap, generated using the RGB-D camera, to enhance the robot's autonomous navigation capabilities.

The proposed system dynamically adjusts the robot's speed based on the current speed, the detected object's class, distance, and the surrounding environment. For instance, when a human is detected, the robot reduces its speed to ensure safety, whereas for less critical objects, it maintains higher speeds to

optimize productivity. This adaptive behavior is achieved through a fuzzy logic module that processes real-time data on object distance, speed, and class, providing a robust and flexible solution for dynamic environments.

Furthermore, the system effectively avoids obstacles in static and dynamic scenarios. The robot can navigate complex environments by integrating object detection, depth sensing, and fuzzy logic while minimizing collisions and maintaining smooth trajectories. The approach is validated through extensive simulations in the Gazebo environment using the Robotic Operating System (ROS), showcasing its ability to balance safety and efficiency in real-world applications.

The main contributions of this research can be summarized as follows:

- **Integration of YOLOv3 and depth sensing**: The system uses YOLOv3 for real-time object detection and classification, combined with a Kinect RGB-D camera to measure distances to detected objects. This enables precise navigation and speed control based on object proximity and class.
- **Enhanced autonomous navigation with costmaps**: The RGB-D camera generates a costmap that feeds into the robot's autonomous navigation algorithm, improving obstacle avoidance and path planning in complex environments.
- **Development of a fuzzy logic-based speed control strategy**: The system dynamically adjusts the robot's speed using real-time data on object distance, speed, and class. This approach provides a robust and adaptable solution for non-linear systems, particularly in dynamic environments.
- **Simulation-based validation**: The proposed strategy is evaluated through simulations in the Gazebo software and ROS environment (KOENIG; HOWARD, 2004). The results highlight the system's ability to navigate complex environments while maintaining smooth trajectories and minimizing collisions.

I.3 Published Works

This dissertation has yielded two significant scientific publications, which complement the research presented herein. These papers cover different but interconnected aspects of the proposed system, contributing to the broader scientific discussion on autonomous navigation and human-robot collaboration.

The first publication was presented at an international IEEE conference:

- Sousa, L. C., Pinto, M. F., & Schettino, V. B. (2023). Autonomous Mobile Manipulator for Obstacle Avoidance and Safe Trajectory Planning in Collaborative Environments. In Proceedings of the 2023 Latin American Robotics Symposium (LARS) and Brazilian Symposium on Robotics (SBR). IEEE. DOI: 10.1109/LARS-SBR57541.2023.10837923

This paper introduced the preliminary architecture of the mobile robotic system, detailing its safe navigation capabilities and integration with ROS and Gazebo simulation environments.

The second contribution was published in a peer-reviewed scientific journal:

- Sousa, L. C., Pinto, M. F., Schettino, V. B., Zachi, A. (2025). A Fuzzy Logic-Based Safe Navigation Framework for Mobile Manipulators in Human-Robot Collaborative Environments. Sensors, 25(8), 2387. DOI: 10.3390/s25082387

  This article extends the research by proposing a fuzzy logic-based control strategy for speed adaptation, integrating object classification via YOLOv3 and distance measurements from an RGB-D camera to enable real-time obstacle avoidance and safety regulation in shared workspaces.

I.4 Organization

The remainder of this dissertation is organized as follows:

- **Chapter II**: Provides a comprehensive review of related works, focusing on computer vision for object detection, depth sensing, fuzzy logic control systems, and autonomous navigation in collaborative environments.
- **Chapter III**: Details the proposed approach, including the integration of YOLOv3 for object detection, the use of the Kinect RGB-D camera for distance measurement, the generation of costmaps, and the design of the fuzzy logic-based speed control system.
- **Chapter IV**: Presents the results of the simulations, including performance metrics such as completion rate, speed, distance, and collision avoidance. The chapter also discusses the implications of the findings for industrial applications.
- **Chapter V**: Concludes the dissertation, summarizing the key contributions and suggesting directions for future research, such as integrating additional sensors (e.g., LiDAR) and testing in real-world industrial environments.

Chapter II Background and Related Works

This chapter will present the background and all mathematical foundations necessary to support understanding the proposal for this master's qualification. Besides, at the end of this chapter, the related works will be presented to support the contributions claimed in this work.

II.1 Mobile Robot

The selected robot for this research is the KUKA Youbot, a mobile robot equipped with a five Degrees of Freedom (DOF) robotic manipulator. This choice stems from its widespread use in industrial research and applications as a result of its versatility, compact design, and seamless integration capabilities with various sensors and peripherals. During this study, we plan to leverage these qualities by incorporating sensors such depth cameras to enhance its functionality and performance.

For this project, we are establishing a foundational model of the KUKA Youbot, which will be further enhanced through hardware integrations and adaptations to meet the specific needs of our objectives. Here's an overview of the hardware configurations employed (SHARMA et al., 2012):

- **Base:** The base of the Youbot is equipped with omnidirectional wheels, enabling smooth and precise movement in any direction. This mobility is essential for navigating dynamic warehouse environments where the robot must avoid obstacles while maintaining efficient operation.
- **Depth Camera:** A Kinect depth camera is integrated into the system to provide real-time 3D perception of the environment. This camera captures both RGB and depth data, enabling the robot to detect obstacles, measure distances, and create a detailed costmap for navigation. The depth camera is mounted on the robot's base or end-effector, depending on the application requirements.
- **Sensors:** The Youbot is equipped with essential sensors, such as odometry, to enhance localization. These work alongside the depth camera to improve obstacle detection and ensure safe navigation. The system prioritizes a minimalistic sensor suite, focusing on a versatile and scalable infrastructure. By relying on the depth camera for perception and odometry for localization, the system reduces hardware complexity while maintaining robust performance. This approach ensures adaptability to various industrial applications and allows for future upgrades.
- **Processing Unit:** A powerful processing unit is integrated into the system to handle the computational demands of real-time obstacle detection and navigation. This unit processes data from the depth camera and other sensors, runs the YOLOv3 algorithm for object detection, and implements fuzzy

logic for adaptive speed control. The processing unit ensures seamless integration of sensor data and decision-making tasks.

- **Software Integration:** The system utilizes the Robot Operating System (ROS) for seamless integration of hardware components and software modules. ROS enables efficient communication between the depth camera, sensors, processing unit, and navigation algorithms, ensuring robust and reliable operation in dynamic warehouse environments.

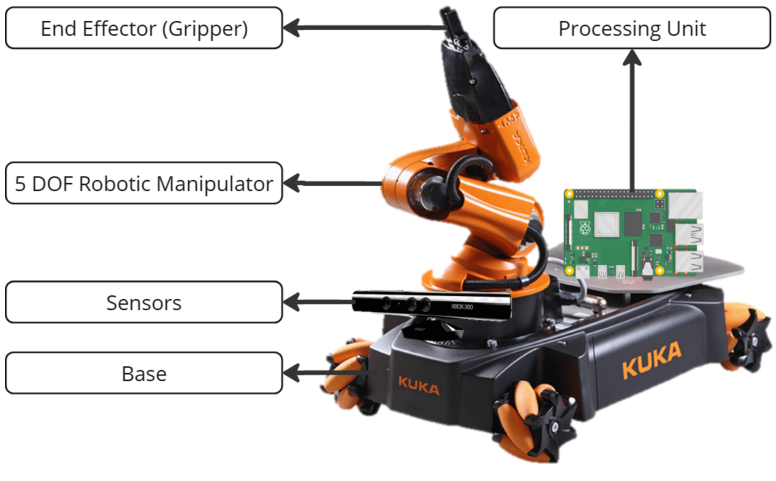


Figure II.1: KUKA Youbot's hardware with implementations. Adapted from (SHARMA et al., 2012).

In addition to being a robust and adaptable Robot, the KUKA Youbot boasts versatile manipulation capabilities owing to its 5-degree-of-freedom robotic arm and customized gripper options. The extending arm enables precise manipulation tasks within confined spaces common in industrial settings, and the gripper grants agile and efficient handling of objects of varying shapes, sizes, and weights, allowing the robot to perform a diverse range of tasks desired in an industrial field.

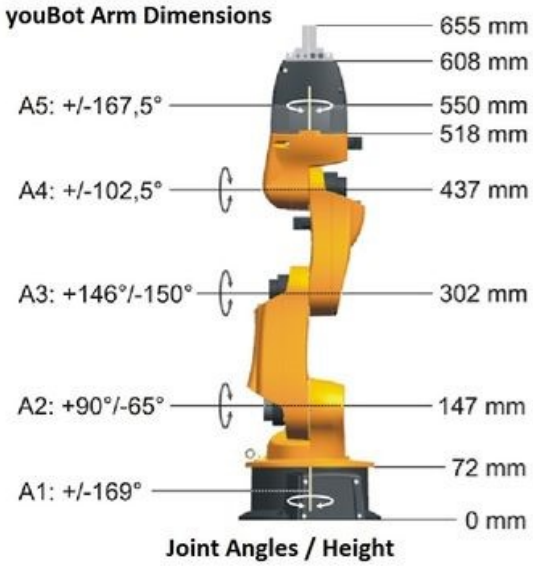


Figure II.2: Robot Joints Image (STORE, 2015).

Joints	Orientation	Max Torque
A1	0° 0° 180°	9.5 Nm
A2	-90° 0° 90°	9.5Nm
A3	-90° 0° 0°	6.0 Nm
A4	0° 0° 0°	2.0 Nm
A5	0° 0° -90°	1.0 Nm

Table II.1: Robot Joints and Characteristics

II.2 ROS and Gazebo

One of the greatest challenges encountered in engineering, particularly in the field of control and automation engineering, lies in the compatibility of equipment, instruments, and electronics in general, concerning system communication, to perform control, supervision, maintenance, optimization, and improvements. It is often necessary, in the vast majority of cases, to use components from similar brands and models, but restricted in their variability due to the possibility of communication through the same protocol, which, due to technical limitations, sometimes results in a lagged choice to compensate for the use of obsolete equipment.

At times, not only in the field of automation and instrumentation, but also in robotics, engineers must assess the positive and negative aspects, sometimes having to forego benefits in favor of an optimized and viable process. Within the realm of robotics, the wide variety of components and controllers complicates compatibility both in the electronic and software parts. With the growing demand caused by specificity, a multi-language architecture was developed to standardize and distribute tools using a robust and reliable development system, named Robot Operating System (ROS).

ROS is a set of libraries and tools contained in an open-source software that allows simplified

communication of various hardware and software controllers, achieving compatibility without the use of low-level programming languages. Developed by Stanford University, the operating system consists of a collection of various software frameworks that enable the creation of modular applications and services, called nodes, which operate variably and flexibly in dynamic systems. In addition to managing packages, ROS primarily acts as an operating system capable of abstracting hardware from a system. In summary, the system adapts tools and software aiming to "simplify the task of creating complex and robust robotic behavior on a wide variety of robotic platforms." II.3 presents a image specifying the relation between the ROS communication and workflow process, including the default workflow and the expanded scope use.

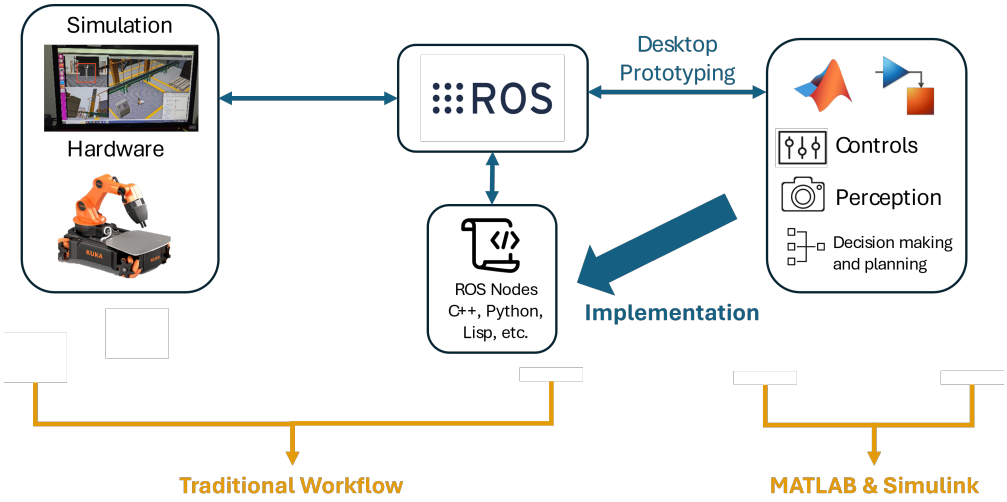


Figure II.3: ROS workflow

Traditional ROS workflow can be divided in process that describes the nodes, topics, services and messages:

- **Nodes** Nodes are individual computational units within ROS. Each node is responsible for a specific task, such as sensor data processing, user interface, control algorithm or just publishing and receiving details, notes or reports about system sections, Nodes can be divided between publisher, in which publishes specific customizable data, and subscriber nodes that receives and process the information regarding user needs;
- **Topics** Topics works discrediting node data and isolating a message. It enables asynchronous communication between nodes, in which can work in parallel with topics, publishing messages to a topic or subscribing to receive it's messages;
- **Services** Services acts defining a pair of messages demanded by the client, providing a synchronous request-response communication mechanism between nodes. Services differ from topics as they requires individual requests to perform specific tasks or retreive information. Services are userful for operations that require direct interaction of immediate feedback.

- **Messages** These messages published on topics are structured data types defines in ROS message. In this project, several message files are structured and called within scripts to define data architecture before sending to a topic;

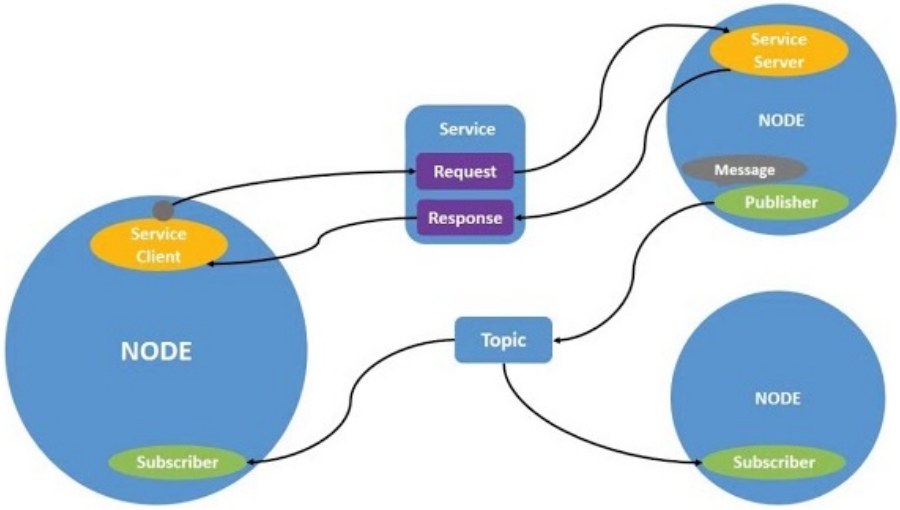


Figure II.4: ROS structured nodes, topics, services and messages process. The image can be found at (ROBOTICS, 2024).

For designing prototypes and developing tools in robotics, researchers commonly use techniques and software to help create a controlled environment in which test subjects and ideas can be widely tested in a simulated aspect, mirroring a real-world scenario. To this extent, graphical and physics engines are integrated with robotics software and sensors. In this project, the Gazebo engine is used. The main reason for this common choice is its complex robotic systems integration, realistic physics simulation of gravity, friction, collisions, and even sensor noise. Another aspect is its ROS integration and Multi-Robot simulation support, making it a perfect candidate for prototyping and testing any kind of robot application under development.

In this project, ROS1 Kinetic was selected as the middleware due to its official support for the KUKA YouBot through the youbot_driver package. According to the ROS Wiki, Kinetic is the latest ROS distribution with stable and maintained support for this robot platform, making it a necessary choice to ensure full hardware and simulation compatibility. Consequently, Python 2.7 and OpenCV 3 were used, as they are the default and compatible versions for ROS Kinetic in Ubuntu 16.04. For object detection, YOLOv3 was integrated, primarily because it maintains reliable performance while being compatible with the OpenCV version supported in this environment. These decisions were driven by compatibility constraints and the need to maintain a stable development workflow while integrating computer vision and robotic control systems under a unified ROS framework.

II.3 Related Works

II.3.1 Computer Vision

Bonci et al. (BONCI et al., 2021) highlighted in their work the evolution of collaborative environments in production settings, moving beyond the traditional concept of Cyber Physical Production Systems (CPPS). While CPPS integrates cyber aspects into physical systems, typically treating humans as intruders in automated tasks, the future trend points towards Cyber Physical Human Systems (CPHS), where the control, communication, and automation technologies converge with human involvement. This evolution signifies a shift from systems that react to the presence of humans to ones that actively engage and collaborate with them, representing a significant advancement in human-robot interaction within production environments.

Computer vision plays a crucial role in autonomous systems by allowing them to capture and comprehend visual information from their immediate environment, which is essential for making informed decisions. Recognizing objects within images and determining optimal grasping strategies are just a few examples of the tasks enabled by computer vision in autonomous systems (BENGTSON et al., 2020)

With the use of exteroceptive sensors, the robot can have the perception of the 3D external environment. However, the sensor arrangements can also be eye-in-hand instead of eye-to-hand setup. In eye-in-hand configurations, sensors are mounted on the robot's end-effector, offering advantages such as a higher resolution point cloud of obstacles and simpler algorithms due to no need for coordinate transformations. However, these setups suffer from limited viewing angles and potential occlusions, resulting in blind spots. Conversely, in eye-to-hand setups, sensors are positioned outside the robot's workspace, continuously surveilling the entire workspace. In our research work, we will use both sensor arrangements to guarantee safety and effectiveness for this HRC (WANG et al., 2017).

Note that in recent years, great attention has been paid to human and robot collaboration, especially vision and distance sensors. The monocular RGB, RGB-D, and Light Detection and Ranging (LiDAR) are the most employed sensors. However, the performance of vision sensors can be very affected by environmental conditions, which makes them generally used to check the workspace, detecting the presence of objects, humans, and robots (BONCI et al., 2021).

Various studies have explored the utilization of Kinect RGB-D sensors, leveraging their capacity to capture RGB and depth space imagery for reconstructing 3D environments. In the work of Flacco et al. (FABRIZIO; LUCA, 2016), the authors developed a method for computing distances among various points and dynamic obstacles, employing multiple depth cameras akin to Kinect. By integrating robot kinematics, they distinguished the robot's point cloud data within the scene, facilitating collision avoidance through repulsive vectors during motion tasks.

Brito et al. (BRITO et al., 2018) used the the Kinect sensor to play a crucial role in incorporating real-

world obstacle data into a virtual environment, simulating scenarios for testing re-planning algorithms and human-robot interaction under safe conditions. This approach aimed to facilitate collaboration between humans and robots by modeling potential scenarios.

Note that computer vision algorithms further augment the robot's capabilities by allowing it to discern and interpret the visual data. This integration empowers the robotic system to not only locate objects but also extract meaningful information for collaborative tasks with humans or other robotic entities (BENGTSON et al., 2020). Incorporating machine vision algorithms is invaluable in this context (ZHANG et al., 2020; WONG et al., 2022). Convolutional Neural Networks (CNN), with their capacity for learning complex patterns and features from visual data, contribute to the efficiency and adaptability of the manipulator's decision-making process. The combination of camera inputs, computer vision algorithms, and neural network capabilities forms a comprehensive framework that enhances the manipulator's overall performance in object recognition and positioning tasks (GIUFFRIDA; MEONI; FANUCCI, 2019; WONG et al., 2022; ZHANG et al., 2020).

At present, there are many methods for object detection based on deep convolutional neural networks, such as YOLO (Redmon et al., 2016), Single shot multi-box detector (SSD) (LIU et al., 2016), and faster R-CNN (GIRSHICK, 2015). These techniques have become increasingly prevalent in practical applications, with YOLO, in particular, being a focal point in this study. Noteworthy contributions in recent years include Ramos et al. (RAMOS et al., 2022), who proposed using YOLOv3 to assist the autonomous operation of a Unmanned Aerial Vehicle (UAV) in an offshore task. Silva et al. (SILVA et al., 2022) utilized a YOLOv3 to facilitate object tracking for UAVs. Giuffrida et al. (GIUFFRIDA; MEONI; FANUCCI, 2019) showcased the application of YOLOv2 in recognizing objects captured by a camera mounted on a robotic arm's end-effector, enhancing user interaction capabilities. Xu, Fei, et al (XU et al., 2024), in particular, explored the use of YOLOv3-tiny for real-time object detection in low-performance systems, such as the Raspberry Pi 5, highlighting its suitability for resource-constrained platforms. This demonstrates the versatility and effectiveness of YOLO across various domains, including robot navigation, object tracking, and human-robot interaction.

The incorporation of cameras and computer vision algorithms is proposed in this research to empower the robotic system with enhanced sensory inputs. The paper specifically suggests using the YOLO algorithm for object detection, contributing to the robustness of the robot's performance. The sensory inputs come from eye-in-hand and eye-to-hand configurations. In this sense, it allows the robot not only to locate objects and recognize obstacles on its path, but also to extract meaningful information for collaborative tasks with humans or other robotic entities;, providing a more robust and adaptable solution, especially in the presence of non-linearities.

II.3.2 Control Approach

Introducing autonomous behavior into mobile robots enhances navigation efficiency and ensures safety in dynamic environments, particularly in human-robot collaborative settings. As stated in Goodrich and Schultz (GOODRICH; SCHULTZ, 2007), autonomous behavior is a multi-stage process that includes perception, decision-making, and action execution. The first stage involves information acquisition, where sensory inputs like depth cameras and odometry gather data about the environment. The second stage, information analysis, consists in processing this data using computer vision algorithms like YOLOv3 to detect obstacles and humans. In the third stage, decision selection, the system determines the appropriate navigation strategy, such as adjusting speed or altering the path to avoid obstacles. Finally, the action implementation stage executes the chosen strategy using a fuzzy logic-based speed controller, ensuring smooth and safe navigation.

In our research, the first stage relies on the Kinect RGB-D camera to capture real-time depth and RGB data. This data is processed by the YOLOv3 algorithm, which detects objects and humans in the environment. A virtual protection zone is created around detected obstacles, ensuring a safety buffer during navigation. When the robot approaches this zone, the decision-making stage activates, determining whether to adjust speed or change trajectory. The action implementation stage employs a fuzzy logic-based speed controller to execute these decisions, dynamically adjusting the robot's speed based on proximity to obstacles and environmental conditions.

Traditional navigation systems often rely on pre-programmed paths and odometry-based control, which can struggle with dynamic and uncertain environments. To address these limitations, our approach integrates fuzzy logic with real-time object detection and depth sensing. Fuzzy logic is particularly effective in handling nonlinearities and uncertainties, making it well-suited for adaptive speed control in complex environments. For instance, Jin and Choi (JIN; CHOI, 2011) demonstrated the effectiveness of a fuzzy logic system for obstacle avoidance in mobile robots, highlighting its ability to adapt to dynamic environments. Similarly, Saidi et al. (SAIDI et al., 2022) proposed a real-time fuzzy-PID controller for mobile robot navigation, combining fuzzy logic with traditional control methods to achieve robust obstacle avoidance in real-world scenarios.

Our proposed system builds on these advancements by integrating fuzzy logic with YOLOv3-based object detection and depth sensing. This combination allows the robot to dynamically adjust its speed and trajectory, ensuring safe and efficient navigation in warehouse environments. The fuzzy logic controller uses inputs such as distance to obstacles, robot speed, and object classification to generate adaptive speed commands. This approach minimizes the need for abrupt braking or sharp turns, resulting in smoother and more efficient navigation.

II.3.3 Safety Aspects

A critical component of the HRC scenario is ensuring human safety in the shareable workspace. Adaptable systems offer significant benefits in terms of flexibility and scalability in manufacturing. However, they bring about new and complex types of faults that may arise from human behaviors as well as from the different system components (MUKHERJEE et al., 2022). In this sense, the inherent complexity in designing such safety frameworks presents a significant challenge for HRC.

Several standards and definitions were proposed to design safety collaborative workspaces. For example, Wang et al. (WANG et al., 2019) outline safety regulations, categorize failures originating from engineering, human, and environmental sources, and conduct a comprehensive literature review on collision detection and avoidance techniques. Similarly, Robla et al. (ROBLA-GÓMEZ et al., 2017) offers a detailed review that introduces a framework for industrial safety across all interaction levels, incorporating both control-based and machine learning-based strategies, along with developing materials and sensors for industrial robots. The researchers in Askarpour et al. (ASKARPOUR et al., 2019) suggested a classification system for errors arising from human behavior and neglecting proper procedures during manufacturing processes.

Many attempts have been made to classify safety control factors in HRC. As stated in Robla et al. (ROBLA-GÓMEZ et al., 2017), the control safety aspects involve psychological and physical issues. Psychological factors concern the human partners' lack of trust in the robot, while physical factors involve elements that could pose risks to the human partner's well-being. In both safety control factors, it is necessary to ensure the reliability and predictability of the robot's actions.

According to Mukherjee et al. (MUKHERJEE et al., 2022), four collaborative modes have been outlined in ISO 10218-1/2, which are: (i) safety-rated monitored stop, (ii) Speed and separation monitoring, and (iii) power and force limiting controls. The first one, a safety-rated monitored stop, is the simplest solution for avoiding a collision. In this approach, the robot stops when it detects a human. In the speed and separation monitoring, the robot performs its mission regularly, but when the human part is closer, it reduces speed (VILLANI et al., 2018). In the power and force limiting controls, the robot must restrict both static and dynamic forces by regulating the motor power to ensure safe interaction with humans (AIVALIOTIS et al., 2019).

The collaborative mode prescribed in our proposed solution is an insertion of speed and separation monitoring mode. Integrating fuzzy logic with a CNN allows the robot to dynamically adjust its speed as it navigates and encounters objects, ensuring a safer interaction within a collaborative environment, particularly when humans are nearby. Several other solutions were proposed in the literature in this kind of mode. For instance, (LONG et al., 2017) designed a solution with a nominal mode (i.e., maximum velocity), a coexistence mode (i.e., reduced velocity), and a gravity compensation mode, where the transitions among modes are triggered by three lasers, two of which are mechanically linked to the

robot. Zang et al. (ZHANG et al., 2016) generated a collision-free path through the potential fields method using data collected from a 3D camera. The relative position and velocity of the human partner are used as a safety metric in Zanchettin et al. (ZANCHETTIN et al., 2015).

II.3.4 Safety and Navigation Strategies in Collaborative Robotics

Traditionally, robots have functioned as isolated systems within restricted production environments. However, with the rise of Industry 4.0, there has been a significant shift towards collaborative robotics, emphasizing human-robot interaction (HRI). An in-depth survey on HRC in industrial applications can be found in (VILLANI et al., 2018), while a comprehensive review of HRC interfaces and interaction methods is detailed in (AJOUDANI et al., 2018). Note that the system engineering methodology outlined in ISO 17757 serves as an effective framework to manage fully autonomous or mixed fleet operations (TIUSANEN; MALM; RONKAINEN, 2020).

Note that the issue of safety in autonomous navigation remains an area with significant gaps (ABDALLAOUI et al., 2022). As mobile robots become more prevalent in industrial settings, particularly where they share space with human workers and undertake critical tasks, ensuring their safe operation has become a pressing concern. Various approaches have been investigated to address this challenge, including sensor-based detection systems (De Ryck; VERSTEYHE; DEBROUWERE, 2020), advanced path-planning algorithms (DU et al., 2022), and robust fail-safe protocols (SOBREIRA et al., 2016). In the past few years, there has been significant research on mobile robots' autonomous navigation in industrial settings. Typically, these applications are employed to improve efficiency, reduce operational costs, and ensure safety in daily operational activities, such as material handling (ZHAO; CHIDAMBARESWARAN, 2023), inspection (SZREK; JAKUBIAK; ZIMROZ, 2022), and assembly (SOBREIRA et al., 2020). Different approaches have been proposed for autonomous navigation, including SLAM algorithms (ABDULSAHEB; KADHIM, 2023), sensor fusion techniques (OU et al., 2024), and perception systems based on machine learning (OU et al., 2024).

As the industrial environment constantly changes, the methods based on pre-defined paths and navigation markers are not well-oriented for safety or efficiency. In order to solve it, the research has explored the integration of Light Detection and Ranging (LIDAR) (AGUIAR et al., 2022), ultrasonic sensors (RACHKOV; WU, 2024), and cameras (DANG et al., 2023) to improve obstacle detection and avoidance capabilities, as well as the use of deep learning models for object recognition and decision-making in real-time (ZHANG et al., 2023).

Besides, several algorithms were proposed to assist robots in autonomous navigation, allowing robots to adjust their path in the presence of statics or moving obstacles (WANG et al., 2024). SLAM algorithms enable mobile robots to determine their position while simultaneously creating a map of their environment using sensors such as LIDAR, cameras, and odometry and have advanced to a

level suitable for deployment in AGVs. LIDARs collision avoidance capabilities can be leveraged for sophisticated, multi-tiered obstacle avoidance during movement (GRAU et al., 2020). There are also being explored predictive models that forecast potential collisions by adapting the robot's speed based on the environment sensing in the distance defined robot zone defined from the workers or objects (BAYONA; SIERRA-GARCÍA; SANTOS, 2023) (ZHOU et al., 2023).

While human-robot collaboration began gaining traction only years after robots were introduced in manufacturing, the issue of robots avoiding obstacles has been a persistent challenge since their early days in the industry. The authors of Ali et al. (ALI; KIM; KIM, 2015) proposed the use of Fuzzy Logic (FL) to calculate collision risk degree to avoid obstacles for unmanned roads. In (TALHA et al., 2019), the authors applied FL to introduce safety aspects to landing. Zhang et al. (ZHANG et al., 2022) introduced a vector field histogram method that builds on the artificial potential field concept. In their approach, each grid in the map is assigned a specific value representing the likelihood of an obstacle being present in that location. During the planning process, this environmental data is transformed into a one-dimensional histogram, which is then used with a cost function to determine the optimal driving direction. The authors of Sun et al. (SUN et al., 2023) developed a collision avoidance control method using the Dynamic Window Approach (DWA), simultaneously generating feasible dynamic paths and collision avoidance control measures while accounting for potential collision risks. However, neither strategy was tested in the HRC scenario.

The authors of (KULIĆ; CROFT, 2006) estimated the danger levels in HRC in real-time and generated an alternative robot trajectory when danger exceeded a determined threshold, which combine factors like distance, velocity, and inertia between robot links and humans. A virtual force was used to redirect the robot away from high-risk zones based on relative distance and effective impedance. Subsequent work (KULIC; CROFT, 2007) enhanced this system by incorporating human monitoring, which improved safety through an integrated path planner and motion controller. An interesting approach was proposed in (AVANZINI et al., 2014). The authors presented a distributed distance sensor mounted on an industrial robot's surface as part of the active control strategy. They also designed an avoid collision control algorithm with the possibly detected humans.

As stated in (FAN; ZHENG; LI, 2022), there is a lack of a holistic consideration of other crucial elements of a working scene, especially when taking a further step towards Proactive HRC. In this sense, they proposed a systematic review of the use of vision-based holistic scene understanding to proactively collaborate in manufacturing environments. They suggested using CNNs for object recognition, human activities, and visual scene understanding. Regarding the use of CNN for obstacle detection, the authors of Alkhaleefah et al. (ALKHALEEFAH et al., 2021) presented a lightweight YOLOv3-mobile network by refining the architecture of YOLOv3-tiny to improve pedestrian detection. In (SOEBHAKTI et al., 2019), the authors used YOLOv3 to detect objects in a mobile soccer robot scenario.

These methods primarily operate on 2D image data and lack depth perception, making it difficult to differentiate between low-lying barriers and overhead obstacles. Henke Dos Reis et al. (REIS et al., 2019) explored a method that integrates YOLO-based object detection with RGB-D images for autonomous navigation, demonstrating that depth information enhances obstacle differentiation. RGB-D sensors have been integrated into navigation frameworks to mitigate this limitation and provide depth-aware obstacle detection. Studies such as (DANG et al., 2023) and (LóPEZ-VELáSQUEZ; ACOSTA-AMAYA; JIMéNEZ-BUILES, 2024) demonstrate that combining CNN-based object recognition with depth information significantly improves navigation accuracy. However, most of these works do not explicitly incorporate height-based decision-making, which is crucial for 3D-aware motion planning.

To handle the uncertainty inherent in dynamic environments, FL has been widely explored for adaptive navigation control (MOON; JAJULWAR et al., 2016). Traditional approaches apply fuzzy controllers for local trajectory adjustments, velocity regulation, and collision risk assessment (ALI; KIM; KIM, 2015). However, these methods often lack integration with modern deep learning-based perception systems. Lopez-Velásquez et al. (LóPEZ-VELáSQUEZ; ACOSTA-AMAYA; JIMéNEZ-BUILES, 2024) proposed a hybrid approach combining deep learning with fuzzy behavior trees to improve autonomous navigation. Their work highlights the benefits of integrating rule-based decision-making with learned object recognition models. Achirei et al. (ACHIREI et al., 2023) proposed a model-predictive control approach using object detection from CNNs to enhance trajectory planning in logistic environments. However, these methods did not explicitly consider obstacle height and clearance as critical factors in motion planning. Our work builds upon these studies by integrating YOLO for object classification, RGB-D sensors for depth perception, and fuzzy logic for adaptive control. This combination allows for more accurate obstacle classification and safer trajectory adjustments.

Table II.2 categorizes each approach based on its type, adaptability, computational complexity, real-time applicability, and suitability, specifically within HRC contexts. This comparative framework allows for a clearer understanding of the strengths and limitations of each method, highlighting how different techniques address the critical challenges in collaborative robotics. Note that advanced approaches like real-time danger estimation and sensor-based methods provide higher adaptability and suitability for HRC but come at the cost of increased computational demand.

Table II.2: Related Works.

Approach	Year	Type of Approach	Adaptability	Computational Complexity	Real-Time Applicability	HRC Suitability
(ALI; KIM; KIM, 2015)	2015	Type-2 Fuzzy for Collision Avoidance	Moderate	Moderate	Low	Low
(MOON; JAJULWAR et al., 2016)	2016	Fuzzy Logic Controller	Moderate	Low	Moderate	Low
(REIS et al., 2019)	2019	YOLO + RGB-D Perception	Moderate	Moderate	Moderate	Moderate
(DANG et al., 2023)	2023	CNN + RGB-D Sensing	High	High	High	High
(ACHIREI et al., 2023)	2023	CNN + Model Predictive Control	High	High	Moderate	Moderate
(LóPEZ-VELáSQUEZ; ACOSTA-AMAYA; JIMéNEZ-BUILES, 2024)	2024	Deep Learning + Fuzzy Behavior Trees	Very High	High	High	Very High
Proposed Approach	2024	YOLO + RGB-D + Fuzzy Logic	Very High	High	Very High	Very High

As can be seen, using CNNs, like YOLO, further improves detection and avoidance capabilities by

recognizing objects in complex settings, allowing for dynamic path adjustments. This technique, along with other technologies, such as RGB-D cameras, robust control algorithms, and costmap functions, such as the proposition of this work, can ensure reliable and adaptive navigation in constantly changing industrial environments. It is possible to verify that the proposed approach distinguishes itself through the use of SLAM combined with Fuzzy Logic and RGB-D cameras integrated with YOLOv3 CNN for dynamic obstacle detection and avoidance. Our proposed work incorporates navigation strategies and safety measures to enhance operational safety in human-robot interactions in complex and dynamic environments.

Chapter III Proposed Methodology

III.1 Overall Idea

The proposed architecture for an autonomous robotic system is designed for a collaborative environment. The system operates in a simulated 3D environment with typical warehouse objects such as cones, human workers, and robots. The chosen robot is a KUKA Youbot, equipped with a Kinect RGB-D camera on its end-effector to capture color and depth information, as shown in Figure III.1 (A). The processing pipeline is divided into several key stages. In stage (B), the robot determines its movement goals and plans a trajectory to reach those points while avoiding obstacles. Real-time spatial mapping and goal point identification assist in the trajectory planning stage, helping the robot define safe paths. Next, the robots sensory inputs are processed in parallel for two main tasks: object detection and trajectory adjustment. In the object detection module, Figure III.1 (C.1), the system uses distance data along with the YOLO algorithm to identify nearby obstacles. This is further explained in Section 3.3. The robots internal speed controller is fed by the fuzzy logic-based mechanism (C.2), which continuously generates velocity reference values based on detected distances, ensuring smooth and efficient movement. The details are explained in Section 3.4. Finally, the robots integrated navigation system (details can be found in Section 3.5), Figure III.1 (D), combines data from previous modules to adjust its speed and trajectory continuously, allowing it to move safely through dynamic environments by maintaining an optimal speed and avoiding potential collisions.

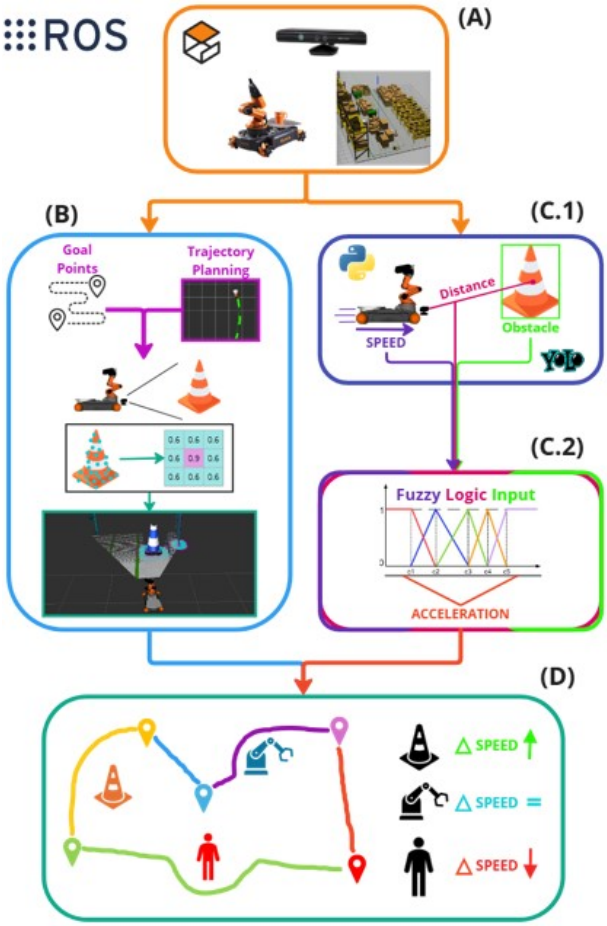


Figure III.1: Overview of the proposed method. The target application scenario is a warehouse with a mobile platform equipped with a camera (**A**). This setup enables the implementation of trajectory planning methods that utilize goal point coordinates. The system enables RGB-D odometry-based path planning by leveraging the Kinect Camera to create a costmap from the depth camera's point cloud (**B**). In parallel, the method employs YOLOv3 for object detection to identify specific obstacles in the path. The solution uses the depth camera to measure the distance to the centroid of detected objects while also gathering speed data from the robot's sensors (**C.1**). This information is input into the fuzzy logic module (**C.2**), where it is processed. Based on the object class, robot speed, and distance, the module adjusts the robot's reference velocity while maintaining its trajectory (**D**).

To better understand the Fuzzy-Speed approach used in this methodology, Figure III.2 shows the overall flow chart of the system.

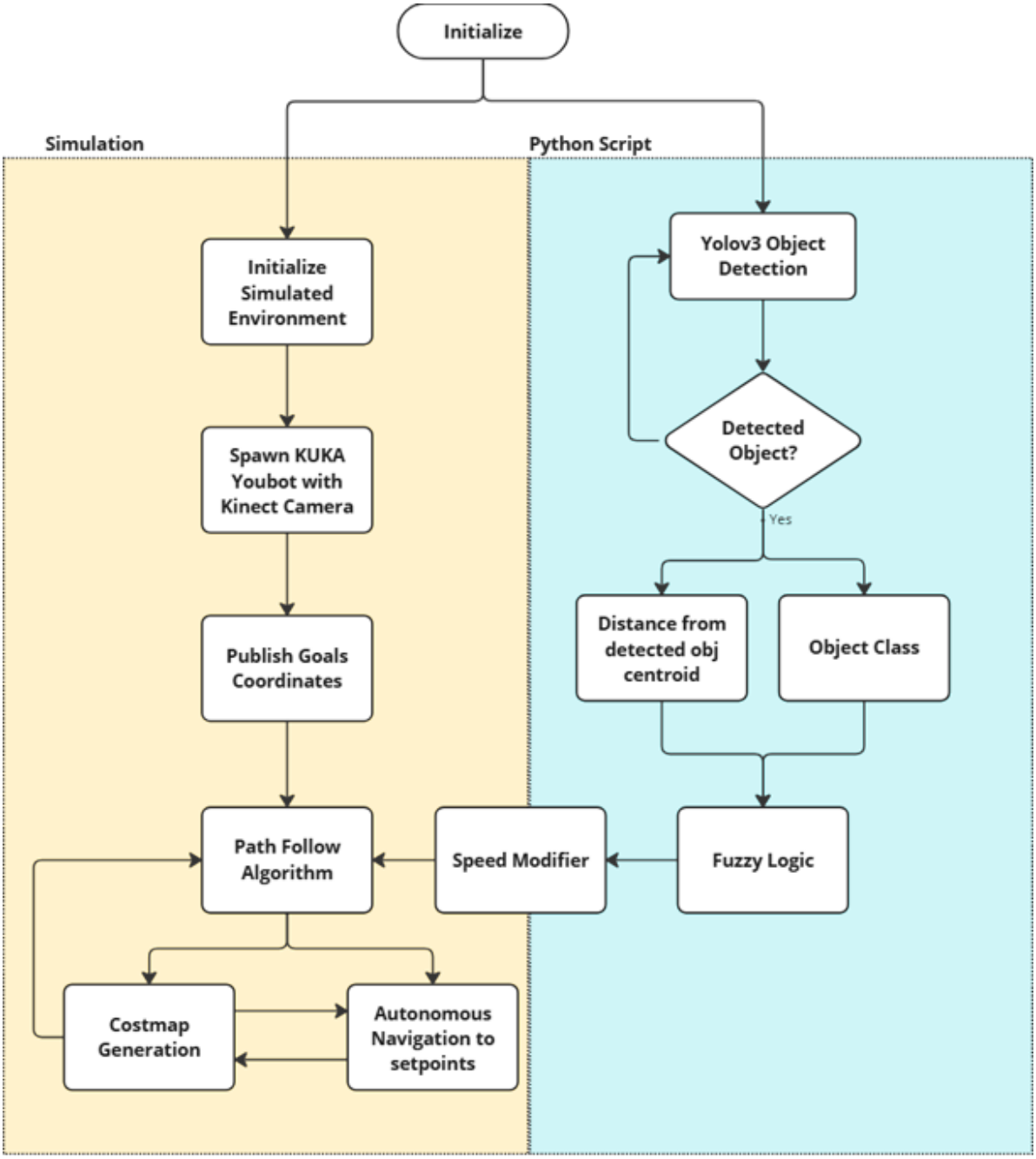


Figure III.2: Scheme of fuzzy addition to the speed controller.

The first step is initializing the 3D simulated environment, which includes the industrial setting, the KUKA Youbot equipped with a Kinect RGB-D camera, and common obstacles found in industrial environments, such as warehouse robots, toolboxes, cones, and humans. Once the controlled environment is set up, the methodology follows the multi-stage model of robotic interaction described by Bengtson et al. (BENGTSON et al., 2020). This model is implemented as follows:

- **Stage 1 - Information Acquisition:** The RGB-D camera captures image and depth data from the environment, which is then transferred to the system for further processing. This data includes information about detected objects and their distances from the robot.

- **Stage 2 - Information Analysis:** The YOLOv3 algorithm processes the RGB-D data to detect and classify objects in the robot's path. The detected objects are represented by bounding boxes, and their centroids are calculated for distance estimation. The depth camera provides the distance z to the object's centroid, which is used for further processing.
- **Stage 3 - Decision Selection:** The processed data, including object class, centroid distance, and robot speed, is fed into the fuzzy logic module. The fuzzy logic system uses a rule set to infer the appropriate speed adjustment for the robot, ensuring safe and efficient navigation.
- **Stage 4 - Action Implementation:** The speed value determined by the fuzzy logic module is converted into a control signal, which adjusts the robot's motion. This ensures that the robot maintains its trajectory while avoiding collisions with detected obstacles.

The sensory inputs acquired through the RGB-D camera provide critical environmental information to the robotic system. These inputs include data on object detection, localization, and distance estimation, which are crucial for the robot's navigation and obstacle avoidance tasks. The YOLOv3 algorithm generates bounding boxes to delineate detected objects within the robot's operational area. From these bounding boxes, the centroid coordinates (X_c, Y_c) are calculated as:

$$X_c = \frac{x_{\text{min}} + x_{\text{max}}}{2}, \quad Y_c = \frac{y_{\text{min}} + y_{\text{max}}}{2}$$

where $(x_{\text{min}}, y_{\text{min}})$ and $(x_{\text{max}}, y_{\text{max}})$ are the minimum and maximum pixel coordinates of the bounding box, respectively.

Additionally, the depth camera provides the distance z to the object's centroid. To map the 3D world coordinates (x, y, z) to 2D image coordinates (u, v), the following equations are used:

$$u = \left(\frac{x \cdot fx}{z}\right) + cx, \quad v = \left(\frac{y \cdot fy}{z}\right) + cy \tag{III.1}$$

Here, fx and fy represent the focal lengths in the x and y axes, respectively, while cx and cy are the optical center coordinates of the camera. These parameters are intrinsic to the camera and define how 3D points are projected onto the 2D image plane.

This information, along with the speed parameters, is fed into the Fuzzy-Speed controller, which integrates fuzzy logic theory with traditional speed control. The fuzzy logic component allows for adaptive adjustments to the control signals based on real-time environmental conditions and system states, enhancing the robot's adaptability and robustness.

III.2 Safety Threshold Methodology

Industrial environments present dynamic hazards that require robust safety measures. Our approach implements danger threshold circles around each significant component in the workspace (Figure III.3). This established methodology, demonstrated effective in (KIM; SUMI, 2020), provides a computationally efficient solution for collision prediction and avoidance in dynamic environments.

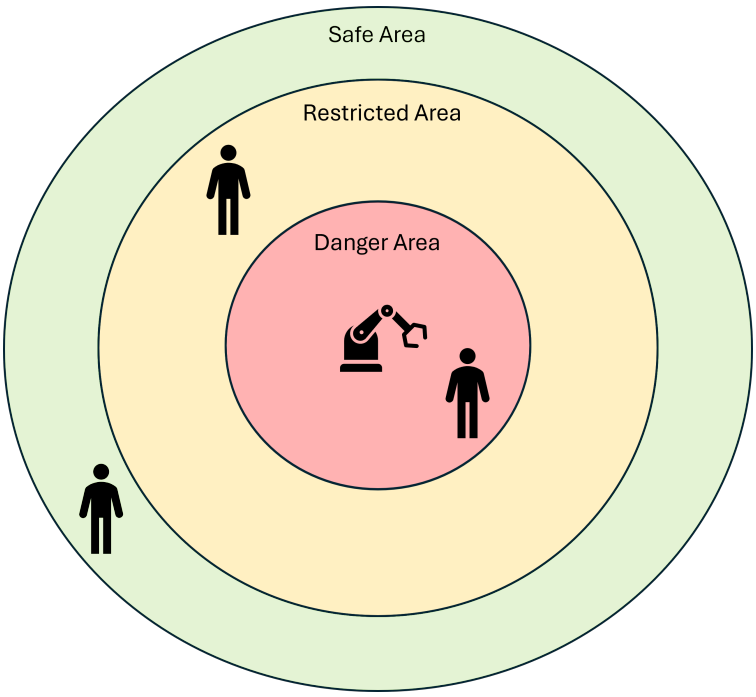


Figure III.3: Safety threshold circle methodology showing protected zones around (A) human workers, (B) static obstacles, and (C) other robots. The concentric circles represent varying danger thresholds that trigger different response behaviors.

The system establishes virtual protection zones with predefined radius around both the robot and environmental objects. These zones operate through three key mechanisms:

- **Proximity Detection**: The robot's depth sensors continuously monitor distances to all objects within the operational space. When any object enters the outermost warning threshold (typically 1.5m radius), the system initiates preliminary speed adjustments.
- **Danger Assessment**: Nested threshold circles enable graduated response strategies. The middle warning zone triggers moderate deceleration, while the inner critical zone initiates emergency braking procedures.
- **Adaptive Response**: The fuzzy-speed controller dynamically adjusts velocity parameters based on:
  - Object classification (human, robot or other common warehouse objects)
  - Relative velocity vectors

  - Zone penetration depth

This multi-layered approach provides several advantages for collaborative warehouse environments:

- **Safety**: Maintains ISO/TS 15066 compliant separation distances
- **Efficiency**: Minimizes unnecessary braking through graduated response zones
- **Adaptability**: Handles both static and dynamic obstacles through continuous zone monitoring.

The implementation integrates seamlessly with the ROS navigation stack, publishing safety zone violations as custom messages that trigger appropriate control responses. Later it will be explained how these threshold events interface with the fuzzy-speed control system to achieve smooth, safe operation in crowded environments.

III.3 Simulation with ROS and Gazebo

The initial stage of developing the simulation within the ROS framework involved establishing a suitable application that could effectively integrate with the robot manipulator with an established routine. The application was designed to use images captured from the robot's end effector and overhead cameras to achieve predefined objectives. The chosen application scenario was a manufacturing environment. The robot would autonomously follow programmed path goals during in which passes by specified detectable objects, such as a tool, a person or anything that can be interpreted as an obstacle. Figure III.4 shows the mobile manipulator KUKA Youbot used in the simulations. The environment used in Gazebo is shown in Figure III.5. The simulation runs in Ubuntu 16.04, OpenCV3.4.5, and TensorFlow 1.4. The computer is an Intel i7-11800H processor, DDR4 16GB memory, and NVIDIA GeForce RTX3050TI 12GB GPU.

The initial phase of our project involves developing a Python 2.7 script with rospy to establish communication with ROS (Robot Operating System) messages, connecting all the 3D simulated environment along with tools to help building and creating the proposed solutions. Rospy is a Python library specifically designed for ROS, enabling seamless interaction with ROS nodes, topics, and messages. Within this framework, we integrate OpenCV and "CvBridge" modules to effectively process imagery captured by the camera within the Gazebo simulation environment. Utilizing OpenCV facilitates sophisticated image manipulation tasks, while "CvBridge" acts as a bridge between ROS image messages and OpenCV workable images, enabling seamless conversion between the two formats. This integration enables us to harness the visual data captured by the simulation's camera, which serves as a foundational component for subsequent analysis. Moreover, we augment this data by incorporating YOLOv3 to perform real-time object detection and recognition tasks, thereby enabling our project's capabilities to include an immense range of potential applications. Through this cohesive integration of technologies,

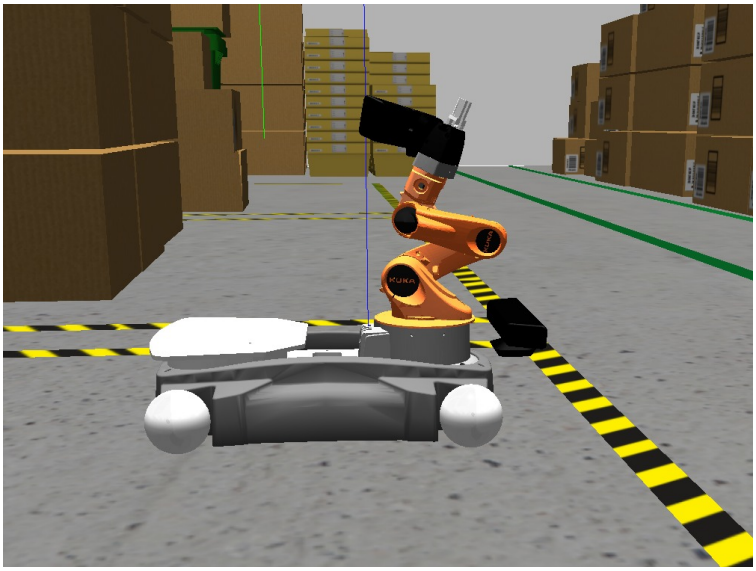


Figure III.4: Robot Youbot used in the proposed scenario with an eye-in-hand camera



Figure III.5: Proposed test scenario

we lay the groundwork for our project's progression toward more advanced functionalities and objectives that will be discussed in the upcoming paragraphs.

In this setup, the robot transmits the images captured by its cameras to a central control station. Figure III.6 illustrates the system's nodes and topics involved in this process. The "camera" node represents attached camera, which detects objects within the simulation environment. It transmits both RGB and RGB-D messages as rostopic messages. These messages are then processed by the system to generate a 2D representation of the environment using the Costmap approach, a technique widely used in robotics to identify static and dynamic obstacles. The Costmap applies an inflation layer that

defines a safety zone around detected obstacles, allowing the robot to adjust its trajectory and ensure safe navigation.

The camera serves as the primary sensor, providing both depth and color images. The depth information from the topic `/camera/depth/points` is the input of the costmap node `/costmap`, which builds a 2D map of the environment. This map identifies obstacles and applies an inflation layer to define a safe buffer around the robot. The color image topic, `/camera/color/image_raw`, is processed by the YOLOv3 node to enhance object detection by identifying potential dynamic obstacle types in the proposed trajectory. The fuzzy logic module topic `/fuzzy_input` enables adaptive decision-making during navigation, improving the safety of operators and the warehouse structure.

To ensure safe and efficient navigation, the `move_base` node plays a critical role in the proposed stack. This node acts as both a global and local planner. The global planner calculates the main trajectory, from the starting position to the goal, and uses the pre-existing environment map to generate a clear route that avoids static obstacles. The local planner adjusts the global path in real time based on sensor and algorithm data, allowing the robot to avoid dynamic obstacles and ensuring smooth movement during trajectory execution. The interface between these planners is based on the costmap node with the topic `/move_base/global_costmap/costmap_updates`. The `move_base` node also generates velocity commands (`/cmd_vel`) based on real-time feedback from the odometry and the RGB-D camera data.

Finally, the velocity references are processed by the fuzzy module, resulting in the `/cmd_vel_safe` topic. This control layer significantly enhances safety and responsiveness, ensuring that the robot operates at safe speeds and behaviors in complex environments.

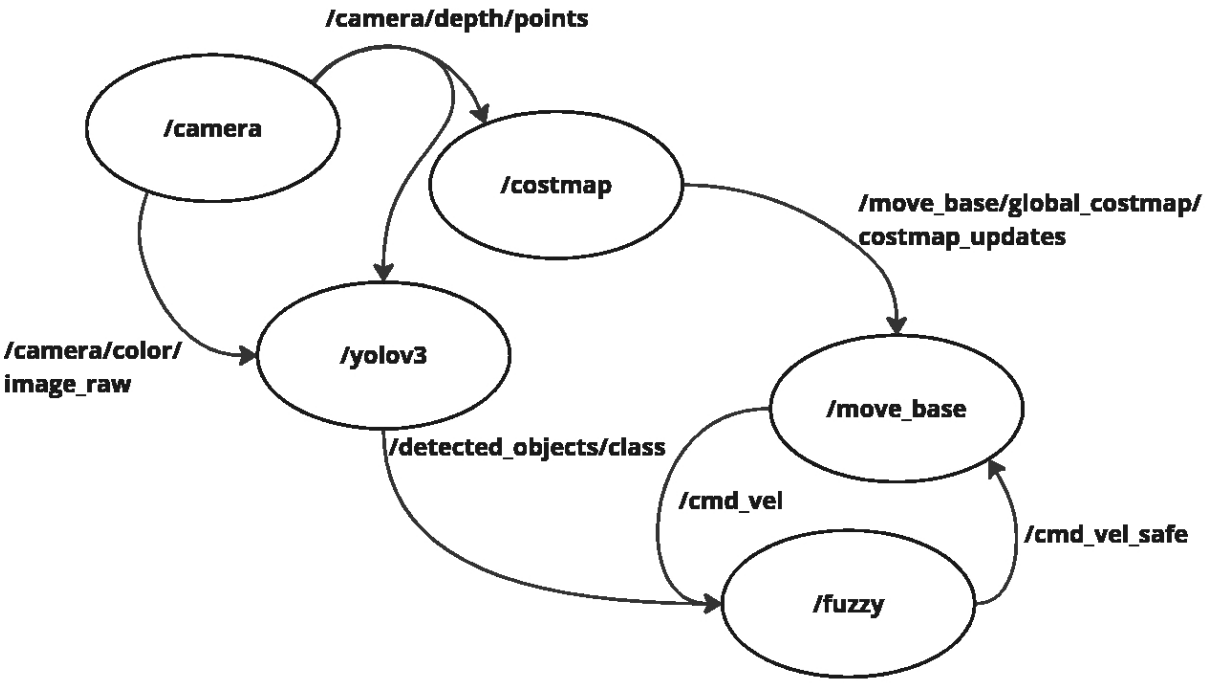


Figure III.6: Active ROS nodes and topics in the simulation.

These components operate within the Gazebo simulation software environment, facilitating a comprehensive simulation experience that closely mimics real-world scenarios. This integrated approach allows for the testing and validation of robotic control algorithms in a controlled virtual environment before deployment in physical systems.

III.3.1 Autonomous Navigation

The autonomous navigation system developed in this project is built around the **ROS 1 (Robot Operating System)** framework, leveraging its robust navigation stack to enable real-time environment mapping, path planning, and obstacle avoidance. The ROS navigation stack is a collection of software packages designed to provide a modular and scalable framework for mobile robot navigation. It integrates sensor data, localization, mapping, and motion planning into a cohesive system, allowing robots to navigate autonomously in dynamic environments.

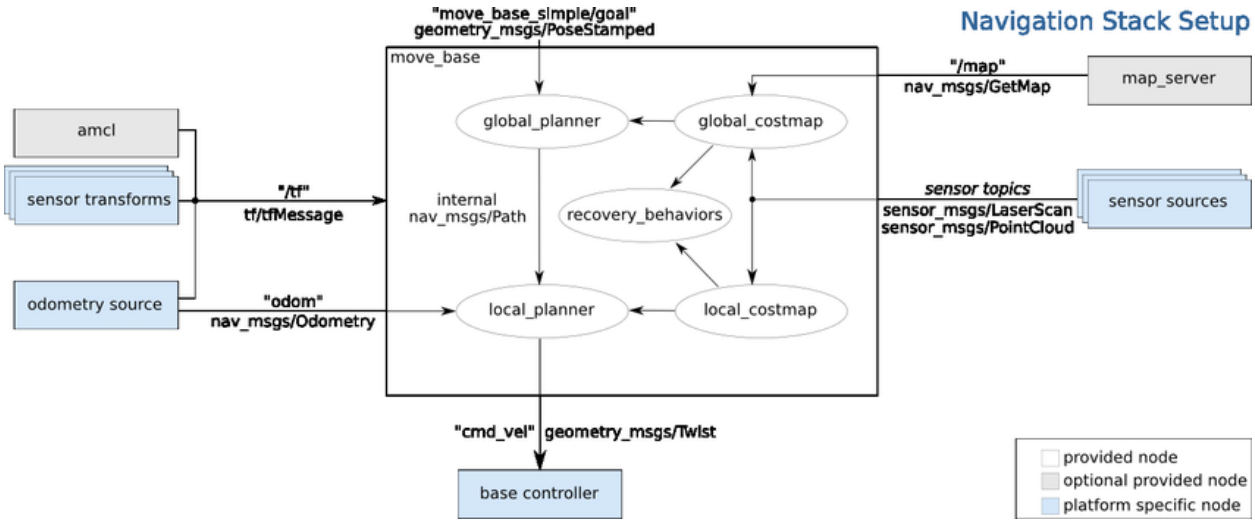


Figure III.7: Standard use of the ROS navigation stack, as described in Granosik et al. (GRANOSIK et al., 2016). The architecture highlights the integration of sensors, planners, and controllers for autonomous navigation.

At the core of the ROS navigation stack is the **Costmap**, a 2D grid-based representation of the robot's surroundings. The Costmap is divided into two primary layers: the **static layer** and the **dynamic layer**. The static layer represents fixed obstacles in the environment, such as walls, shelves, and other permanent structures. This layer is typically generated from a pre-existing map or through SLAM (Simultaneous Localization and Mapping) techniques during the initial exploration phase. The dynamic layer, on the other hand, captures moving obstacles, such as humans or other robots, using real-time sensor data from the Kinect RGB-D camera's `/camera/depth/points` PointCloud2 topic. This 3D point cloud data enables the system to:

- Determine obstacle height for proper clearance assessment

- Identify navigable free space beneath suspended objects
- Track dynamic changes in the environment over time
- Assess vertical obstacle extent for comprehensive collision avoidance



Figure III.8: A suspended toolbox (red circle) placed on a table in front of the robot's path during the navigation phase. Note that while the table itself and the object above it are detected, they are not marked as obstacles in the costmap. However, the table's legs are correctly identified and registered as obstacles (green circles).

By combining these layers with 3D perception capabilities, the Costmap provides a comprehensive view of the environment, enabling the robot to navigate safely and efficiently.

One of the key features of the Costmap is the **inflation layer**, which defines a safety zone around detected obstacles. This layer assigns higher costs to cells closer to obstacles, effectively creating a gradient that guides the robot's path planning. The inflation radius determines the size of the safety buffer, ensuring that the robot maintains a safe distance from obstacles while navigating. This approach not only prevents collisions but also allows the robot to plan smoother trajectories by avoiding sharp turns or sudden stops.

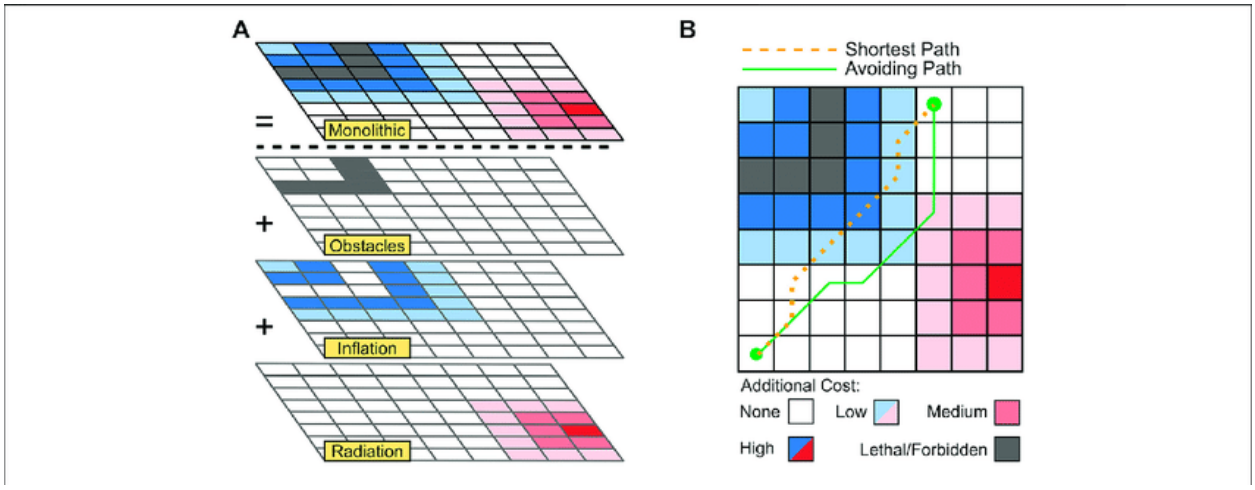


Figure III.9: Example of a costmap generated by the system, showing static and dynamic obstacles with inflation layers. The inflation radius defines a safety buffer around detected obstacles, ensuring safe navigation. Adapted from West et al. (WEST et al., 2022).

The ROS navigation stack relies on several key components to achieve autonomous navigation:

1. **Sensor Integration:** The system integrates data from multiple sensors, including the Kinect RGB-D camera, which provides both depth and color images. Depth information from the topic `/camera/depth/points` is used to build the Costmap, while color images from the topic `/camera/color/image_raw` are processed by the YOLOv3 algorithm to detect and classify dynamic obstacles. This sensor fusion ensures that the robot has a comprehensive understanding of its environment.

2. **Localization:** Localization is achieved using the **AMCL (Adaptive Monte Carlo Localization)** package, which estimates the robot's position within a pre-existing map. AMCL uses a particle filter to track the robot's pose based on sensor data and odometry. This information is critical for accurate path planning and navigation.

3. **Global and Local Planning:** The navigation stack employs a two-tiered planning approach, consisting of a **global planner** and a **local planner**. The global planner calculates the optimal path from the robot's current position to the goal, using the static layer of the Costmap to avoid fixed obstacles. The local planner, on the other hand, adjusts the global path in real time based on the dynamic layer, allowing the robot to avoid moving obstacles and adapt to changes in the environment. The local planner also considers the robot's kinematic constraints, ensuring smooth and feasible motion.

4. **Move_Base Node:** The `move_base` node is the central component of the ROS navigation stack. It coordinates the global and local planners, Costmap updates, and sensor inputs to generate velocity commands for the robot. The `move_base` node publishes these commands to the `/cmd_vel` topic, which controls the robot's motion. The node also interfaces with the Costmap via the `/move_base/global_costmap/cost` topic, ensuring that the navigation system operates with up-to-date environmental data.

5. **Fuzzy Logic Module:** To further enhance safety and responsiveness, the system incorporates a **fuzzy logic module**, which processes velocity commands from the `move_base` node. This module

dynamically adjusts the robot's speed based on proximity to obstacles, ensuring smooth and safe navigation in complex environments. The fuzzy logic controller uses inputs such as distance to obstacles, robot speed, and object classification to generate adaptive speed commands, which are published to the `/cmd_vel_safe` topic.

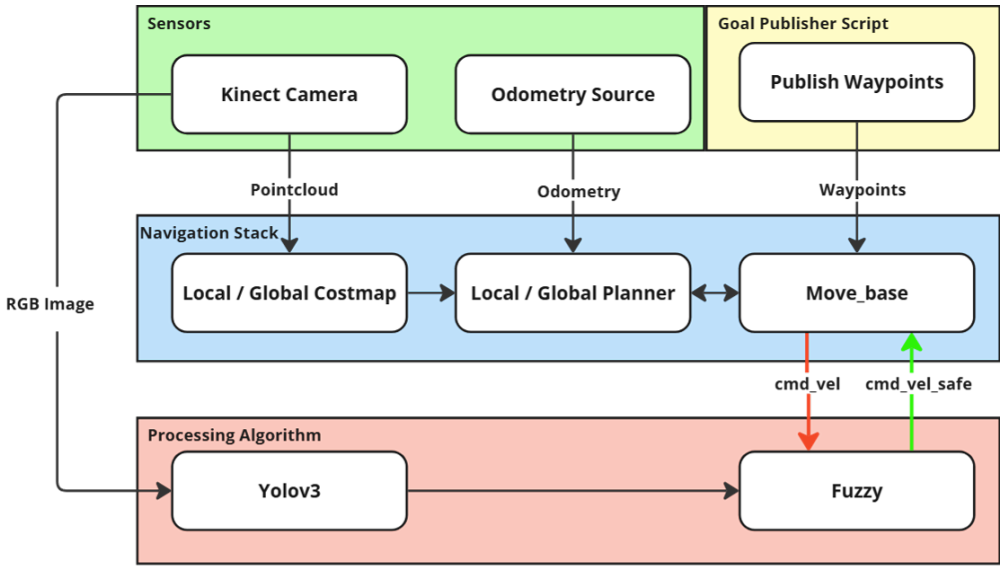


Figure III.10: Architecture based on the ROS navigation stack and adapted to use object identification in order to provide safer autonomous movement.

The ROS navigation stack is designed to be modular and extensible, allowing for the integration of additional sensors and algorithms. For example, LiDAR sensors can be added to improve SLAM-based navigation and obstacle detection in more complex scenarios. This scalability ensures that the system can be adapted to a wide range of applications and environments.

The two-tiered planning approach, explained in the key components, can be described as the global planner and the local planner: The global planner computes the optimal path from the robots current position to the target goal, avoiding static obstacles. It uses the A* (A-star) algorithm, which enhances the classical Dijkstra approach by combining the accumulated cost with a heuristic estimate of the remaining distance to the goal, significantly improving efficiency in complex maps.

The local planner continuously adjusts the path in real time based on dynamic changes in the environment and the robots motion constraints. For this, the Dynamic Window Approach (DWA) algorithm was used. DWA evaluates multiple trajectory options by simulating the robots movement over a short time window, considering dynamic constraints such as velocity and acceleration limits. Each trajectory is scored according to its safety (distance to obstacles), progress toward the goal, and adherence to the global path. The configuration used in this work maintained default parameter values defined in the navigation stack, which are presented in Table III.1.

To optimize the system's performance, several parameters are fine-tuned, including the robot's footprint, inflation radius, and minimum obstacle height. These parameters ensure that the Costmap

Table III.1: DWA Local Planner Parameters.

Parameter	Value
Maximum Linear Velocity (`max_vel_x`)	0.5 m/s
Maximum Angular Velocity (`max_vel_theta`)	1.57 rad/s
Linear Velocity Samples (`vx_samples`)	6
Angular Velocity Samples (`vtheta_samples`)	20
Path Distance Bias (`path_distance_bias`)	0.75
Goal Distance Bias (`goal_distance_bias`)	1.0
Inflation Radius	0.30 m

accurately represents the environment and provides reliable navigation guidance. Table III.2 presents the set of parameters used in the costmap configuration.

Table III.2: Costmap Parameters.

Parameter	Value
Robot Footprint	0.21, 0.165
Footprint Padding	0.05
Inflation Radius	0.30
Minimum Obstacle Height	0.05

III.3.2 Object Detection

The autonomous navigation system proposed in this work relies on onboard cameras to provide real-time images to calculate the object's position and provide the information itself for obstacle detection and avoidance in warehouse environment. Computer vision algorithms can be used to provide useful information to assist in the feedback for automatic control. The selection of the You Only Look Once Version 3 (YOLOv3) convolutional neural network for this study was driven by its computational efficiency for real-time operation on mobile platforms, robust detection performance, versatility, ease of training, established results evaluation metrics, highly used in the reasearch community, avaiability and adptability in new or old ROS versions and recognized speed in detection. Additionally, its utilization of only convolutional layers ensures the Convolutional Neural Network (CNN)'s invariance to variations in the input image's resolution and size (HIEMANN et al., 2021). Further details can be found in (CHOI et al., 2019).

The object detection system in this work builds upon the YOLOv3 architecture, which employs Darknet-53 as its core feature extractor. This backbone network consists of 53 carefully designed layers that incorporate residual connections (ResNets) alongside convolutional layers with batch normalization and leaky ReLU activation functions. Through this architecture, the system achieves efficient hierarchical processing of visual information, progressively extracting both basic visual features like edges and textures in the early layers, followed by more complex semantic features such as object parts and contextual relationships in deeper layers.

A key strength of YOLOv3 lies in its multi-scale detection capability. The architecture makes

simultaneous predictions at three different grid resolutions: 52×52, 26×26, and 13×13 pixels. This design allows the system to effectively detect objects across various size ranges - the finer 52×52 grid excels at identifying smaller objects, while the coarser 13×13 grid handles larger items. When combined with Darknet-53's robust feature extraction, this multi-scale approach proves particularly valuable in complex industrial environments like warehouses, where objects of different sizes may appear at varying distances from the robot's cameras (REDMON; FARHADI, 2018).

Figure III.11 presents a simplified overview of the YOLOv3 structure implemented in this study, highlighting both the backbone network and the multi-scale detection components. The architecture's balance between computational efficiency and detection accuracy makes it well-suited for real-time operation on mobile robotic platforms, a crucial requirement for dynamic warehouse environments where processing delays could compromise safety and navigation performance.

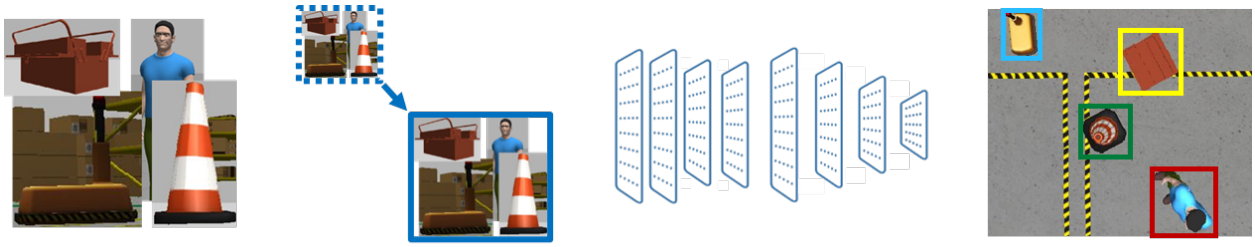


Figure III.11: YOLOv3 simplified structure. (**a**) Input image. (**b**) Image resizing. (**c**) Convolutional layers. (**d**) Object detection.

To address the specific perception requirements of mobile robots operating in shared warehouse spaces, this work developed a specialized object detection dataset comprising 250 annotated images. The dataset focuses on four critical object categories that represent both operational infrastructure and dynamic elements in collaborative industrial environments:

- **Human workers (Person)**: The highest-priority detection class, where collision avoidance takes precedence over all other navigation objectives. The system implements a velocity overlap focusing on increasingly higher braking protocols when humans approaches the robot's operational radius to ensure compliance with safety standards.
- **Toolboxes**: Representing static objects that pose no immediate danger, toolboxes allow the robot to maintain higher speeds and require less braking mid-course.
- **Cones**: Delineation markers that signal restricted zones or pathway boundaries. The control system implements progressive braking when approaching cones to maintain safe operational margins.

- **Other robots**: Mobile platforms requiring dynamic response strategies. They require careful navigation to avoid collisions, prioritizing balancing speed and caution during the route.

The training process was carefully designed to develop a robust perception system capable of optimizing both operational efficiency and safety in warehouse environments. Using exclusively synthetic imagery generated through Gazebo simulations, we constructed a comprehensive dataset that captures the key visual elements of collaborative industrial spaces. Each simulation was configured to reproduce realistic warehouse lighting conditions, object arrangements, and camera perspectives.

The dataset consists of 250 carefully curated synthetic images, each annotated using LabelImg (TZUTALIN, 2015) to identify four critical classes: **person**, **toolboxes**, **cones**, and **warehouse robots**. These categories were selected to represent both the dynamic and static elements commonly encountered in warehouse operations. The annotation process followed strict consistency guidelines to ensure uniform labeling across all training samples.

To address the challenges of training with a limited synthetic dataset, we implemented several key strategies (i.e., flipping, scaling, rotations, and brightness) to improve generalization and simulate real-world variability.

These techniques significantly enhanced the model's ability to generalize from synthetic data to real-world operational conditions, particularly for challenging cases involving partial obstructions or variable lighting - common occurrences in active warehouse environments. The complete set of training parameters, including learning rates, batch sizes, and optimization settings, is provided in Table **??**.

Table III.3: Hyperparameters used in YOLO's training.

Optimizer	SGD (Learning rate = 0.01)
Epochs	10000
Batch size	64
Batch subdivisions	16
Classes	4
Filters	27
Image Size	416x416p
Weight Decay	0.0005

The evaluation of the YOLOff network was conducted using the average precision methodology (mAP) and the $Precision \times Recall$ ($P \times R$) curve (Padilla; Netto; da Silva, 2020). While the mAP metric offers insights into the correlation between detected objects and their respective ground truth, it may lead to potential misinterpretations if the network detects annotated and non-existing objects within the image. To address this concern, the $P \times R$ curve was employed, where P denotes the ratio of true positives to the total detected objects, and R signifies the ratio of true positives to the total ground truths in an image (RAMOS et al., 2022). Ideally, as R improves, P should decrease, but a proficient object detector should maintain high values for both P and R throughout the process.

To modify the required parameters of the simulated robot, a number of Python scripts will be utilized to construct the environment, manage the intended consequences, and regulate the computer vision settings. The configuration file and the weights are two distinct parameters that determine how the neural network architecture for YOLOv3 is implemented in the system.

- Configuration File (".cfg"): A text document that contains the architecture of the neural network, detailing and arranging each layer and parameter that the convolutional network must pass through to forecast and identify each occurrence of a detectable object within the image;
- Weights (".weight"): A text file that holds the numerical value for each labeled data set that the algorithm acquires throughout its learning phase. It is predicated on prior image collection training that associates and dissociates the data from the ground truth.

The files are imported into the script as a "Model", while the classes that designate each detection are loaded from a ".name" file. Following the establishment of layers, parameters, and image conversions, the subsequent stage of the script involves iterative processing the classes within each detection. This entails assigning the appropriate class identifier and confidence score to each detection, reflecting its proximity to the ground truth.

The hyper-parameters used to train the model differ from the YOLOV3 default structure due to only three classes being used. This means that the Configuration File ".cfg" used for training must be adapted to cover this discrepancy.

Regarding depth data acquisition, the Kinect camera captures RGB-D data, where each pixel contains both color information and depth data up to 3 meters. This depth data is obtained through infrared light and an infrared receiver emission. The infrared light emits a structured pattern with noise, allowing the object's deformation to be identifiable by the receiver. After processing, the distance data is obtained.

The Xbox 360 Kinect requires specific software components to function in real-world robotics applications. Unlike newer Kinect models that use Microsoft's official SDK, the legacy Xbox 360 version relies on open-source solutions for depth data processing and ROS integration.

For depth sensing, the system primarily uses `OpenNI` (Open Natural Interaction), an open-source framework that provides standardized APIs for processing depth data from 3D sensors. This middleware handles the raw infrared pattern interpretation and converts it into usable depth information.

In ROS environments, the following tools are essential:

- The `ros-kinetic-freenect-launch` package (part of the ROS Kinetic toolchain) provides the main interface between the Kinect hardware and ROS
- The `openni_launch` ROS package creates standardized topic interfaces
- Proprietary firmware embedded in the Kinect hardware handles low-level sensor operations

For Gazebo simulations, these physical components are replaced by:

- The `libgazebo_ros_openni_kinect.so` plugin, which emulates the Kinect's behavior
- Depth calculation included on simulation based on Gazebo's environment mesh data
- Simulated ROS topics that maintain identical naming and structure to the real hardware outputs (`/camera/depth/points`, `/camera/color/image_raw`)

The original Microsoft Kinect (Xbox 360 version) utilizes a structured light approach for depth perception, distinct from both stereo vision and time-of-flight methods. The system operates by projecting an infrared speckle pattern consisting of approximately 500,000 unique dots onto the scene using an IR projector. A dedicated monochrome CMOS IR camera then captures how this pattern deforms when it hits objects in the environment.

The depth calculation relies on analyzing the displacement of these speckle patterns. As objects move closer to the sensor, their reflected pattern shows greater displacement compared to the reference pattern, while farther objects show less displacement. This deformation is processed using sophisticated correlation algorithms that match local speckle constellations between the projected and observed patterns. Each unique speckle arrangement acts as a natural marker, allowing the system to perform accurate triangulation without requiring distinct visual features on objects themselves.

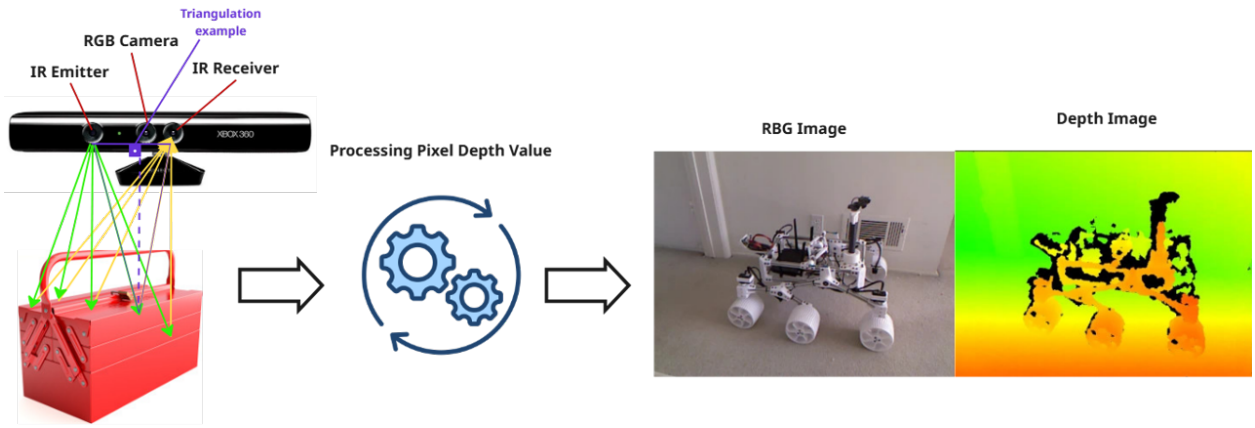


Figure III.12: Kinect V1 depth processing method simplified structure.

The technology provides several advantages over traditional stereo vision, particularly its ability to work on textureless surfaces and uniformly colored objects. However, it has specific operational constraints: it functions best indoors as sunlight contains infrared radiation that can overwhelm the projected pattern, and multiple Kinect devices may interfere with each other due to pattern cross-talk. The system delivers VGA resolution (640Œ480) depth maps at 30Hz with an effective range of approximately 3-4 meters, where closer objects appear brighter in the depth image output.

It's important to note that while later Kinect versions adopted different technologies, this structured light approach remains unique to the Xbox 360 version, making it particularly suitable for certain robotic

applications that benefit from its precise short-range measurements and resistance to visual ambiguity in featureless environments.

Within the project, the depth image topic is converted using CV.Bridge. Then, a bounding box is applied to the object via YOLOv3 detection using the color image (since the coordinates of the depth and color images are the same). This establishes a Region of Interest (ROI) within these bounding boxes to improve processing. Additionally, an average is taken of the distance values obtained within this region of interest. The integration of YOLOv3 object detection with depth data offers a comprehensive approach to scene understanding

Thus, the depth data is printed along with the object's label detected via YOLOv3. Afterward, these distance data are continuously updated in a text file. In another script, they are read and sent as a message on a new topic with other parameters such as robot speed. The distance from the robot to the centroid of the objects is measured by drawing a line between the centroid of the robot.

The Kinect v1 (Xbox 360) RGB-D perception system provides environmental data through these modalities:

- **RGB**: 640×480 resolution @ 30Hz (8-bit VGA)
- **Depth**: 320×240 resolution @ 30Hz (11-bit, 0.8-4m range)
- **Noise**: $\sigma_z = 0.0013z^2 + 1.08$ mm (empirical measurement)

The method consists of first using the Kinect RGB camera to gather raw image data to feed the YOLOv3-tiny detector, processing this input to generate bounding boxes $\mathcal{B}_i = (x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}, c)$ .Centroid coordinates are computed with subpixel accuracy:
where:

- $(x_{\text{min}}, y_{\text{min}})$ are the top-left pixel coordinates
- $(x_{\text{max}}, y_{\text{max}})$ are the bottom-right pixel coordinates
- c is the detection confidence score between 0 and 1

III.3.3 Camera Intrinsics Fundamentals

Camera intrinsic parameters define how points in 3D space project onto the 2D image plane. For RGB-D cameras like the Kinect v1, these include:

- **Focal Length** (f_x, f_y): The distance between the camera's optical center and image plane (in pixels), determining the field of view
- **Principal Point** (c_x, c_y): The optical center's pixel coordinates where the optical axis intersects the image plane

- **Distortion Coefficients**: Parameters modeling lens distortion effects

Table III.4: Kinect v1 Intrinsic Parameters (BURRUS, 2011)

Parameter	RGB Camera	Depth Sensor
Resolution	640×480	320×240
Focal Length (f_x)	525.0 px	570.3 px
Focal Length (f_y)	525.0 px	570.3 px
Principal Point (c_x)	319.5 px	159.5 px
Principal Point (c_y)	239.5 px	119.5 px
Depth Range	N/A	0.8-4.0 m

Example Calibration Matrix

For the depth sensor, the intrinsic matrix takes the form:

$$K = \begin{bmatrix} 570.3 & 0 & 159.5 \\ 0 & 570.3 & 119.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{III.2}$$

Centroid Calculation

The center point of each detected object is calculated as:

$$X_c = \frac{x_{\text{min}} + x_{\text{max}}}{2}, \quad Y_c = \frac{y_{\text{min}} + y_{\text{max}}}{2} \tag{III.3}$$

This gives the pixel coordinates of the bounding box center. For example, a box from (100,150) to (200,250) would have center at (150,200).

Depth Measurement with ROI Search

The system implements a robust depth estimation method that:

1. Creates a Region of Interest (ROI) around the centroid (X_c, Y_c)
2. Converts 3D point cloud to image coordinates:

$$u = \left\lfloor \frac{x \cdot f_x}{z} + c_x \right\rfloor, \quad v = \left\lfloor \frac{y \cdot f_y}{z} + c_y \right\rfloor \tag{III.4}$$

3. Finds the nearest valid depth point by minimizing:

$$d = \sqrt{(X_c - u)^2 + (Y_c - v)^2} \tag{III.5}$$

4. Returns the depth z of the closest valid point

This matches the implementation in `DepthImageHandler.point_cloud_callback()`:

```
points = pc2.read_points(msg, field_names=('x', 'y', 'z'), skip_nans=True)
for point in points:
    u = int((x * fx / z) + cx)
    v = int((y * fy / z) + cy)
    dist = sqrt((centroid_x - u)**2 + (centroid_y - v)**2)
    if dist < min_distance:
        min_distance = dist
        nearest_point = point
```

Temporal Filtering

A first-order low-pass filter reduces depth sensor noise:

$$z_t = 0.7 \cdot z_{t-1} + 0.3 \cdot \mathcal{D}(X_c, Y_c) \tag{III.6}$$

where:

- z_t is the current filtered depth
- z_{t-1} is the previous frame's filtered depth
- The 0.7/0.3 ratio provides optimal noise reduction while maintaining responsiveness

3D Position Calculation

Using the Kinect v1 depth sensor intrinsics (BURRUS, 2011):

$$x = \frac{(X_c - 159.5) \cdot z}{570.3}, \quad y = \frac{(Y_c - 119.5) \cdot z}{570.3} \tag{III.7}$$

where:

- 159.5, 119.5 are the optical center coordinates (in pixels)
- 570.3 is the focal length (in pixels) for Kinect v1

Fuzzy Control Input Normalization

The fuzzy controller receives three normalized inputs:

$$\tilde{d} = \frac{d}{5.0}, \quad \tilde{v} = \frac{v}{2.0}, \quad \tilde{c} = \frac{c - 0.6}{0.4} \tag{III.8}$$

where:

- $\tilde{d}$ is the normalized distance (5m max range)
- $\tilde{v}$ is the normalized velocity (2m/s max speed)
- $\tilde{c}$ is the normalized confidence (threshold at 0.6)

Subsequently, all distances between the robot and the detected objects are compiled and stored for future use within subsequent iterations of the overarching procedure.

III.3.4 Hybrid-Fuzzy Speed Approach

The centroid distance data, processed from the camera in the preceding step, undergoes an additional Python-based fuzzy logic controller implemented with the scikit-fuzzy library. Subsequently, it is transposed into a readable format and converted into a ROS topic using preconfigured message types, facilitating integration within the ROS encapsulated framework. Once distances and speeds are published, this topic is subscribed by the script, where it assumes the role of a parameter influencing the topic's value. Subsequently, this value acts as input for the Fuzzifier step in the Fuzzy Logic Controller box. This process is shown in Figure III.13. This scheme enables real-time adjustments to the speed parameter, facilitating the robot's swift integrated motion during its approach to the detected. This agility is achieved without compromising the robot's movements within predefined threshold boundaries.

The fuzzy implementation of this work can be divided into three main stages, which are fuzzification, rule evaluation, and defuzzification. **Fuzzification** involves transforming precise (CRISP) input values into degrees of membership within a fuzzy set. In the current work, this stage applies to the velocity, the distance, and the class data, collected in the trained network model. These CRISP inputs are mapped to determine the truth degree, indicating how closely the variables fit within specific groups, while the distance is measured based on the obstacle's proximity to the camera. Finally, the velocity is based on the Kuka Youbots default speed, for which zero indicates stopped, and one is the maximum speed. The class value represents the object type.

For the velocity control subsystem, the defuzzification process employs the centroid method (also known as the center of area method), which calculates the weighted average of all rule outputs according to the equation:

$$v_{out} = \frac{\sum_{i=1}^{N} \mu_i \cdot v_i}{\sum_{i=1}^{N} \mu_i} \tag{III.9}$$

where μ_i represents the firing strength of the i^{th} rule and v_i is the corresponding output velocity

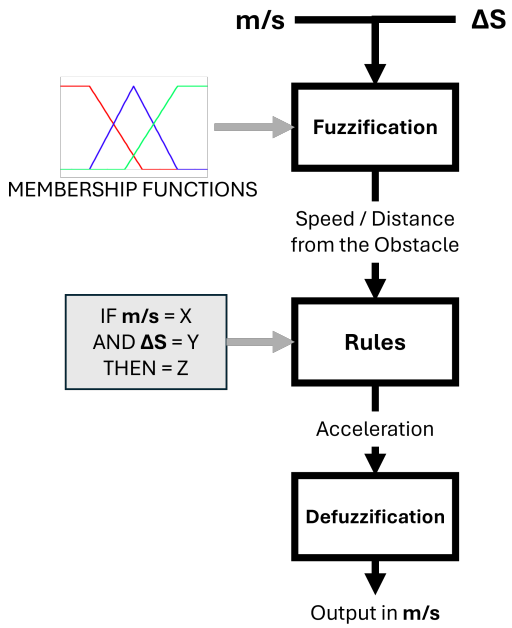


Figure III.13: Fuzzy flowchart implementation in this study. The diagram illustrates the process where precise inputs, such as speed (m/s) and distance (ΔS), are fuzzified into degrees of membership. Fuzzy rules are then applied to determine velocity, and defuzzification converts the fuzzy output into a precise speed value (m/s).

value. This approach provides smooth velocity transitions while maintaining computational efficiency suitable for real-time operation;

Table III.5 presents the complete rule base governing the robot's angular velocity control. The rules incorporate five distinct linguistic variables for turn rate intensity:

- **VL (Very Low)**: Minimal turning adjustment
- **L (Low)**: Gentle turning
- **M (Medium)**: Moderate turning
- **H (High)**: Aggressive turning
- **VH (Very High)**: Maximum turning rate

The rule base follows a systematic design methodology where each rule combines proximity information with relative object position to determine appropriate turning behavior. For instance, rules governing obstacle avoidance generate stronger turning commands (H or VH) when objects appear in immediate proximity to the robot's path, while more distant obstacles trigger gentler adjustments (L or VL). This graduated response ensures smooth navigation while maintaining safety margins.

This implementation has shown superior performance compared to conventional controllers in scenarios requiring rapid adaptation to unexpected obstacles, as documented in our experimental validation (Section IV).

Table III.5: Fuzzy rules for angular velocity.

Distance/Turn [m]	**H Clockwise**	**Clockwise**	**L Clockwise**	**No Turn**	**L Counter-Clockwise**	**Counter-Clockwise**	**H Counter-Clockwise**
0–1	VH Left Turn	VH Left Turn	VH Left Turn	No Movement	VH Right Turn	VH Right Turn	VH Right Turn
0.5–2	H Left Turn	VH Left Turn	VH Left Turn	No Movement	VH Right Turn	VH Right Turn	VH Right Turn
1.5–3	M Left Turn	H Left Turn	M Left Turn	No Movement	M Right Turn	M Right Turn	M Right Turn
2.5–5	M Left Turn	H Left Turn	M Left Turn	No Movement	M Right Turn	M Right Turn	M Right Turn
4–6	M Left Turn	H Left Turn	M Left Turn	No Movement	M Right Turn	M Right Turn	M Right Turn

Subtitles: VH = Very High, H = High, L = Low, M = Moderate.

Initially, suitable speed values are predefined for the speed controller. Over time, these gains are dynamically updated by the fuzzy algorithm, relying solely on obstacle distance and objective distance values. Membership Functions (MF) choice is based on distance values that result in the desired closed-loop response. By avoiding crisp functions, the final gains may slightly deviate from those at the center of the membership function.

Meanwhile, the **rule evaluation** phase can be presumed as an IF-THEN statement, which is applied to each fuzzified input. A corresponding set of rules is generated that defines the systems response. This step classifies both distance and velocity groups to produce an arbitrary danger zone, which will be the input of the next stage.

$$\text{IF } d \text{ is } D_i \text{ AND } v \text{ is } V_j \text{ THEN } a \text{ is } A_{ij} \tag{III.10}$$

where D_i and V_j represent fuzzy sets for distance and velocity respectively, and A_{ij} denotes the consequent speed action.

Table III.6: Speed rules for the robot.

Distance	**Velocity**				
	Very Slow	**Slow**	**Medium**	**Fast**	**Very Fast**
0–1 m	zero	-0.1 m/s	-0.2 m/s	-0.4 m/s	-0.4 m/s
0.5–2 m	zero	zero	-0.1 m/s	-0.2 m/s	-0.4 m/s
1.5–3 m	$+0.1$ m/s	zero	zero	-0.1 m/s	-0.2 m/s
2.5–5 m	$+0.1$ m/s	$+0.1$ m/s	zero	zero	-0.1 m/s
4–6 m	$+0.1$ m/s	$+0.1$ m/s	$+0.1$ m/s	zero	-0.1 m/s

The rule table is developed through individual analyses of speed instances while near the objects.

Conversely, the **defuzzification** step receives the danger zone value and translates it into a precise

value. This involves converting acceleration and deceleration rates into speed adjustments for the robot. Then, the resulting output is modified by a class magnitude factor, which adjusts the speed based on the type of obstacle, e.g., if the robot detects a human, it reduces the speed more significantly than it would when encountering a toolbox or cone. Table III.7 defines some arbitrary acceleration multiplier factors. These values are chosen based on the hazard of a collision if any of the classes are detected in the robot's path. The authors consider both static (e.g., cone or toolbox) and dynamic issues (e.g., a person or robot). Based on this framework for the fuzzy module, the robot can dynamically navigate its environment by continuously adjusting its speed to avoid obstacles and optimize its trajectory. This real-time object detection system with a fuzzy module is key in ensuring the robot reaches its target destination safely.

Table III.7: Speed multipliers.

Class	Speedup Factor	Slowdown Factor
Toolbox	1.7	0.5
Robot	1.5	0.5
Cone	1.0	0.7
Person	0.7	1.0

The fuzzy logic implementation is visually represented in Figures III.14 and III.15. Figure III.14 displays the complete response surface of the fuzzy controller, mapping all possible combinations of distance and velocity inputs to corresponding velocity outputs. This three-dimensional plot reveals the nonlinear relationship between system variables

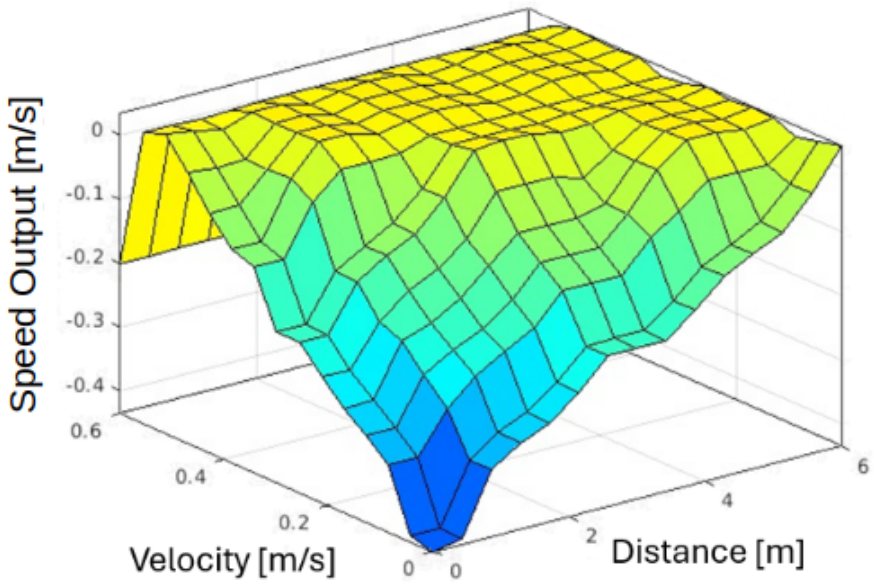


Figure III.14: Fuzzification process surface plot.

Figure III.15 specifically details the distance membership functions that transform crisp sensor measurements into fuzzy variables. The plot shows how:

- Triangular functions partition the input space into linguistic categories (Very Near to Very Far)
- Overlapping regions enable smooth transitions between adjacent categories
- Function widths are tuned to match the Kinect's depth measurement characteristics

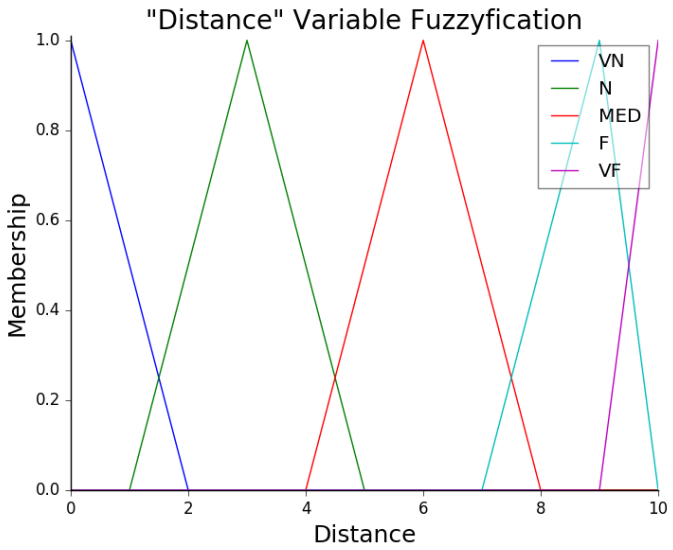


Figure III.15: Membership function of the distance.

This graphical representation complements the rule base in Table III.6, providing complete visibility into how sensor inputs propagate through the fuzzy system to generate motion commands. The surface plot's smooth contours confirm the controller's ability to produce gradual speedup and slowdown changes, while the membership functions demonstrate the careful calibration of input quantization for responsive yet stable operation.

III.3.5 Scripts and programming

As explained in the previous section, for this project is being used Python programming language in the version 2.7 for compatibility reasons, following rospy library in which enables node creation, topic subscription, publication and message exchange meanwhile adapting all the technological ground, implementation and experimentation provided by the most used programming language in the world.

In the subsequent section, the methodologies assumed earlier will be deployed across several scripts in parallel. This approach is sustained by the utilization of multitasking, which capitalizes on the enhanced performance capabilities of a multi-core processor. It's important to note that the efficacy and overall performance of rospy publisher and subscriber nodes can significantly fluctuate based on the frequency rates at which they operate. For instance, in the context of YOLOv3 detection, optimal object detection outcomes are achieved when the publisher node operates at a high frequency rate. Alternatively, the

subscriber node and topic publisher demonstrate superior performance in transmitting messages and updating topics when operating at a lower frequency publishing rate.

The system has three main parts working together: (1) YOLOv3 for obstacle detection, (2) depth estimation for measuring distances, and (3) fuzzy logic for speed control. Each part runs separately but shares information through ROS topics. The speed multipliers from Table III.7 are used to adjust the speed based on what kind of object is detected.

Below is the complete pseudocode showing how these components work together in parallel. The algorithm starts by setting up YOLOv3, point cloud data, and the fuzzy logic module. Then it enters a continuous loop where all three processes run simultaneously. The first process handles object detection, the second calculates distances, and the third determines the safe speed using fuzzy logic rules. The final speed is adjusted using the multipliers from the table before being published.

Algorithm 1 Autonomous Collaborative Environments Obstacle Avoidance Algorithm

x_0,y_0: Pixel coordinates centroid
ROI: Region of interest
u,v: Pixel coordinates of nearest point to the center
fx, fy: Focal lengths in x, y-axis
cx, cy: Optical center coordinates
D_o: Distance from detected object
V_i, V_o: Input and Output Velocity from Fuzzy process
f_a: Multiplicative factor separated by object class
ΔV: Speed difference to be applied
`/cmd_vel`: Default speed of the robot
`/cmd_vel_safe`: Solution's speed output
Initialize YOLOv3 model and load class labels
Initialize Point Cloud data distance estimation
Initialize Fuzzy Logic Speed module
while Running **do**
    // Parallel Block Begins
    fork:
        [Process 1] YOLOv3 obstacle detection
            Gets `/camera/color/image_raw`
            Applies to resize and blob creation
            Extract bounding box
            Calculate the centroid of the bounding box
            $x_0, y_0 \leftarrow$ bounding box centroid
        [Process 2] Obstacle distance estimation
            Gets `/camera/depth/points`
            $ROI = (x_0, y_0) + -Threshold$
            Extract (x, y, z) for each point in ROI
            Calculate distance from the camera to point:

$$u = \left(\frac{x \cdot fx}{z}\right) + cx, \quad v = \left(\frac{y \cdot fy}{z}\right) + cy \tag{1}$$

            if u, v coordinates are nearest to (x_0, y_0)
                z = Object distance
            `/fuzzy_input` $\leftarrow [u, v]$, Class label
            Publish `/fuzzy_input`
        [Process 3] Fuzzy module
            Gets `/fuzzy_input`
            Append `/fuzzy_input` as D_i and $(class)$
            Append `/cmd_vel` as V_i
            // General Fuzzy Logic Rules
            Apply Acceleration Rule based on D_i, V_i
            // Class-based adjustment function
            Multiplicative Factor $f_a \leftarrow$ Value from Table 6
            $\Delta V \leftarrow V_o \times f_a$
            `/cmd_vel_safe` $\leftarrow$ `/cmd_vel` $+ \Delta V$
            Publish `/cmd_vel_safe`
    join
    // Parallel Block Ends
end while

Figure III.16: Complete pseudocode describing the algorithm's logic pipeline.

Overall, the robotic navigation system, identified at **??** is implemented through three core algorithmic

components that will be described below:

- **Object Detection Node (YOLOv3 Processor)**: This Python script utilizes CvBridge to process RGB-D imagery from the */camera/color/image_raw* and */camera/depth/points* topics. It performs real-time object detection using YOLOv3-tiny, calculating both pixel coordinates (x_c, y_c) and 3D distances to detected obstacles. The node publishes classified objects with their respective distances to the */entrada_fuzzy* topic while maintaining a log file of detection events for system verification. Specialized safety radii are applied based on object class (human, robot, toolbox, etc.) through the dynamic parameter server.

- **Autonomous Navigation Core**: Comprising two interconnected components:

  - *Costmap Generator*: Constructs a 3D occupancy map from PointCloud2 data (*/camera/depth/points*), implementing ground plane removal and obstacle inflation layers. The costmap is published to */move_base/global_costmap/costmap* with configurable parameters including:

    * Robot footprint: 0.21×0.165 meters
    * Inflation radius: 0.3 meters $\angle$Z-axis filtering: 0.05-0.5 meters

  - *Waypoint Follower*: Reads sequential goals from a YAML configuration file, publishing target coordinates to */move_base_simple/goal*. The node subscribes to */odom* for real-time position feedback, advancing through waypoints when within ± 0.15 meter tolerance.

- **Fuzzy Velocity Modifier**: This ROS node subscribes to */entrada_fuzzy* (object data) and */cmd_vel* (base velocity), implementing a Mamdani-type fuzzy controller with:

  - Inputs: Distance (0-6 m), Current Velocity (0-2 m/s), Object Class
  - Output: Adjusted velocity (-0.4-$+0.1$ m/s^2)
  - Rule Base: 25 rules weighted by obstacle type

  The processed velocity commands are published to */cmd_vel_safe* after applying class-specific multipliers (Toolbox:1.7, Human:0.7, etc.).

This modular architecture ensures decoupled development while maintaining tight integration through standardized ROS topics and services. The system's performance stems from the synergistic operation of these components - where object detection informs navigation constraints, costmapping enables 3D obstacle avoidance, and fuzzy logic optimizes velocity profiles in real-time.

Following the implementation of core navigation capabilities, the research focused on evaluating system performance in realistic operational scenarios. To validate the system's performance under such conditions, we first evaluated its response to a moving human obstacle. When the robot reaches a

predefined trigger position, a simulated person begins moving along the negative Y-axis towards the robot's path (Figure III.17). This scenario tests the navigation stack's ability to detect and track dynamic obstacles in real-time, recalculate paths while maintaining safety margins, adjust velocity profiles through the fuzzy controller and preserve operational efficiency during avoidance maneuvers.

The system demonstrated effective collision avoidance by dynamically braking when the human entered the danger zone, while maintaining optimal clearance distances through continuous costmap updates. Figure III.17C shows the costmap trail generated by the moving person, illustrating how the depth camera and navigation stack collaboratively track dynamic obstacles.

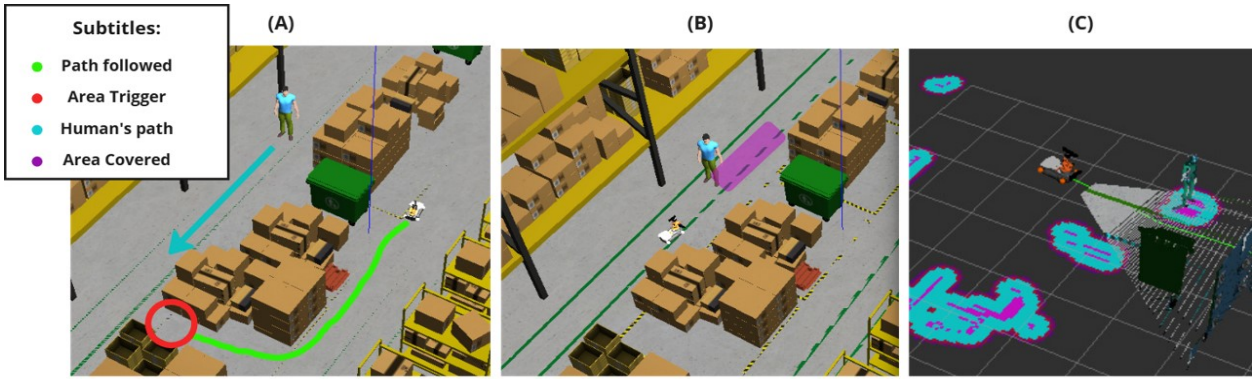


Figure III.17: (**A**) Initial test phase showing detection zones and safety margins. (**B**) Human movement trajectory during testing. (**C**) Costmap visualization of the moving person, showing real-time obstacle tracking and inflation layers.

To further evaluate reactive capabilities, we developed a second test simulating sudden obstacle appearances - a critical requirement for industrial safety, for example, trying to mimic an toolbox falling in front of the robot during path following. For this step, a background script monitors the robot's odometry and spawns static obstacles (e.g., toolboxes) at predetermined trigger points along the path (Figure III.18). This evaluates Immediate object detection latency, emergency braking performance, path re-planning efficiency and system recovery after unexpected stops.

The sudden obstacle test complements the dynamic human evaluation by verifying the system's response to instantaneous environmental changes. Both scenarios collectively validate the navigation stack's adaptability across the spectrum of potential workplace hazards, from gradually approaching workers to immediately appearing objects.

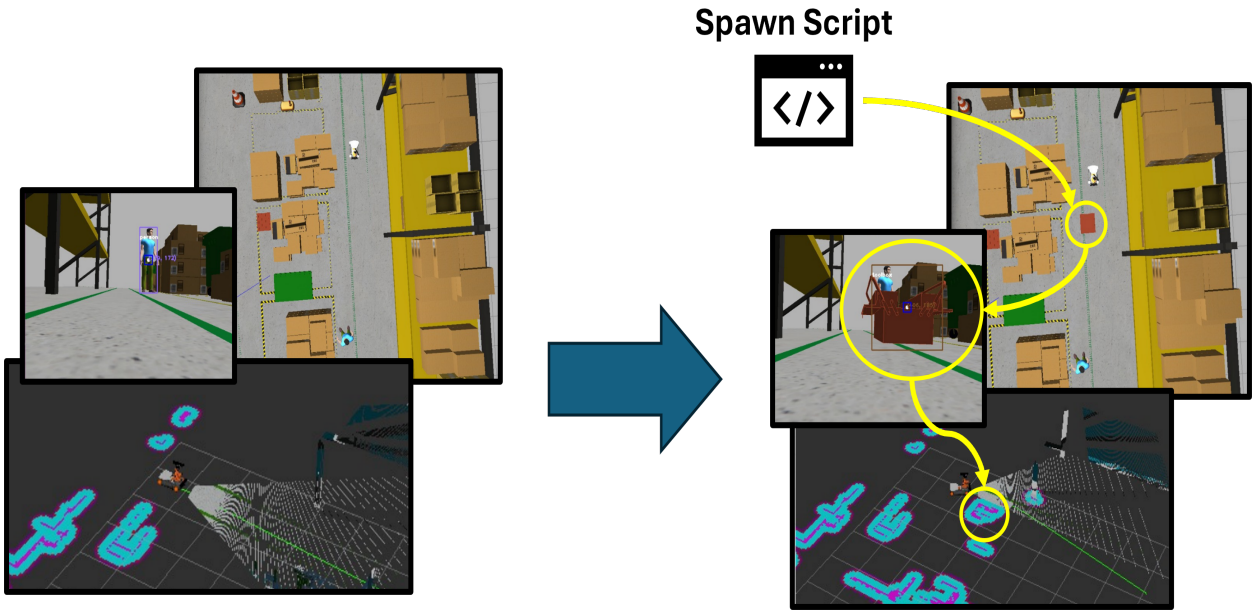


Figure III.18: Obstacle spawning mechanism: (**Left**) Normal operation, (**Right**) Toolbox appearing at the trigger point, testing emergency response capabilities.

Chapter IV Results and Discussion

IV.1 Neural Network Performance

The trained YOLOv3 model effectively detected key objects during the environment tests, prioritizing the accuracy of the person's class for safety. The model was trained on a custom dataset of 250 images using data augmentation and transfer learning. The model achieved a mean Average Precision (mAP) of 76.00% across five classes: cone, person, robot, toolbox, and Youbot. mAP is a commonly used evaluation metric in object detection that reflects both the accuracy and localization quality of predicted bounding boxes. Specifically, mAP represents the mean of the Average Precision (AP) computed across all object classes, where AP accounts for both precision (how many of the detected objects are correct) and recall (how many of the actual objects were detected). A high AP (e.g., 89.60% for the person class) does not imply that 10.4% of persons are entirely missed, but rather that, across varying confidence thresholds, there is a 10.4% discrepancy in achieving perfect precision and recall. Moreover, part of this discrepancy may be due to minor inaccuracies in bounding box localization or confidence score variations, rather than complete detection failures. To provide further context, the dataset contained the following distribution of labeled instances: Youbot (36), Toolbox (67), Robot (64), Cone (75), and Person (140). Figure IV.1 provides a visualization of the mAP across all classes and showcases the system's ability to balance accurate detection and efficient navigation. Furthermore, class-specific adjustments during training should enhance both the robustness and reliability of the detection system.

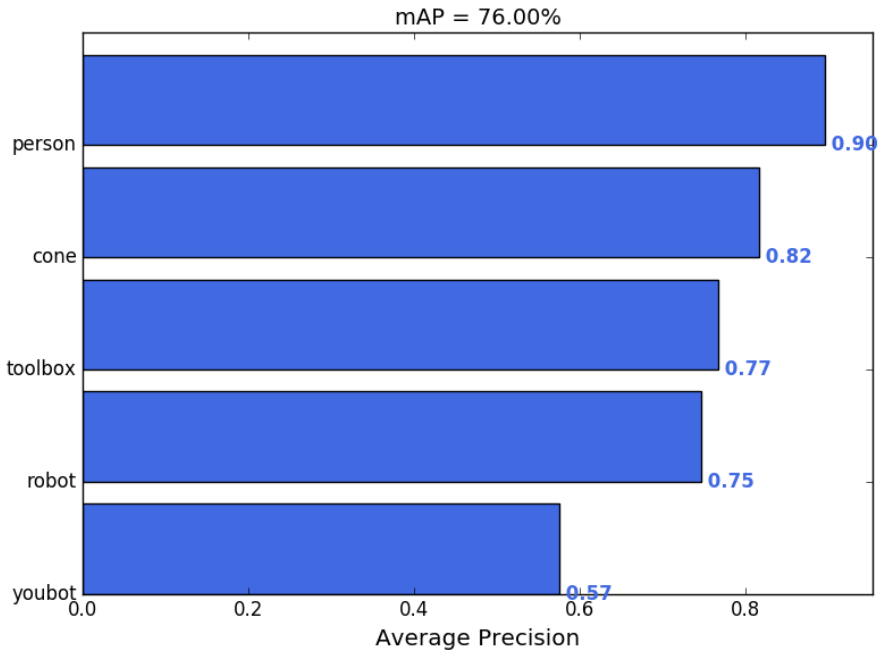


Figure IV.1: The mAP values corresponding to the trained Yolov3 Custom Model.

The person class achieved the highest Average Precision (AP) at 89.60%, followed by the cone

(81.57%) and toolbox (76.69%). These results reflect the training's emphasis on safety-critical classes, where high precision (93%) and recall (94%) for the person class ensure consistent and reliable detection. Note that the Youbot class displayed the lowest AP at 57.47%. However, it is important to observe that this class was not the primary focus of the training process. The Youbot, being the robot utilized in the project, was not expected to exhibit the same detection accuracy as other common warehouse robots, which were represented by the "robot" class in the training dataset. Therefore, the detection performance for the Youbot should not be seen as a theoretical limitation but rather as an intended outcome, as its detection was less critical compared to other object classes.

Figure IV.2 presents the validation loss curve throughout the YOLOv3 training process. The loss decreases significantly in the initial iterations, demonstrating rapid learning, and progressively stabilizes as training advances. This behavior indicates effective parameter optimization and convergence. The corresponding training Colab notebook, detailing the entire process, is available at the following link: YOLOv3 Training Notebook.

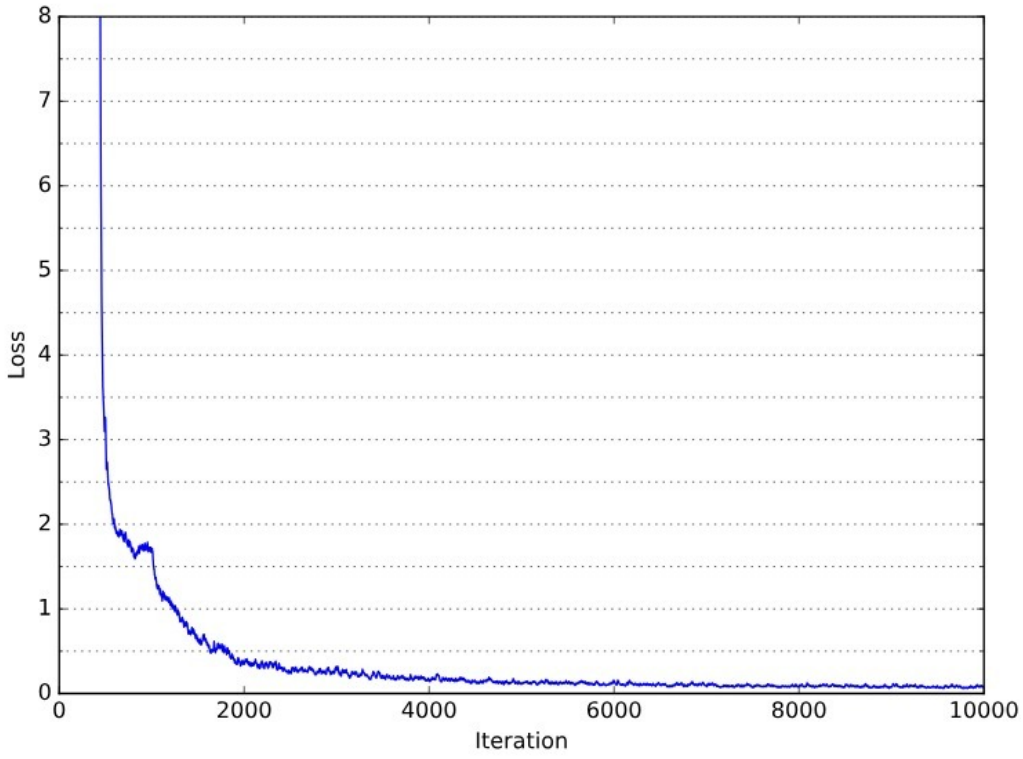


Figure IV.2: Evolution of the validation loss during the YOLOv3 training. The steady decrease and stabilization of the curve indicate effective learning and convergence.

IV.2 Autonomous collaborative results

This section presents the results of the proposed autonomous collaborative solutions performance in a warehouse environment. Figure IV.3 illustrates the map generated by the robot. This costmap effectively identifies potential obstacles and available spaces within the workspace. Furthermore, the continuous map updating enables the robot to maintain a high level of spatial awareness, which is crucial to guarantee smooth and efficient operations in collaborative scenarios.

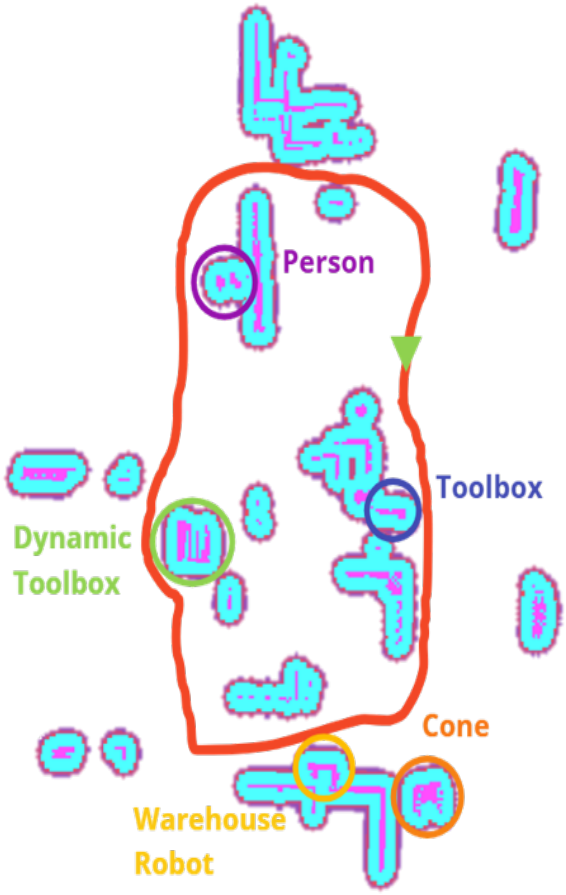


Figure IV.3: A Real-time costmap generated by the robots Kinect RGB-D camera with objects, highlighting objects, the start of the test, and the average path.

Figure IV.4 compares 2D paths derived from odometry data, highlighting the effectiveness of different situations. The red and blue lines show the robots trajectories under different configurations, demonstrating the systems effectiveness in obstacle avoidance and the effectiveness of object-aware navigation using Fuzzy Logic in simulated environments. In this context, the Default configuration refers to the standard ROS Navigation Stack setup, which performs 2D costmap-based path planning and obstacle avoidance without any semantic perception or adaptive control mechanisms. On the other hand, the Solution corresponds to our proposed approach, which integrates object classification through

YOLOv4, depth perception via Kinect RGB-D sensing, and a fuzzy logic controller that dynamically adjusts the robots navigation strategy based on the type and proximity of detected obstacles.



Figure IV.4: Comparison of navigation paths in a dynamic warehouse environment. The blue trajectory corresponds to the Default configuration (standard ROS Navigation Stack without semantic perception), while the red trajectory represents the proposed Solution, which integrates object detection (YOLOv4), depth sensing (RGB-D), and fuzzy logic-based adaptive control.

After conducting several tests comparing the standard navigation stack and the proposed method for an autonomous obstacle-avoiding mobile robot in human-robot collaborative warehouse environments, the robustness of the algorithm has been demonstrated, particularly in settings where safety and adaptability are the biggest concerns. To ensure the reliability and representativeness of the results, only the best runs out of 32 tests were considered for averaging. Runs that exhibited atypical deviationssuch as abnormally low speeds, sudden stops, or irregular braking patternswere excluded, as they were deemed outliers that could introduce noise and distort the overall assessment of performance. This filtering process ensured that the averages reflected the typical behavior of the system rather than being disproportionately influenced by anomalous cases. Table IV.1 summarizes the results of these experiments, indicating that this approach not only enhances safety but also improves travel efficiency.

The data shows that the Solution achieved faster average lap times than the Default setup in static, sudden toolbox appearance, and dynamic tests (moving person). The braking logic, based on object proximity, provides smoother speed control, leading to quicker and more consistent laps. Controlled deceleration minimizes the need for sharp turns or abrupt corrections, allowing the path-planning algorithm to function with less strain and deliver a more consistent route. This also enhances

Table IV.1: Test Results for Static, Sudden, and Dynamic Environments.

Static Test Results (average best runs from 32 tests)					
Type	**Completion Rate**	**Speed (m/s)**	**Distance (m)**	**Path Variation (m)**	**Collisions**
Default	100%	0.463	25.62	4.57	0
Solution	100%	0.441	25.22	4.51	0
Largest Speed Difference (m/s)		0.124			
Sudden Toolbox Test Results (average best runs from 32 tests)					
Type	**Completion Rate**	**Speed (m/s)**	**Distance (m)**	**Path Variation (m)**	**Collisions**
Default	100%	0.448	27.31	4.41	0
Solution	100%	0.422	25.87	4.64	0
Largest Speed Difference (m/s)		0.182			
Moving Person Test Results (average best runs from 32 tests)					
Type	**Completion Rate**	**Speed (m/s)**	**Distance (m)**	**Path Variation (m)**	**Collisions**
Default	100%	0.484	25.23	4.39	0
Solution	100%	0.451	24.86	4.55	0
Largest Speed Difference (m/s)		0.207			

the robot's cargo transportation capability, reducing the chance of an overturn in the load.

The outcomes also show that path variation metrics indicate a slightly smoother trajectory with the Solution, especially in static environments. In sudden tests, where a toolbox unexpectedly appeared in front of the robot during the track, the Solution effectively balanced speed and safety, maintaining collision-free performance. Similarly, in dynamic tests involving a moving human obstacle, the Solution demonstrated superior adaptability and safety margins, as evidenced by the larger distances maintained compared to the default navigation stack. This improvement in trajectory smoothness and safety confirms that the vision-fuzzy system offers a more robust and adaptive solution than traditional odometer-based navigation.

IV.3 Safety Conduct Analysis

Safety is the biggest concern in autonomous robots operating in collaborative environments, as real-world applications involve dynamic and constantly changing scenarios. To evaluate the safety performance of the proposed system, the authors conducted both static and dynamic tests. During these tests, odometry data was collected as the robot navigated through environments with obstacles, providing distance measurements from detected objects. This data was crucial for assessing the effectiveness of the proposed system in maintaining a larger safety buffer without significantly compromising task efficiency.

Figures IV.5 and IV.6 demonstrate that, in nearly all instances, our solution maintained a greater distance from detected objects compared to the default navigation method. These results highlight the ability of our approach to allow the robot to operate with a larger error margin while ensuring a safe distance from obstacles. Notably, during the sudden appearance toolbox tests, the average minimum distance consistently increased across all trials, underscoring the effectiveness of our object-aware navigation system in enhancing safety without sacrificing operational efficiency.

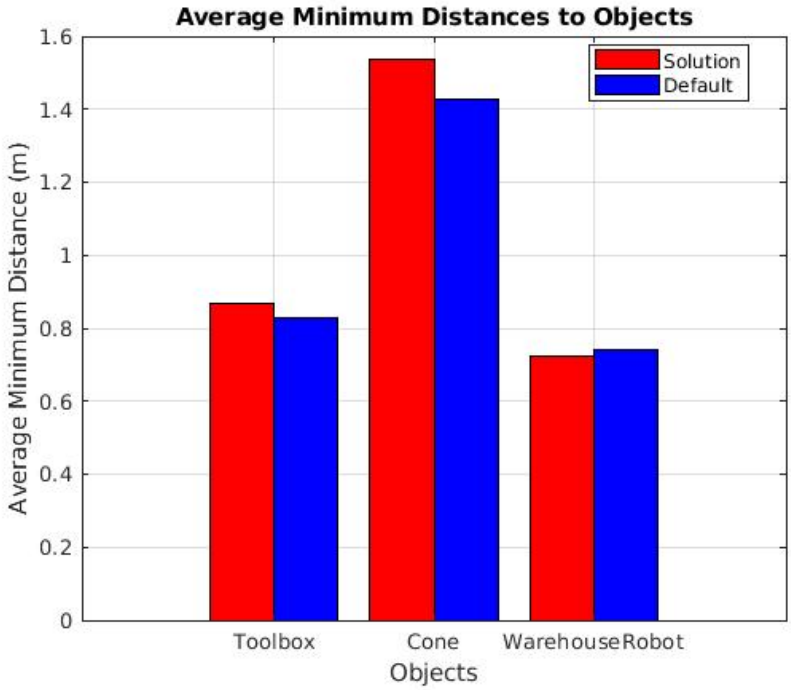


Figure IV.5: Minimum distances from the robot to the respective objects. Data was acquired during static toolbox tests.

In the dynamic obstacle tests, specifically Figure IV.7, a moving obstacle (a human) was introduced when the robot entered a specific area. However, due to variations in the robot's speed across tests, the time taken to approach the human after the trigger point differed, resulting in inconsistent and random relative positions between the robot and the obstacle. Since the human obstacle lacked an odometry sensor and was not in a fixed position, directly tracking the minimum distance between the robot and the obstacle proved to be nearly impossible. To address this challenge, an alternative approach was adopted: the minimum distance was estimated by identifying a coordinate where both the solution and default tests exhibited the most significant change in direction along the robot's path. This change in direction is assumed to correspond to the moment when the robot detects the moving human, brakes, and alters its trajectory. While this method introduces some noise into the results, it does not significantly impact the overall evaluation of the system's performance, as no collisions occurred in any of the tests.

The results revealed a substantial difference between the distances maintained by our solution and those of the default navigation stack. Our solution consistently demonstrated significantly larger

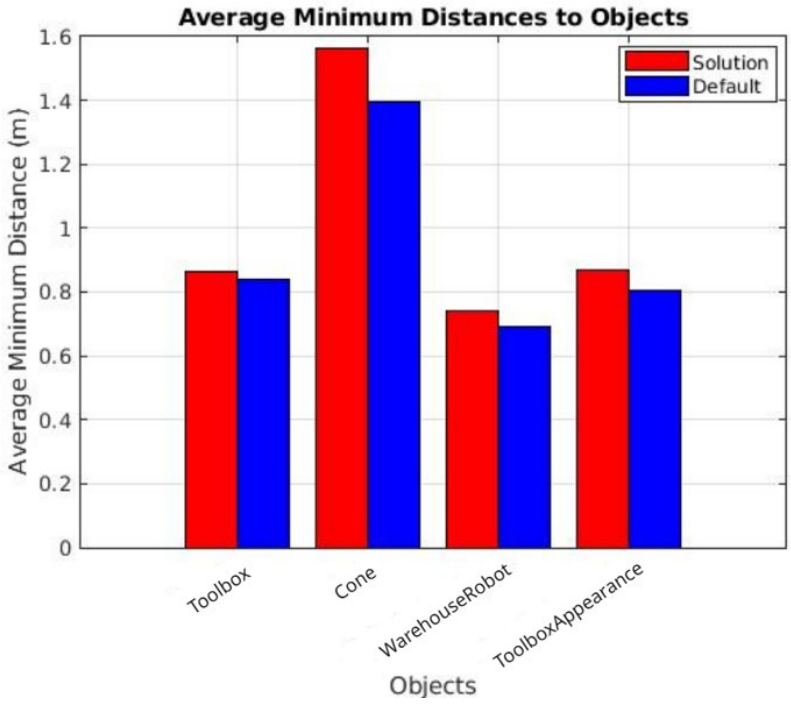


Figure IV.6: Average minimal distances from the robot to the objects. Data was acquired during sudden toolbox appearance tests.

safety margins, which not only validates the effectiveness of our approach but also highlights its prioritization of safety in hazardous situations involving moving humans in the robot's autonomous path. This improvement in maintaining safe distances shows the system's ability to handle dynamic and unpredictable environments with a higher degree of safety.

Figures IV.8, IV.9, and IV.10 illustrate the most significant differences in braking behavior between the default and proposed solutions applied in static, sudden obstacle appearing, and dynamic tests. The default navigation path is shown in blue, and the red marks illustrate the points where the highest braking intensity was observed. Meanwhile, the purple arrow indicates the moving direction of the dynamic person obstacle in the dynamic tests. These figures demonstrate that our integrated system finds the best braking point based on object detection, especially as the robot approaches obstacles. Notably, there is an expressive difference when the robot is approaching a human being, particularly in the dynamic test, as even at minimal proximity to a person (i.e., the closest detectable distance), the system still engaged in meaningful braking. This underscores the crucial role of object detection in the robots speed controller and overall safety.

While our solution, applied specifically for Warehouse robot detection, performed slightly lower than the default technology in certain tests, particularly in scenarios involving human approaches, these results can be attributed to the limitations of the sensor setup. The depth camera, mounted at the base of the Kuka Youbot, encounters challenges when navigating curves, as detection occurs immediately after a turn. Furthermore, the cameras height limits its field of view, often cropping the legs of a tall person,

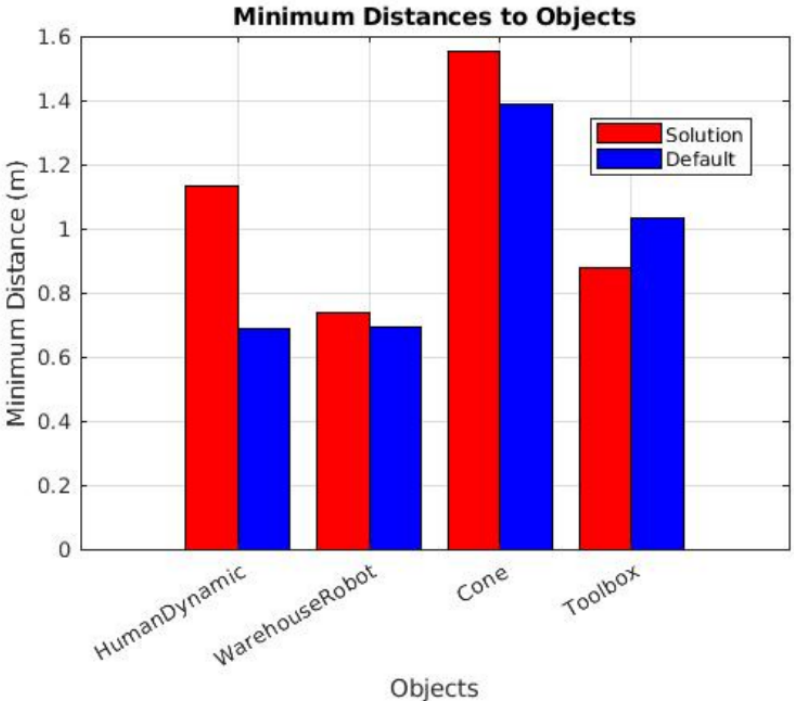


Figure IV.7: Estimated minimum distances from the robot to the moving human obstacle. Data was acquired during dynamic tests by analyzing the point of maximum directional change.

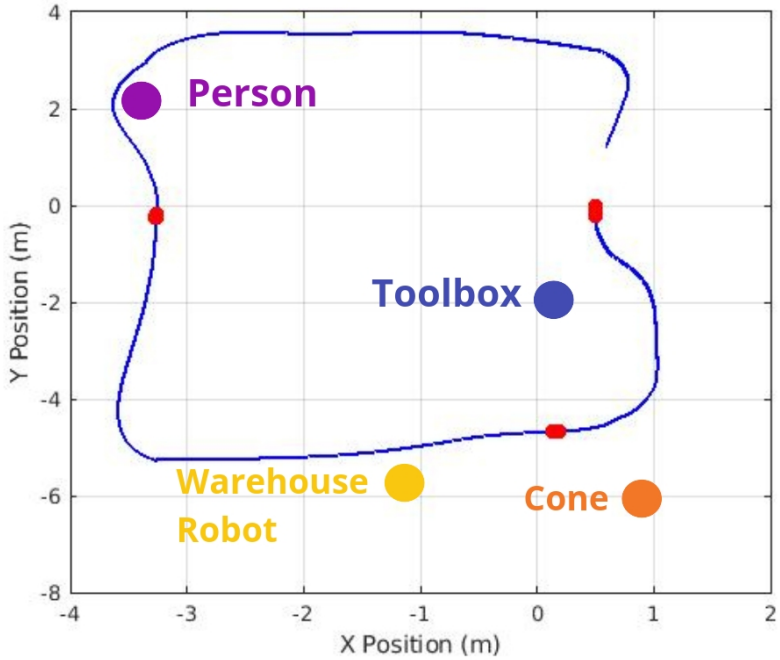


Figure IV.8: Path of the robot during static tests (blue). The highest brake intensity (marked in red) happens when a large gap between the standard velocity value and the object detection solution with fuzzy logic output is detected.

which does not allow the algorithm to detect the full body. Consequently, these constraints prevented the system from improving performance during all tests and scenarios.

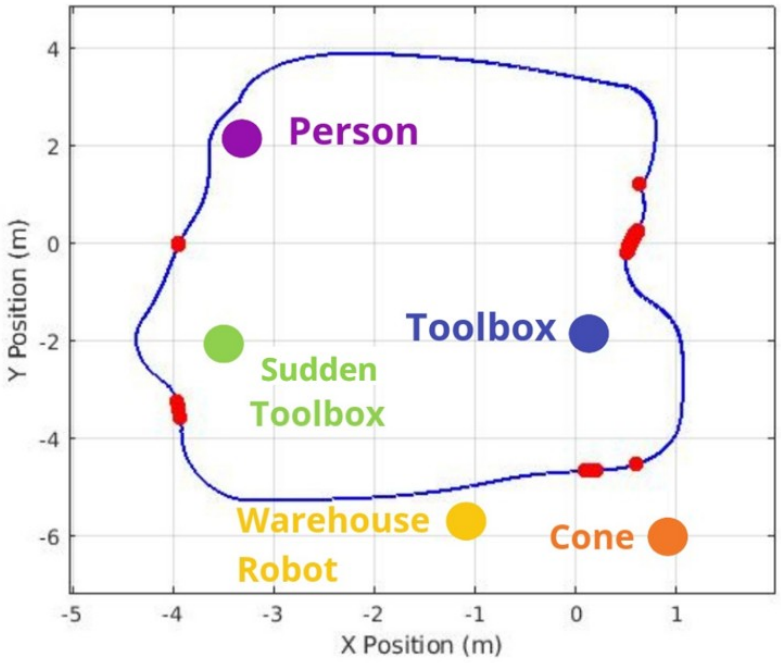


Figure IV.9: Second braking intensity plot now applied to the sudden obstacle results, adding the sudden toolbox location.

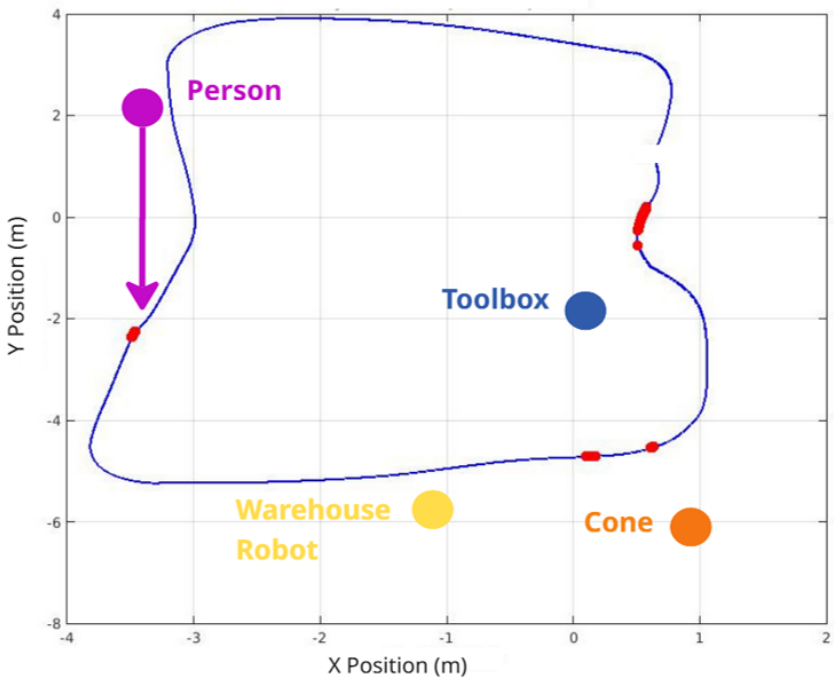


Figure IV.10: Braking intensity plot results on a dynamic obstacle environment, in which the person moves in the negative Y-axis position, creating a notably difficult test that highlights the robustness of the solution in the simulation.

Chapter V Conclusions and Future Work

V.1 Concluding Remarks

This work presents a novel and effective approach for enhancing the navigation and safety capabilities of mobile manipulators operating in collaborative environments. By integrating a Fuzzy-Speed control strategy with real-time object detection using YOLOv3 and leveraging both eye-in-hand and eye-to-hand Kinect RGB-D sensor configurations, the system achieved significant improvements in dynamic obstacle avoidance and adaptive speed regulation. The robot demonstrated the ability to adjust its behavior in real-time based on the class, distance, and speed of detected objects, contributing to safer interactions within shared workspaces.

Simulation results validated the systems efficacy in various scenarios, particularly those involving human presence, a critical challenge in human-robot collaboration (HRC). The proposed framework consistently exhibited safe braking behavior and maintained greater safety margins compared to the default ROS Navigation Stack, without compromising operational efficiency. These findings underscore the feasibility of implementing hybrid perception-based control strategies for real-time decision-making in industrial environments.

Furthermore, the study highlighted key insights regarding the importance of sensor positioning and field-of-view limitations, especially in scenarios requiring human detection. Despite some constraints introduced by the sensor layout, the system maintained robust performance, confirming its suitability for applications demanding high safety standards.

Overall, the integration of fuzzy logic and CNN-based perception proved to be a promising solution for achieving responsive and context-aware robotic behavior. The research provides a solid foundation for future developments targeting more sophisticated collaborative robotic systems, capable of operating safely and autonomously in complex, real-world industrial settings.

V.2 Future Ideas

New implementations are foreseen for future work. For instance, although more recent object detection models exist (e.g., YOLOv5, YOLOv8, and Transformer-based architectures), YOLOv3-Tiny was selected in this study for its adequate accuracy and reliable real-time performance. However, we intend to explore the integration of more advanced models in future real-world deployments to enhance detection robustness and generalization, potentially. Another important future improvement includes the integration of 3D object detection techniques to estimate object shapes and volumes better. Although the current system uses depth data and PointCloud2-based costmaps for 3D-aware navigation, the explicit reconstruction of object geometries through 3D bounding boxes or mesh models may further improve obstacle representation, particularly in complex or cluttered environments. This capability could enhance motion planning, particularly when the object's shape affects clearance and safety margins. Additionally, the authors will focus on enhancing the robots perception capabilities by incorporating sensors such as LiDAR and expanding testing to real-world industrial environments to validate the system's scalability and performance under diverse conditions. Additionally, future work will include more comprehensive experiments by incorporating complex scenarios with additional obstacles, barriers, and dynamic elements, such as moving persons. These enhancements aim to ensure more realistic evaluations and further refine the systems adaptability and robustness in collaborative industrial settings. Further research will also explore the integration of more advanced deep learning algorithms to enhance obstacle detection and real-time decision-making. Finally, future work will include a theoretical framework to model and analyze the relationships between control inputs, system dynamics, and environmental uncertainties.

Bibliography

ABDALLAOUI, S. et al. Thorough review analysis of safe control of autonomous vehicles: Path planning and navigation techniques. *Energies*, v. 15, n. 4, 2022. ISSN 1996-1073. Disponível em: <https://www.mdpi.com/1996-1073/15/4/1358>https://www.mdpi.com/1996-1073/15/4/1358. Citado na página 28.

ABDULSAHEB, J. A.; KADHIM, D. J. Real-time slam mobile robot and navigation based on cloud-based implementation. *Journal of Robotics*, v. 2023, n. 1, p. 9967236, 2023. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2023-/9967236>https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/9967236. Citado na página 28.

ACHIREI, S.-D. et al. Model-predictive control for omnidirectional mobile robots in logistic environments based on object detection using cnns. *Sensors*, MDPI, v. 23, n. 4992, 2023. Citado na página 30.

AGUIAR, A. S. et al. Localization and mapping on agriculture based on point-feature extraction and semiplanes segmentation from 3d lidar data. *Frontiers in Robotics and AI*, v. 9, 2022. ISSN 2296-9144. Disponível em: <https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2022-.832165>https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2022.832165. Citado na página 28.

AIVALIOTIS, P. et al. Power and force limiting on industrial robots for human-robot collaboration. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 59, p. 346–360, 2019. Citado na página 27.

AJOUDANI, A. et al. Progress and prospects of the human–robot collaboration. *Autonomous robots*, Springer, v. 42, p. 957–975, 2018. Citado na página 28.

ALI, F.; KIM, E. K.; KIM, Y.-G. Type-2 fuzzy ontology-based semantic knowledge for collision avoidance of autonomous underwater vehicles. *Information Sciences*, Elsevier, v. 295, p. 441–464, 2015. Citado 2 vezes nas páginas 29 e 30.

ALKHALEEFAH, M. et al. Yolov3-mobile for real-time pedestrian detection on embedded gpu. In: *Proceedings of the 5th International Conference on Graphics and Signal Processing*. [S.l.: s.n.], 2021. p. 27–31. Citado na página 29.

ANDRADE, F. A. et al. Unmanned aerial vehicles motion control with fuzzy tuning of cascaded-pid gains. *Machines*, MDPI, v. 10, n. 1, p. 12, 2021. Citado na página 14.

ASKARPOUR, M. et al. Formal model of human erroneous behavior for safety analysis in collaborative robotics. *Robotics and computer-integrated Manufacturing*, Elsevier, v. 57, p. 465–476, 2019. Citado na página 27.

AVANZINI, G. B. et al. Safety control of industrial robots based on a distributed distance sensor. *IEEE Transactions on Control Systems Technology*, IEEE, v. 22, n. 6, p. 2127–2140, 2014. Citado na página 29.

BAYONA, E.; SIERRA-GARCÍA, J.-E.; SANTOS, M. Keeping safe distance from obstacles for autonomous vehicles by genetic algorithms. In: BRINGAS, P. G. et al. (Ed.). *18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023)*. Cham: Springer Nature Switzerland, 2023. p. 300–310. ISBN 978-3-031-42536-3. Citado na página 29.

BENGTSON, S. H. et al. A review of computer vision for semi-autonomous control of assistive robotic manipulators (arms). *Disability and Rehabilitation: Assistive Technology*, Taylor & Francis, v. 15, n. 7, p. 731–745, 2020. Citado 3 vezes nas páginas 24, 25 e 34.

BERX, N. et al. Assessing system-wide safety readiness for successful human–robot collaboration adoption. *Safety*, MDPI, v. 8, n. 3, p. 48, 2022. Citado na página 14.

BI, Z. M. et al. Safety assurance mechanisms of collaborative robotic systems in manufacturing. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 67, p. 102022, 2021. Citado 2 vezes nas páginas 14 e 15.

BIUNDINI, I. Z. et al. Coverage path planning optimization for slopes and dams inspection. In: SILVA, M. F. et al. (Ed.). *Robot 2019: Fourth Iberian Robotics Conference*. Cham: Springer International Publishing, 2020. p. 513–523. Citado na página 13.

BONCI, A. et al. Human-robot perception in industrial environments: A survey. *Sensors*, MDPI, v. 21, n. 5, p. 1571, 2021. Citado na página 24.

BRITO, T. et al. Dynamic collision avoidance system for a manipulator based on rgb-d data. In: SPRINGER. *ROBOT 2017: Third Iberian Robotics Conference: Volume 2*. [S.l.], 2018. p. 643–654. Citado na página 24.

BURRUS, N. *Kinect Calibration*. 2011. OpenKinect Project. Online resource for Kinect v1 calibration parameters. Disponível em: <https://openkinect.org/wiki-/Kinect_calibration>https://openkinect.org/wiki/Kinect_calibration. Citado 3 vezes nas páginas viii, 50 e 51.

CASTRO, G. G. de et al. Dynamic path planning based on neural networks for aerial inspection. *Journal of Control, Automation and Electrical Systems*, Springer, v. 34, n. 1, p. 85–105, 2023. Citado na página 13.

CHEN, C. et al. Interval type-2 fuzzy disturbance observer-based t–s fuzzy control for a pneumatic flexible joint. *IEEE Transactions on Industrial Electronics*, IEEE, v. 69, n. 6, p. 5962–5972, 2021. Citado na página 14.

CHETTY, R. K.; PONNAMBALAM, S. A heuristic approach towards path planning and obstacle avoidance control of planar manipulator. In: SPRINGER. *International Conference on Intelligent Robotics, Automation, and Manufacturing*. [S.l.], 2012. p. 1–11. Citado na página 14.

CHOI, J. et al. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 502–511. Citado na página 44.

COELHO, F. O. et al. Autonomous uav exploration and mapping in uncharted terrain through boundary-driven strategy. *IEEE Access*, v. 12, p. 92464–92483, 2024. Citado na página 16.

DANG, C. V. et al. Collision-free navigation in human-following task using a cognitive robotic system on differential drive vehicles. *IEEE Transactions on Cognitive and Developmental Systems*, v. 15, n. 1, p. 78–87, 2023. Citado 2 vezes nas páginas 28 e 30.

De Ryck, M.; VERSTEYHE, M.; DEBROUWERE, F. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, v. 54, p. 152–173, 2020. ISSN 0278-6125. Disponível em: <https://www.sciencedirect.com/science/article-/pii/S0278612519301177>https://www.sciencedirect.com/science/article/pii/S0278612519301177. Citado na página 28.

DIWAN, T.; ANIRUDH, G.; TEMBHURNE, J. V. Object detection using yolo: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, v. 82, n. 6, p. 9243–9275, 2023. Disponível em: <https://doi.org/10.1007/s11042-022-13644-y>https://doi.org/10.1007/s11042-022-13644-y. Citado na página 16.

DONG, J.; HE, B. Novel fuzzy pid-type iterative learning control for quadrotor uav. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 1, p. 24, 2019. Citado na página 15.

DU, H. et al. A path planning approach for mobile robots using short and safe q-learning. *PLoS ONE*, v. 17, 2022. Citado na página 28.

DU, J. Understanding of object detection based on cnn family and yolo. *Journal of Physics: Conference Series*, IOP Publishing, v. 1004, n. 1, p. 012029, apr 2018. Disponível em: <https://dx.doi.org/10-.1088/1742-6596/1004/1/012029>https://dx.doi.org/10.1088/1742-6596/1004/1/012029. Citado na página 16.

ELGUEA-AGUINACO, Í. et al. A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 81, p. 102517, 2023. Citado na página 13.

ELSHEIKH, E. A.; ABDO, M. I. Performance analysis for fractional order fuzzy pid controller: A comparative study with various pid controller structures. In: IEEE. *2022 18th International Computer Engineering Conference (ICENCO)*. [S.l.], 2022. v. 1, p. 80–85. Citado na página 14.

FABRIZIO, F.; LUCA, A. D. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, IEEE, v. 2, n. 1, p. 56–63, 2016. Citado na página 24.

FAN, J.; ZHENG, P.; LI, S. Vision-based holistic scene understanding towards proactive human–robot collaboration. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 75, p. 102304, 2022. Citado na página 29.

GAGO, J. et al. Uavs challenge to assess water stress for sustainable agriculture. *Agricultural water management*, Elsevier, v. 153, p. 9–19, 2015. Citado na página 13.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 25.

GIUFFRIDA, G.; MEONI, G.; FANUCCI, L. A yolov2 convolutional neural network-based human–machine interface for the control of assistive robotic manipulators. *Applied Sciences*, MDPI, v. 9, n. 11, p. 2243, 2019. Citado 2 vezes nas páginas 15 e 25.

GOODRICH, M. A.; SCHULTZ, A. C. Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, Now Publishers Inc., v. 1, n. 3, p. 203–275, 2007. Citado na página 26.

GRANOSIK, G. et al. Using robot operating system for autonomous control of robots in eurobot, erc and robotour competitions. *Acta Polytechnica CTU Proceedings*, v. 6, p. 11, 11 2016. Citado 2 vezes nas páginas ix e 40.

GRAU, A. et al. Robots in industry: The past, present, and future of a growing collaboration with humans. *IEEE Industrial Electronics Magazine*, IEEE, v. 15, n. 1, p. 50–61, 2020. Citado na página 29.

HAN, D. et al. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, MDPI, v. 23, n. 7, p. 3762, 2023. Citado na página 13.

HENTOUT, A.; MAOUDJ, A.; AOUACHE, M. A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots. *Artificial Intelligence Review*, Springer, v. 56, n. 4, p. 3369–3444, 2023. Citado 3 vezes nas páginas 13, 14 e 15.

HIEMANN, A. et al. Enhancement of speed and accuracy trade-off for sports ball detection in videosfinding fast moving, small objects in real time. *Sensors*, v. 21, n. 9, 2021. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/21/9/3214>https://www.mdpi.com/1424-8220/21/9/3214. Citado na página 44.

JIN, S.; CHOI, B.-J. Fuzzy logic system based obstacle avoidance for a mobile robot. In: *Communications in Computer and Information Science*. [S.l.: s.n.], 2011. v. 256, p. 1–6. ISBN 978-3-642-26009-4. Citado na página 26.

JIN, Z.; PAGILLA, P. R. Collaborative operation of robotic manipulators with human intent prediction and shared control. In: IEEE. *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*. [S.l.], 2020. p. 1–6. Citado 2 vezes nas páginas 13 e 14.

KIM, B.; SUMI, Y. Vision-based safety-related sensors in low visibility by fog. *Sensors*, v. 20, p. 2812, 05 2020. Citado na página 36.

KODATE, N. et al. Home-care robots–attitudes and perceptions among older people, carers and care professionals in ireland: A questionnaire study. *Health & Social Care in the Community*, Wiley Online Library, v. 30, n. 3, p. 1086–1096, 2022. Citado na página 13.

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. [S.l.], 2004. v. 3, p. 2149–2154. Citado na página 17.

KULIC, D.; CROFT, E. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, Boston: Kluwer Academic Publishers, 1994-, v. 22, n. 2, p. 149–164, 2007. Citado na página 29.

KULIĆ, D.; CROFT, E. A. Real-time safety for human–robot interaction. *Robotics and Autonomous Systems*, Elsevier, v. 54, n. 1, p. 1–12, 2006. Citado na página 29.

KUMAR, A.; KUMAR, V. Evolving an interval type-2 fuzzy pid controller for the redundant robotic manipulator. *Expert Systems with Applications*, Elsevier, v. 73, p. 161–177, 2017. Citado na página 13.

KUMAR, A. et al. Design of a novel mixed interval type-2 fuzzy logic controller for 2-dof robot manipulator with payload. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 123, p. 106329, 2023. Citado na página 13.

LABBÉ, M.; MICHAUD, F. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, Wiley Online Library, v. 36, n. 2, p. 416–446, 2019. Citado na página 15.

LI, W. et al. Safe human–robot collaboration for industrial settings: A survey. *Journal of Intelligent Manufacturing*, Springer, v. 35, n. 5, p. 2235–2261, 2024. Citado na página 15.

LIU, L. et al. Application, development and future opportunities of collaborative robots (cobots) in manufacturing: A literature review. *International Journal of Human–Computer Interaction*, Taylor & Francis, v. 40, n. 4, p. 915–932, 2024. Citado na página 15.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. [S.l.], 2016. p. 21–37. Citado na página 25.

LIU, Y. et al. Self-tuning control of manipulator positioning based on fuzzy pid and pso algorithm. *Frontiers in Bioengineering and Biotechnology*, Frontiers, v. 9, p. 817723, 2022. Citado na página 15.

LOCHAN, K.; ROY, B. K. Control of two-link 2-dof robot manipulator using fuzzy logic techniques: a review. In: SPRINGER. *Proceedings of Fourth International Conference on Soft Computing for Problem Solving: SocProS 2014, Volume 1*. [S.l.], 2014. p. 499–511. Citado na página 15.

LONG, P. et al. An industrial security system for human-robot coexistence. *Industrial Robot: An International Journal*, Emerald Publishing Limited, v. 45, n. 2, p. 220–226, 2017. Citado na página 27.

LóPEZ-VELáSQUEZ, J.; ACOSTA-AMAYA, G. A.; JIMéNEZ-BUILES, J. A. Enhancing mobile robot navigation: Integrating reactive autonomy through deep learning and fuzzy behavior. *Revista EIA*, Universidad EIA, Envigado, Colombia, v. 21, n. 42, p. 1–14, 2024. Citado na página 30.

MOON, A. P.; JAJULWAR, K. et al. Design of adaptive fuzzy tracking controller for autonomous navigation system. *International Journal of Recent Trend in Engineering and Research*, v. 2, n. 2, p. 268–275, 2016. Citado na página 30.

MUKHERJEE, D. et al. A survey of robot learning strategies for human-robot collaboration in industrial settings. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 73, p. 102231, 2022. Citado na página 27.

NTAKOLIA, C.; MOUSTAKIDIS, S.; SIOURAS, A. Autonomous path planning with obstacle avoidance for smart assistive systems. *Expert Systems with Applications*, Elsevier, v. 213, p. 119049, 2023. Citado na página 14.

OU, Y. et al. Autonomous navigation by mobile robot with sensor fusion based on deep reinforcement learning. *Sensors*, v. 24, n. 12, 2024. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/24/12/3895>https://www.mdpi.com/1424-8220/24/12/3895. Citado na página 28.

Padilla, R.; Netto, S. L.; da Silva, E. A. B. A survey on performance metrics for object-detection algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.: s.n.], 2020. p. 237–242. Citado na página 46.

PINTO, M. F. et al. Arcog: An aerial robotics cognitive architecture. *Robotica*, Cambridge University Press, p. 1–20, 2020. Citado na página 13.

PINTO, M. F. et al. A robotic cognitive architecture for slope and dam inspections. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 16, p. 4579, 2020. Citado na página 13.

RACHKOV, M. Y.; WU, Q. Ultrasonic location system of a reconfigurable mobile robot. In: *2024 International Russian Smart Industry Conference (SmartIndustryCon)*. [S.l.: s.n.], 2024. p. 994–999. Citado na página 28.

RAMOS, G. S. et al. Hybrid methodology based on computational vision and sensor fusion for assisting autonomous uav on offshore messenger cable transfer operation. *Robotica*, Cambridge University Press, v. 40, n. 8, p. 2786–2814, 2022. Citado 2 vezes nas páginas 25 e 46.

Redmon, J. et al. You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788. Citado na página 25.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. Disponível em: <http://arxiv.org/abs/1804.02767>http://arxiv.org/abs/1804.02767. Citado na página 45.

REIS, D. H. D. et al. Mobile robot navigation using an object recognition software with rgbd images and the yolo algorithm. *Applied Artificial Intelligence*, Taylor Francis, v. 33, n. 14, p. 1290–1305, 2019. Citado na página 30.

ROBLA-GÓMEZ, S. et al. Working together: A review on safe human-robot collaboration in industrial environments. *Ieee Access*, IEEE, v. 5, p. 26754–26773, 2017. Citado na página 27.

ROBOTICS, O. *Understanding nodes*. 2024. Disponível em: <https://docs.ros.org/en-/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes-.html>https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html. Citado 2 vezes nas páginas ix e 23.

SAIDI, S. M. et al. Real-time fuzzy-pid for mobile robot control and vision-based obstacle avoidance. *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, v. 13, n. 1, p. 1–32, 2022. Citado na página 26.

SCALERA, L. et al. Enhancing fluency and productivity in human-robot collaboration through online scaling of dynamic safety zones. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 121, n. 9, p. 6783–6798, 2022. Citado na página 14.

SHARMA, S. et al. Unified closed form inverse kinematics for the kuka youbot. In: . [S.l.: s.n.], 2012. p. 1–6. ISBN 978-3-8007-3418-4. Citado 3 vezes nas páginas ix, 19 e 20.

SILVA, Y. M. da et al. Computer vision based path following for autonomous unmanned aerial systems in unburied pipeline onshore inspection. *Drones*, MDPI, v. 6, n. 12, p. 410, 2022. Citado 2 vezes nas páginas 15 e 25.

SOBREIRA, H. et al. Robust mobile robot localization based on a security laser: an industry case study. *Industrial Robot: An International Journal*, Emerald Group Publishing Limited, v. 43, n. 6, p. 596–606, 2016. Disponível em: <https://doi.org/10.1108/IR-01-2016-0026>https://doi.org/10.1108/IR-01-2016-0026. Citado na página 28.

SOBREIRA, H. et al. Autonomous robot navigation for automotive assembly task: An industry use-case. In: SILVA, M. F. et al. (Ed.). *Robot 2019: Fourth Iberian Robotics Conference*. Cham: Springer International Publishing, 2020. p. 645–656. ISBN 978-3-030-35990-4. Citado na página 28.

SOEBHAKTI, H. et al. The real-time object detection system on mobile soccer robot using yolo v3. In: IEEE. *2019 2nd International Conference on Applied Engineering (ICAE)*. [S.l.], 2019. p. 1–6. Citado na página 29.

SPONG, M. W.; VIDYASAGAR, M. *Robot dynamics and control*. [S.l.]: John Wiley & Sons, 2008. Citado na página 13.

STORE youbot. *YouBot Detailed Specifications*. 2015. Disponível em: <http://www-.youbot-store.com/wiki/index.php/YouBot_Detailed_Specifications>http://www.youbot-store.com/wiki/index.php/YouBot_Detailed_Specifications. Citado 2 vezes nas páginas ix e 21.

SUN, X. et al. Collision avoidance control for unmanned surface vehicle with colregs compliance. *Ocean Engineering*, Elsevier, v. 267, p. 113263, 2023. Citado na página 29.

SZREK, J.; JAKUBIAK, J.; ZIMROZ, R. A mobile robot-based system for automatic inspection of belt conveyors in mining industry. *Energies*, v. 15, n. 1, 2022. ISSN 1996-1073. Disponível em: <https://www.mdpi.com/1996-1073/15/1/327>https://www.mdpi.com/1996-1073/15/1/327. Citado na página 28.

TALHA, M. et al. Fuzzy logic-based robust and autonomous safe landing for uav quadcopter. *Arabian Journal for Science and Engineering*, Springer, v. 44, n. 3, p. 2627–2639, 2019. Citado na página 29.

TIUSANEN, R.; MALM, T.; RONKAINEN, A. An overview of current safety requirements for autonomous machines–review of standards. *Open Engineering*, De Gruyter, v. 10, n. 1, p. 665–673, 2020. Citado na página 28.

TZUTALIN. *labelImg*. 2015. https://github.com/tzutalin/labelImg. GitHub repository. Citado na página 46.

VILLANI, V. et al. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, Elsevier, v. 55, p. 248–266, 2018. Citado 2 vezes nas páginas 27 e 28.

WANG, L. et al. Symbiotic human-robot collaborative assembly. *CIRP annals*, Elsevier, v. 68, n. 2, p. 701–726, 2019. Citado na página 27.

WANG, X. et al. Robot manipulator self-identification for surrounding obstacle detection. *Multimedia Tools and Applications*, Springer, v. 76, p. 6495–6520, 2017. Citado 2 vezes nas páginas 15 e 24.

WANG, Z. et al. Apf-cpp: An artificial potential field based multi-robot online coverage path planning approach. *IEEE Robotics and Automation Letters*, IEEE, 2024. Citado na página 28.

WEST, A. et al. Real-time avoidance of ionising radiation using layered costmaps for mobile robots. *Frontiers in Robotics and AI*, v. 9, p. 862067, 03 2022. Citado 2 vezes nas páginas x e 42.

WONG, C.-C. et al. Moving object prediction and grasping system of robot manipulator. *IEEE Access*, IEEE, v. 10, p. 20159–20172, 2022. Citado 2 vezes nas páginas 16 e 25.

XU, F. et al. Research on yolov3 model compression strategy for uav deployment. *Cognitive Robotics*, Elsevier, v. 4, p. 8–18, 2024. Citado na página 25.

YU, Y.-H.; ZHANG, Y.-T. Collision avoidance and path planning for industrial manipulator using slice-based heuristic fast marching tree. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 75, p. 102289, 2022. Citado na página 13.

ZANCHETTIN, A. M. et al. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 13, n. 2, p. 882–893, 2015. Citado na página 28.

ZHANG, G. et al. Intelligent vector field histogram based collision avoidance method for auv. *Ocean Engineering*, Elsevier, v. 264, p. 112525, 2022. Citado na página 29.

ZHANG, G. et al. Target detection and position measurement based on machine vision for agv. In: *2023 8th International Conference on Automation, Control and Robotics Engineering (CACRE)*. [S.l.: s.n.], 2023. p. 343–347. Citado na página 28.

ZHANG, H. et al. A fast detection and grasping method for mobile manipulator based on improved faster r-cnn. *Industrial Robot: the international journal of robotics research and application*, Emerald Publishing Limited, v. 47, n. 2, p. 167–175, 2020. Citado na página 25.

ZHANG, P. et al. Ensuring safety in human-robot coexisting environment based on two-level protection. *Industrial Robot: An International Journal*, Emerald Group Publishing Limited, v. 43, n. 3, p. 264–273, 2016. Citado na página 28.

ZHAO, X.; CHIDAMBARESWARAN, T. Autonomous mobile robots in manufacturing operations. In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. [S.l.: s.n.], 2023. p. 1–7. Citado na página 28.

ZHOU, Z. et al. A safe reinforcement learning approach for autonomous navigation of mobile robots in dynamic environments. *CAAI Transactions on Intelligence Technology*, 2023. Disponível em: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2-.12269>https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12269. Citado na página 29.