POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Development of a Human-Centered Framework for Safe Object Handling with Mobile Manipulators

Supervisors

Prof. Marina INDRI

Dr. Pangcheng David CEN CHENG

Candidate

Angela RIPI

December 2024

Summary

In recent years, the use of robots has become increasingly common in people's daily lives. The ability to delegate simple tasks, such as floor cleaning or lawn mowing, simplifies and enhances the quality of life. The fundamental idea behind this thesis is based on this premise. The project focuses on developing an assistant robot that can coexist in a domestic environment alongside people.
For the development of this thesis, a mobile manipulator was used, programmed to: (i) receive user requests, (ii) locate itself in a known closed environment and navigate safely in presence of people, (iii) identify the object requested by the user, grasp it and place it in a predefined position, and (iv) deliver the object to the user safely.
The idea is to create a robot that can assist people with mobility difficulties, including disabled and elderly individuals. This robot will be utilized in indoor domestic settings. However, its capabilities can also be adapted for use in industrial settings, where it could serve as an assistant for workers. The development of the robot is designed to be versatile, thus facilitating its use in different contexts.
In order to achieve such versatility and effectiveness in a variety of settings, recent advances in artificial intelligence and machine learning have been employed. The robot processes data in order to determine its actions in real time. In this case, neural networks from Ultralytics were employed, specifically YOLOv8 for object detection and YOLOv8 Segmentation for image segmentation. The networks were used for the purposes of tracking individuals during the navigation process and for the computation of the grasping pose of the requested object. Additionally, MediaPipe's Gesture Recognition was used to develop a safe object handover maneuver. The software architecture of the system was developed using ROS.
An Android app was developed to facilitate human-robot communication. The robot lacks an integrated microphone, so the user-friendly nature of a smartphone or tablet was employed in the development of software capable of communicating directly with the robot.
Each developed algorithm and strategy was tested on the Locobot wx250 in simulation and laboratory.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial Intelligence

ML
Machine Learning

ST
Speech-to-Text

E2E
End-To-End

TTS
Text-To-Speech

APP
Mobile Application

HRI
Human Robot Interaction

RL
Reinforcement Learning

ROS
Robot Operating System

API
Application Programming Interface

URDF

Unified Robot Description Format

SRDF

Semantic Robot Description Format

GUI

Graphical User Interface

SLAM

Simultaneous Localization and Mapping

Chapter 1

Introduction

The integration of robots into daily lives has seen a remarkable increase, gaining widespread acceptance in modern society. Robots have been initially employed for simple but challenging tasks, such as cleaning floors, and thanks to the technological advances, they are able to offer support in the care and medical fields as well. These advanced devices, equipped with perceptual, cognitive, and physical assistance capabilities, have become invaluable allies, especially for individuals with motor and sensory challenges, such as the elderly and disabled, significantly enhancing their quality of life [1].
Beyond domestic and assistive roles, social robotics is expanding into other areas of daily life. As evidenced in [2], small robots, such as Little Sophia and Buddy PRO, are being deployed in educational settings for children and for personal assistance. Concurrently, larger robots are being utilised primarily in healthcare, where they perform tasks such as drug dispensing and act as receptionists in hospitals.
Moreover, the integration of robots extends to the workplace. The advancement of technologies and the integration of artificial intelligence have enabled significant advances in both productivity and worker satisfaction. A study reported in [3] shows that, in an industrial setting characterized by repetitive and physically demanding tasks, such as assembly, workers prefer to collaborate with robots. These robots, known as collaborative robots or cobots, are designed to work safely alongside humans, improving efficiency and reducing physical workload.
The main objective of these technologies is the interaction with non-specialized individuals. This implies that, as a crucial part of the entire project, a user-friendly and intuitive interface must be developed. Such an interface must allow easy interaction between human and robot, ensuring that requests are formulated in a simple and understandable way. Moreover, it is essential that the robotic assistant can correctly interpret these requests, thus guaranteeing an efficient and satisfying user experience.

The term "service robot" is used to describe a device that is capable of providing tangible and practical assistance to the user. An effective system must be capable of integrating a variety of tools and technologies. This integration should commence with the user's initial request, proceed through the navigation process, and conclude with the handover of the object concerned.

This thesis presents a software design and implementation for an assistance robot based on the Locobot wx250, a mobile manipulator equipped with a single robotic arm and a parallel gripper.
It examines the specific challenges of integrating the different functionalities required for robotic assistance and proposes an approach that takes into account the capabilities of the Locobot wx250. It offers an analysis of the implemented solutions and their practical applications.

1.1 Thesis Goal

As previously stated, the principal aim of this thesis is to develop a software architecture that is integrated seamlessly with the available hardware platform, specifically the Locobot wx250. The overarching objective is to enable the Locobot to function as an effective assistance robot, capable of autonomously performing tasks that directly benefit the user. This necessitates not only the assurance of seamless interaction between the hardware and software, but also the evaluation of the robot's performance in authentic operational settings.
The core of this work revolves around the successful implementation of four key functionalities that are fundamental to the robot's assistance capabilities: (i) autonomous social navigation, which allows the robot to move efficiently and safely in shared environments alongside humans; (ii) object detection and grasping, enabling the robot to identify, locate, and pick up various objects using its robotic arm; (iii) object delivery to the user, commonly referred to as the handover process, which involves transferring the object to the user; and (iv) the management of a user interface designed to facilitate easy and intuitive communication between the robot and the user, ensuring that the system remains accessible to individuals with minimal technical expertise.

1.2 Environment description

The robotic system has been developed with the primary objective of providing assistance to elderly or disabled people in domestic environments. The technology enables such individuals to receive the objects they require without having to

move from their position, thereby enhancing their quality of life and promoting personal autonomy. This type of assistance is particularly beneficial in contexts where individuals may experience motor difficulties or other physical limitations, enabling them to perform everyday tasks with minimal intervention.
Despite its initial orientation towards domestic applications, the robot's technical and functional characteristics make it suitable for use in a variety of different contexts, including laboratories, warehouses and industrial environments. Indeed, these environments share similar characteristics, such as the presence of static obstacles that the robot is able to effectively handle. In a domestic setting, the robot must navigate around obstacles such as furniture and furnishings. In contrast, in a laboratory or warehouse, it may have to avoid shelves, machinery, or heavy equipment. The system's adaptability renders it particularly versatile and suitable for operation in diverse environments, while ensuring the maintenance of high standards of safety and precision.
A significant feature of the system is its capacity to function in shared environments with human operators. This means that the robot is equipped with technologies for secure navigation and collision avoidance, thereby guaranteeing the protection of users and human operators, even in confined or crowded environments. The robot is capable of discerning the presence of humans and modifying its conduct to prevent accidents, rendering it suitable not only for static environments but also for those characterised by a high level of interaction between robots and humans.

1.3 Thesis structure

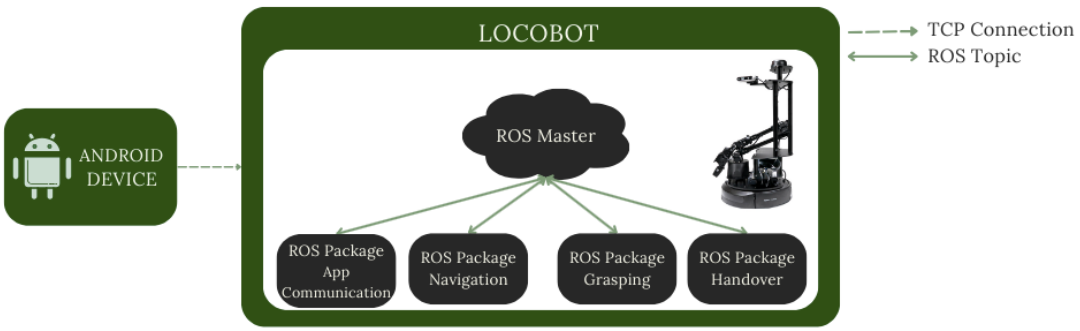


Figure 1.1: Components of the development of an Assistance Robot for Object Retrieval Task

The aim of this thesis is to provide a comprehensive description of each fundamental element of the project.

Firstly, an overview of the existing literature on the various techniques through which the objective can be achieved is provided. Thereafter, an in-depth analysis of each individual component of the system is carried out.
Following this, a chapter is devoted to illustrate the project's fundamental framework, namely the ROS (Robot Operating System). Subsequently, the hardware employed is described in detail, with particular attention devoted to the sensors utilized in the development of the tasks. The subsequent Chapters present a detailed examination of each task, illustrating the methodologies employed and the challenges encountered during the development process. This part is divided into four Chapters, corresponding to the four packages illustrated in Figure 1.1. The project commences with an examination of the techniques and ML models applied to achieve the final goal of social navigation. The next stage is grasping, with a particular focus on the machine learning models used to recognise the object and describe in detail the different positions assumed by the individual components of the robot, in accordance with the different situations that can occur. The handover Chapter addresses the initial method by which humans can communicate with the robot, emphasizing the importance of safety during these interactions. It concludes with the development of an Android application, which facilitates interconnection of the system's constituent modules and serves as an accessible interface through which users can communicate with the robot.
The final Chapter will conclude the work and discuss potential future developments for the project.

Chapter 2

State Of the Art

Technological advancements in robotics and artificial intelligence have resulted in the significant progression observed in this field. The concept of social robotics represents an integration of a number of different activities. The combination of these elements results in the creation of highly useful and functional tools, capable of enhancing the interaction between humans and machines in a variety of contexts, including domestic, professional, and social settings.

A review of the literature reveals that these activities are extensively analysed and developed, with significant improvements in both functionality and safety being highlighted. All fields of research have been oriented towards the advancement of robotic technology, with the objective of enhancing the reliability and performance of these activities. The following paragraphs will provide an overview of the state of the art, exploring the potential and results achieved so far in this field.

2.1 Human-Robot Communication

As previously stated, effective communication is a pivotal element in the domain of collaborative robotics, as it enables seamless collaboration between the user and the robot. Indeed, a variety of communication methods can be employed, contingent on the particular requirements and attributes of the available hardware. Among the most prevalent methods are voice commands, telecontrol, and hand gesture recognition. Each method offers unique advantages and challenges, which must be carefully considered when designing a system for optimal user experience. Voice commands, for example, allow for hands-free operation and can be highly intuitive, but they require sophisticated natural language processing algorithms to accurately interpret user intentions. Telecontrol, involving the use of a mobile or tablet application to remotely control the robot, provides users with flexibility and

convenience, enabling them to operate the robot from a distance. This method is particularly useful for tasks that require monitoring or intervention from various locations, allowing users to issue commands and make adjustments on the go. However, it also depends on the reliability of the wireless connection and the intuitive design of the application interface to ensure effective operation. Hand gesture recognition combines the benefits of natural interaction with the precision of visual tracking, yet it depends heavily on the environment's lighting conditions and the robustness of gesture detection algorithms. The selection of an appropriate communication technique by developers can facilitate the enhancement of robot usability and functionality, thereby ensuring that the collaboration between human and machine is as effective and natural as possible.

2.1.1 Voice Command

Speech-To-Text One straightforward yet efficacious approach is to utilise voice interface. This mode of communication emulates the nuances of interpersonal communication, rendering this approach particularly intuitive and immediate, especially for non-specialised users. The primary challenge in this context is to identify the human voice from the surrounding environmental noise. It is imperative to comprehend the user's requests in order to facilitate an appropriate response.
In the field of robotics, the simplest approach is the utilisation of neural networks for the conversion of speech to text.
Fabian Falck et al. [4] address this problem by testing several audio recognition methods, including Amazon Web Services (AWS) Alexa speech-to-text conversion service, Google Cloud technology, and the open source CMUSphinx toolkit. AWS was tested using Amazon's Echo dot hardware, encountering frequent connection problems and the need for special configurations that slowed down the process. Google's solution, while effective, was very sensitive to ambient noise and required a microphone for accurate transcription. In contrast, CMUSphinx did not require an Internet connection, ran on low-resource platforms, and improved accuracy by masking ambient sounds through its built-in library.
The use of the JSpeech Grammar (JSGF) format with CMUSphinx reduced the space of possible audio inputs, providing an expected grammar that improved the speed and accuracy of speech recognition.
The speech-to-text (ST) method stands out for its great versatility. Thanks to research advances in this field, ST is capable of understanding queries in different languages. For example, Nivedita et al. [5], demonstrate how it is possible to transform a speech signal in one language into written text in another language. They present a review of models developed in End-to-End (E2E) mode, highlighting the details of their development and the performance achieved.

Text-to-Speech Conversely, audio feedback from the social robot can also facilitate communication and comprehension for the user, thereby enhancing the user experience. This is achieved by reversing the aforementioned technology and utilising a process known as text-to-speech.
Additionally, there are numerous models that enable text to be converted into speech signals, which can be readily transmitted through speakers integrated into the device.
Fabian Falck et al. [4] propose the use of eSpeak as a TTS synthesizer [6]. With eSpeak, any string of English text can be converted to audio. The speech is clear and can be used at high speeds, but it is not as natural or fluid as larger synthesizers based on human voice recordings, resulting in a mechanized and formulated voice. There are also alternatives provided by Google or Amazon, but they have some limitations. In addition to not being open source, they only work with a Wi-Fi connection.
In addition to these, there are advanced models such as Tacotron [7], an end-to-end TTS model that synthesizes speech directly from characters. Tacotron achieved a MOS[1] score of 3.82.
WaveGlow [8], used with Tacotron 2, converts spectrograms to audio signals, boasting simplicity of training and speed of inference with a MOS of 3.961. Transformer TTS [9] uses multi-head attention mechanisms to improve training efficiency and audio quality, achieving a MOS score of 4.39 and superior training speed.
FastSpeech2 [10] generates audio directly from text, overcoming the speed and robustness issues of autoregressive models, with MOS scores comparable to those of autoregressive models, demonstrating the effectiveness of variation information such as pitch and energy.

It should be noted, however, that this communication mode is not without limitations. Firstly, it necessitates the utilisation of specific hardware, including a microphone for the capture of incoming audio and a speaker for the playback of the robot's voice output. Furthermore, in the context of mobile robots, issues may emerge if the user and the device are situated at a distance that hinders the accurate acquisition of information, thereby compromising the fulfilment of requests.

[1]Mean Opinion Score (MOS) is a measure used to evaluate the perceived quality of synthesized speech or audio recordings. It is obtained by asking a group of listeners to rate audio quality on a predefined scale, typically from 1 to 5, where 1 indicates very poor quality and 5 indicates excellent quality. The MOS score represents the average of the listeners' individual ratings.

2.1.2 Telecontrol

Another method for developing efficient human-robot communication is the utilisation of mobile devices, which have become a significant aspect of contemporary society. Inspired by the authors M.S. Ghute et al. [11], the idea of creating an Android application that connects to robots allows them to perform numerous tasks that ensure proper control of the device. The only requirement is that both are connected to the same WiFi network. This approach exploits the pervasive presence and sophisticated capabilities of contemporary mobile devices, repurposing them as versatile controllers capable of executing a diverse array of robotic operations. The application facilitates the transmission of commands to the robot and also leverages the diverse sensor capabilities of the mobile device to ensure comprehensive control. For instance, mobile devices are equipped with accelerometers, gyroscopes, and GPS, which can be employed to augment the robot's navigation and orientation capabilities. Furthermore, it is possible to view images captured by the robot's camera via the mobile phone and employ a voice command system, thus avoiding hardware and spatial limitations through the utilisation of the inherent capabilities of a conventional mobile device. Such integration serves to streamline the interaction between the user and the robot, while simultaneously offering a more intuitive and accessible interface. This enables users with varying degrees of technical expertise to operate the robot with greater efficacy.
Another significant example is the one presented by Eduard Clotet et al. [12], which describes a robot capable of providing home care, fully controlled through a mobile application. This APP first offers a joystick interface, which enables the user to control the movement of the robot's moving parts. This facilitates precise manipulation and direct control over the robot's actions. Additionally, the app incorporates a video conferencing service, enabling remote communication with caregivers or family members. This functionality is of particular value in a home care setting, where the capacity to monitor, interact with, and assist remotely can markedly enhance the quality of care provided.

By leveraging the capabilities of mobile devices, these systems are not only enhancing the versatility and accessibility of robots but also expanding their potential applications across various fields, from healthcare and home assistance to industrial automation and beyond. This synergy between mobile technology and robotics represents a forward-thinking approach to human-robot interaction, harnessing the power of everyday devices to make robotic technology more accessible and user-friendly.

2.1.3 Hand-Gesture Recognition

In interpersonal communication, speech is often complemented by gestures to emphasize and reinforce the message being conveyed. For deaf individuals, gestures serve as the primary mode of communication, enabling them to interact effectively. This principle also applies to human-robot communication, where gestures can significantly enhance interaction.
Human-robot interaction (HRI) benefits from the ability to detect and interpret human gestures, as it allows robots to understand requests and perform tasks accurately, leading to more effective and precise collaboration. According to Qi, J., Ma, L., Cui, Z. et al. [13], vision-based hand gesture recognition is crucial for natural HRI, because it facilitates intuitive and creative communication. The study explains that gesture recognition can be achieved either through sensors, such as specialized gloves, or through vision-based systems that use camera-captured images, with RGB-D cameras offering superior performance.
Gesture recognition technology has a wide range of applications, including healthcare, safe driving, sign language recognition, virtual reality, and device control. In healthcare, it enables hands-free control of equipment in sterile environments, such as operating theatres, thereby reducing the risk of cross-contamination. In the automotive industry, gesture recognition systems, like those in the BMW 7 Series, allow drivers to control certain functions without diverting their attention from the road. For the hearing impaired, sign language recognition technology translates gestures into text and speech, enhancing communication. In virtual reality, gesture recognition improves user interaction and immersion, benefiting gaming and IoT applications. For device control, it simplifies the use of smart home technologies and robots, providing a more intuitive alternative to traditional input methods.
As robots become increasingly prevalent in daily life, gesture recognition stands out for its naturalness, expressiveness, and versatility, making it an essential component of human-robot communication.
Hand gesture recognition is generally performed in three main steps: data acquisition, gesture detection and segmentation, and gesture recognition. Currently, there are many algorithms and software in research that can integrate this feature into various projects. The authors Köpüklü et al. [14] address the challenges of real-time dynamic hand gesture recognition from video streams - such as accurately detecting the start and end of gestures, ensuring that gestures are recognised only once, and optimising the use of memory and computational resources - by proposing a hierarchical structure. This structure includes a lightweight CNN detector and a deep CNN classifier, with performance evaluated using Levenshtein distance to comprehensively measure errors. The proposed architecture, tested on the EgoGesture and NVIDIA Dynamic Hand Gesture datasets, achieves state-of-the-art accuracy in both offline and real-time applications.

Bigalke et al. [15] proposes a dual-stream model for hand gesture recognition that directly processes 3D point cloud sequences to capture both fine-grained local posture variations and global hand movements. By decoupling the learning of local and global features and fusing them with an LSTM for temporal modelling, the model exploits different 3D learning architectures in each stream, achieving leading performance on the Shrec'17 and DHG datasets while reducing computational cost.
The proposed Temporal Decoupling Graph Convolutional Network (TD-GCN)[16] solves the problem of hand gesture recognition using a skeleton-based approach, specifically addressing the limitation of previous methods that use the same adjacency matrix for skeletons from different frames. TD-GCN uses different adjacency matrices for skeletons from different frames, significantly improving the modelling capability and achieving state-of-the-art results on gesture datasets such as SHREC'17 Track and DHG-14/28.
Overall, the integration of gesture recognition into HRI and various technological applications represents a significant advancement in making communication more intuitive and effective.

2.2 Navigation

Navigation is a pivotal element in the domain of collaborative robotics, as it enables robots to interact with their surroundings and users in a more dynamic and versatile manner. However, it should be noted that social robotics does not necessarily require the robot to be mobile. In many cases, a stationary device, such as a robotic arm, may be sufficient to interact safely and effectively with users. The ability to move is an additional feature that depends on the specifications of the available hardware. This feature becomes particularly useful in contexts of assistive robotics, where the robot must often move around to assist users in different situations.
The capacity for mobility not only renders the robot more dynamic and versatile, but also enables it to undertake additional tasks that enhance the overall experience for the robot user. Firstly, it permits dynamic interaction with the surrounding environment, thereby enhancing the accessibility and flexibility of task performance. For example, the robot is capable of relocating to disparate locations in order to provide assistance in whichever area is required. Moreover, the ability to be mobile allows for more effective management and response in emergency situations. This enables robots to rapidly reach the requisite location and provide immediate assistance, such as the delivery of a defibrillator.
It is important to distinguish between the concepts of mobility, which includes navigation, and social navigation in robotics. A device capable of navigating an environment does not necessarily have the ability to operate effectively in the presence of dynamic obstacles, such as humans, nor does it inherently have the

ability to interact safely with them. Basic navigation involves a robot determining its position within a pre-mapped environment and avoiding fixed, known obstacles. In contrast, social navigation involves a robot's ability to coexist in an environment shared with humans who act as dynamic obstacles. In such scenarios, the robot must be able to move in a way that ensures the safety of both itself and its environment.

2.2.1 Social Navigation

Francis et Al. [17] define a socially navigating robot as a device that acts and interacts with humans or other robots, achieving its navigation goals while modifying its behavior so that the experience of agents around the robot is not degraded or is even enhanced.
The work that the authors present outlines the *Principles of Social Navigation*, which serve as criteria for assessing the quality of a robot's social behavior. These principles include: *(1) safety, (2) comfort, (3) legibility, (4) politeness, (5) social competency, (6) understanding other agents, (7) proactivity, and (8) appropriate contextual response*, as illustrated in Figure 2.1.

In order for a robot to be considered safe, it must be equipped with a set of

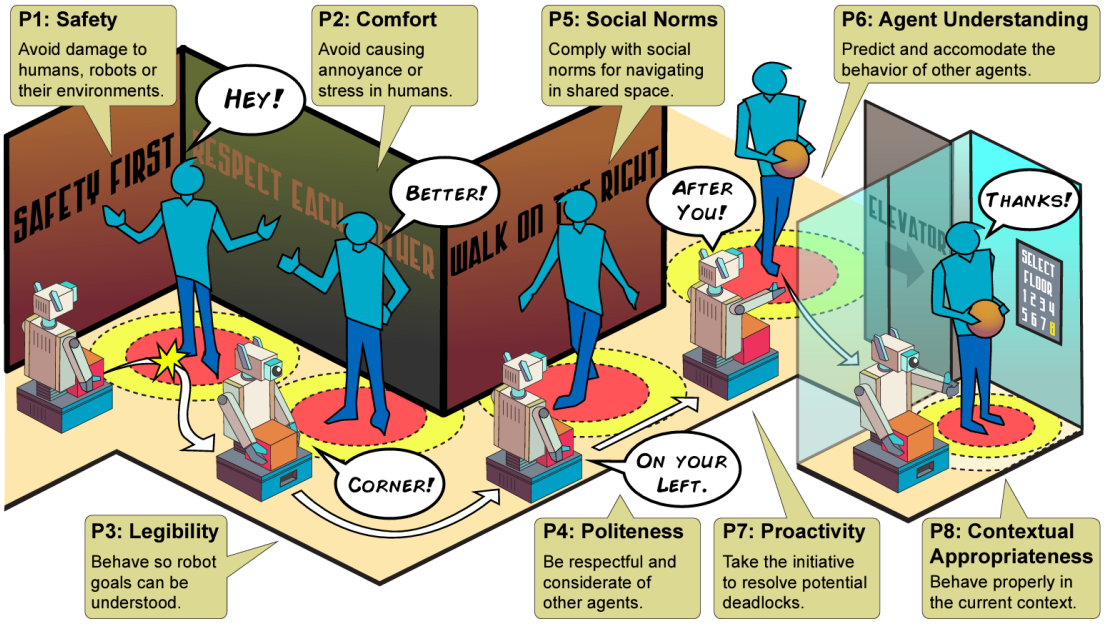


Figure 2.1: Principles of Social Navigation.
Adapted by Francis et Al. [17]

sensors and algorithms that allow it to control its movements in a way that best

meets the specified requirements.
For basic navigation, which simply consists of moving the robot within an environment, sensors such as LIDAR or depth cameras (RGB-D) and SLAM algorithms are used. These tools allow the robot to gather information about its environment, create a map, locate itself within it and determine the shortest path to its final destination using minimum path algorithms. However, while these elements are necessary, they are not sufficient for social navigation. In this context, it is essential to use more advanced ML algorithms that analyse the data collected by the sensors to assess the situation and decide on the most appropriate behaviour for the robot.
The development of sophisticated algorithms for the social navigation of robots frequently employs the use of Reinforcement Learning (RL). This field of study, which is situated at the nexus of machine learning and optimal control, focuses on how an intelligent agent can execute actions within a dynamic environment in order to achieve the greatest overall reward. In a recent contribution to the field, Shuijing Liu et al. [18] proposed a novel approach for robotic navigation in crowded environments, introducing the Structure-Recurrent Decentralised Neural Network (DS-RNN). In terms of performance, the DS-RNN has been demonstrated to markedly outperform existing navigation methods. Despite representing a significant advance, the DS-RNN method is not without limitations, many of which were addressed in subsequent work with the development of *CrowdNet++* [19]. This novel approach seeks to enhance robot navigation in dense, interactive crowds, considering the intentions and interactions between all agents, whether human or robotic. In contrast to preceding methodologies, it incorporates a recurrent graphical neural network with attention mechanisms to capture the intricate dynamics of crowds in both spatial and temporal domains. This technique has exhibited superior navigation performance and augmented social awareness in simulated environments relative to traditional methods such as ORCA, Social Force (SF), and DS-RNN.
However, it is noteworthy that the use of Reinforcement Learning (RL) is not the sole method for addressing this issue. In a recent publication, Phani Teja Singamaneni et al. [20] presented the Cooperative Human-Aware Navigation (CoHAN) planner, a navigation system designed with the specific intention of being aware of human presence and adaptable to different human-robot interaction scenarios in both indoor and outdoor environments. The system, which is based on the ROS navigation stack, has been modified to include costmaps for human safety and visibility, and employs the Human-Aware Timed Elastic Band (HATEB) local planner, as described by Singamaneni et al. (2020) [21], to effectively handle dynamic interactions with humans.

2.3 Grasping

The ability to grasp objects is a fundamental skill in the field of assistive and collaborative robotics. One of the primary functions demanded to these types of robots is the ability to accurately grasp and manipulate objects. Grasping is not merely the act of lifting and picking up an object; it is a more expansive and intricate concept. The grasping ability is fundamental for enabling a wide range of more advanced tasks, such as opening doors, assembling or disassembling objects, performing intricate manipulations in industrial settings, and offering assistance in daily activities or healthcare environments. This functionality allows for increased user autonomy, with a potential application being in the assistance of an elderly person with mobility difficulties. In these scenarios, the robot can approach a walker or directly transport essential objects, such as medication, greatly improving the user's quality of life. Moreover, in certain circumstances, grasping can also facilitate social and emotional interaction. A robot that is capable of performing a gentle grasping action could participate in games or other forms of playful activity, or even provide physical comfort in emotional situations, such as taking a user's hand.
Considerable research has been directed towards the development of technologies that ensure accurate and repeatable grasping, such as force and tactile sensors. The aforementioned sensors permit the robot to discern and grasp fragile objects, and to regulate the force exerted during interaction with the user, thereby ensuring the safe and gentle utilisation of the robot.
The process of determining the optimal grip for a given object is a complex one. In practice, when an individual decides to grasp an object, already possess knowledge of its shape, position and texture. However, this information is not readily accessible to a robot. It is therefore necessary to develop strategies for calculating the possible positions that the robot gripper can assume and then selecting the most optimal one.
Rhys Newbury et al. [22] suggest that the quality of each gripping hypothesis can be evaluated based on several criteria. One important factor is the stability of the grip, which is influenced by the geometry of the object being grasped. Additionally, the geometry and kinematics of the gripper itself play a crucial role in determining how well a grip can be executed. Furthermore, the suitability of the grip for a specific manipulation task is also considered, as different tasks may require different types of grips to be effective. Finally, the grasp is defined by the position and orientation of the coordinate frame attached to the gripper or the robot wrist.
In their survey, Kleeberger et al. [23] posit that vision-based robotic grasping approaches can be classified according to a variety of criteria. In general, these approaches can be classified into two main categories: analytical or geometric methods and data-based methods. Analytical methods entail the analysis of the

shape of the target object with a view to identify a suitable gripping position. Conversely, data-driven methods employ machine learning algorithms, powered by data proliferation, to detect a proper gripper position.
Moreover, as postulated by Kleeberger et al. [23], grasping approaches can be further categorised according to their model-based or model-free nature. Model-based approaches make use of object-specific knowledge (e.g., a CAD model or a previously scanned model) in order to solve the task at hand. In contrast, model-free approaches propose candidates for taking or aiming, with the objective of generalising to novel objects. Additional classification criteria include the type of object to be manipulated (e.g., rigid, articulated, or deformable), the degree of familiarity with the object (e.g., known, familiar, or unfamiliar), the type of learning (e.g., supervised or reinforced), and the training environment (e.g., simulation, real-world, or both). Furthermore, some approaches operate in an open-loop mode, without feedback, while others utilise continuous feedback based on visual features, a technique known as "visual servoing" [23].
This extensive range of criteria and approaches reflects the complexity and diversity of strategies available to address the challenges of robotic grasping, emphasising the importance of selecting an appropriate method based on the specific operational context.
The problem of grasping an object in real life is more complex than it may initially appear. In addition to identifying the correct position for the gripper, the robot must be able to distinguish the desired object from a group of other objects that are not relevant to the task. To address this challenge, it is common to include object detection algorithms in the process. These algorithms assist in isolating the target object, thereby ensuring that the gripper pose calculation is correctly applied to the appropriate object.
In their study, Sundermeyer et al. [24] address the challenge of generating 6-DoF grasps from any viewpoint in cluttered environments with unknown objects. The approach employs raw depth images, which may be optionally enhanced with object masks, to propose potential 6-DoF grasp positions. To this end, the researchers developed an asymmetric U-shaped neural network based on PointNet++, which processes random point clouds as input and predicts grasp points for a specific subset of these points. The network outputs the key variables necessary for successful grasping, including the likelihood of a successful contact, the approach direction for the gripper, the baseline direction for the grasp, and the appropriate grasp width (Figure 2.2).

In a similar way, Pelleschi et al. [25] put forth a data-driven grasp planning algorithm that generates grasp candidates from partial and incomplete point clouds. In this instance, the point cloud is decomposed into a specified number of Minimum Volume Bounding Boxes (MVBBs). A Decision Tree Regressor (DTR) trained

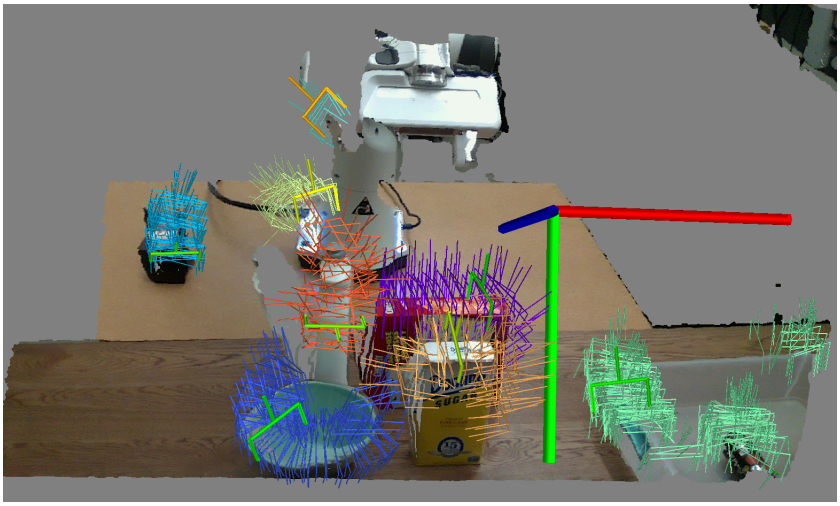


Figure 2.2: An example of the output obtained from the Sundermeyer et al.[24] proposed solution.

on data collected from a skilled human performing grasps with the same gripper on sample boxes is then used to generate a set of candidate grasps based on this bounding box decomposition. These candidate grasp poses are subsequently ranked according to a specific metric.
Bergamini et al. [26] present a framework based on Deep Convolutional Neural Networks (DCNN) to predict both single and multiple grasp poses for multiple objects all at once, using a single RGB image as input. The work introduces new architectures and a novel loss function to enable faster training and unprecedented real-time grasping performance.
The majority of studies on 6-Degrees-of-Freedom (6-DoF) robotic grasping employ point clouds as the input data, yet these are frequently influenced by noise from depth sensors. To address this issue and enhance scene representation, a number of alternative approaches have been investigated. For instance, Haoxiang Ma and Di Huang [27] put forth a methodology to enhance the identification of robotic grips, accounting for objects of disparate dimensions. During the training phase, the researchers employ a data augmentation technique, designated as "Noisy-clean Mix", which entails the combination of clean and noisy point clouds with the objective of enhancing the model's robustness. Subsequently, candidate grip points are selected and encoded using a transformer-based encoder. In the process of inference, the Object Balanced Sampling method examines samples at varying scales and utilises a 3D segmentation network to enhance grip accuracy, effectively adapting to objects of disparate sizes and complex scenes.
In summary, robotic grasping is a complex task requiring a blend of analytical, data-driven, and vision-based techniques. Advances in machine learning and neural

networks are improving the precision and adaptability of these systems, opening new possibilities for their use in both industrial and assistive applications.

2.4 Handover

One of the most critical and fundamental tasks in human-robot collaboration is object transfer, which plays a key role in enabling robots to integrate seamlessly into human daily life. This capability is essential to facilitate the effective participation of robots in various tasks in both everyday and manufacturing environments, thereby enhancing their ability to serve and assist humans in a wide range of scenarios.

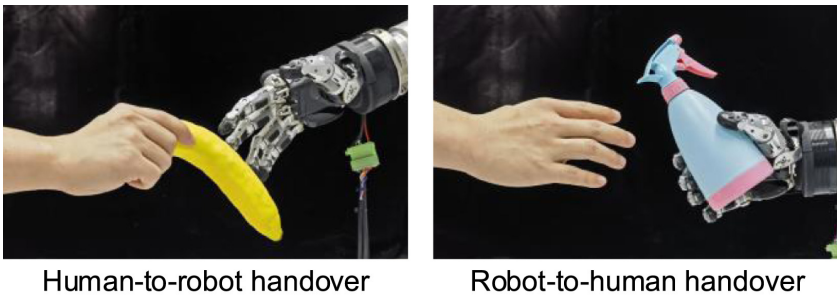


Figure 2.3: Human-Robot object handover.
Adapted by Duan et al. [28]

Duan et al.[28] propose a taxonomy of human-robot object handover methods categorises research into three main dimensions: the role of the robot, the end-effector, and the abilities of the robot (Figure 2.4). Robots can act either as a receiver (human-to-robot handover) or as a giver (robot-to-human handover), each of which presents unique challenges. For example, human-to-robot handover presents difficulties due to uncertainties in human behaviour and real-time response requirements, while robot-to-human handover requires attention to human comfort and timing. Technological advances have evolved robot end-effectors from simple parallel jaw grippers to multi-finger hands, improving their ability to interact with different objects. The handover process itself is divided into pre-handover and physical handover phases, with research focusing on improving motion planning and grasping strategies, as these determine the success of the handover. In addition, while most studies explore single-object scenarios, recent work aims to improve robot generalisation to multiple objects, which is crucial for real-world interactions.

Lehotsky et al. [29] present methods and systems for optimising the handover

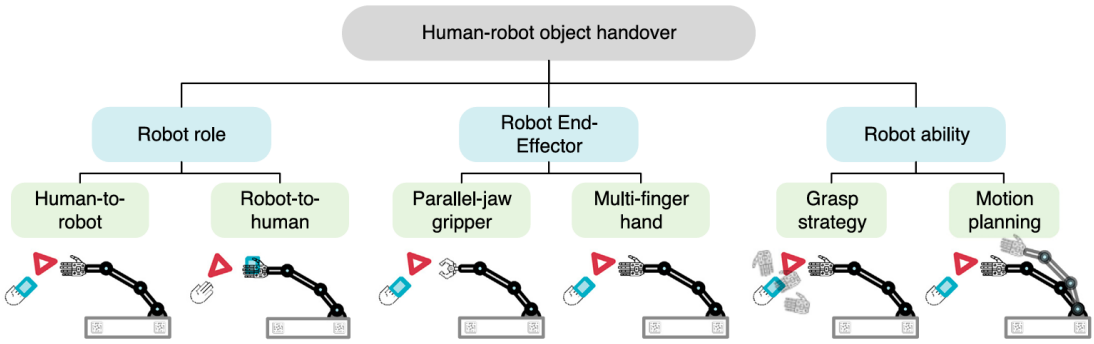


Figure 2.4: Taxonomy of human-robot object handover.
Adapted by Duan et al. [28]

process from robots to humans using vision-based affordance information. The authors evaluate two different approaches for computing object handover orientations: (i) the observation-based method and (ii) the affordance-aware rule-based method. The observation-based method collects and annotates datasets from human-to-human handovers, where object orientations are calculated based on real-world data. In contrast, the rule-based method simplifies computation by relying on object affordances, such as grasping or utility functions, to determine optimal handover orientations. Tested on twelve objects, the rule-based method offers a more straightforward approach by applying predefined rules, such as orienting handles towards the receiver or positioning objects to avoid spillage.
One of the most prominent approaches in the literature is that proposed by Christensen et al. [30], which introduces a novel framework to improve robotic handover tasks through enhanced object affordance segmentation using synthetic data. The method proposed by Christensen et al. [30] involves the segmentation of the object image and the classification of the areas that correspond to the parts that should be grasped for safe and effective use, distinguishing them from other potentially hazardous areas. This approach ensures that the robot grasps the object in a way that presents the appropriate part to the user, thereby optimising both safety and efficiency in the handover process (Figure 2.5).
F. Iori et al. [31] put forth an innovative approach that enables continuous modulation of the interaction, acting directly on the generation of the trajectory rather than utilising symbolic signals. This type of control makes the system more resistant to perturbations or interruptions that could interfere with the human user, while simultaneously enhancing safety. The ability to adapt behaviour is crucial when considering changes in the environment or the behaviour of other individuals. In light of the aforementioned considerations, the authors put forth an online method for trajectory generation. Some methods employ explicit cues, such as

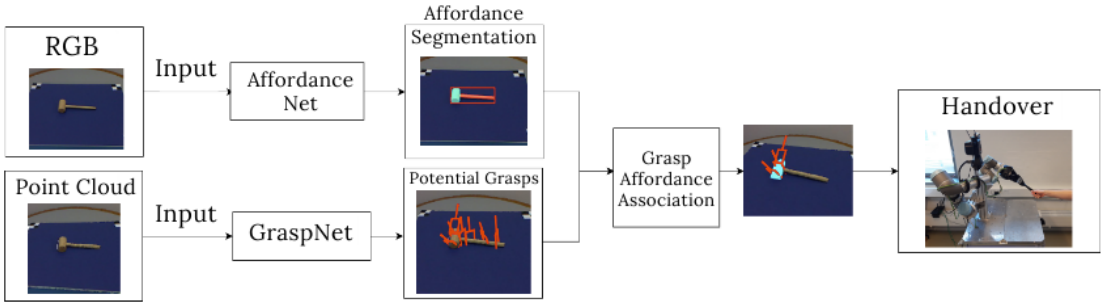


Figure 2.5: Two-stage task-oriented grasping method proposed by Christensen et al. [30].

speech or gestures, while others rely on implicit cues, such as human movement, to predict intentions. In this context, the authors emphasise the use of Dynamic Movement Primitives (DMP), which facilitate real-time trajectory generation and enable robots to adapt to environmental changes without the necessity for complete recalculation of movements, thus enhancing responsiveness and efficiency in the completion of tasks.

The field of human-robot collaboration is undergoing a period of rapid evolution, encompassing not only the software domain, where the development of increasingly sophisticated and high-performance neural networks is advancing rapidly, but also the hardware domain. The incorporation of sophisticated sensors facilitates the delivery of increasingly precise feedback, providing detailed information that enhances the precision of robotic movements and actions. These advances in both software and hardware are facilitating the development of interactions between robots and humans that are more natural and efficient. They are also bringing robotic capabilities closer to human ones in terms of readiness and adaptability.

Chapter 3

Robot Operating System (ROS) Overview

The robotics industry often faces common problems such as a lack of defined standards, limited code reusability and the need to continuously rewrite drivers, communication protocols and basic algorithms. These obstacles slow down progress, as the integration of a new robot often requires starting almost from scratch. ROS was created precisely to address these problems, providing an open source platform that simplifies robot software development by adapting to a wide range of systems. Despite its nomenclature, ROS is not an operating system per se, but rather a set of libraries and tools that facilitate the creation of robotic applications in a modular and reusable manner [32]. Initially developed at the Stanford Artificial Intelligence Laboratory in 2007, and then further developed by Willow Garage, ROS has been managed by the Open Source Robotics Foundation (OSRF) since 2013 [33].

ROS is characterised by two main components: on the one hand, it provides "operating system"-style services, such as hardware abstraction, low-level device control, common functionality implementation, package management and inter-process communication. On the other hand, it provides community-developed packages containing modules for advanced tasks, such as SLAM (Simultaneous Localisation and Mapping), planning, perception, computer vision and manipulation.

The structure of a ROS project is mainly based on a peer-to-peer architecture, consisting of a large number of autonomous programs, called nodes, which directly exchange messages to communicate with each other. These nodes can be developed using different programming languages. In fact, ROS supports multiple languages, including C++, Python, Java, MATLAB and many others, making it a versatile framework for robotics development [33].

ROS also encourages the development of standalone packages, facilitating the

integration of disparate modules that can communicate with one another via message exchange. The distributed architecture of ROS enables the execution and development of code on multiple machines simultaneously, thereby facilitating the creation of scalable applications suitable for complex systems. This approach addresses numerous traditional challenges in robotics, promoting code reusability and markedly reducing development time. Currently, ROS is compatible with Unix-based platforms, including Ubuntu and Mac OS X, while a version compatible with Microsoft Windows is under development. The various ROS versions are distributed in packages, analogous to a Linux distribution, which provide ready-to-use tools and libraries [34].

3.1 ROS Architecture

In order to gain an understanding of ROS and its constituent parts, it is necessary to begin by introducing the building blocks of this framework. ROS is structured in a well-defined manner.
A ROS project is composed of a series of individual units, or packages, each of which performs a specific task. These units communicate with each other in order to generate a workflow and facilitate communication. However, these components can be integrated into any other workflow, thereby facilitating both development and research in robotics.

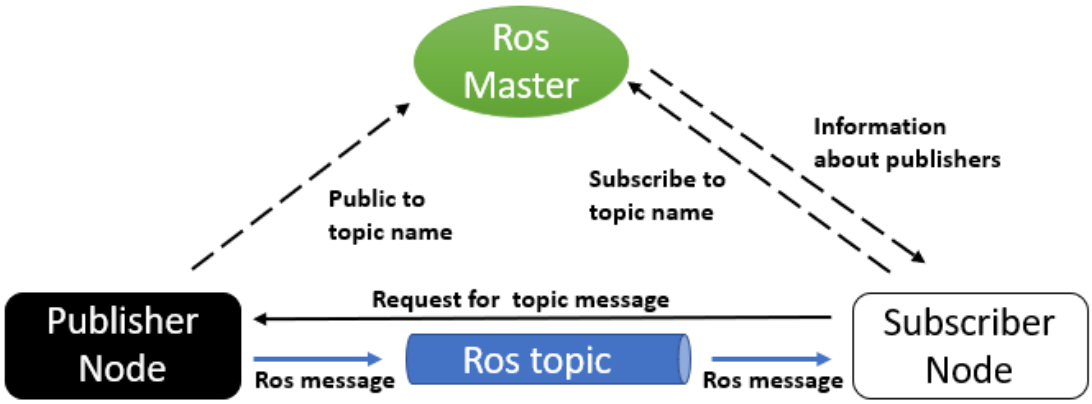


Figure 3.1: Introduction of main element of ROS [35]

Nodes

The fundamental component of any ROS project is the **Node**.
In ROS, each node represents an executable unit of the project, that is to say, a

portion of code that handles a specific task or sensor.
For the implementation of these Nodes, ROS provides support for two principal programming languages: C++ and Python, thus enabling developers to select the approach that best aligns with their requirements [36].
Nodes are configured to communicate with each another via the transmission of messages, utilising communication channels designated as "topics" (which will be discussed in greater detail below). This enables the establishment of a communication network that is truly interconnected.
In accordance with the function they perform, two principal categories of nodes can be identified:

- **Publisher:** whose function is to generate and publish messages (also called Talker) [37];
- **Subscriber:** a node that listens in on a topic and is activated when it receives messages sent by the publisher [37].

Messages and Topics

As previously stated, the nodes implementing disparate functions within a ROS project are able to communicate with each another via communication channels designated as **"topics"**. Topics can be conceptualised as "tunnels" through which publishers broadcast messages, while subscribers await their receipt [32].
However, topics are not constrained to a single pair of nodes (publishers and subscribers), but can connect an unlimited number of them, thus enabling multiple publishers and subscribers to interact simultaneously without restriction.
The communication protocol that is usually used to connect the different nodes in the network is the TCPROS, which uses standard TCP/IP sockets [38].

The role of each topic is to convey messages, which represent the data shared between nodes. The messages may be defined using the predefined data types provided by ROS or alternatively, custom data types may be created. The utilisation of custom messages enables the data structure to be tailored to the particular requirements of the project. Furthermore, a message may comprise a combination of disparate data types, thereby facilitating the organisation and transmission of information in a structured and comprehensive manner.
The definition of customised messages is accomplished through the utilisation of text files with the *.msg* extension. These files delineate the structure of the message by listing the various components thereof. It is typical for messages to comprise a particular data type called "Header", which contains a timestamp and information regarding the coordinate frame. This is a common feature of ROS, facilitating the synchronisation and management of spatial coordinates [39].

Master

The final element in Figure 3.1 that has yet to be delineated is the **ROS Master**. This component functions as a central server, containing all the information required for the management of the communication network that constitutes the entire **ROS Computation Graph**.

The ROS Master is responsible for maintaining a database of all topic, message, and service details, as well as managing the registration of nodes within the network [40]. The primary function of the ROS Master is to identify and locate nodes, thereby facilitating peer-to-peer communication between them during execution, in a manner analogous to that of a DNS server [38].

Services

In scenarios where request/response communication is necessary, the deployment of publisher and subscriber nodes is not the optimal solution. In such circumstances, the utilisation of **services** is the best solution.
A service is a ROS element that comprises two distinct messages: one for the request and one for the response. The messages are defined within a text file with the extension *.srv*, where the format of the request message is first specified, followed by the response message, separated by a set of dashes.
Once the messages have been defined, the node responsible for managing the requested service is implemented. Each service is associated with a specific **name**, which will be used by the client to invoke it. The client sends the necessary parameters and awaits the response from the node providing the service [38].

Parameter Service

The **parameter server** represents a fundamental component of the ROS Master. It functions as a database, wherein all data and configuration parameters required by the nodes during runtime are stored, with each item associated to a unique key. It may be compared to a dictionary, which is readily accessible via an application programming interface (API). The parameter server is implemented using the XMLRPC protocol, which allows it to be accessed via XMLRPC-compatible libraries.
As in the case of topics and nodes, a hierarchical naming convention is employed in the parameter server. This approach is of purely practical value, as the use of a hierarchical structure prevents the various parameters from conflicting with each other, thereby avoiding errors and interpretation problems [41].

ROS Package

The aforementioned elements are organised within **packages**. In ROS, a package is defined as a simple directory containing all the components necessary to perform a specific task. In this way, a ROS project is structured in a manner analogous to that of a file system, comprising a collection of packages.
The intention behind this organisational structure is to facilitate **reuse**. The lightweight nature of packages, coupled with their narrow focus on a single task, facilitates their reuse in other projects. This enhances the efficiency of development and streamlines workflows [42].
In addition to the files and directories created by the programmer, each ROS package includes a number of specific core files by default:

- **CMakeLists.txt**: This is a basic text file that contains all the requisite information for the CMake build system, which is employed for the compilation of software packages. This file contains instructions on the location and method of code compilation and installation. It is imperative that each CMakeLists.txt file adheres to the prescribed standard; failure to do so may result in compilation errors [43].

- **package.xml**: This file, formally designated a "package manifest", is an XML document that delineates the attributes of the package, including its designation, the identity of its creator, and the dependencies associated with its construction. Similarly, the package.xml file adheres to a defined format and contains the requisite information for the construction and organisation of the package [44].

3.2 Visualization and Simulation Tools

ROS provides a number of valuable tools for visualisation and simulation, which are indispensable for the advancement and assessment of robotic applications. The term "visualisation" is used to describe the capacity to observe the behaviour of a system in real time during its execution, thereby facilitating the process of debugging. In contrast, the term "simulation" denotes the generation of virtual environments in which actions can be evaluated prior to their implementation in the actual robot.

Such tools are especially beneficial when developing tasks that involve dangerous objects or require collaboration with humans. The utilisation of a simulation environment for testing purposes allows for the generation of reliable results, thereby increasing the probability of success in a real-world setting. Furthermore, it enables the maintenance of high safety standards and the avoidance of any unexpected

movements that could potentially be dangerous.

In contrast, visualisation tools permit the intuitive planning of robot movements via a graphical interface, obviating the necessity for manual operation. This greatly facilitates comprehension and planning of operations.

From the perspective of debugging, the capacity to visualize system behavior in real time, in conjunction with error logs, facilitates the identification and correction of bugs, thereby promoting the development of more reliable software.

3.2.1 RViz

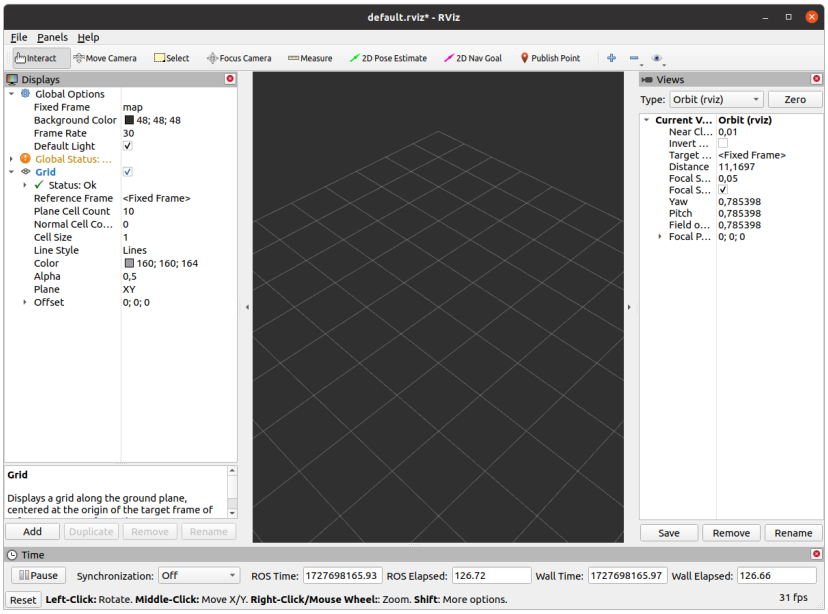


Figure 3.2: RViz interface

ROS Visualization (RViz) is a three-dimensional visualization tool that is provided directly by ROS. The graphical user interface facilitates straightforward and intuitive interaction, thereby facilitating comprehension of the visualised data. The utilisation of RViz is of significant benefit in the field of robot development and programming, as it enables the comprehensive planning and visualisation of the robot's components, including sensors, in a meticulous manner.
The real-time observation of the robot's perception, along with the analysis of its movement planning during execution, is made possible by subscribing to the relevant communication topic. This process facilitates the monitoring, analysis, and optimization of the robot's behaviour during the developmental and testing

phases [45].
Figure 3.2 illustrates the default configuration that is displayed upon launching RViz. A variety of distinctive working elements are present:

- **Display area**, situated in the centre of the screen and presented against a black background, allows the user to visualise the data in a clear and concise manner. This section enables the user to observe a range of elements, including the LaserScan output, the point cloud, the robot model and the various reference systems. Furthermore, it enables direct interaction with these elements in real time;

- **Display Panel** (vertical column on the left): This section lists all elements that can be added to the display. The user is able to modify the configuration according to their requirements, adding or removing elements of interest;

- **Toolbar**: Provides rapid access to functions such as setting or estimating the initial pose and setting a navigation goal directly on the displayed map;

- **Time Panel**, situated at the base of the interface, is a particularly valuable tool during simulations, as it enables the duration of the test to be monitored and compared with the actual elapsed time [45].

As previously mentioned, the data collected by the robot's sensors can be visualised via RViz. The tool subscribes to the various communication topics and provides a real-time visualisation of the collected data. Supported sensors include the camera, laser scanning, inertial measurement units (IMUs) and global positioning systems (GPSs).
In the case of the camera, RViz enables the robot's perspective to be displayed directly on the main screen, with the output shown in the lower section of the Display Panel. In the case of the LaserScan, the collected data are used to generate a point cloud, which is then displayed in the central display area. This graphical representation provides a comprehensive illustration of the surrounding environment and the position of the robot within the scene. The data obtained from the IMU (inertial measurement unit) are presented in the form of acceleration, angular velocity and, in some instances, magnetic field. Additionally, the Global Positioning System (GPS) provides information regarding the robot's latitude, longitude, and altitude. In certain contexts, data regarding the robot's speed is also displayed [45].

URDF and SRDF

In order to visualise the entire robot and interact with it, RViz utilises two fundamental files, in the Unified Robot Description Format (URDF) and in the Semantic

Robot Description Format (SRDF), respectively. These XML files are complementary and provide fundamental information on the structure and functionality of the robot.
The **URDF** file provides a comprehensive description of the physical structure of the robot, delineating the manner in which the links and joints are connected. This file is of great importance for the definition of the robot's geometry, inertial properties and collision patterns [46].
The **SRDF** file, instead, incorporates semantic data, including the designation of groups of joints and links (e.g., a robotic arm), potential poses, zones susceptible to collision, and other more complex kinematic attributes. The SRDF is especially beneficial for defining intricate movements and preventing collisions, thereby ensuring that the simulated movements are as close to reality as possible [47].

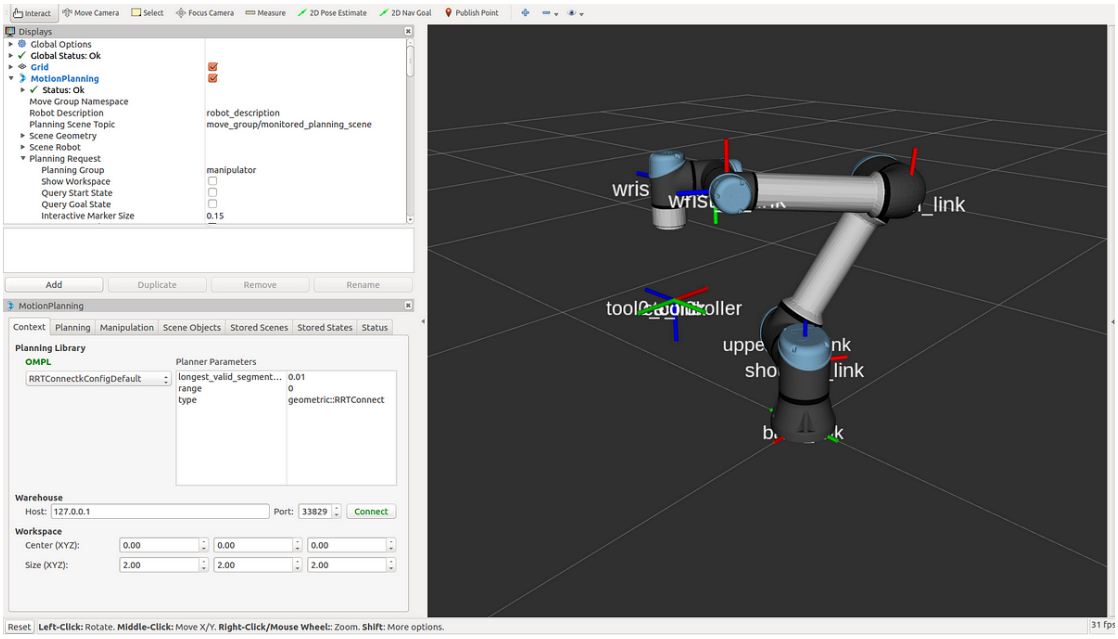


Figure 3.3: RViz Visualization of the UR5 Robot Model [48]

Tf

The RViz tool enables the visualisation of the reference frames associated to all the joints of a robot, including also the ones relative to the surrounding environment and the map. The aforementioned reference frames are organised in a tree structure, in accordance with the ROS convention. ROS's **tf** package enables the monitoring of reference frames, facilitating the observation of their temporal evolution and the management of the required transformations for transitions between different

frames.
Additionally, the tf package provides a graphical integration in RViz, which enables the visualization of the interconnections between the various frames and the manner in which they evolve in accordance with the robot's movements. This functionality is particularly useful for understanding the kinematics of the robot and for monitoring frame changes during motion execution [49]. Figure 3.3 illustrates an example of the frame visualisation in RViz.

3.2.2 Rqt

Another visualisation tool in ROS is **Rqt**, which enables the creation of graphical user interface (GUI) for the visualisation and analysis of the activities and interactions in the ROS network, thereby facilitating the debugging process.
Rqt offers three principal visualisation tools:

- **rqt_graph**: This tool provides a graphical representation of the interaction between active nodes in the network. It visualises both the nodes and the topics that connect them, thereby creating a clear diagram of the communication within the ROS network. In this way, the network topology and current interactions can be rapidly understood;
- **rqt_console**: This GUI displays all messages sent to `rosout`, which is the mechanism responsible for log reporting. The utility of this tool lies in its capacity to facilitate the monitoring of log messages and the diagnosis of underlying issues;
- **rqt_plot**: offers a graphical representation in the form of a plot. The tool allows the plotting of trends in values published on specific topics, such as the speed of a mobile robot. It is particularly suited to the visualisation of the evolution of critical parameters in real time [50].

3.2.3 Gazebo

One of the most widely used simulation platforms in ROS is **Gazebo** [51], which plays a pivotal role in the testing of algorithms and designs, particularly in scenarios where their failure could result in hazardous circumstances or damage to the robot. The utilisation of simulation permits the testing of a design in a virtual environment, thereby facilitating the refinement of the latter and the avoidance of potential risks, prior to its implementation in the actual operational context. This guarantees a high level of safety and a greater probability of success in real conditions.

Gazebo is a highly versatile platform that offers a plethora of tools for the creation

Figure 3.4: Example of Gazebo simulation environment [51]

of both indoor and outdoor environments. In addition to utilising pre-configured models, developers are able to insert customised meshes or create robots and objects de novo, defining them via SDF (Simulation Description Format) files.

A further advantage of Gazebo is its capacity to incorporate dynamic elements into scenes by accurately reflecting the physical laws that govern the interactions between objects. This further enhances the precision of the simulation, enabling the replication of authentic physical behaviour with remarkable fidelity.

Gazebo is not solely employed for the simulation of motion dynamics; it is also utilised for the simulation of sensors, thereby facilitating the generation of virtual measurements that are beneficial for the purposes of testing and calibration. The aforementioned capabilities render Gazebo an effective tool for robot design, offering a safe and flexible environment for design and validation.

3.3 Mapping and Navigation Stack

In the field of mobile robotics, one of the fundamental elements is the ability of the robot to orientate itself and navigate autonomously within its surrounding environment. In order to achieve this, it is necessary to equip the device with the requisite tools to enable it to **map** the environment, locate itself within it and move around safely and efficiently. ROS provides a set of advanced tools to address these challenges. The robot is able to map its surroundings and construct a detailed representation of its environment, which is then used for localisation purposes. Subsequently, **ROS's navigation stack** exploits this information to

plan and guide the robot towards its goals, ensuring safe and precise movement. This integrated system allows the robot to operate autonomously, avoiding obstacles and optimising its path within the environment.

3.3.1 Mapping

The ability of the robot to locate itself within an environment is contingent upon the knowledge it acquires about its surroundings. In order for the robot to gather this information, it is essential that it is initially guided through the work area so that it can generate a detailed map of the space in which it will operate.
ROS offers a number of specific software and tools that facilitate the creation of precise and well-defined maps, thereby enhancing the robot's autonomous navigation.
The following sections present some of the main tools used for this purpose.

Gmapping

Figure 3.5: Mapping of the environment using gmapping package [52]

Gmapping [53] is a highly popular package within ROS that implements a SLAM (Simultaneous Localization and Mapping) algorithm to construct a 2D map of the environment in the form of an occupancy grid. This results in the generation of a representation of the environment in which each grid cell can be designated as either 'occupied', 'vacant' or 'unknown', according to the data collected by the sensors.

SLAM indicates the ability of a robot to generate a map of an unknown environment while simultaneously determining its own position within it. There are

numerous approaches to solve this problem, which can be divided into two main categories: those based on extended Kalman filters (EKF-SLAM) and those that use particle filters. The choice of the approach to be adopted depends on how the map is represented. The EKF-SLAM approach is based on the use of distinctive landmarks in the environment for mapping purposes, whereas the particle filter method to represent the environment uses occupancy grids.

RTAB-Map

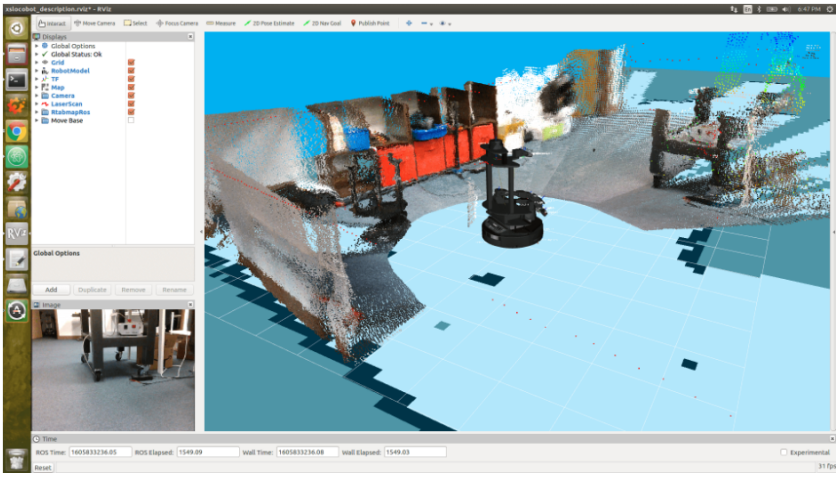


Figure 3.6: RTAB-Map visualization in RViz [54]

Real-Time Appearance-Based Mapping (RTAB-Map) [55] is an advanced mapping and localization tool that operates using **multisensor SLAM**. In contrast to other tools that rely primarily on LIDAR data, RTAB-Map also utilises information from supplementary sensors, including stereo cameras and RGB-D cameras (depth and colour), thereby enhancing the quality and precision of the generated map.

The core technique employed by RTAB-Map is referred to as "Loop Closure Detection". This approach enables the robot to discern when it has previously visited a given area, even if its position is different. To achieve this, RTAB-Map employs a technique known as "bag-of-words", which entails a comparison of the images captured by the sensors with a pre-existing dictionary of images that have been previously detected. In practice, the system compares each new image captured with those already stored, thereby establishing links (or constraints) between different portions of the environment.
Once these constraints have been created, the map is optimised to correct any errors and remove unnecessary links, thus resulting in an accurate and definitive map of the environment.

Octomap

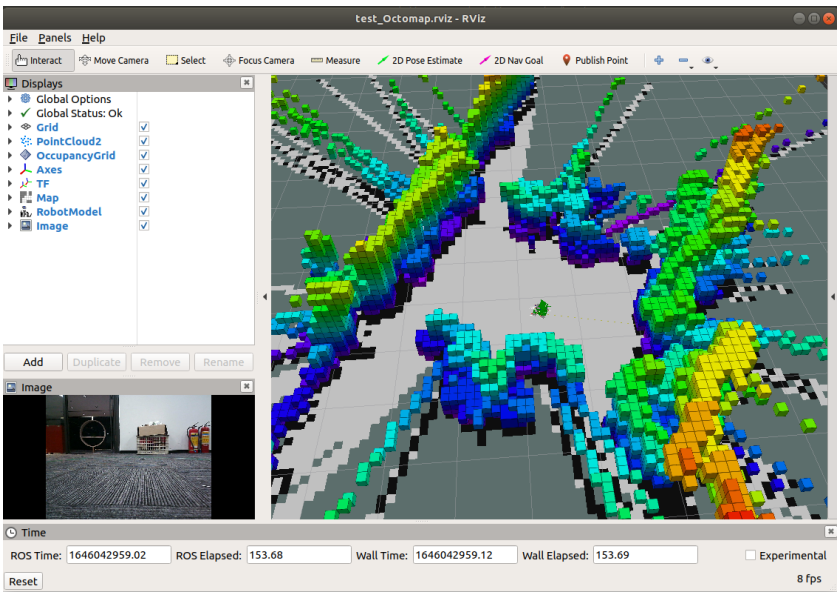


Figure 3.7: Octomap visualization in RViz

OctoMap [56] is a library that generates a dynamic 3D map of an unknown environment. Implemented in C++, OctoMap is particularly suitable for robotic applications due to its efficiency and flexibility. The library is based on the concept of the **octree**, a tree structure that divides space into three-dimensional volumes called voxels, representing the environment in a hierarchical way. This structure allows the construction of a 3D model of the environment that is dynamically updated when new spaces are discovered or when there are mobile agents in the scene.

Modelling and updating of the map are done probabilistically, taking into account uncertainties from sensors and measurements. This approach allows an accurate representation of the environment even in presence of noisy or inaccurate data.

3.3.2 Navigation Stack

Although robotic navigation may appear to be a relatively straightforward process in theory, in practice it result to be considerably more complex. The ability of a robot to move safely within an environment depends on its capacity to avoid obstacles and reach desired destinations. These processes require the robot to process a large amount of information in real time. ROS's Navigation Stack [57] plays a

crucial role in autonomous navigation, enabling the robot to acquire the information it needs to locate itself and plan routes. This system combines data from sensors, odometry and predefined maps, allowing the robot to move autonomously, avoiding obstacles and adapting to any changes in the environment. In order to operate

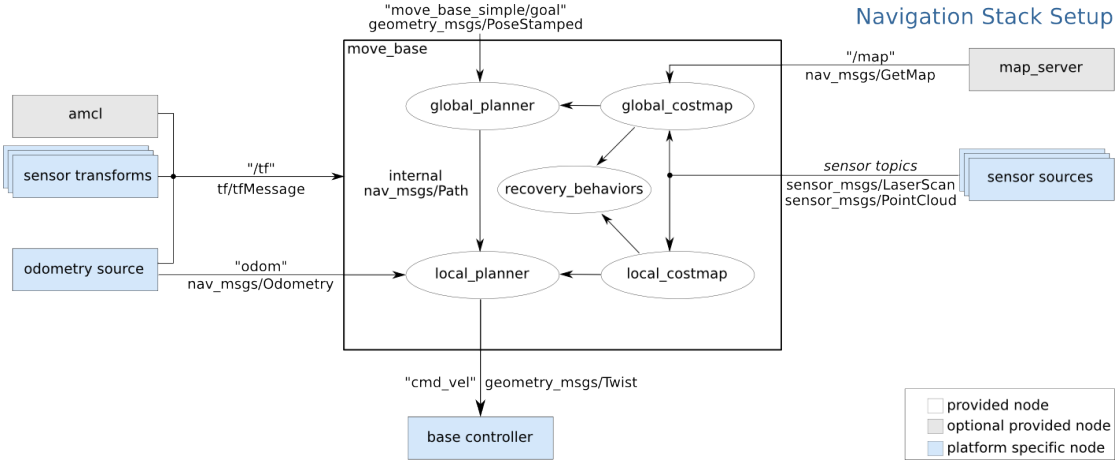


Figure 3.8: Navigation Stack setup [57]

effectively, the navigation stack adopts a clearly delineated structure, as depicted in Figure 3.8. The diagram displays a central rectangle representing the navigation tool, with boxes inside describing the various components that constitute it. The aforementioned components process the inputs provided to the system, including odometry, sensor data, the default map of the environment and tf transformations. The tf transformations are pivotal in ensuring the consistency and applicability of the information within a single reference system. They facilitate the alignment of data from disparate sources, thus establishing a unified representation applicable to the base of the robot.

The four vertices of the central rectangle represent the primary components that influence the robot's movement and path calculation. These include the local and global planners, as well as the related local and global costmaps. These elements function collectively to compute optimal navigation, determining secure routes and handling obstacles at both local and global levels.

3.3.3 Global and Local Costmap

As the name suggests, the function of costmaps is to assign a numerical value, or "cost", to each cell on the map in order to quantify the level of "danger" present at that location. The costs are employed in the formulation of a safe trajectory, thereby enabling the robot to circumvent obstacles. The cost assigned to each cell

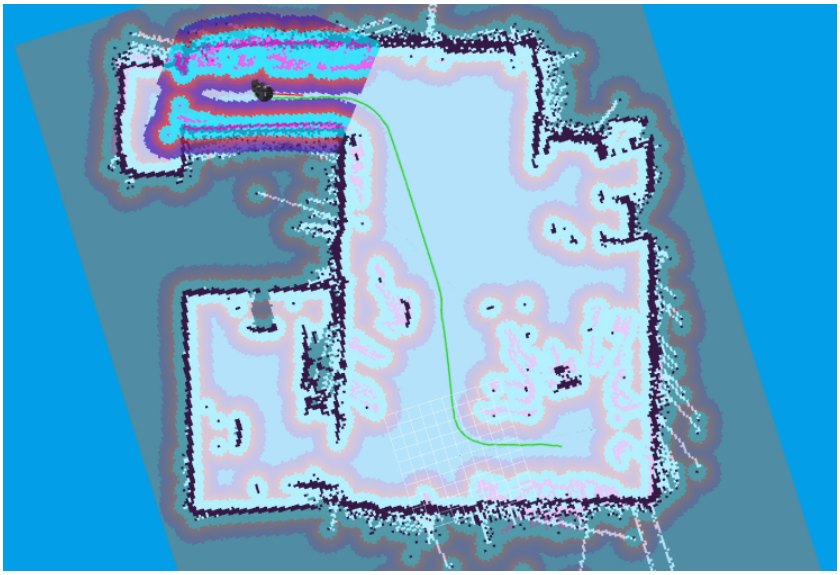


Figure 3.9: Representation of navigation stack elements as depicted in RViz [54]

is reflective of the distance to the obstacle. Consequently, the closer the cell is to the obstacle, the higher the cost, allowing to distinguish between safe and unsafe areas.

In the Navigation Stack, two principal costmaps are employed: the global costmap and the local costmap [57]. The global costmap offers an overview of the environment based on the complete map provided by the map server, and is useful for long-term planning. As illustrated in Figure 3.9, the global cost map is represented by the colour variations shown in the background, which cover the entire map area. In contrast, the local costmap is more dynamic, focusing on the area immediately surrounding the robot. Furthermore, it considers the presence of moving obstacles and alterations in the surrounding environment, enabling the robot to respond promptly to unforeseen circumstances. This is also illustrated in Figure 3.9 through the use of brighter colours in the area close to the robot.

Some information is common to both costmaps, including the subscription to sensors required for obstacle detection, the desired distance between the robot and obstacles, and the size of the robot's base. This information is useful for correctly calculating the space needed for navigation. The principal distinctions between the two costmaps pertain to the frequency of updates, the reference frames employed for spatial transformations, and the manner in which the map's data are updated.

3.3.4 Global and Local Planner

In order to guarantee the safe and precise reaching of a target point, it is necessary to integrate all the data obtained about the environment and calculate an optimal route. In this phase, planners are employed, which can be divided into two categories: global planners and local planners.

The global planner is tasked with calculating the overall route from the robot's current position to the desired destination. This calculation is performed using the complete map of the environment, with the objective of ensuring that the chosen route is as direct and safe as possible, taking into account static obstacles that have already been mapped. In contrast, the local planner assumes a more dynamic role, continuously updating the route in the proximity of the robot in response to unanticipated or mobile obstacles that may not have been identified in the static map.

This combination of global and local planning is crucial for maintaining uninterrupted movement and ensuring the robot's capacity to adapt swiftly to environmental changes. Figure 3.9 illustrates the two planners in operation. The global planner is shown in green, indicating the primary route to the destination, while the local planner, depicted in red, highlights changes in real time around the robot.

3.3.5 move_base

The ROS package that implements the aforementioned description is designated as **move_base** [58]. The package permits interaction with the configuration files, thereby enabling the robot to reach the target within a tolerance level specified by the user. In the event of unexpected obstacles or location errors, the move_base package provides a **recovery mode**. This mode comprises a series of progressive steps, contingent on the severity of the situation, up to the termination of navigation if necessary.

In the event that the robot encounters an obstacle that precludes further movement, it will initially attempt to remove the obstacle from outside a region specified by the user. It will then begin to rotate on itself in order to update the surrounding information. In the event of a further unsuccessful attempt, the robot progresses to a more radical phase, whereby all obstacles situated outside a rectangular area in which it is able to rotate are removed. In the event of a failure to free the robot from its current position, the system will terminate the mission and notify the user of the failure.

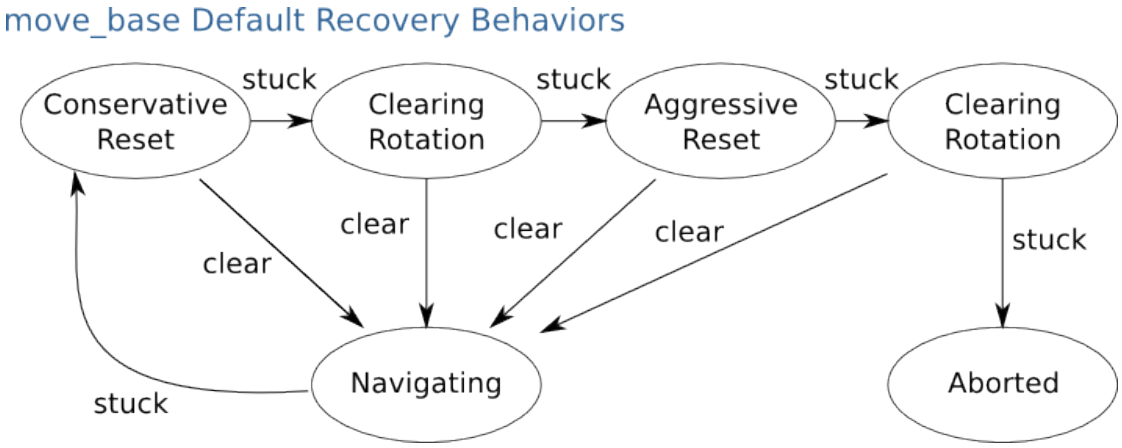


Figure 3.10: Recovery plan of move_base [58]

3.4 MoveIt

In the context of manipulator robotics, it is of pivotal importance to accurately plan the movements of the robotic arm in order to fully leverage its capabilities. To this end, ROS offers **MoveIt** [59], a platform comprising a range of plugins and interfaces that facilitate interaction with the robot and enable the planning of optimal movements for the intended tasks.

3.4.1 Motion Planning

Motion Planning is a plugin interface that can be visualised via RViz, thereby enabling direct interaction with the manipulator robot. The interface utilises data delineated in URDF and SRDF files to comprehend the configuration of the robot and its potential movements. The interface enables the planning of movements that will guide the robot to a predefined position or, alternatively, allows the user to manually adjust the joints in order to achieve a customised pose.

Once the desired position or pose has been defined, the plugin generates a movement plan that considers the kinematic and dynamic limitations of the robot, as well as any obstacles present in the environment. This planning process is employed to determine whether the desired position can be reached in a safe and collision-free manner. Once the plan has been validated, it is possible to simulate the movement in order to verify its feasibility prior to executing it in the real world.
Figure 3.11 illustrates the RViz interface for motion planning. The initial state of the robot is represented in green, while the goal position is represented in orange. In the event of a collision, this will be highlighted in red.

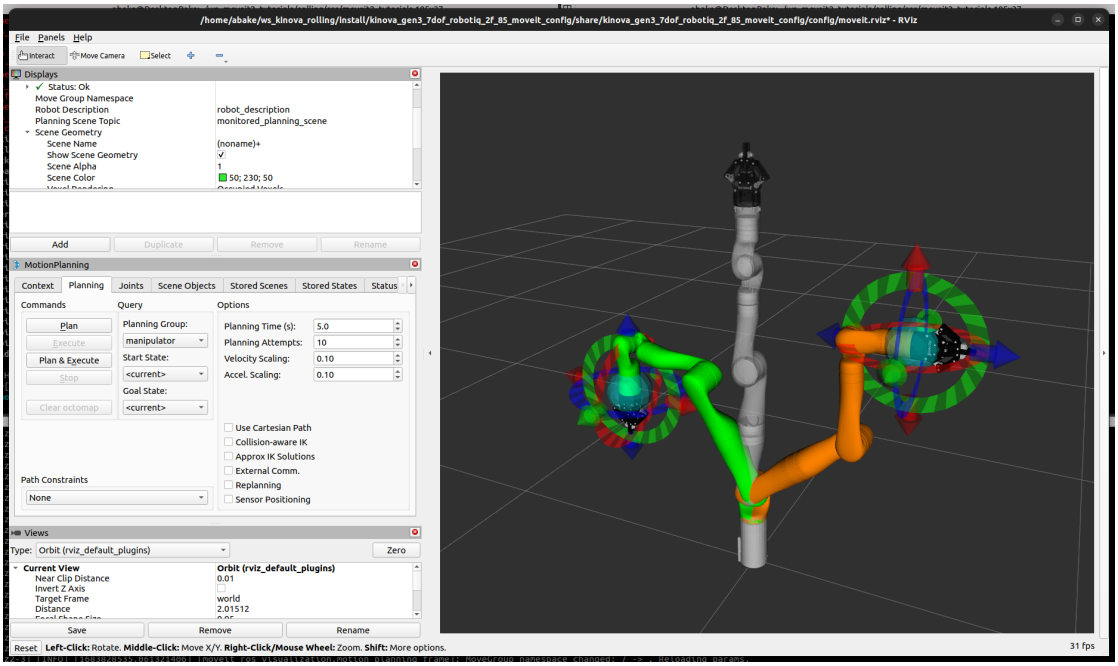


Figure 3.11: RViz visualization of Motion Planning Plungin [59]

The motion planning interface is realised through a series of calls to *actions* and *services* that are provided by the ROS **move_group** library. This library manages motion planning by taking into account a number of predefined constraints, including the position, orientation, visibility and joints of the robot. Additionally, it allows the user to define a range of customised restrictions, such as the inclusion of forbidden zones or areas to be avoided.

The default motion planner employed by move_group is **OMPL** (Open Motion Planning Library), an open-source library with a specialisation in motion planning. The OMPL employs randomised planning techniques to explore the configuration space of the robot and determine valid paths, even in complex environments. However, alternative planners are available that are compatible with the move_group library, including the Pilz Industrial Motion Planner and CHOMP (Covariant Hamiltonian Optimisation for Motion Planning). These may be employed in accordance with the specific requirements of the application.

The OMPL is specifically based on randomised planning algorithms, including RRT (Rapidly-exploring Random Tree) and PRM (Probabilistic Roadmap), which facilitate the efficient resolution of complex motion planning problems.

move_group library The move_group library is a ROS-compliant library that can be utilized in two distinct ways: directly within the code, through a C++ library import, or via a graphical interface in RViz. The library enables access to the information stored in the parameter server, including the URDF, SRDF and MoveIt configuration parameters present within the robot configuration package.

The library collates data regarding the specific components of the robot, including the present status of the joints and the current positions, by monitoring the relevant topics. Furthermore, it facilitates the handling of tf transformations, ensuring the consistency and validity of information within a unified reference system. This enables the planning and control of robot movement.

Chapter 4

Locobot wx250 Description

The LoCoBot wx250 [60] is the robot employed in the present thesis work for experimental purposes (see Figure 4.1).
LoCoBot (Low Cost Robot) is a mobile manipulator based on ROS that offers a series of open-source libraries to support the development of complex algorithms and advanced tasks. The robot has been mainly designed for research and educational applications, rendering it an optimal choice for academic projects.

Figure 4.1: LoCoBot wx250 [60]

The specific hardware characteristics relative to the robot in exam are shown in detail in Table 4.1.
The LoCoBot wx250, manufactured by Trossen Robotics and part of the Interbotix

X series, is equipped with a six degrees-of-freedom (DOF) manipulator, a Kobuki moving base, an RGB-D camera and an RPLidar sensor for mapping and navigation. The following sections will examine the technical and functional characteristics of each of these components in detail.

Feature	**Value**
Total Weight (kg)	11.25
Height (mm)	63
Width (mm)	35
Degrees of Freedom	6
Reach (mm)	750
Span (mm)	1500
Repeatability (mm)	1
Working Payload (g)	250

Table 4.1: LoCoBot wx250 Specification [60]

4.1 Mobile Base

One of the fundamental components of a mobile manipulator is the mobile base [60]. The LoCoBot wx250 offers two options for the mobile base: the Kobuki Mobile Base, manufactured by Yujin Robotics, and iRobot's Create3. In the context of the aforementioned experimental project, the Kobuki base was employed. The base has a circular configuration, as illustrated in Figure 4.2, which lends itself to high versatility and a compact footprint, rendering it particularly suitable for use in indoor and complex environments.

Figure 4.2: Kobuki mobile base [60]

The Kobuki base is equipped with an accurate odometry system and a gyroscope, which are essential tools for ensuring precise navigation. The base is equipped with two motorised wheels positioned at the extremes of the diameter, which are supported by two additional wheels, known as "casters", which are positioned diametrically opposite and perpendicular to the motorised wheels. This configuration provides greater stability. While the wheels are of a compact size, which makes the base ideal for use in indoor environments, this feature renders it less suitable for outdoor applications.

The base is equipped with a built-in battery that provides approximately 90 minutes of operational time. Moreover, it permits the connection to an external computer and the incorporation of supplementary structures and sensors, as exemplified in this project, thereby facilitating the robot's functionality beyond mere navigation.

4.2 Robotic Arm

Figure 4.3: WidowX-250 6DOF [61]

The manipulation functionality of the robotic system is enabled by a robotic arm that is mounted directly on the structure, situated above the mobile base. The LoCoBot model is equipped with three distinct manipulator options. The LoCoBot PincherX-100, LoCoBot WidowX-200 and LoCoBot WidowX-200 6DOF are the three models available [60]. The manipulators exhibit a distinction in the number

of available degrees of freedom (DOF), which are 4, 5, and 6, respectively.

Degrees of Freedom	6
Reach	650mm
Total Span	1300mm
Repeatability	1mm
Accuracy	5 - 8mm
Working Payload	250g*
Total Servos	9
Wrist Rotate	Yes

Table 4.2: WidowX-250 6DOF Specification Summary [61]

In the context of the aforementioned experimentation, the model with six degrees of freedom was utilized [61], as illustrated in Figure 4.3, with its specifications outlined in Table 4.2.

The manipulator is characterised by a high level of manoeuvrability, which is enabled by the six degrees of freedom and the ability to perform full 360-degree rotations. The position of the robotic arm is typically defined by six parameters. The spatial position is defined by (x, y, z), while the orientation is defined by (roll, pitch, yaw) angles. In manipulators with a limited number of degrees of freedom, the parameters that can be adjusted are constrained, which restricts the manipulators' manoeuvrability. A manipulator with six degrees of freedom, conversely, permits greater flexibility, enabling interaction with a larger number of points in space, with each parameter being modifiable independently.

Joint	**Min**	**Max**	**Servo ID(s)**
Waist	-180	180	1
Shoulder	-108	114	2+3
Elbow	-123	92	4+5
Forearm Roll	-180	180	6
Wrist Angle	-100	123	7
Wrist Rotate	-180	180	8
Gripper	30mm	74mm	9

Table 4.3: Joint Limits and Corresponding Servo IDs [61]

It should be noted that the various joints in the system have limitations for safety

reasons, as illustrated in the data presented in Table 4.3. These limitations are intended to preclude the performance of dangerous operations and to guarantee the safe utilisation of the manipulator.

The robotic arm belongs to the Interbotix X-Series family, which utilises Robotis' DYNAMIXEL X-Series actuators. These actuators offer higher torque and superior heat dissipation, enabling more efficient and prolonged operation.

4.3 Sensors

Figure 4.4: Locobot wx250 on board sensors [60]

In order to guarantee the optimal functionality of the previously described hardware, it is fundamental that sensors are installed directly on the LoCoBot structure. The sensors provide the robot with the requisite feedback to enable it to interact with its surroundings and become aware of the space in which it moves. The data collected can be used to develop advanced algorithms that enhance the device's intelligence and integration into its operational context.

The LoCoBot is equipped with two principal sensors: an RGB-D camera and an RPLidar sensor, as can be seen in Figure 4.4, the characteristics of which are delineated in the following sections.

4.3.1 RGB-D Camera

The first sensor to be discussed is the RGB-D camera, with a particular focus on the model integrated within the LoCoBot wx250, which is the Intel RealSense Depth Camera D435 [62].

The principal distinction between a conventional camera and an RGB-D camera, such as the aforementioned model, resides in the nature of the data it is capable of capturing. In contrast to a conventional camera, which captures only RGB

Figure 4.5: Intel® RealSense™ Depth Camera D435 [62]

(red, green, blue) images, an RGB-D camera also provides information about the depth of objects in the scene. This additional capability is made possible by the integration of an infrared sensor, which measures the distance between the camera and surrounding objects.

In the case of the Intel RealSense D435, the compact and lightweight design provides considerable versatility, rendering it particularly well-suited to robotic applications. The extensive field of view and capacity to detect objects at distances up to 10 metres render it highly functional for artificial intelligence tasks. These characteristics, in conjunction with the presence of a "global shutter" sensor, render the D435 an optimal solution for applications such as robotic navigation and object recognition. Furthermore, the global shutter sensors enhance sensitivity in low-light conditions, enabling the robot to navigate in poorly illuminated environments with precision and safety.

Figure 4.6: Camera tilt movement [60]

Figure 4.6 exemplifies a further significant attribute of the camera: the incorporation of the Dynamixel 2XL servomotor enables the camera to rotate around two reference axes, facilitating the so-called pan and tilt movements, or rotations on

the X and Z axes [60].

The servomotor is capable of simultaneously executing both movements, thereby providing a more comprehensive and expansive view of the surrounding environment. This simultaneous control capability markedly enhances the robot's spatial perception, enabling it to adapt dynamically to the demands of exploration and navigation.

4.3.2 RPLidar

The second sensor in the LoCoBot wx250 is a LiDAR, which is based on a technology that employs light to quantify the distance between the sensor and the surrounding objects. Laser radar emits beams of light, which, after being reflected by objects in the surrounding environment, are detected by the sensor. By calculating the time taken for the light to return, the distance to the objects can be determined with a high degree of accuracy.

The LiDAR model installed on the robot is the RPLIDAR A2 360° Laser Range Scanner [60]. This is a laser scanner with the ability to rotate 360 degrees, thereby providing a comprehensive representation of the surrounding environment. The device is capable of performing up to 8,000 samples per second, due to its high rotation speed. Furthermore, the onboard system is capable of performing two-dimensional 360-degree scans with a range of 12 metres, which can be extended up to 18 metres through a firmware upgrade.

Figure 4.7: RPLIDAR A2 360° Laser Range Scanner [60]

The output of the sensor is a two-dimensional point cloud, which can be conceptualised as a two-dimensional map representing the spatial distribution of objects

in the vicinity of the robot. LiDAR is therefore an indispensable instrument for activities such as mapping, localisation and navigation, ensuring that the robot can move with precision and security within its environment.

4.4 OnBoard Computer

The system's computational capabilities are provided by a computer that is directly installed on the mobile manipulator. The model employed for the development of this project is the Intel NUC Mini PC, which can be accessed remotely via an SSH connection.

Figure 4.8: Intel NUC Mini PC [60]

The system is equipped with an eighth-generation Intel Dual-Core i3 central processing unit (CPU) with two cores and four threads, supported by 32 GB of double data rate 4 (DDR4) random-access memory (RAM) and a 256 GB solid-state drive (SSD) for storage. In terms of graphical performance, the NUC is equipped with an Intel Iris Plus Graphics 655 GPU, which is capable of handling the processing of graphical data in an efficient manner [60].

These specifications render the NUC a highly functional and efficient device for the completion of tasks of medium complexity. In order to guarantee optimal autonomy and sustained performance during operation, the system is equipped with an external MAXOAK K2 battery with a capacity of 50,000 mAh.

The MAXOAK K2 battery [60] has been designed to provide power to electronic devices, such as laptops, for extended periods of time. With a capacity of 50,000 mAh, the battery is capable of sustaining the NUC and all peripheral devices for approximately four hours under conditions of moderate use. The battery is equipped with a variety of output options, including 20V 5A and 12V 2.5A barrel connectors, in addition to two 5V 2.1A and two 5V 1A USB-A ports.

Chapter 5

Social Navigation

One of the main objectives of this thesis project is the development of **social navigation** capabilities. The term "social navigation" refers to the robot's ability to map, locate and navigate in enclosed and structured environments, taking into account the presence of people and ensuring safety through collision avoidance. The objective is to develop a system that enables the robot to be incorporated into environments that are shared with humans, such as homes, offices, workplaces, or laboratories. In addition to ensuring safety, this approach also aims at making the system adaptable to contexts that are becoming increasingly prevalent.

In order to achieve this objective, the project started with the integration of the ROS Navigation Stack, specifically the library provided by Trossen Robotics that furnishes a specific configuration adaptable to the Locobot wx250. Furthermore, a **social navigation layer** and an algorithm for detecting and tracking users in the scene were incorporated.

The Navigation Stack provides a fundamental implementation for navigation, including local and global planners, and local and global costmaps. However, it does not take into account the potential presence of moving obstacles. To enhance the precision and efficiency of the mapping and localisation processes, RTABMap was employed, resulting in a more robust and optimised structure.

The principal sensors employed are the LiDAR, which generates a comprehensive point cloud representation of the surrounding environment, and a RGB-D Camera that furnishes supplementary data beneficial for the purposes of localisation and social navigation. The camera allows for tracking and identification of individuals within the robot's trajectory.

With regard to global trajectory planning, the **A*** algorithm, which is a *best-first*

search algorithm, was selected. This search method, based on graphs, determines the optimal path as a sequence of nodes at a lower cost, thereby enabling the robot to identify the most efficient route from the starting point to the goal.
An additional aspect of social navigation is the ability to recognise people as moving obstacles, influencing the local planner to adjust the robot's trajectory in the presence of users.

The following sections will provide detailed descriptions of the additional components of the strategy, offering explanations of the work conducted within the project.

5.1 Social Navigation Layers

As previously stated in Chapter 3, in particular in Section 3.3.3, both global and local costmaps, are fundamental tools for navigation planning. These maps serve as the primary means of ensuring the robot's awareness of its surrounding environment and the maintenance of safety during movement.

Costmaps are constituted by a set of cells, each of which is associated to a cost determined by layers. The layers are overlapping structures that provide supplementary data on each cell, thus enabling their categorisation according to a range of criteria. This classification is made visible through a colour-coding system displayed on the map in RViz.

In standard navigation, fundamental layers are employed to identify and categorise objects within the static map. The most commonly occurring layers are as follows:

- The **static layer** furnishes data regarding static obstacles, such as walls or furnishings, that are identified during the preliminary mapping of the environment;
- The **obstacle layer** incorporates sensor data, including LiDAR and camera information, to identify and update the map in real time with information about dynamic obstacles;
- The **inflation layer** is a mechanism that introduces a safety zone around obstacles, increasing the cost of cells in that area to ensure that the robot maintains a safe distance while navigating.

However, to address the issue of mobile obstacles, such as people, ROS offers supplementary layers of functionality. These layers, designated as **Social Navigation Layers** [63], enhance the robot's behaviour, enabling it to adapt to dynamic

scenarios and safely interact with individuals within the environment. The aforementioned layers are based on models of human behaviour, which are designed to avoid collisions, respect personal distance and plan socially acceptable routes.
The existing social layers are two:

- `social_navigation_layers::ProxemicLayer`
- `social_navigation_layers::PassingLayer`

The two layers manage and exchange information about individuals in the vicinity of the robot, modifying the costmap by adding a Gaussian distribution around each person, represented in the left image of Figure 5.1 as a blue shape.

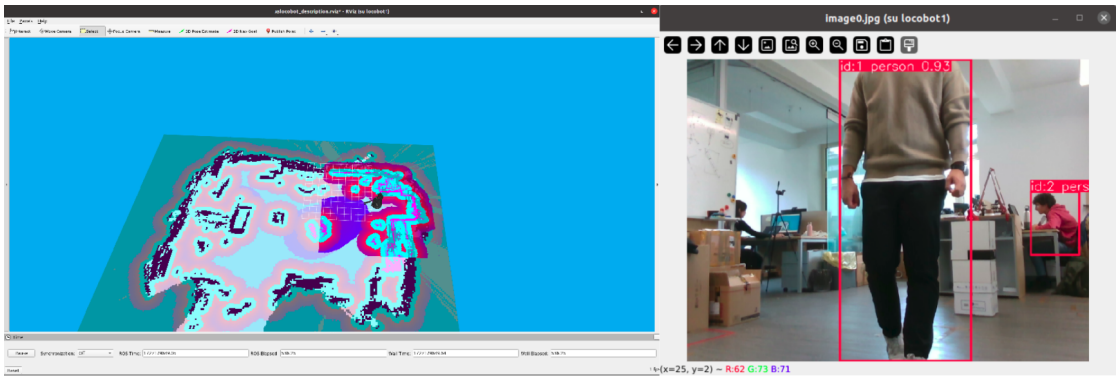


Figure 5.1: The left image depicts the RViz visualisation of the `ProxemicLayer` output on the map, whereas the right one shows the output of the camera mounted on the robot.

Both layers subscribe to the topic `/people (people_msgs/People)` [63] in order to obtain data pertaining to the individuals present within the scene. The `People` message is an array of messages of the `Person` type, structured as follows:

```
people_msgs/Person.msg:
    string              name
    geometry_msgs/Point position
    geometry_msgs/Point velocity
    float64             reliability
    string[]            tagnames
    string[]            tags
```

The fundamental information utilized for the operation of the layers is derived from the tracking of individuals, the recording of their position and speed at each frame, and the subsequent analysis of these data points. This enables the robot to

implicitly deduce the direction and speed of movement of the people in its vicinity, thereby allowing it to react in an appropriate manner.

`social_navigation_layers::ProxemicLayer` This layer [63] employs a Gaussian distribution to predict the movement of the person in question. In the event that the monitored individual is at rest, the Gaussian distribution assumes the form of a circle around the user. Conversely, if the person is in motion, the Gaussian distribution expands in accordance with the direction of movement, thereby anticipating the trajectory.
Figure 5.2 illustrates this phenomenon. It demonstrates that when a person is moving away from the robot, the Gaussian is stretched in the opposite direction to the robot's position. Figure 5.1 depicts a scenario in which the person is approaching the robot. In this instance, the Gaussian is observed to be stretched in the direction of the robot.

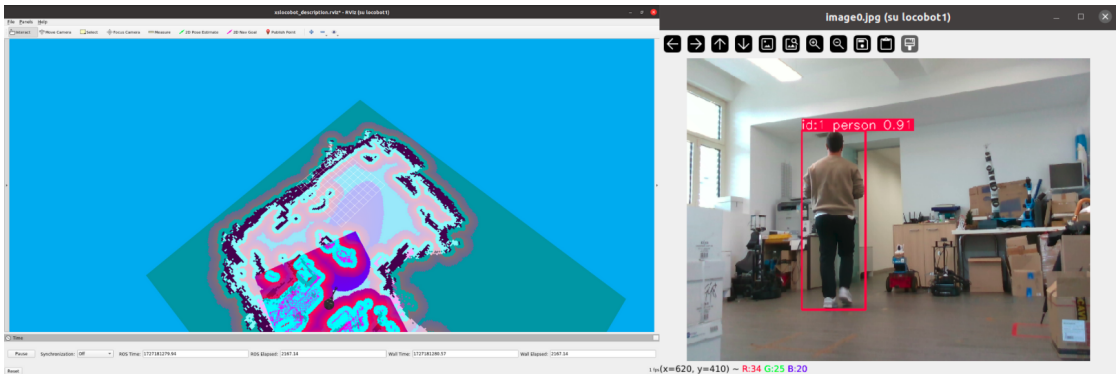


Figure 5.2: The illustration depicts a Gaussian shape of a person walking away from the robot

In the thesis project, the specified layer was enabled and configured to detect people within a radius of 3 metres (`cutoff: 3.0`). The ability to perceive distance is a fundamental aspect of the robot's operation, as it directly influences its behaviour in accordance with the set amplitude value. This parameter renders distance variations a particularly salient factor in determining the robot's reactions.
The system's accuracy is guaranteed by a low covariance (`covariance: 0.25`), which minimises uncertainties in the measurements and enhances the precision of perception. Moreover, the impact of the proximity layer is calibrated by an average weight factor (`factor: 5.0`), which strikes a balance between the influence of proximity and other navigation priorities. In this case, the high value means the higher importance of this layer with respect to the other ones.

Ultimately, the "keep_time" of the proximity reaction is relatively brief, at 0.5 seconds (`keep_time: 0.5`), thus ensuring a rapid and adaptive response to changes in the surrounding environment.

```
proxemicLayer:
    enable: true
    cutoff: 3.0
    amplitude: 80.0
    covariance: 0.25
    factor: 5.0
    keep_time: 0.5
```

`social_navigation_layers::PassingLayer` This layer [63] is responsible for the management of the robot's behaviour when passing in close proximity to a person, ensuring that it always passes to the left of the user. In order to achieve this, the cell cost to the right of the person is increased, thereby prompting the robot to select a path to the left.

It is important to note that, in the context of this project, the Passing Layer was not employed, as it is not always necessary for the robot to pass to the left of a person. Indeed, in a multitude of scenarios, it is plausible that the robot may alternatively proceed to the right, on the basis of the specific circumstances.

5.2 People Detection

The correct functioning of social navigation layers necessitates the acquisition of data regarding the robot's surrounding environment. It is of particular importance to determine the presence of individuals in the immediate vicinity of the device, in order to implement the appropriate safety measures and ensure the security of the operational environment.

As previously outlined, in order to define the optimal behaviour of the robot when it shares a space with other mobile agents, it is essential to have precise information on the position and speed of any potential mobile obstacle (in this case, people).

The strategy adopted is based on the use of YOLOv8 for object tracking, a machine learning algorithm that will be explored in detail in the following paragraph. This decision proved to be appropriate, despite the numerous alternatives in the modern literature, due to the satisfactory balance between the excellent performance obtained and the required computational resources, which were compatible with

the robot's capabilities.

The node tasked with detecting and tracking individuals operates as both a publisher on the topic `/people` and a subscriber to receive the required data, including colour images from the RealSense camera, depth images, and other information from the same camera.

The images are subject to a series of processing techniques to ensure compatibility with the neural network, which will utilise them as input. Subsequently, the YOLO system is employed to identify the individuals present within the scene. The data obtained comprises the coordinates of the bounding boxes for each individual, along with an ID number for tracking purposes. Consequently, the position of the individual is determined by calculating the midpoint of each bounding box.

At this juncture, the two-dimensional coordinates of the pixels are transformed into three-dimensional coordinates with respect to the robot's reference system through the utilisation of the tf package. In this manner, people are identified and tracked throughout the course of their journey. The final piece of information required is the speed of the individuals, which is calculated as the variation of their positions over time.

5.2.1 Ultralytics YOLOv8 for Object Tracking

YOLOv8 [64], developed by Ultralytics, represents an advanced neural network belonging to the YOLO family, which is widely used for real-time object detection. As illustrated in Figure 5.3, this model demonstrates superior performance compared to previous versions, largely due to significant enhancements in both accuracy and speed. This combination renders the model highly versatile and suitable for a wide range of applications. One of the key strengths of YOLOv8 is the incorporation of sophisticated backbone and neck architectures, which optimise feature extraction, thereby improving performance in object detection. Furthermore, the novel anchor-free Ultralytics split head facilitates more efficacious detection than conventional methodologies, thereby conferring superior accuracy.

YOLOv8 has been designed to maintain an optimal balance between accuracy and speed, rendering it an ideal choice for real-time detection tasks in a variety of industrial contexts. The availability of pre-trained models allows for the selection of the most appropriate version for a given task.

The YOLOv8 series comprises a diverse range of models, each of which has been optimised for a specific machine vision task. The models support a variety of

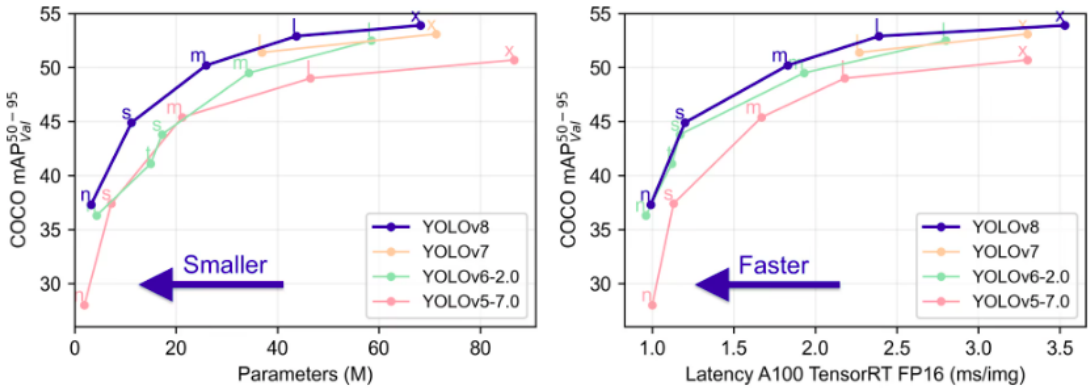


Figure 5.3: Comparison between YOLOv8 and the preceding versions of the model developed by Ultralytics [64]

operating modes, including Inference, Validation, Training and Export, thereby facilitating their use at all stages of the development and implementation process.

In order to fulfil the specified task, the YOLOv8n model was utilised, having been pre-trained on the COCO dataset. The specifications of the YOLOv8n model, which has been pre-trained on the COCO dataset, are provided in detail in Table 5.1.

Feature	**Value**
Model	YOLOv8n
Size (pixels)	640
mAPval 50-95	37.3
Speed CPU ONNX (ms)	80.4
Speed A100 TensorRT (ms)	0.99
Params (M)	3.2
FLOPs (B)	8.7

Table 5.1: YOLOv8n model Performance [64]

COCO Dataset COCO (Common Objects in Context) [65] is a comprehensive, large-scale dataset designed for the detection, segmentation and caption generation of visual content. The dataset includes 330,000 images, 200,000 of which have been annotated. These images are divided into 80 object categories, ranging from common objects such as cars and animals to more specific categories such as bags

and sports equipment. Each image is accompanied by detailed annotations in the form of bounding boxes, segmentation masks and captions.

The COCO dataset employs standardised metrics, including mean Average Precision (mAP) for object detection and mean Average Recall (mAR) for segmentation. This renders it an optimal instrument for the comparison of model performance.

The COCO dataset is divided into three principal subsets:

- **Training:** The Train2017 subset includes a total of 118,000 images, which are used for training the models for object detection, segmentation and captioning;
- **Validation:** Additional 5,000 images have been allocated to the Val2017 subset with the objective of validating the model during the training phase;
- **Test:** Further 20,000 images have been designated for the purposes of testing and benchmarking the trained models and have been allocated in the Test2017 subset.

COCO is a highly utilised resource for the training and evaluation of deep learning models for object detection, including YOLO and Faster R-CNN, as well as for segmentation and key point detection. It is regarded as an indispensable resource for computer vision researchers and practitioners alike.

Object Tracking Ultralytics' YOLOv8 can track objects in real time [66], in addition to its traditional object detection capabilities. This functionality makes it extremely versatile and efficient, especially in areas such as security, video surveillance or sports analysis.
The information provided by YOLOv8 is similar to that of a normal object detec-

Figure 5.4: Real time tracking object with YOLOv8 [66]

tion algorithm: the coordinates of the bounding box that delimits the object, the name of the detected object and a percentage of accuracy are provided. However,

in this case an additional field is added, the ID, which allows the same object or person to be identified and tracked over time.

YOLOv8 uses two main object tracking algorithms that can be selected in the configuration:

- **ByteTrack**: is a multi-object tracking method that assigns a unique identifier to each object within a scene, thereby enabling the tracking of its movements. Firstly, the most certain detections are associated with existing tracks, utilising either IoU (Intersection over Union) or re-identification (Re-ID). Subsequently, the less certain detections are matched in order to recover any objects that have been lost due to the presence of obstacles or blurring. In the event that some tracks are not matched, they are deleted, and the new unmatched detections result in the creation of new tracks. Consequently, ByteTrack is capable of maintaining accurate tracking even in challenging circumstances [67];
- **BoT-SORT**: similar to ByteTrack, this is another multi-object tracking algorithm, but with some improvements. It integrates a Kalman filter to predict the movement of objects and a Re-ID system to detect them in subsequent frames. It also compensates for camera movement and combines visual similarity (Re-ID) with Bounding Box IoU to improve object tracking [68].

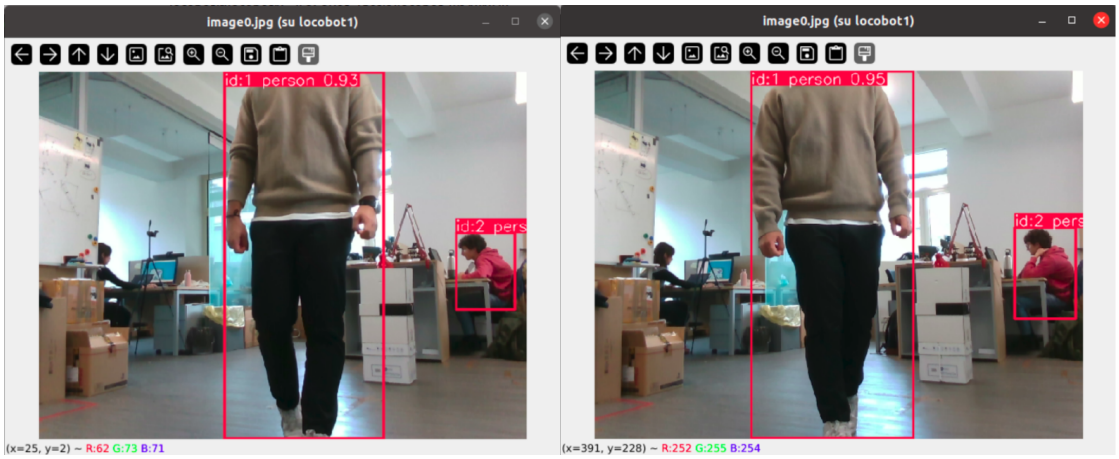


Figure 5.5: Two consecutive frames illustrate the tracking of individuals. Two distinct individuals are present in the scene, and both frames demonstrate that the identification number assigned to each person remains consistent.

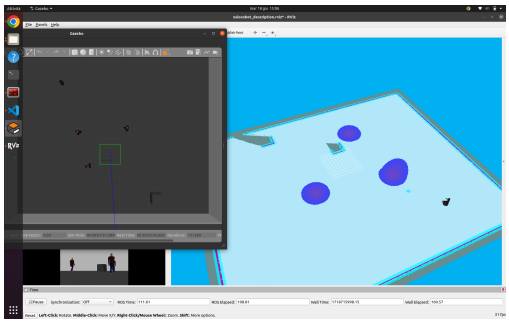
The algorithm used in the project was BoT-SORT. Although it is a multi-object tracking algorithm, in this particular project it was only used to track people.

However, the ability to track different types of objects means that the algorithm can also identify and track other elements in the scene, such as service robots, so that they can be easily avoided.

5.3 Simulation and Test

To validate the strategy and verify its functionality, the task was first tested in a simulation environment and then in a real-world setting.
The decision to conduct preliminary tests in a simulated environment is based on the premise that the robot is designed to navigate in locations that are shared with humans. It is possible that the robot will encounter one or more failures, which could result in the human operator being placed at risk of collision. The use of a simulated environment can help to create a safe version of the algorithm that can then be tested in the real situation. A basic model of the simulated environment was constructed in Gazebo, incorporating dynamic actors, as illustrated in Figure 5.6a.

(a) *Animated actor in Gazebo*

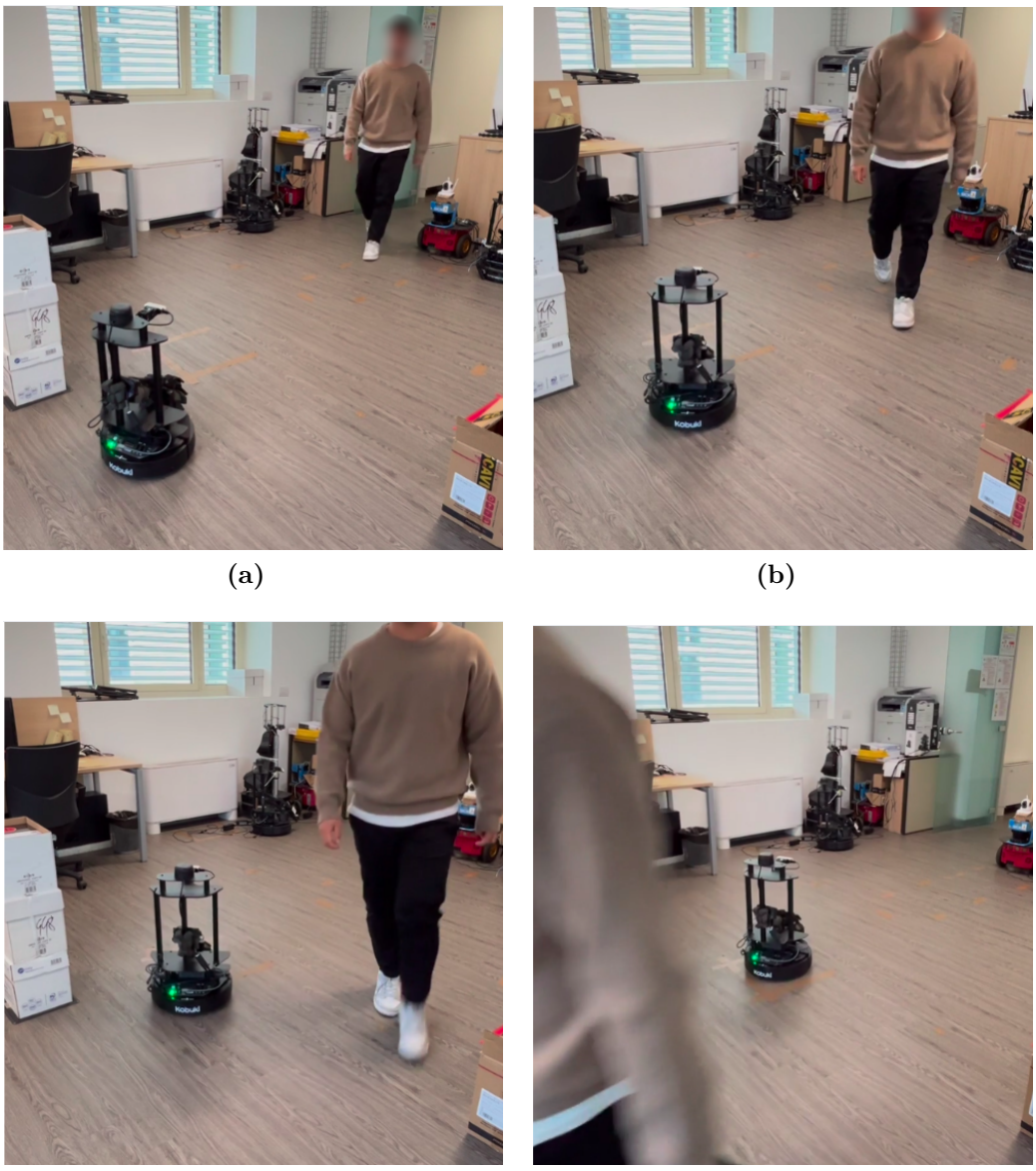


(b) *RViz Visualization of tracking and detection of 3 people in the scene*

Figure 5.6: Testing phase conducted in a simulated environment.

As demonstrated in Figure 5.6b, when three individuals approach the robot from three different directions, the robot is able to locate the actors and update the costamps in the corresponding layers. Furthermore, during the navigation, the local planner is capable of using the acquired information and instructing the robot to act in an appropriate manner. This leads to the successful implementation of the adopted strategy, thereby enabling the test to be conducted in a real-world setting at the Robotics Laboratory of the Department of Electronics and Telecommunications (DET) of Politecnico di Torino. As demonstrated in Figures 5.1, 5.2, the results obtained evidenced that the robot was not only capable of correctly

detecting people, but was also able to avoid them while maintaining a sufficient safety margin.
Figure 5.7 demonstrates the behaviour of the Locobot in a real-world setting. Initially, the robot and the user move in a forward direction towards each other, see Figure 5.7a. Once they have crossed, however, the robot deviates slightly from its original trajectory in order to maintain an appropriate safety distance from the user, as illustrated in Figures 5.7b and 5.7c. When the robot has successfully avoided the moving actor, it can resume its trajectory along the global path (Figure 5.7d).



(a) **(b)**

(c) **(d)**

Figure 5.7: Safe social navigation.

Chapter 6

Object Grasping

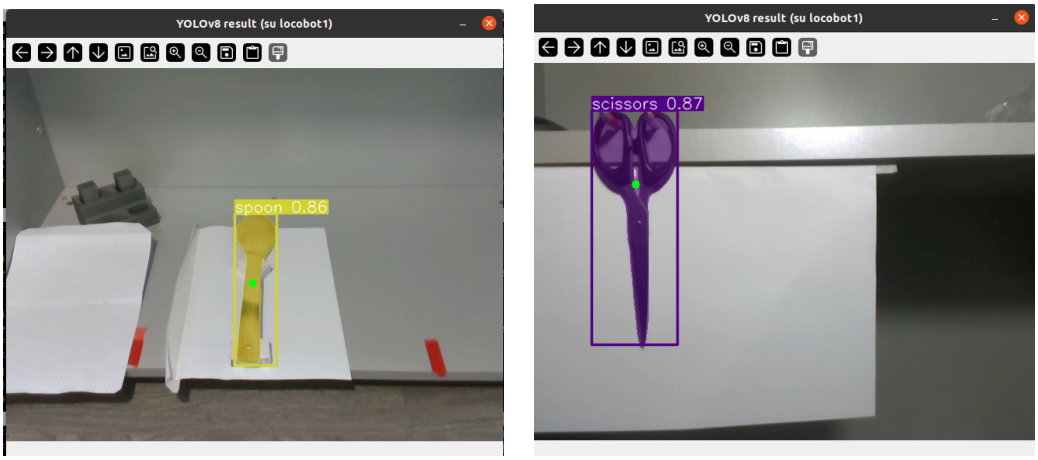
The objective of this project is to develop a service robot that can retrieve an object and deliver it to the user. The primary objective is to develop the capability to grasp objects. In this context, the term "grasping" refers to the robot's ability to identify the target object, determine its position, define a grasping point, and finally perform the movement to grasp it. Once the object is successfully grasped, the robot keeps the object within its operating area in a safe manner. The chosen position ensures that the robot can navigate safely, avoiding collisions with people or objects in its vicinity, while maintaining the correct distance from obstacles.

6.1 Object Segmentation

The essential data for the successful completion of the task are derived from the RGB-D camera. It is necessary for the robot to be aware of the objects in its vicinity, to identify the correct object, and to calculate its position in order to reach it.

In order to obtain such data, the images generated by the camera are processed and then transmitted as input to the YOLOv8-seg neural network [69]. This network, which is part of the Ultralytics family and is described in detail in Section 5.2.1, is concerned with the detection of objects. However, rather than tracking objects, it is focused on the process of object segmentation.

The process of object segmentation enables the classification and assignment of each individual pixel of an object to a specific class. This enables the precise identification of the area occupied by the object within the image. This feature enables the precise delineation of the object's shape, thereby facilitating the calculation of its position and enabling the robot to perform a correct grasp, as illustrated in Figure 6.1.



(a) *Wooden spoon* **(b)** *Scissor*

Figure 6.1: Image segmentation results

Once more, one of the pre-trained models provided by Ultralytics was employed, specifically **YOLOv8s-seg**, the characteristics of which are outlined in Table 6.1. To train this model, the COCO dataset was utilised, which is particularly well-suited for training networks specialising in image segmentation due to its extensive collection of annotated images.

Feature	**YOLOv8s-seg**
Size (pixels)	640
mAPbox 50-95	44.6
mAPmask 50-95	36.8
Speed CPU ONNX (ms)	155.7
Speed A100 TensorRT (ms)	1.47
Params (M)	11.8
FLOPs (B)	42.6

Table 6.1: YOLOv8s-seg Model Performance [69]

6.2 Grasping Pose Computation

In order to accurately determine the position of the end effector during the grasping process, a methodology was developed that considers the spatial location of the object to be grasped.

Initially, the node that is responsible for the processing of the pose subscribes to the camera topic, thus enabling the robot to receive images of the area in which the objects are situated. The initial phase of the process entails the Locobot searching for the object, which may be located either vertically, in a hanging position, or horizontally, resting on the base of the shelf.

Subsequently, the robot initiates an exploration phase with the objective of finding the object. The search procedure starts with a slight tilt of the camera of -0.1 degrees (upwards), see Figure 6.2a, enabling the robot to view the hanging objects. The Locobot maintains this position for a period of 10 seconds, during which an object detection algorithm, YOLOv8, analyses the image in order to identify the object of interest. If the object is not located, the camera is tilted further 0.7 degrees, as illustrated in Figure 6.2b, enabling it to frame the shelf and resume the search.

(a) *-0.1°* **(b)** *0.7°*

Figure 6.2: Camera tilt inclination

Upon locating the object, the camera is maintained in a fixed tilt, and the captured images are transmitted to the YOLOv8s-seg network, which is tasked with segmenting and defining the shape of the object. To reduce the margin of error due to the uncertainty of the depth camera, eleven consecutive frames are analysed and multiple measurements are stored.

For each frame, a series of operations are conducted with the objective of determining both the inclination of the object and its precise location. The inclination is essential for determining whether the end-effector should be positioned at 0 or

80 degrees, while the position is crucial for performing the grasping manoeuvre correctly, as illustrated in Figure 6.3.

(a) *0.0°* **(b)** *80°*

Figure 6.3: End-effector inclination with respect to the inclination of the object to grasp.

The image segmentation serves to determine the centre of mass of the object of interest, which is calculated on the basis of the spatial moments of the considered image. With regard to the binary mask, the following equations may be used in order to determine the centre of mass:

$$\mathrm{cX} = \frac{M_{10}}{M_{00}}, \quad \mathrm{cY} = \frac{M_{01}}{M_{00}} \tag{6.1}$$

Where the variables M_{00}, M_{10} and M_{01} represent the spatial moments of the mask. The aforementioned calculation is illustrated in Figure 6.1, wherein the centre of mass is represented by the green point within the mask of both objects.

Subsequently, two additional points belonging to the object are randomly selected, provided that they are part of the binary mask. The random selection of points serves to circumvent collinearity among them, which could otherwise result in erroneous calculations of the object's normal. It is imperative that the normal be determined in order to ascertain the inclination of the object, thus ensuring that the end-effector is oriented correctly during the gripping operation.

Subsequently, the three selected points (the centre of mass and the two randomly

selected points) are transformed from the two-dimensional coordinates of the image to three-dimensional coordinates, utilising the depth information provided by the camera, as:

$$Z = \frac{\text{depth}}{1000} \tag{6.2}$$

$$X = \frac{(u - cX) \cdot Z}{fX}, \quad Y = \frac{(v - cY) \cdot Z}{fY} \tag{6.3}$$

where u and v represent the coordinates of the points in the image, cX and cY denote the intrinsic parameters of the camera, and fX and fY are the focal distances. Subsequently, the three-dimensional points are transformed into the robot's reference system: `/locobot/base_footprint`.

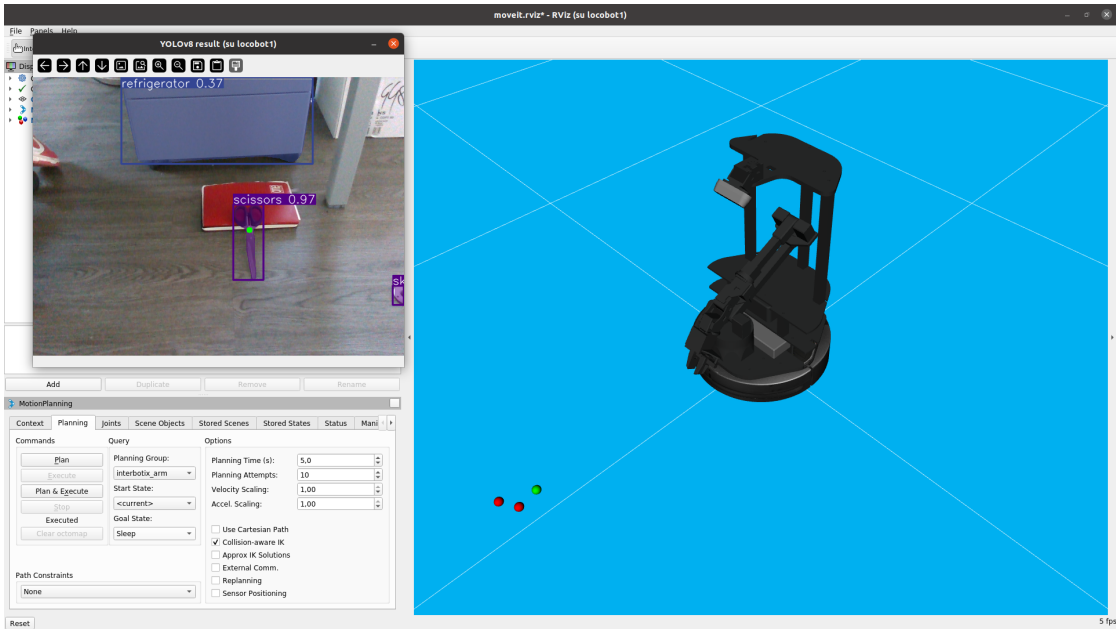


Figure 6.4: RViz visualization of captured points

The transformed 3D points, are visualised using RViz, as illustrated in Figure 6.4, to verify the accuracy of the coordinates and the calculated plane. From the three 3D points $P1$, $P2$, and $P3$, two vectors lying on the plane are calculated as:

$$\vec{v1} = P2 - P1, \quad \vec{v2} = P3 - P1 \tag{6.4}$$

The normal vector to the plane is obtained by their vector product as:

$$\vec{n} = \vec{v1} \times \vec{v2} \tag{6.5}$$

The computed vector is then expressed in the following form:

$$\vec{n} = (a, b, c) \tag{6.6}$$

At this point of the calculation, the only remaining component to be determined is the term d, required to complete the equation for the plane containing the object's mask, $ax + by + cz + d = 0$. This value can be obtained by using the coordinate of the object's centre of mass in the reference system `/locobot/base_footprint`:

$$CM = (x_{cm}, y_{cm}, z_{cm}) \tag{6.7}$$

Subsequently, the value of d is computed as:

$$d = -(a \cdot x_{cm} + b \cdot y_{cm} + c \cdot z_{cm}) \tag{6.8}$$

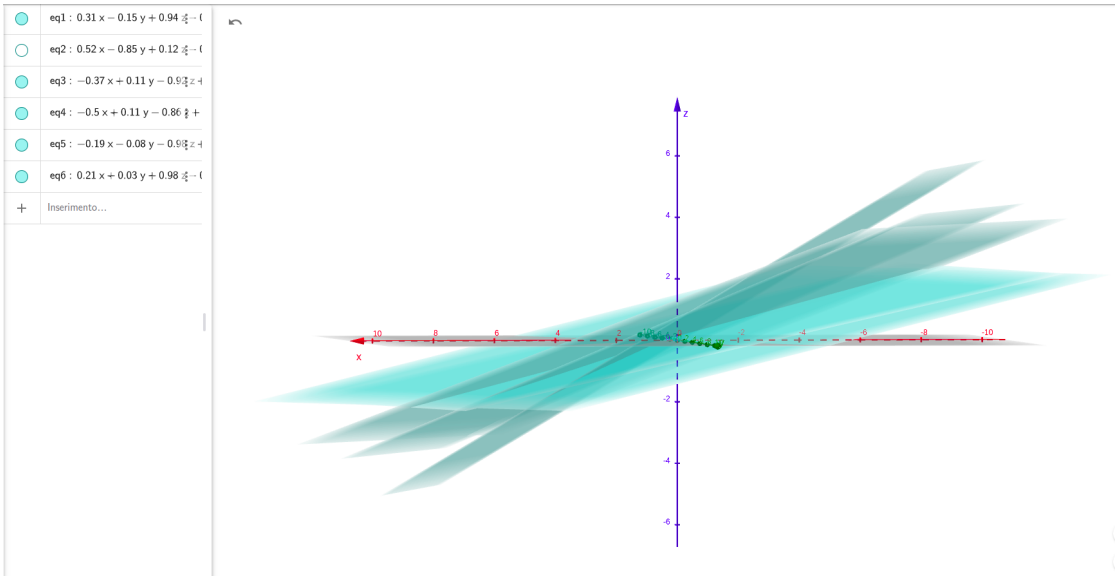
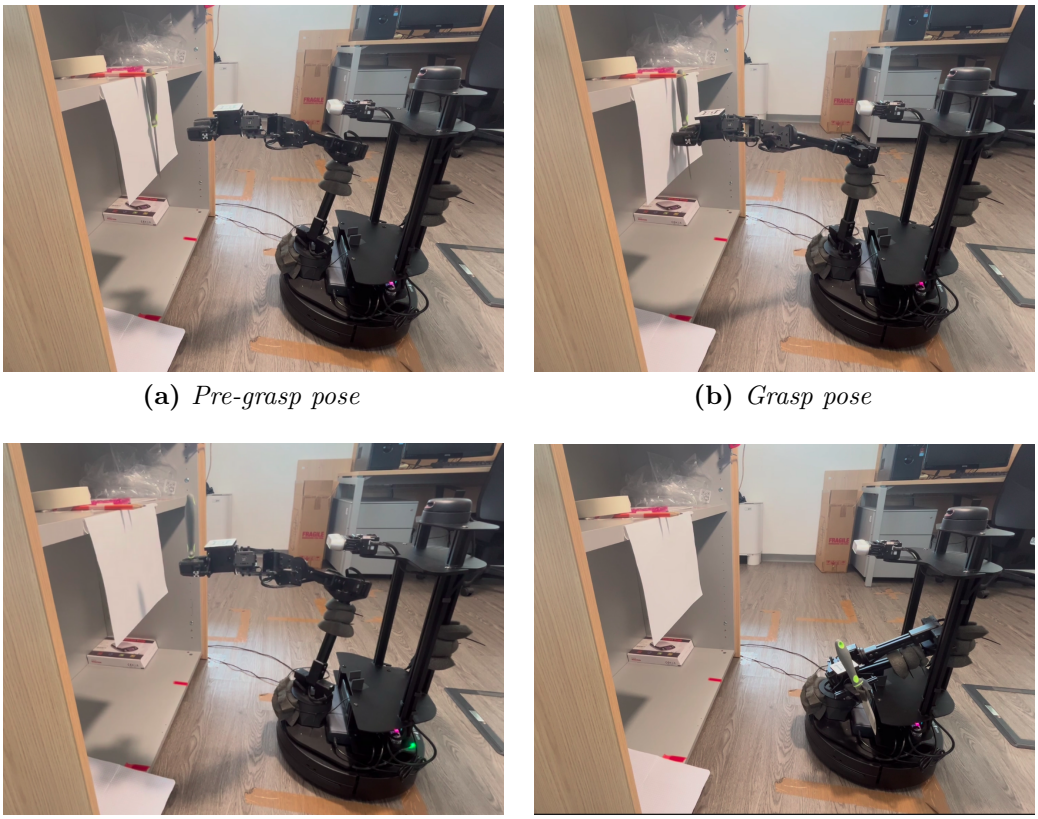


Figure 6.5: Planes containing the object mask

The representation of the plane allows for the computation of the object's inclination, thereby establishing whether the object is in a supine or hanging position. The position is associated with the calculated centre of mass, whereas the orientation requires a more complex evaluation.

The roll and yaw angles of the gripper are maintained at 0 degrees, while the pitch angle is determined based on the inclination of the plane. In the event that the object is in a state of suspension, the pitch angle is maintained at 0 degrees.



(a) *Pre-grasp pose* **(b)** *Grasp pose*

(c) *Post-grasp pose* **(d)** *Final position*

Figure 6.6: Grasping process of a knife in a hanging position.

Conversely, if the object is in a horizontal position with a 0-degree inclination, the pitch angle is set at 80 degrees. To ensure the accuracy of the calculation, the analysis of the data is conducted on the basis of 11 consecutive frames, thus reducing the influence of any errors.

The inclination of the plane is evaluated according to the following criteria: if the inclination is less than 45°, the object is considered to be lying down and the pitch angle will be 80°; if the inclination is between 45° and 90°, the object is considered to be in a vertical position and the pitch angle will be 0°.

Ultimately, this angle is employed to differentiate between the various operational scenarios and to delineate the pre- and post-grasping poses, as well as to determine the requisite behavior for attaining the secure position during the conveyance of the object to the receiver.

In order to guarantee safe and efficient navigation, the object is positioned so to avoid any increase in the overall size of the robot. This guarantees that the navigation system remains fully operational and that the efficiency of the robot is not compromised. Furthermore, this prevents any protruding parts from coming into contact with or interfering with obstacles in the surrounding environment, thereby reducing the risk of accidents or damage. The correct arrangement of the object is illustrated in detail in the Figure 6.7, which serves to illustrate and support the instructions provided.

(a) **(b)**

Figure 6.7: Safe position after the grasping of a knife

6.3 Limitations

It should be noted, however, that the performance of this task is not without limitations. During the grasping phase, the gripper is fully closed in order to guarantee a secure grip on the object. Nevertheless, this frequently results in the motors becoming blocked as they attempt to close the gripper beyond the permitted limit, due to the presence of the object within the gripper panels. To circumvent this issue, a methodology was developed to calculate the optimal closing angle. In

particular, the width of the object is evaluated, considering a weighbonhood around its Centre of mass (COM) 200 pixel large. Once the minimum width is found, the two extremities of the object mask are adopted as reference points. Subsequently, these points are transformed into the /locobot/base_footprint reference system, and the width of the gripper closure is determined based on the actual distance between them.

Once the amplitude has been calculated, the closure angle is proportionally established. Nevertheless, despite this approach, motor lock can still occur due to hardware and precision limitations, leading to an excessively narrow or insufficient grip. To enhance the process, it would be necessary to refine the precision in the calculation of the width and angle of closure, as well as to upgrade the gripper hardware to ensure greater accuracy. The utilisation of force sensors in the gripper could be a further solution, providing real-time feedback to improve the quality of grasping.

Chapter 7

Robot to Human Object Handover

The handover constitutes the concluding and most intricate phase of the entire process. This task is particularly complex, due to the necessity for the Locobot to interact directly with the user, thereby requiring maximum precision and safety. Typically, the development of such a task necessitates the integration of sophisticated sensors, including force sensors, additional cameras, and even microphones, to facilitate prompt response to user feedback. However, in this specific context, the only sensor available for development is the RGB-D camera.
To address this limitation, the handover task was designed in a standardised manner, utilising only the visual information collected by the camera. In this way, the user is informed in advance of the exact movements of the Locobot, ensuring predictability. Each subsequent movement of the robot will only occur in response to explicit feedback provided by the user, ensuring a controlled and safe interaction.

The rationale behind the design of the handover in these circumstances is to create basic movements that are easily predictable by the user. Upon reaching the user's position, the Locobot's objective is to manoeuvre the robotic arm into one of three standard positions, designated as "home" (see Figure 7.1b).

At this phase, the robot will await user feedback to indicate the completion of the task and proceed with the subsequent movement. In the absence of force sensors, which could signal dangerous situations, or microphones to listen to any voice communication from the user, the video camera serves the function of monitoring the user's actions. In accordance with an approach inspired by sign language, the user is able to utilise gestures to communicate the status of the action to the robot.

The Locobot has been programmed to recognise the **thumbs-up** gesture as an indication of a positive signal from the user (Figure 7.1c). In particular, the user should indicate to the robot, via this gesture, that he or she has grasped the object that the robot is extending towards him or her. Upon detecting the gesture, the robot is thereby informed that the object has been taken by the user and that it is no longer necessary for it to maintain its grasp. At this point, the gripper is opened and the robot assumes a predefined rest position, designated as "sleep", as illustrated in Figure 7.1d.

Upon reaching this position, the robot has successfully completed the task and will await further instructions from the user.

(a) *Locobot arrives to the user position*

(b) *"Home" position*

(c) *Thumb up signal*

(d) *"Sleep" position*

Figure 7.1: Handover maneuver.

7.1 Gesture Recognition

In order to facilitate the aforementioned scenario, it is necessary to equip the Locobot with a gesture recognition algorithm, thereby ensuring accurate interpretation of the displayed content. To this end, the MediaPipe library, which has been specifically designed for this type of task, was employed.

MediaPipe [70] is an open-source platform that provides a set of tools and libraries with the objective of simplifying the incorporation of artificial intelligence and machine learning algorithms. In addition to providing pre-trained models, MediaPipe enables the customisation of neural networks to align with the specific requirements of a given application, thereby offering considerable flexibility.

A significant benefit of the MediaPipe platform is its capacity to facilitate the optimisation of processing pipelines for acceleration on central processing units (CPUs), graphics processing units (GPUs), and tensor processing units (TPUs), enabling real-time operation on a range of devices, including those with comparatively limited processing capabilities [71]. The tasks offered by MediaPipe include machine learning algorithms for object recognition, segmentation, classification, facial recognition and, of course, gesture recognition. The applications are extremely versatile and, given that they can be implemented on mobile devices such as smartphones and tablets, they ensure optimal performance even on hardware with limited processing power.

The gesture recognition task with MediaPipe [72] is capable of processing both static inputs, such as photographs, and continuous streams of data, such as video. At the frame level, the library enables the identification of gestures and the retrieval of the associated hand key points. This facilitates the association of a response with the received input, thereby ensuring a real-time response to the displayed results.

The library is capable of supporting a multitude of platforms and languages, including Python, Android, and web applications, thereby ensuring its versatility and facilitating its integration in a wide range of contexts. The operations performed by the model include image processing, such as rotation, normalisation, and lighting adjustment, as well as score filtering. The latter function enables the user to set an accuracy threshold, thereby filtering out results that do not exceed this value. Furthermore, the model provides a list of labels corresponding to the identified gestures.

The output of the network is not merely a list of detected gestures with their

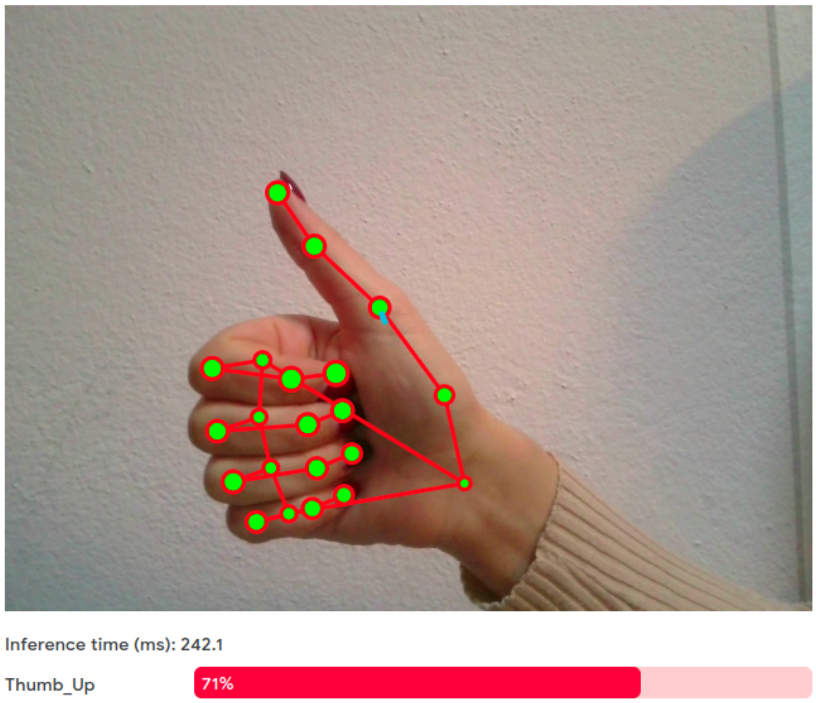


Figure 7.2: Thumb-up recognition.

corresponding accuracies; rather, it also includes information regarding the dominant hand and the reference points for each detected hand. This is provided with coordinates relative to two distinct reference systems: one based on the image itself and another with respect to the global reference system.

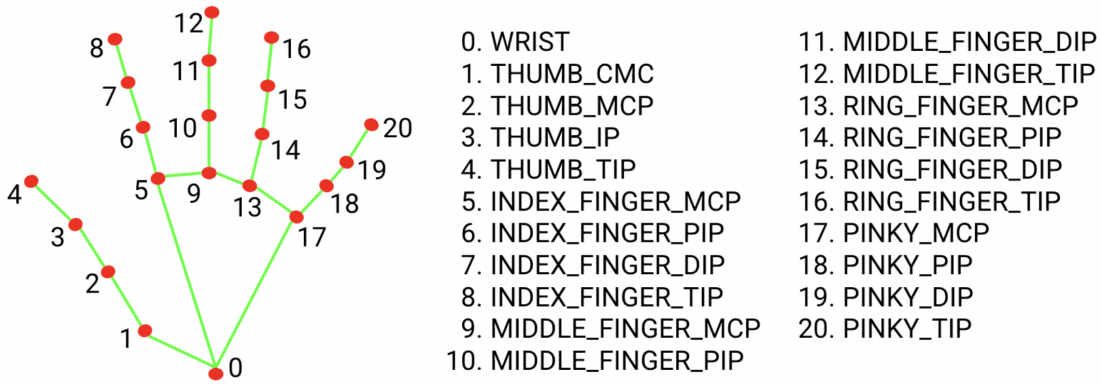


Figure 7.3: Hand key-points recognizable by the hand gesture recognition model [72]

The workflow of the model is divided into two distinct phases. The initial phase of the process pertains to the localisation of the hand within the frame. Once the hand has been identified, the localisation of the key points and the recognition of the gesture are conducted by processing only the area limited to the hand, thus avoiding to process the entire image. This approach is employed in order to reduce the computational load. In the case of video streams, to further optimise performance and accelerate processing, the hand is localised solely at the beginning of the process, rather than in each individual frame, unless there is an absence of identification within the initially calculated frame.

The gesture classification model [72] is comprised of two distinct pipelines. The initial step involves the embedding of gestures, which entails the analysis of the image and the generation of a vector of numerical values to assess the content depicted in the image.
Subsequently, the vector is transferred to the classification model, which is responsible for comparing the vectors of values with the predefined gestures and identifying the one represented.
The available default gestures are the following seven [72]:

```
0 - Unrecognized gesture, label: Unknown
1 - Closed fist, label: Closed_Fist
2 - Open palm, label: Open_Palm
3 - Pointing up, label: Pointing_Up
4 - Thumbs down, label: Thumb_Down
5 - Thumbs up, label: Thumb_Up
6 - Victory, label: Victory
7 - Love, label: ILoveYou
```

However, the model is customisable, and through the application of an appropriate training process, it is possible to incorporate additional gestures and make them available in accordance with the requirements of the context.

Chapter 8

User Interface and Communication with the Robot

The previous sections have presented a list of tools that, when used in combination, are capable of achieving the final goal of this thesis. Nevertheless, the interconnection of these components is contingent upon the data provided by the user. It is imperative that the latter is able to communicate with the robot in a clear and unambiguous manner, specifying with precision the exact requirements.

Indeed, communication represents the fundamental starting point of the entire project. In the absence of this input, the robot would be unable to receive the necessary data, thereby preventing the user from making effective use of the tool, which in turn would be unable to fulfil the requests.

While there are communication techniques that are more immediate and reflect human behaviour, such as voice communication, this solution has been found to have limitations. It is evident that voice communication necessitates the presence of sensors capable of detecting and interpreting audio. In this particular instance, the device lacks the requisite sensors, rendering this option inoperable. It is also important to emphasise that, in general, for mobile robot applications, audio communication is not the appropriate means of communication with the device. Indeed, in the event that the robot is situated at a distance from the user, it is unable to capture the voice input due to the distance.

It was thus decided to develop an Android application. This choice was based on the widespread use of mobile devices globally, with almost everyone owning at least

one smartphone or tablet and being familiar with their basic use. The application not only allows the user to communicate, but also to create a specific workflow, ordering the sequence of actions to be performed by the robot. The structure, workflow and interface layout of the application will be described in detail later.

8.1 WorkFlow

The operational flow that governs the robot's behavior is visually represented in Figure 8.1.

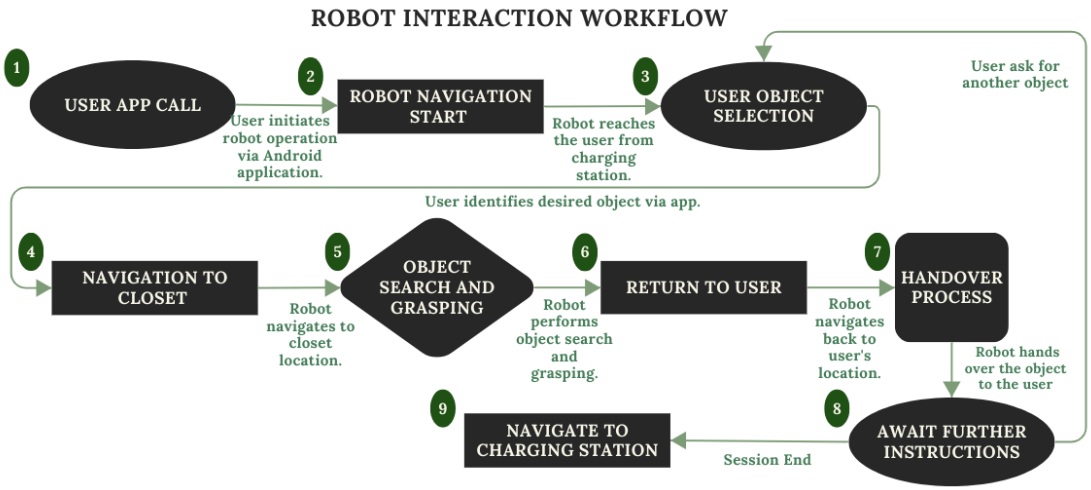


Figure 8.1: Robot interaction workflow

The process begins when the user requires the robot's assistance by means of a dedicated Android application. The application provides a user-friendly interface that allows the user to quickly communicate his/her needs to the robot. Once activated, the robot autonomously navigates to the user's location and awaits further instructions. Through the app, the user can specify the object they wish to retrieve, selecting from a range of predefined options that the robot is capable of handling.
At this stage, the robot proceeds to the designated storage location, which may be a closet or storage cabinet, where the object is stored. At this phase of the process, the robot deploys devoted object detection algorithms to identify the item of interest. Thereafter, it employs its robotic arm to grasp and retrieve the object. Once the object has been successfully grasped, the robot then navigates back to the user's location, maintaining spatial awareness to avoid obstacles or dynamic elements in the environment, such as other people.

Upon reaching the user, the robot initiates the handover process, carefully delivering the object to the user in a manner that prioritizes the ease of use.
Following the successful completion of the handover, the user has the option to either request an additional object, repeating the entire process, or to dismiss the robot. In the latter case, the robot will autonomously navigate back to its charging station, where it will enter standby mode, awaiting further instructions from the user.

8.2 App Layout

The application is structured into three principal screens, each of which provides the user with guidance on the action to be performed by the robot. Prior to the aforementioned screens, users are presented with an initial interface displaying the name of the application and a brief description of the project's objective (Figure 8.2).

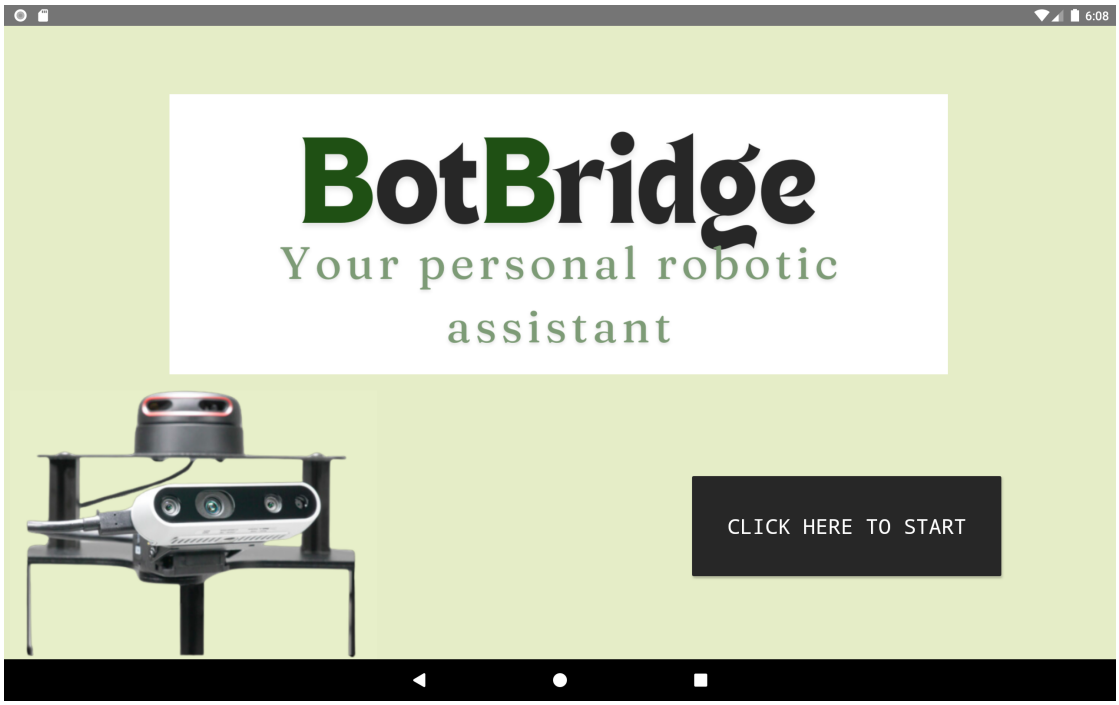


Figure 8.2: Initial interface.

Once the `"Click here to start"` button is pressed, the user is directed to the second screen, which is designed to capture the robot's attention. In many instances where the robot is not required to perform a specific task, it is left in its charging

station. This is done to ensure that its performance is well-preserved for the moment it is needed. The station is typically situated in an area with minimal user traffic, in order to avoid any obstacles to daily activities. However, given that the station is located in a less accessible area, the user can request the robot's presence by pressing the `"Hi robot"` button (Figure 8.3). This allows the robot to approach and activate its functionality.

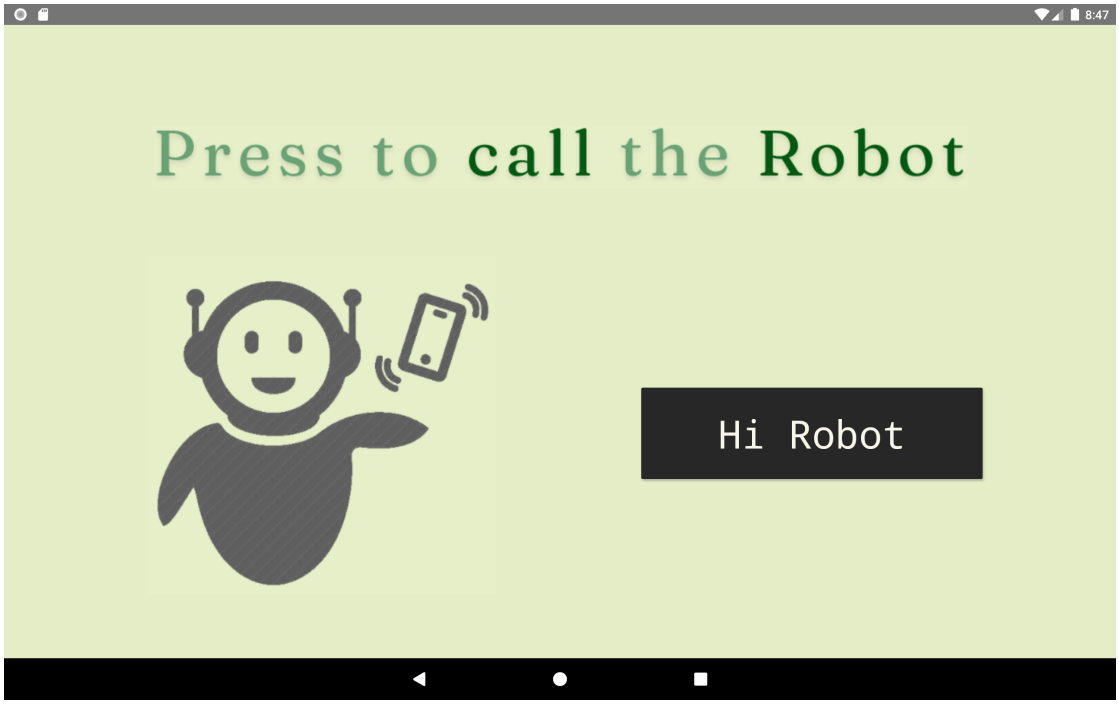


Figure 8.3: Call the robot

Upon calling the robot, the user is redirected to the third screen, which presents a menu of objects for selection. Upon reaching the user, the robot awaits further instructions. On this screen, the user can instruct the robot on the next task to be performed. By selecting the desired object from the list and pressing the corresponding button (Figure 8.4), the robot will activate to retrieve it and deliver it to the user.

Subsequently, the user is directed to the fourth and final screen, which is intended to conclude the interaction cycle with the robot (Figure 8.5). This screen becomes active once the robot has successfully completed all required operations, including the final handover action, that is to say, the physical delivery of the selected object into the user's hands. At this juncture, the interface presents two

Figure 8.4: List of All the object available

options for the user to select from, thereby enabling them to determine the optimal course of action according to their specific requirements.

In the event that the user requires additional objects, the first option becomes available. It is important to note that the robot, which is equipped with a single end-effector, i.e., a single arm or gripping mechanism, is only capable of transporting a single object at a time. It is therefore not feasible to ask the robot to transport multiple objects simultaneously. Should the user require another object, the `"Choose another item"` option has to be selected. This option will redirect the user to the preceding screen, which is the object menu, thus enabling the selection of a new item for the robot to retrieve. This process may be repeated as necessary until the user has obtained all the required items.

The second option is intended for users who have completed their requests and are satisfied with the robot's performance. In this instance, the user is able to select the option designated as `"I'm all set"`, which serves to indicate to the robot that no further action is required. Upon selection of this option, the robot will automatically return to its charging station. Even when the robot returns to its base, it remains active and listening, thus ensuring that it is able to respond to any

new requests that may arise in the future. This behaviour is essential to guarantee that the robot is perpetually available and prepared for action, while sustaining its optimal performance through the charging process when not in use.

This final stage of interaction serves to complete the robot's work cycle, while simultaneously ensuring a seamless and uninterrupted experience for the user. The user is afforded the option to conclude the interaction or prolong it, in a straightforward and intuitive manner, without the inconvenience of interruptions or technical complications. The transparent interface and clearly delineated options provide users with a substantial degree of control and customisation, thereby facilitating efficient and accessible operation of the robot.

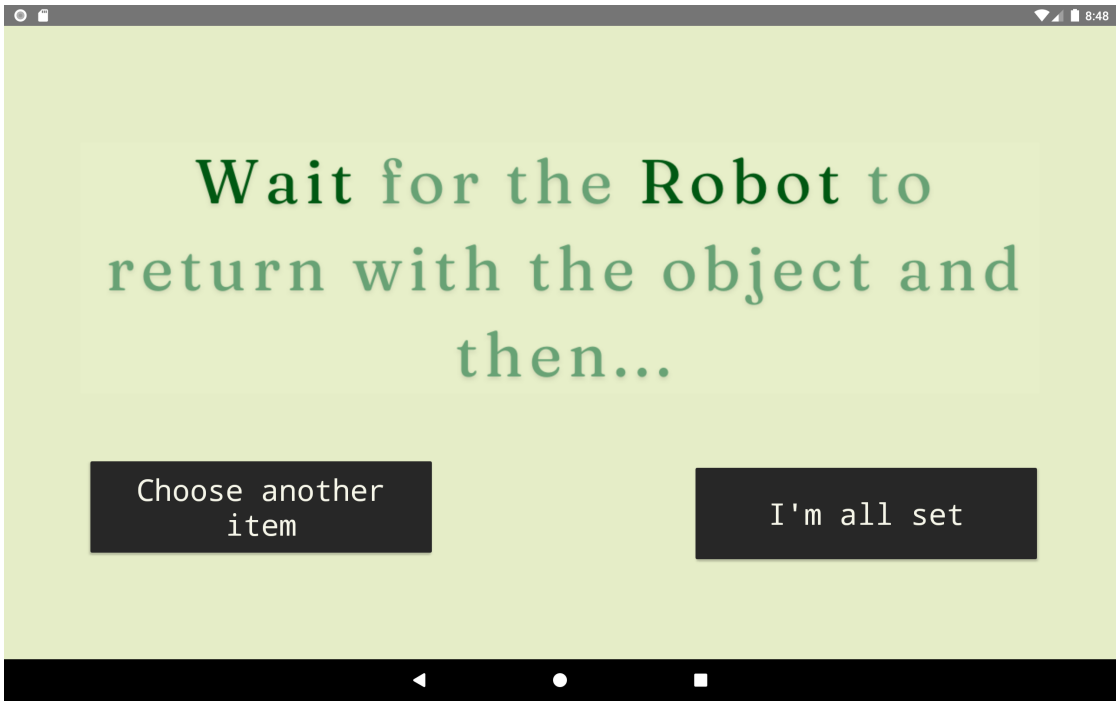


Figure 8.5: Concluding screen

8.3 Android Studio

This Android application was developed using **Android Studio** [73], which serves as the reference integrated development environment (IDE) for Android app development. Android Studio provides developers with a comprehensive suite of sophisticated tools that leverage the capabilities of IntelliJ IDEA.

One of the most significant features is the flexible Gradle-based build system, which facilitates advanced and bespoke project management. Gradle enables the configuration of the entire build process, from dependency management to the creation of different variants of the application, which can be adapted to different configurations, devices and Android versions.

Another noteworthy advantage of Android Studio is the rapid, comprehensive emulator, which enables the assessment of applications on a diverse assortment of virtual devices. The emulator accurately replicates the hardware characteristics of smartphones and tablets, and incorporates sophisticated capabilities such as network condition simulation, GPS event generation, and sensor emulation. The accelerated processing speed and enhanced efficiency of the system enable developers to reduce the time required for testing and to promptly identify any anomalies or bugs.

Furthermore, Android Studio provides a unified environment for the development of applications for all Android-powered devices, including smartphones, tablets, smartwatches, televisions, and automobiles. This enables the code to be written once and readily adapted to multiple platforms, thereby optimising the workflow.

Furthermore, Android Studio incorporates predefined code templates and a robust integration with GitHub, which streamline the development of common app features, such as layouts and User Interfaces (UI) components, thereby accelerating the development process. Furthermore, Android Studio facilitates the development of applications in C++ through the use of the NDK (Native Development Kit). This enables the creation of specific components within the application in native code, which is optimal for enhancing performance in scenarios where high-level efficiency is paramount, such as in gaming or graphical applications.

8.3.1 Kotlin

At the 2019 Google I/O conference, it was revealed that the advancement of Android projects will be gradually transitioning towards the utilisation of the **Kotlin** programming language [74]. Consequently, Android Studio has been designated as **"Kotlin-first"**. This signifies that the main support of the IDE, including libraries, examples and content, will be developed primarily in Kotlin, while maintaining compatibility with Java, which remains supported.

From the perspective of the developer, there are a number of advantages associated with the use of Kotlin. The language is characterised by a clear and concise syntax, and its interoperability with Java facilitates seamless integration into existing projects that were originally developed in Java. This permits a gradual

evolution of the code without the necessity for its complete rewrite.

The data collected indicated that 67% of Android developers reported an increase in productivity when using Kotlin. This is due to the fact that Kotlin offers a number of advanced features that enhance the security of the language, thereby reducing the probability of encountering common programming errors. Furthermore, applications developed in Kotlin are 20% less prone to crashing than those developed in other languages, thereby enhancing the overall stability of the software.

8.4 MQTT

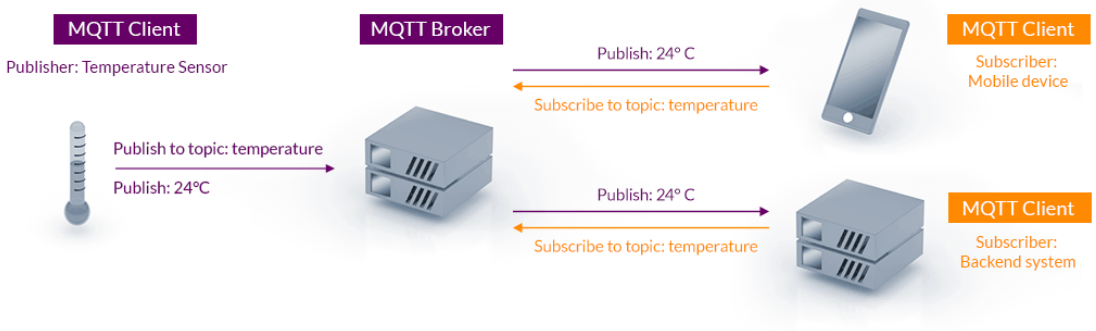


Figure 8.6: MQTT publisher and subscriber architecture [75].

Communication between the Android application and the Locobot is achieved through the utilisation of the **MQTT communication protocol**. MQTT (Message Queuing Telemetry Transport) [76] is an OASIS standard messaging protocol based on a client/server strategy that employs a publish/subscribe message transmission system. The transmission of messages is primarily conducted over TCP/IP channels. However, the protocol also supports other two-way network protocols, which ensure the maintenance of orderly and lossless connections.

The MQTT protocol is particularly well-suited to applications with limited resources, such as communication between machines (M2M) and within the Internet of Things (IoT), where the consumption of resources, both in terms of code and bandwidth, must be minimal. The aforementioned characteristics render it an optimal choice for industrial contexts, including manufacturing, automotive, and smart home solutions, due to its lightweight nature, open-source foundation, and straightforward implementation.

One of the most distinctive characteristics of the MQTT protocol is its publisher/subscriber strategy, which facilitates one-to-many publishing and decouples applications. Furthermore, the transmission of messages is independent of the data payload, as the protocol does not interpret or modify the transmitted data. This implies that the payload can comprise any type of data (text, binary, etc.), thereby ensuring flexibility, simplicity and enhanced independence between system components.

The MQTT protocol [76] offers three quality of service (QoS) levels for the administration of message delivery, each with a distinct degree of reliability:

- **QoS 0 - "At Most Once":** represents the least reliable level, offering no guarantee of delivery. Messages are sent without any assurance of successful transmission, relying on the inherent limitations of the operating environment. It is therefore possible that messages may be lost. This level of service is appropriate for scenarios where the loss of a message is not a significant issue. For instance, in the case of environmental sensor data, any missing readings can be promptly replaced by new ones, thus mitigating the impact of potential message loss;
- **QoS 1 - "At Least Once"**, guarantees the delivery of messages but does not preclude the possibility of the same message being received multiple times. This level is applicable in contexts where it is imperative that the message is delivered, while acknowledging the possibility of duplicates, which necessitate management at the application level;
- **QoS 2 – "Exactly Once":** this level of quality of service ensures that each message is delivered only once, without any duplication or loss. This level is the most reliable and is frequently employed in contexts where the consequences of a delivery error could be financially or operationally significant, such as in billing applications.

The aforementioned QoS levels provide the flexibility to adapt the MQTT protocol to varying requirements in terms of reliability and efficiency.

The MQTT protocol is comprised of two principal components: the MQTT Client and the MQTT Broker. The following sections describe these components in detail.

MQTT Client

MQTT Clients [77] are essentially analogous to traditional executable files, comparable to the nodes utilized in ROS. The clients are capable of acting as both publishers and subscribers, contingent upon the necessity to publish or receive

information, respectively. It should be noted that the clients do not communicate directly with each other; rather, they connect to the MQTT Broker, which acts as a switch to ensure the correct routing and delivery of messages to the appropriate clients.

MQTT Clients can be developed using a variety of programming languages, including C#, C++, Java, Websockets, and Python, although other options may also be available. This flexibility enables developers to select the most appropriate language in accordance with the specific environment or system in which they intend to operate.

MQTT Broker

The function of the MQTT Broker [77] is to act as an actual intermediary between the disparate clients. A node's subscription to the broker implies its assent to the communication from specific topic, which may be considered as thematic channel of communication. The primary function of the broker is to receive messages from multiple publishers and categorize them according to their associated topics, subsequently forwarding them to the intended subscribers. In this manner, the broker guarantees that each message is conveyed solely to the pertinent clients, thereby maintaining an efficacious and systematic communication network among all participants.

The most well-known broker and one that has been utilised in the development of this project is **Mosquitto** [78], an open-source MQTT broker developed by Eclipse. Mosquitto is widely used due to its user-friendly interface and the fact that it can be installed on low-power Linux servers, such as Raspberry Pi or Mini PCs, as is the case under consideration. This software is particularly lightweight and offers good performance, requiring approximately 3 MB of RAM to manage up to 1000 connected clients. These features make it an ideal solution for home automation applications and other contexts requiring efficiency and low resource consumption.

Chapter 9

Experimental Results

In order to illustrate the techniques described in the preceding chapters, a video demonstration is provided here [79]. The demonstration displays the Locobot's behaviour in a series of tasks, illustrating its ability to complete each target. It begins with the Locobot awaiting charging at the charging point. In the bottom left of the screen, the Android application serves as the user interface (see Figure 9.1).

Figure 9.1: Starting situation with the robot at the charging station and the initial screen of the application in the bottom left

The user starts the application and calls the robot through the designated button

(see Figure 8.3), resulting in the robot's movement.
The robot is initially confronted with the challenge of navigating around a person in its path. The solution, which is outlined in Chapter 5, was found to be effective in practice by modifying the local path and computing a trajectory that allowed the robot to avoid the moving agent while maintaining the safety of the situation. Upon reaching the user station, the Locobot awaits a command from the user. The user is then able to request the desired object via the application. In the example illustrated in the video, the user commands the robot to retrieve the scissors (see Figure 9.2).

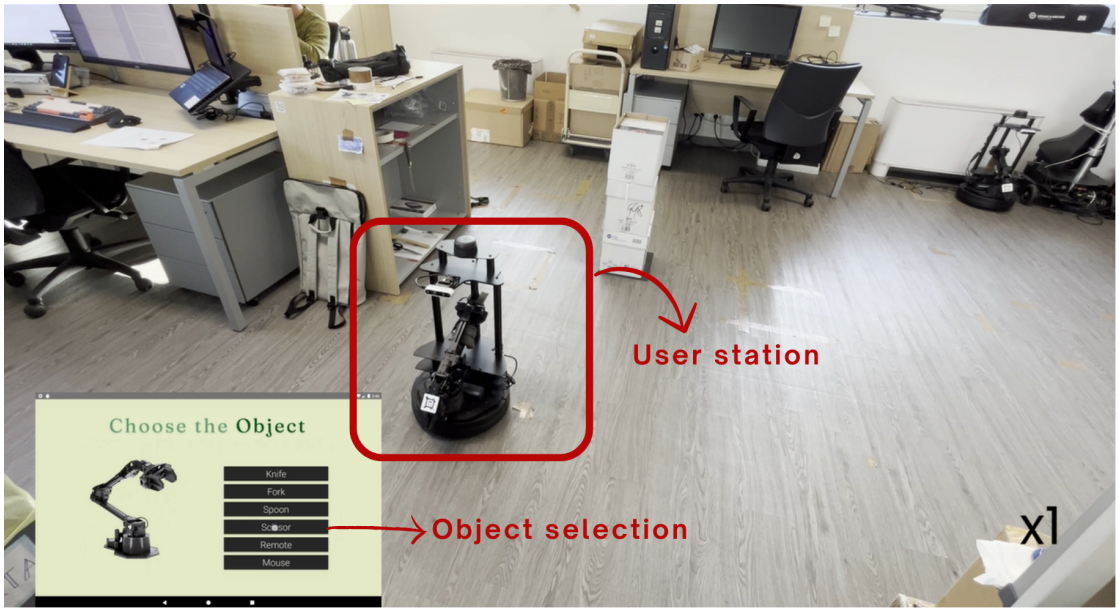


Figure 9.2: The Locobot is waiting at the user station and the user requests the scissors though the list provided by the application

Once the message has arrived, the Locobot starts to move towards the cabinet. The arrival at the destination marks the beginning of the search for the object. At the initial stage of the process, the Locobot searches for the scissors that are placed in a suspended position. Upon determining that the desired object was not present, the camera was tilted to scan the bottom of the cabinet. The robot's point of view is displayed as a pop-up window on the screen, which shows the identification of the object in the cabinet, the relative masks, and the grasping point of the scissors as a green point (see Figure 9.3), as explained in Section 6.2.

Once the scissors are grasped, the robot positions them in a predefined manner, thereby ensuring safe navigation and enabling the robot to reach the user station and proceed with the handover maneuver, as highlighted in Figure 9.4.

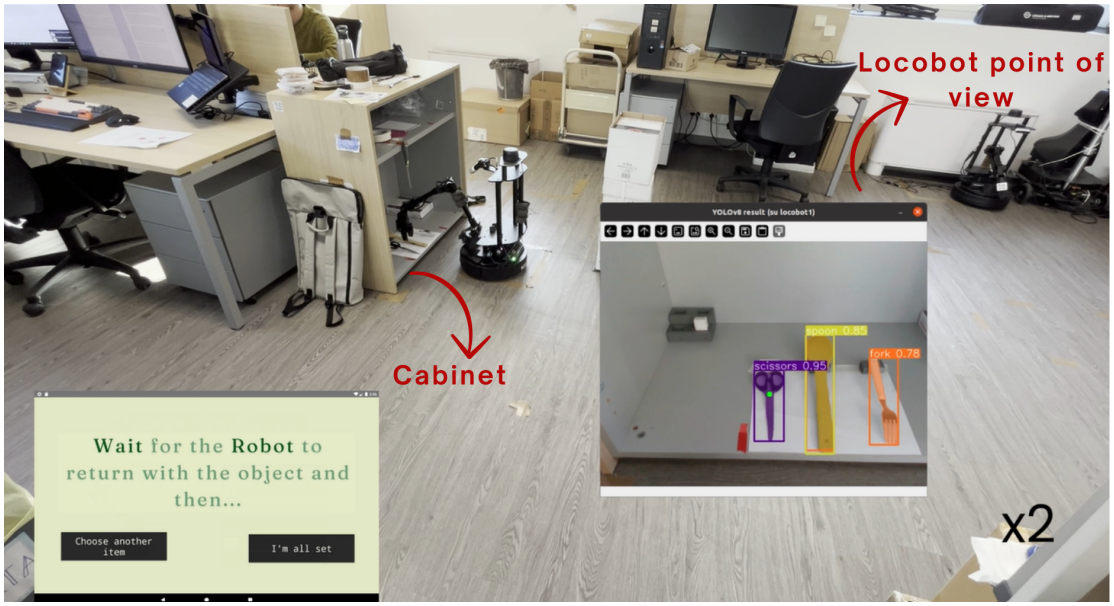


Figure 9.3: Grasping maneuver with the robot point of view and the image processing

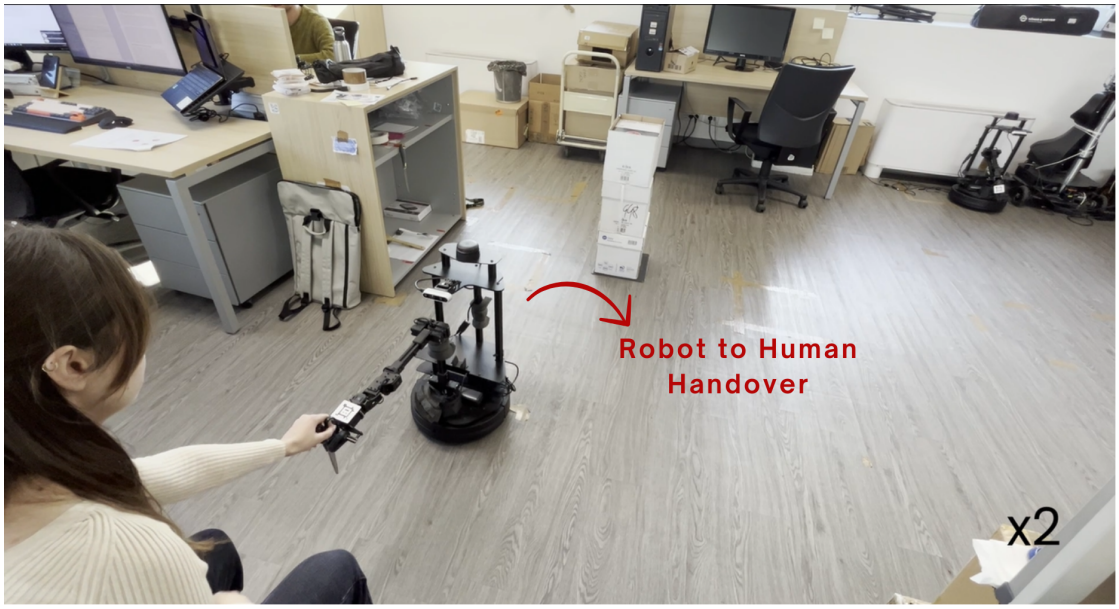


Figure 9.4: The Locobot comes back to the user and proceed with the handover maneuver

The video demonstration concludes with the final command issued by the user. Once the requested object has been successfully received, the Locobot is no longer required and can therefore be terminated by selecting the `"I'm all set"` option in the interface (see Figure 8.5). Subsequently, the robot is permitted to return to the charging station, where it will await further instructions.

Chapter 10

Conclusions and Future Works

The objective of the thesis project was to develop a framework capable of safe object manipulation in a shared environment with humans. This objective was achieved through a series of phases, during which advanced strategies for social navigation, object manipulation and user interaction were developed.

At the beginning of the project, a strategy for **social navigation** was designed and implemented, thereby enabling the robot to operate safely and efficiently in environments shared with people. The utilisation of data from video cameras and its processing by machine learning algorithms (in this specific case, YOLOv8 for tracking people) enabled the system to identify and track the position of people, who were considered as potential obstacles, thus ensuring their avoidance in real time. This component is of significant importance in order to ensure safe interaction between the robot and people in the environment.

Subsequently, the component responsible for **object manipulation** was developed. Once more, the utilisation of sophisticated machine learning methodologies enabled the development of a grasping system with the capacity to grasp objects in both vertical and horizontal resting positions on surfaces. The system was designed to manipulate a range of objects, from small items such as kitchen cutlery, scissors and a mouse, to larger items such as a remote control. However, the maximum weight of the objects the robot was capable of lifting imposed certain limitations, which restricted the scope of the project to light and small objects.

A further crucial aspect of the project was the creation of a standardised **handover system**, with the objective of ensuring safe and predictable interaction between the

robot and the user. In light of the hardware limitations, including the absence of sophisticated haptic feedback sensors, a strategy was employed that promoted the standardisation of the robot's movements, thereby minimising the risk of unanticipated actions. This enabled the user to anticipate the robot's movements, thereby facilitating the maintenance of a high level of safety during interaction.

At this point in the project, a **communication interface** was developed with the objective of facilitating interaction between the user and the robot. A dedicated Android application enables users to issue commands via a TCP/IP connection, thereby controlling the robot's operational sequence. The interface has been designed with intuitive and user-friendly functionality, allowing users with varying degrees of technological expertise to interact with the system with ease.

In consideration of the totality of the work performed, it can be concluded that a significant contribution was made to the development of a safe and intuitive robotic system for the handling of objects in shared environments with people.
Despite the presence of certain limitations, primarily associated with hardware constraints and the robot's capacity, the outcomes achieved provide a robust foundation for future enhancements and expansions of the project.

In terms of communication with the robot, it is possible to integrate supplementary functionalities that enhance the user experience, thereby rendering it more comprehensive and intuitive. One potential approach to enhance the user experience would be to implement a response system on the robot, enabling the user to receive immediate feedback. Moreover, the utilisation of the sensors intrinsic to the mobile device could facilitate the development of a voice command system, thereby enabling a more natural and immediate mode of interaction. Concurrently, integration with vision systems via the robot's cameras could facilitate advanced interface management based on visual recognition for control and interaction.

From the perspective of object manipulation, a possible improvement could be achieved by training the neural network on a more extensive and diverse set of objects. This would enable the system to recognise and manipulate a greater number of objects, thereby expanding the robot's operational capabilities. The utilisation of novel machine learning techniques or the integration of more sophisticated tactile sensors could further enhance the accuracy and reliability in gripping objects of varying shapes and sizes.

Bibliography

[1] Cory-Ann Smarr, Cara Bailey Fausset, and Wendy A Rogers. «Understanding the potential for robot assistance for older adults in the home environment». In: *Georgia Institute of Technology* (2011).

[2] Luca Ragno, Alberto Borboni, Federica Vannetti, Cinzia Amici, and Nicoletta Cusano. «Application of Social Robots in Healthcare: Review on Characteristics, Requirements, Technical Solutions». In: *Sensors* 23.15 (2023). ISSN: 1424-8220. DOI: 10.3390/s23156820. URL: https://www.mdpi.com/1424-8220/23/15/6820.

[3] Christina Schmidbauer, Setareh Zafari, Bernd Hader, and Sebastian Schlund. «An Empirical Study on Workers' Preferences in Human–Robot Task Assignment in Industrial Assembly Systems». In: *IEEE Transactions on Human-Machine Systems* 53.2 (2023), pp. 293–302. DOI: 10.1109/THMS.2022.3230667.

[4] Fabian Falck et al. «Robot DE NIRO: A Human-Centered, Autonomous, Mobile Research Platform for Cognitively-Enhanced Manipulation». In: *Frontiers in Robotics and AI* 7 (2020). ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00066. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.00066.

[5] Nivedita Sethiya and Chandresh Kumar Maurya. *End-to-End Speech-to-Text Translation: A Survey*. 2024. arXiv: 2312.01053 [cs.CL]. URL: https://arxiv.org/abs/2312.01053.

[6] eSpeak. *eSpeak: Speech Synthesizer*. https://espeak.sourceforge.net/. [First Access: 07-03-2024].

[7] Yuxuan Wang et al. «Tacotron: Towards End-to-End Speech Synthesis». In: *Interspeech 2017*. Vol. 2017-August. Interspeech 2017. ISCA, 2017, pp. 4006–4010.

[8] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. «Waveglow: A Flow-based Generative Network for Speech Synthesis». In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3617–3621. DOI: 10.1109/ICASSP.2019.8683143.

[9] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, Ming Liu, and Ming Zhou. *Neural Speech Synthesis with Transformer Network*. 2019. arXiv: 1809.08895 [cs.CL].

[10] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. *FastSpeech 2: Fast and High-Quality End-to-End Text to Speech*. 2022. arXiv: 2006.04558 [eess.AS].

[11] M.S. Ghute, K.P. Kamble, and M. Korde. «Design of military surveillance robot». In: *Proceedings of the IEEE First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. Jalandhar, India, Dec. 2018, pp. 270–272.

[12] Eduard Clotet, Dani Martínez, Javier Moreno, Marcel Tresanchez, and Jordi Palacín. «Assistant Personal Robot (APR): Conception and Application of a Tele-Operated Assisted Living Robot». In: *Sensors* 16.5 (2016). ISSN: 1424-8220. DOI: 10.3390/s16050610. URL: https://www.mdpi.com/1424-8220/16/5/610.

[13] Jing Qi, Li Ma, Zhenchao Cui, and Yushu Yu. «Computer vision-based hand gesture recognition for human-robot interaction: a review». In: *Complex & Intelligent Systems* 10.1 (2024), pp. 1581–1606. DOI: 10.1007/s40747-023-01173-6. URL: https://doi.org/10.1007/s40747-023-01173-6.

[14] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. *Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks*. 2019. arXiv: 1901.10323 [cs.CV]. URL: https://arxiv.org/abs/1901.10323.

[15] Alexander Bigalke and Mattias P. Heinrich. «Fusing Posture and Position Representations for Point Cloud-Based Hand Gesture Recognition». In: *2021 International Conference on 3D Vision (3DV)*. 2021, pp. 617–626. DOI: 10.1109/3DV53792.2021.00071.

[16] Jinfu Liu, Xinshun Wang, Can Wang, Yuan Gao, and Mengyuan Liu. «Temporal Decoupling Graph Convolutional Network for Skeleton-Based Gesture Recognition». In: *IEEE Transactions on Multimedia* 26 (2024), pp. 811–823. DOI: 10.1109/TMM.2023.3271811.

[17] Anthony Francis et al. *Principles and Guidelines for Evaluating Social Robot Navigation Algorithms*. 2023. arXiv: 2306.16740 [cs.RO]. URL: https://arxiv.org/abs/2306.16740.

[18] Shuijing Liu, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. «Decentralized Structural-RNN for Robot Crowd Navigation with Deep Reinforcement Learning». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 3517–3524.

[19] Shuijing Liu, Peixin Chang, Zhe Huang, Neeloy Chakraborty, Kaiwen Hong, Weihang Liang, D. Livingston McPherson, Junyi Geng, and Katherine Driggs-Campbell. «Intention Aware Robot Crowd Navigation with Attention-Based Interaction Graph». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 12015–12021.

[20] Phani Teja Singamaneni, Anthony Favier, and Rachid Alami. «Human-Aware Navigation Planner for Diverse Human-Robot Ineraction Contexts». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[21] Phani Teja Singamaneni and Rachid Alami. «HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation». In: *International Conference on Robot & Human Interactive Communication*. 2020. DOI: 10.1109/RO-MAN47096.2020.9223463.

[22] Rhys Newbury et al. *Deep Learning Approaches to Grasp Synthesis: A Review*. 2023. arXiv: 2207.02556 [cs.RO]. URL: https://arxiv.org/abs/2207.02556.

[23] Kevin Kleeberger, Richard Bormann, Wolfgang Kraus, and Tamim Asfour. «A Survey on Learning-Based Robotic Grasping». In: *Current Robotics Reports* 1.4 (2020), pp. 239–249. DOI: 10.1007/s43154-020-00021-6. URL: https://doi.org/10.1007/s43154-020-00021-6.

[24] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. *Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes*. 2021. arXiv: 2103.14127 [cs.RO]. URL: https://arxiv.org/abs/2103.14127.

[25] Alessandro Palleschi, Franco Angelini, Chiara Gabellieri, Do Won Park, Lucia Pallottino, Antonio Bicchi, and Manolo Garabini. «Grasp It Like a Pro 2.0: A Data-Driven Approach Exploiting Basic Shape Decomposition and Human Data for Grasping Unknown Objects». In: *IEEE Transactions on Robotics* 39.5 (2023), pp. 4016–4036. DOI: 10.1109/TRO.2023.3286115.

[26] Luca Bergamini, Mario Sposato, Marcello Pellicciari, Margherita Peruzzini, Simone Calderara, and Juliana Schmidt. «Deep learning-based method for vision-guided robotic grasping of unknown objects». In: *Advanced Engineering Informatics* 44 (2020), p. 101052. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2020.101052. URL: https://www.sciencedirect.com/science/article/pii/S1474034620300215.

[27] Ma Haoxiang and Di Huang. «Towards Scale Balanced 6-DoF Grasp Detection in Cluttered Scenes». In: *Conference on Robot Learning (CoRL)*. 2022.

[28] Haonan Duan, Yifan Yang, Daheng Li, and Peng Wang. «Human–robot object handover: Recent progress and future direction». In: *Biomimetic Intelligence and Robotics* 4.1 (2024), p. 100145. ISSN: 2667-3797. DOI: https://doi.org/10.1016/j.birob.2024.100145. URL: https://www.sciencedirect.com/science/article/pii/S2667379724000032.

[29] Daniel Lehotsky, Albert Christensen, and Dimitrios Chrysostomou. «Optimizing Robot-to-Human Object Handovers using Vision-based Affordance Information». In: *2023 IEEE International Conference on Imaging Systems and Techniques (IST)*. 2023, pp. 1–6. DOI: 10.1109/IST59124.2023.10355704.

[30] Albert D Christensen, Daniel Lehotsky, Marius W Jørgensen, and Dimitris Chrysostomou. «Learning to Segment Object Affordances on Synthetic Data for Task-oriented Robotic Handovers». In: *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, 2022. URL: https://bmvc2022.mpi-inf.mpg.de/0544.pdf.

[31] Francesco Iori, Gojko Perovic, Francesca Cini, Angela Mazzeo, Egidio Falotico, and Marco Controzzi. «DMP-Based Reactive Robot-to-Human Handover in Perturbed Scenarios». In: *International Journal of Social Robotics* 15.2 (Feb. 2023), pp. 233–248. ISSN: 1875-4805. DOI: 10.1007/s12369-022-00960-4. URL: https://doi.org/10.1007/s12369-022-00960-4.

[32] Spiceworks. *What is Robot Operating System (ROS)?* Accessed: 2024-09-29. 2024. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-robot-operating-system/.

[33] Wikipedia contributors. *Robot Operating System — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-September-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Robot_Operating_System&oldid=1241894721.

[34] ROS Community. *ROS/Introduction - ROS Wiki*. https://wiki.ros.org/ROS/Introduction. Accessed: 2024-09-25. 2024.

[35] Robert Troj. *Hands-on Introduction to Robot Operating System (ROS)*. Accessed: 2024-09-29. 2024. URL: https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system(ros)/.

[36] ROS Wiki. *Understanding ROS Nodes*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/ROS/Tutorials/UnderstandingNodes.

[37] ROS Wiki. *Writing a Simple Publisher and Subscriber (C++)*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29.

[38] ROS Wiki. *ROS Concepts*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/ROS/Concepts.

[39] ROS Wiki. *Creating a ROS msg and srv*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv.

[40] ROS Wiki. *ROS Master*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/Master.

[41] ROS Wiki. *Parameter Server*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/Parameter%20Server.

[42] ROS Wiki. *ROS Packages*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/Packages.

[43] ROS Wiki. *catkin/CMakeLists.txt*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/catkin/CMakeLists.txt.

[44] ROS Wiki. *catkin/package.xml*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/catkin/package.xml.

[45] Pranathi Meegada. *Getting Started with RViz: A Beginner's Guide to ROS Visualization*. Accessed: 2024-09-29. 2024. URL: https://medium.com/@pranathi.meegada/getting-started-with-rviz-a-beginners-guide-to-ros-visualization-2e5f4156e410.

[46] ROS Wiki. *URDF XML: Model*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/urdf/XML/model.

[47] ROS Wiki. *SRDF (Semantic Robot Description Format)*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/srdf.

[48] CSE-468/568 Robotic Algorithms. *Object Recognition and Object Location Using UR5e and Kinect V2 (Part 2/3)*. Accessed: 2024-09-29. 2024. URL: https://medium.com/cse-468-568-robotic-algorithms/object-recognition-and-object-location-using-ur5e-and-kinect-v2-part-2-3-771f1f4f6252.

[49] ROS Wiki. *tf - ROS Wiki*. Accessed: 2024-09-29. 2024. URL: http://wiki.ros.org/tf.

[50] Industrial Training Master. *Using rqt Tools for Analysis*. Accessed: 2024-09-29. 2024. URL: https://industrial-training-master.readthedocs.io/en/latest/_source/session6/Using-rqt-tools-for-analysis.html.

[51] Gazebo Classic. *Guided Tutorial: Beginner Level 1*. Accessed: 2024-09-29. 2024. URL: https://classic.gazebosim.org/tutorials?tut=guided_b1.

[52] Mansoor Alam. *What is Mapping in Robotics? How to Create Map in ROS*. Accessed: 2024-09-29. 2024. URL: https://medium.com/@mansooralam129047/what-is-mapping-in-robotics-how-to-create-map-in-ros-8c002d409c07.

[53] ROS Wiki. *gmapping - ROS Wiki*. Accessed: 2024-09-29. 2024. URL: http://wiki.ros.org/gmapping.

[54] Trossen Robotics. *Interbotix XSeries Locobots - Navigation Stack Configuration*. Accessed: 2024-09-29. 2024. URL: https://docs.trossenrobotics.com/interbotix_xslocobots_docs/ros1_packages/navigation_stack_configuration.html.

[55] IntRoLab. *RTAB-Map (Real-Time Appearance-Based Mapping)*. Accessed: 2024-09-29. 2024. URL: https://introlab.github.io/rtabmap/.

[56] OctoMap. *OctoMap 3D Mapping Framework*. Accessed: 2024-09-29. 2024. URL: https://octomap.github.io/.

[57] ROS Wiki. *Robot Setup for Navigation*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/navigation/Tutorials/RobotSetup.

[58] ROS Wiki. *move_base - ROS Wiki*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/move_base.

[59] MoveIt. *MoveIt - Motion Planning Framework for ROS*. Accessed: 2024-09-29. 2024. URL: https://moveit.ai/.

[60] Trossen Robotics. *Interbotix XSLocoBot Specifications*. Accessed: 2024-09-29. 2024. URL: https://docs.trossenrobotics.com/interbotix_xslocobots_docs/specifications.html#.

[61] Trossen Robotics. *Interbotix WX250S Specifications*. Accessed: 2024-09-29. 2024. URL: https://docs.trossenrobotics.com/interbotix_xsarms_docs/specifications/wx250s.html.

[62] Intel RealSense. *Intel RealSense Depth Camera D435*. Accessed: 2024-09-29. 2024. URL: https://www.intelrealsense.com/depth-camera-d435/.

[63] ROS Wiki. *social_navigation_layers - ROS Wiki*. Accessed: 2024-09-29. 2024. URL: https://wiki.ros.org/social_navigation_layers.

[64] Ultralytics. *YOLOv8 Models - Ultralytics Documentation*. Accessed: 2024-09-29. 2024. URL: https://docs.ultralytics.com/models/yolov8/.

[65] Ultralytics. *COCO Pretrained Models - Ultralytics Documentation*. Accessed: 2024-09-29. 2024. URL: https://docs.ultralytics.com/datasets/detect/coco/#coco-pretrained-models.

[66] Ultralytics. *Tracking Features - Ultralytics Documentation*. Accessed: 2024-09-29. 2024. URL: https://docs.ultralytics.com/modes/track/#features-at-a-glance.

[67] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2022. arXiv: 2110.06864 [cs.CV]. URL: https://arxiv.org/abs/2110.06864.

[68] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. *BoT-SORT: Robust Associations Multi-Pedestrian Tracking*. 2022. arXiv: 2206.14651 [cs.CV]. URL: https://arxiv.org/abs/2206.14651.

[69] Ultralytics. *Segmentation Task - Ultralytics Documentation*. Accessed: 2024-09-29. 2024. URL: https://docs.ultralytics.com/it/tasks/segment/.

[70] Google AI. *MediaPipe Solutions Guide*. Accessed: 2024-09-29. 2024. URL: https://ai.google.dev/edge/mediapipe/solutions/guide?hl=it.

[71] Google AI. *MediaPipe Solutions Tasks*. Accessed: 2024-09-29. 2024. URL: https://ai.google.dev/edge/mediapipe/solutions/tasks?hl=it.

[72] Google AI. *MediaPipe Vision: Gesture Recognizer*. Accessed: 2024-09-29. 2024. URL: https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer?hl=it.

[73] Android Developers. *Introduzione ad Android Studio*. Accessed: 2024-09-29. 2024. URL: https://developer.android.com/studio/intro?hl=it.

[74] Android Developers. *Your first Kotlin program on Android*. Accessed: 2024-09-29. 2024. URL: https://developer.android.com/kotlin/first.

[75] MQTT.org. *MQTT - The Standard for IoT Messaging*. Accessed: 2024-09-29. 2024. URL: https://mqtt.org/.

[76] OASIS. *MQTT Version 3.1.1*. Accessed: 2024-09-29. 2024. URL: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[77] HiveMQ. *How to Get Started with MQTT*. Accessed: 2024-09-29. 2024. URL: https://www.hivemq.com/blog/how-to-get-started-with-mqtt/.

[78] Indomus. *Broker MQTT Eclipse Mosquitto*. Accessed: 2024-09-29. 2024. URL: https://indomus.it/componenti/broker-mqtt-eclipse-mosquitto/.

[79] Angela Ripi. *Development of a Human-Centered Framework for Safe Object Handling with Mobile Manipulators*. 2024. URL: https://www.youtube.com/watch?v=wsEJ8mzYE2Q.