Capstone Project I Report

Human Avoidance by Mobile Platform

Dinmukhamet Murat, Aina Shilikbay

A thesis submitted in part fulfilment of the degree of

BS in Robotics Engineering

Supervisor: Matteo Rubagotti

Instructor: Anara Sandygulova

Department of Robotics and Mechatronics
School of Engineering and Digital Sciences
Nazarbayev University

May 5, 2025

Table of Contents

Abstract

The increasing demand for autonomous mobile robots has led to their widespread use across various industries, including healthcare, logistics, and manufacturing. Effective human avoidance is crucial for safe navigation, necessitating sophisticated collision-avoidance techniques. This project focuses on the development and implementation of a Model Predictive Control (MPC) algorithm to enhance mobile robot navigation in dynamic environments, specifically addressing the challenge of human avoidance. Unlike traditional obstacle avoidance techniques, which often rely on reactive measures, MPC offers a predictive approach that allows robots to anticipate human movement and adjust their trajectory in real time. The project is implemented on jackal robot with LiDAR and RGB-D camera using Clearpath existing sources. The project involves integrating a customized MPC controller into the existing Nav2 framework, as well as incorporating human detection YOLOv8 package. By forecasting human movement and incorporating constraints related to human-robot interaction, this project aims to create safer, more efficient robots capable of operating in complex, human-populated environments across various industries.

Acknowledgments

We would like to express our deepest gratitude to our supervisor, Professor Matteo Rubagotti, for his continuous support, guidance, and encouragement throughout this project. His extensive knowledge, insightful feedback, and commitment to our academic growth were instrumental in the successful completion of our work.

We are also sincerely grateful to Artemiy Oleinikov, MSc in Robotics Engineering, for his outstanding technical assistance and mentorship. Artemiy played a vital role in helping us refine our approach, review and improve our code, and navigate complex challenges related to control systems and the ROS2 Nav2 framework. His patience, clarity, and collaborative spirit significantly enhanced the quality of our project.

In addition, we would like to thank Professor Almas Shintemirov for his kind support and valuable guidance during the course of this work. His availability and constructive input helped us stay focused and confident during key stages of the project.

Chapter 1: **Introduction**

With the increasing demand for autonomous mobile robots, they are now found in a broad range of applications across industries, such as healthcare, delivery, banking, warehousing, retail, hospitality, smart cities, logistics, the public sector, manufacturing, and agriculture [1]. Mobile robot control and navigation techniques continuously evolve, incorporating different machine learning-based strategies including reinforcement learning, neural networks, and deep learning for path planning, obstacle avoidance, and decision-making [2]. Nevertheless, mobile robotic systems also present significant challenges: they must be able to function in unpredictable and uncontrolled environments, which are frequently shared with humans; additionally, they frequently have to work in tandem with humans to complete challenging missions [3]. This way, human avoidance is an essential component of safe robot navigation and calls for sophisticated techniques to avoid collisions.

Conventional obstacle avoidance techniques work well for static objects but they are ineffective in dealing with dynamic obstacles, especially people, leading to the introduction of novel methods. The method in [4] addresses the challenge of mobile robots operating in dynamic environments by estimating typical configurations of dynamic areas through clustering local grid maps and using a Rao-Blackwellized particle filter for improved localization. Another approach proposes an occupancy grid mapping algorithm that detects changes over time using map differencing and the expectation-maximization algorithm to model non-stationary objects [5]. Additionally, a method that interleaves mapping and localization with a probabilistic technique to identify spurious measurements, generating accurate maps in dynamic environments is presented in [6]. Further research introduces a perception mechanism using particle filters and joint probabilistic data association filters to track multiple moving objects and distinguish between dynamic and static features [7]. Looking forward, we aim to implement Model Predictive Control (MPC) to enhance navigation in such dynamic environments, allowing the robot to not only avoid obstacles but also maintain uninterrupted movement by selecting the most efficient path in real time.

Model Predictive Control (MPC) has gradually emerged as a powerful control method for dynamic systems in recent years due to its flexibility in handling various types of constraints and multi-variable input [8]. As an advanced control algorithm, the key advantage of MPC is its ability to explicitly consider the future consequences of current control input variation on the outputs, as well as the impact of such outputs on the specified cost function. By implementing MPC, we can effectively model the dynamics of our mobile platform while also considering various constraints related to human interactions.

The main goal of this project is to develop a prediction model that helps the robot avoid humans by recognizing their movement and adjusting its speed based on the distance to them. The robot will slow down to prevent collisions, considering the time needed to detect the distance, send a command, and stop completely at a given speed. In this project, we will primarily focus on implementing the MPC and its integration with the existing Nav2 framework, comparing it to the current models. Unlike traditional obstacle avoidance strategies, which often rely on reactive measures, MPC allows for a more anticipatory approach by forecasting human movement and adapting the robot's trajectory accordingly.

This project can be applied in a variety of future scenarios where robots operate in close proximity to humans, ensuring safer and more comfortable human-robot interaction. For example, commercialized delivery robots navigating crowded pedestrian areas would benefit

from our algorithm by aligning with safety measures and reducing risks of collision, making them compliant with public safety concerns [9]. Similarly, personal care robots and house-cleaning robots could use this approach to dynamically adjust their movements in response to human presence, enhancing user comfort and safety in homes or healthcare environments [10], [11]. Additionally, tour guide robots in public spaces, like museums or airports, could use the predictive capabilities of our algorithm to navigate efficiently while maintaining appropriate distances from humans [12]. Overall, our work ensures that mobile robots can safely interact with people over long-term use, meeting both safety regulations and human expectations in a variety of applications [13].

Chapter 2: **Literature Review**

Model Predictive Control (MPC) has become a popular approach for controlling mobile robots because of its ability to handle complex dynamics and constraints. Essentially, MPC works by solving an optimization problem at each time step, allowing robots to predict their future behavior [14]. This makes it especially useful for tasks like trajectory tracking and avoiding obstacles. Early studies in MPC focused on its basic principles and applications, showing its ability to manage constraints and optimize robot performance [15]. Recent advancements have explored various specialized MPC approaches to tackle challenges like uncertainty and nonlinearity, successfully implementing MPC-based controllers on mobile platforms.

Various techniques have been developed for human avoidance in mobile robotics, including traditional obstacle detection systems that rely on sensors to identify and navigate around obstacles. For instance, a study [16] introduces a new virtual force field-based navigation algorithm (QVFF) that effectively calculates safe zones and avoids collisions, even in the presence of multiple unpredictably moving humans. Another paper [17] discusses several key techniques for mobile robot navigation, including Neuro-Fuzzy methods like ANFIS, sensor-based controllers for adaptive obstacle avoidance, neural networks for path planning, and reinforcement learning-based navigation. A real-time obstacle avoidance approach for manipulators and mobile robots is presented in [18] using artificial potential fields, which distribute collision avoidance across different control levels for complex environments. This method has been successfully demonstrated in real-time using visual sensing on a PUMA 560 robot, addressing both moving obstacles and manipulator control in operational space. Furthermore, the paper [19] presents a hybrid navigation approach called "Roaming Trails" for mobile robots in human environments, integrating prior knowledge with local perceptions to ensure efficient task execution and deadlock avoidance. In comparison, MPC enhances mobile robot navigation in human environments by enabling anticipatory decision-making, which allows robots to predict future states and reduce the likelihood of collisions with humans.

As research has progressed, there's been a push to make MPC more robust when dealing with disturbances. For example, a paper [20] introduced a Disturbance-Rejection Model Predictive Control (DRMPC) framework that combines a Disturbance Observer (DOB) with MPC to enhance tracking performance even when there are disruptions. However, the complexity involved in designing these systems and the limited research on DOB-based MPC are still challenges that need to be tackled. In a different study [21], researchers presented an improved method for path following in wheeled mobile robots using a disturbance observer-based MPC approach. This method estimates disturbances that could affect the robot's path, allowing it to track its trajectory accurately despite any variations in input. Additionally, a novel method that combines behavior-based strategies with MPC was introduced in [22]. This approach optimizes control parameters and the timing of behavior switches, helping robots adapt to changing environments with less recalibration. While this strategy has benefits like improved trajectory alignment and less computational load, it also faces challenges related to its computational intensity.

Comparative studies also highlighted the effectiveness of different control strategies for mobile robots. The study [23] described a controller that uses an MPC scheme along with a switching algorithm to track moving targets and avoid obstacles. Another interesting approach, called Economic Model Predictive Control (EMPC), was explored for Wheeled Mobile Robots (WMR) in [24]. This method incorporates an event-triggered mechanism and adjusts

the prediction horizon to improve trajectory tracking and obstacle avoidance, proving to be both efficient and practical. Another study [25] compared PID and Linear Model Predictive Control (LMPC) for path-following tasks. Although LMPC is more computationally demanding, it delivers better accuracy in following paths. In paper [26], Nonlinear Model Predictive Control (NMPC) is compared with LMPC, showing that while NMPC can be computationally intensive due to its non-convex optimization, LMPC offers a good balance between lower computational effort and effective performance. This makes linear MPC appealing for practical applications where efficiency matters, while NMPC is more precise but at a higher complexity. This way, MPC is increasingly becoming a suitable method for mobile robots due to its inherent flexibility and ability to manage complex dynamics and constraints in real time.

The robot model used in our project is a nonholonomic skid-steer platform - Clearpath Jackal UGV. Controlling nonholonomic robots presents unique challenges, as conventional smooth and time-invariant feedback control laws are insufficient for stabilizing such systems at a specific configuration, as established by Brockett's theorem. This limitation necessitates the application of advanced control strategies, such as MPC, to achieve the desired stability and performance in trajectory tracking and navigation tasks. In this context, the work of Nascimento et al. [27] provides a comprehensive review of MPC applied to nonholonomic mobile robots, focusing on trajectory tracking problems. The paper highlights how MPC is particularly well-suited for such systems due to its ability to handle nonholonomic constraints, incorporate state and input limitations, and optimize control actions over a prediction horizon. Compared to classical linear control methods, MPC offers greater flexibility and effectiveness in managing the inherent complexities of mobile robots with non-integrable motion constraints.

Building on this, Worthmann et al. [28] present an MPC scheme for nonholonomic mobile robots that achieves asymptotic stability without requiring stabilizing constraints or terminal costs. Their approach employs tailored nonquadratic stage costs, which better capture the behavior of nonholonomic systems compared to traditional quadratic costs. By designing suitable maneuvers and deriving bounds on the value function, they ensure stability through an appropriate choice of prediction horizon. Another noteworthy advancement in the field is the passivity-based nonlinear MPC (PB-MPC) approach proposed by Tahirovic and Magnani [29]. Their framework generalizes traditional navigation strategies by incorporating dynamic models suitable for outdoor robots operating on rough terrain. By integrating passivity theory with receding horizon control, the method enhances convergence and robustness, offering a flexible alternative to classical planners like the dynamic window approach (DWA). In their work, Lafmejani and Berman [30] proposed an efficient nonlinear MPC method for multi-robot navigation of nonholonomic systems, ensuring collision- and deadlock-free trajectories without relying on terminal constraints or costs. Their approach, based on a unicycle model, was validated in simulation and real-world tests with up to six robots.

One more proposed MPC-based approach was introduced in [31] - a randomised MPC-based motion planning algorithm for real-time obstacle avoidance in unknown environments. By combining MPC with tree expansion and random sampling, similar to RRTs, the method efficiently generates kinodynamically feasible trajectories. Maurović, Baotić, and Petrović [32] proposed an explicit MPC method for trajectory tracking of differential-drive mobile robots. By linearizing the robot's kinematics and solving the MPC problem offline, they obtained a piecewise affine control law that enables fast online implementation via a lookup table. The method showed strong tracking performance in both simulations and experiments. Bertoncelli et al. [33] explore a linear time-varying MPC approach for nonprehensile object manipulation with a nonholonomic mobile robot, specifically focusing on pushing manipulation tasks. Their technique incorporates unilateral friction constraints to prevent object slippage, validated through simulations of a Pioneer 3-DX robot. Prado et al. [34] propose an adaptive control approach integrating a Nonlinear Moving Horizon Estimator and NMPC to estimate states

and model parameters for vehicle motion across varied terrains. Evaluated in both simulations and real-world tests with a mini-loader, the system successfully adjusts to terrain variations and maintains high-speed performance for autonomous mining tasks.

Song et al. [35] propose a self-avoidance model predictive controller (SA-MPC) for mobile robots to follow humans while avoiding dynamic obstacles in crowded environments. The approach incorporates an obstacle avoidance optimization item and adaptive waypoint selection, enhancing safety and robustness in dense corridors with random obstacles, as demonstrated in simulations on a pedestrian platform. For example, NMPC was also used as an approach for human-interaction problems. This way, Hu et al. [36] propose an NMPC scheme for mobile medical robots, combining learning-by-imitation and motion control. The system features a multivirtual spring-damper system for trajectory imitation and a varying-parameter one-layer projection neural network to solve online quadratic programming for path tracking, tested in various scenarios on a mobile medical robot.

Another shared control framework using MPC blends human inputs with autonomous behaviors, enhancing teleoperation by accounting for both collision threats and input divergence [37]. Tested on a high-speed robot, it reduced collisions by 66% compared to pure human control while granting 26% more control than simple switching methods. Another recent approach [38], HuMAN-MPC, presents a fast embedded MPC formulation for human-aware navigation in densely populated environments. By combining predictive planning with a real-time optimization backend, it enables scalable, legible robot motion while maintaining low computation times and high-quality avoidance behavior. This work [39] presents a human-robot co-transportation framework using an augmented MPC that incorporates human uncertainty and optimizes the robot's full-body pose, including arm configurations. By solving a human uncertainty-aware Discrete Algebraic Riccati Equation at each planning step, the system improves control performance and adaptability, as validated through both simulations and real-world tests.

This way, the literature highlights the increasing importance of MPC for mobile robots, particularly in addressing complex dynamics, trajectory tracking, and obstacle avoidance. The studies reviewed demonstrate MPC's effectiveness in optimizing performance while managing system constraints, with advancements incorporating techniques like disturbance rejection, hybrid control strategies, and integration with neural networks. MPC has proven particularly valuable in nonholonomic systems, where it manages motion constraints and ensures stability. Building on these findings, the next section outlines the methodology for applying MPC to the Clearpath Jackal UGV, focusing on enhancing trajectory tracking and navigation in both static and dynamic environments.

Chapter 3: **Methodology**

The project follows a structured methodology aimed at developing an autonomous mobile robot capable of human detection, dynamic path planning, and obstacle avoidance in complex environments. The core of this approach lies in the integration of real time human-detection, MPC, and continuous adaptation to the robot's surroundings using sensor data and robust control strategies.

3.1 System Overview and Planning

The project is divided into four major phases. The primary focus in the first half of the project is learning about Robot Operating System 2 (ROS2), preparation of object detection model and simulation environment, while the second half emphasizes implementation of NMPC solver for Jackal robot in the simulated environment and integration with object detection model. The methodology is designed to ensure a progressive development process that allows for constant validation and refinement through simulations before physical deployment. This structured breakdown of tasks maximizes efficiency and adaptability, enabling parallel workflows where necessary.

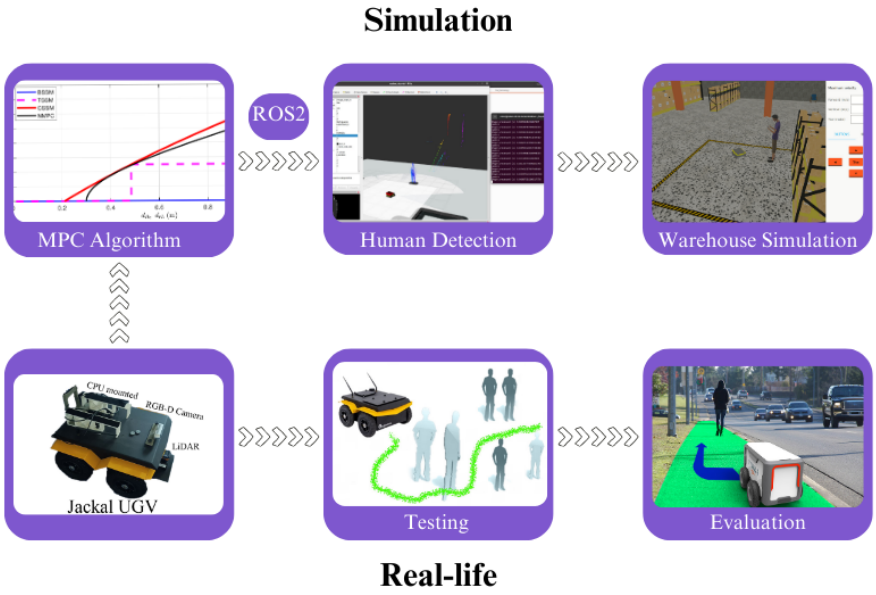


Figure 3.1: Methodology Block Diagram.

3.2 Project Preparation

The first stage involves setting up the necessary software environment, including the installation of ROS2 and the Nav2 package, which provides the foundational components for navigation, localization, and path planning in a ROS2 ecosystem. ROS2's modular framework allows us to integrate various sensor inputs and control algorithms while leveraging the powerful communication infrastructure that ROS provides. The Nav2 package will be central to setting up basic navigation tasks before the MPC module is integrated. A vital part of the

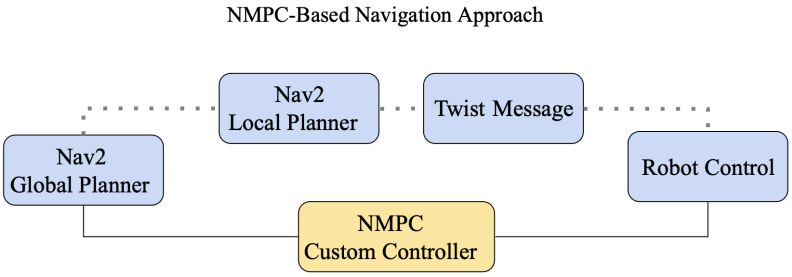


Figure 3.2: Block diagram of the custom NMPC controller implementation.

project preparation involves constructing an occupancy map using the Python3 occupancy mapping package. The occupancy map represents the static obstacles within the environment and will be dynamically updated as the robot detects new obstacles. To achieve accurate mapping, we will use data from the LiDAR and RGB-D cameras, which provide real-time spatial information and human detection capabilities. These sensors ensure that both static and moving entities, such as humans, are accurately represented in the environment map.

Simultaneously, the literature review continues throughout the preparation stage, with team members identifying cutting-edge methodologies in MPC, human detection, and path planning for mobile robots. The review informs our choice of algorithms and control strategies, ensuring that our solution aligns with the latest research trends and industry standards.

3.3 Custom NMPC Controller

In the standard Nav2 navigation stack, the global planner generates a path of waypoints, which the local planner uses to produce velocity commands (Twist messages) for the robot. However, traditional local planners are often reactive and may not fully consider the robot's dynamics.

In our implementation, we replace the local planner with a custom Nonlinear Model Predictive Control (NMPC) solver. This controller processes the global path, selects a goal point about one meter ahead, and solves an optimization problem to compute the optimal wheel velocities. These commands are sent directly to the robot, enabling more predictive and adaptive control compared to conventional methods.

3.4 Simulation in Gazebo and RViz

Once the software environment and occupancy map are in place, we will develop a virtual environment using Gazebo and RViz for simulation. This environment allows us to test the robot's control algorithms without the risks associated with real-world experiments. The first step here is parameter tuning, ensuring that the robot's physical properties (such as kinematic constraints) and sensor characteristics are accurately represented in the simulation.

At this stage, the MPC framework will be integrated into the system. MPC is particularly well-suited for this project due to its ability to handle complex constraints while optimizing the robot's future trajectory in real time. MPC's predictive capabilities ensure that the robot not only reacts to immediate obstacles but also plans ahead to maintain a smooth and efficient path through the environment. This feature is especially crucial for navigating dynamic environments where both static obstacles and human presence are involved. The MPC framework will undergo extensive tuning and simulation-based validation before it is applied to the physical robot.

3.5 Implementation on the Mobile Robot

Human detection is a central component of the project, enabling the robot to recognize and avoid human presence dynamically. Open-source repositories containing human detection algorithms will be adapted for this purpose. The system will be designed to work in conjunction with the occupancy map and other sensor inputs to ensure that humans are not only detected but also tracked within the environment. This information will be fed into the MPC module, which will adjust the robot's velocity and trajectory to avoid potential collisions with humans while maintaining efficiency in its path planning.

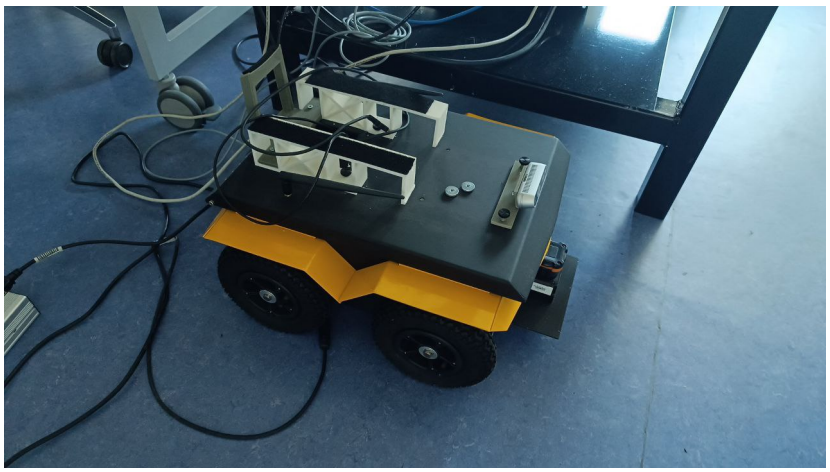


Figure 3.3: The mobile robot

Once the virtual testing is complete, the project will transition to the physical mobile platform. The focus here will be on integrating the software stack with the mobile robot's hardware, ensuring that the control strategies and sensor data are properly interfaced with the robot's actuators and feedback systems. Initial tests will involve simple obstacle avoidance scenarios to ensure that the MPC's output commands are accurately executed by the robot.

After the initial integration, we will perform several tuning sessions to optimize both the control parameters and the human detection system. This phase will include real-world tests in various environments to assess the robustness of the system. Debugging and refinement will continue iteratively until the robot exhibits reliable performance in all expected conditions.

3.6 Final Evaluation

The final phase of the project will focus on data collection, performance evaluation, and the preparation of the final presentation. The results of the tests, including trajectory accuracy, human-avoidance performance, and system robustness, will be documented through data plots and graphs. This evaluation will highlight the system's capabilities and its alignment with the project objectives, forming the basis for the final presentation and report.

Overall methodology employed in this project is highly modular and adaptable, allowing for simultaneous development and testing of various components such as human detection and MPC. By combining state-of-the-art sensor technology, robust control strategies, and detailed simulations, we ensure that the final system is capable of navigating complex environments while maintaining high performance in terms of both efficiency and safety.

Chapter 4: **Implementation and Execution**

4.1 Installation and Initial Setup of ROS2 Humble

The project commenced with the successful installation of ROS2 Humble, an essential step for enabling the development of robotic systems. Following this, the team completed a fundamental tutorial that provided a comprehensive introduction to ROS2. This tutorial covered crucial aspects such as node management, topic handling, and communication protocols, laying a solid foundation for subsequent development tasks. This initial setup was vital in preparing the team for more advanced phases of the project.

4.2 Familiarization with the Nav2 Package

With ROS2 Humble installed, the focus shifted to the Nav2 package, which is integral for navigation and path planning in autonomous robotics. The team engaged with both a detailed tutorial and the full documentation of the Nav2 package. This in-depth study covered key functionalities including localization, mapping, and trajectory planning. The acquired knowledge from these resources has been essential for implementing and customizing the navigation solutions required for the project.

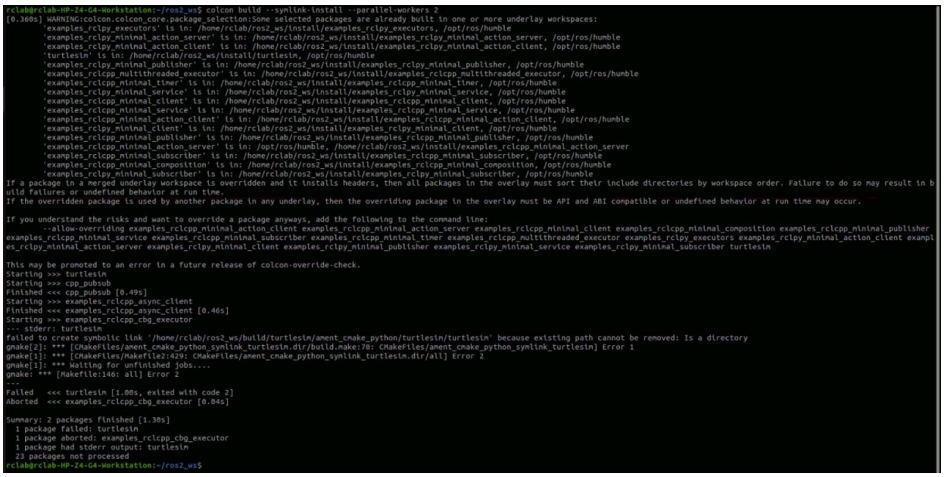


Figure 4.1: Tutorial process

4.3 Development of an Occupancy Map Analysis Tool

A significant achievement was the development of a custom Python package designed to interact with ROS2 occupancy map topics. This tool performs detailed analysis by calculating the percentage of the map occupied by obstacles. Such analysis is crucial for understanding environmental layouts and obstacle distribution, which directly impacts navigation and decision-making processes. The implementation of this tool has enhanced the team's ability to interpret occupancy maps and refine navigation strategies.

4.4 Simulation of the Jackal Robot

We successfully conducted a simulation with the Jackal robot in a warehouse environment. Utilizing the Nav2 package, the robot was programmed to drive in a square trajectory through a custom Python node. This simulation was pivotal for testing and validating the integration of the Nav2 navigation stack with the Jackal robot's control systems. The controlled virtual environment provided valuable insights into the robot's performance and the effectiveness of the implemented navigation algorithms.

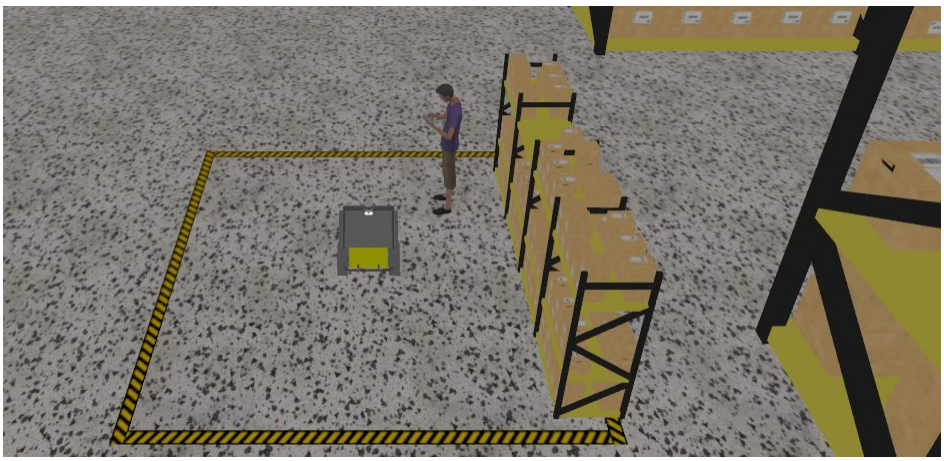


Figure 4.2: Jackal robot in Gazebo simulation

In summary, the completed tasks have established a strong foundation for the project. The successful installation and setup of ROS2 Humble, coupled with a thorough understanding of the Nav2 package, have prepared the team for the next stages of development. The creation of the occupancy map analysis tool and the effective simulation with the Jackal robot demonstrate the team's progress and readiness for further advancements.

4.5 Integration of LiDAR Sensor and Intel Realsense Camera with Clearpath Simulation

The project advanced by integrating both the LiDAR sensor and Intel Realsense camera into the simulation using the Clearpath platform. The primary objective was to mount a 2D

Hokuyo LiDAR sensor on the front of the robot for obstacle detection and environmental mapping, as well as installing the Intel Realsense camera on top for depth sensing and RGB data capture.

The process involved configuring the robot's URDF and YAML configuration files to properly mount the LiDAR on a horizontal bracket at the front of the robot and the Realsense camera on the top. We successfully spawned the LiDAR sensor, the mounting bracket, and the Intel Realsense camera in the Gazebo simulation environment. Both sensors were integrated with the ROS2 system, allowing real-time data publishing and visualization.

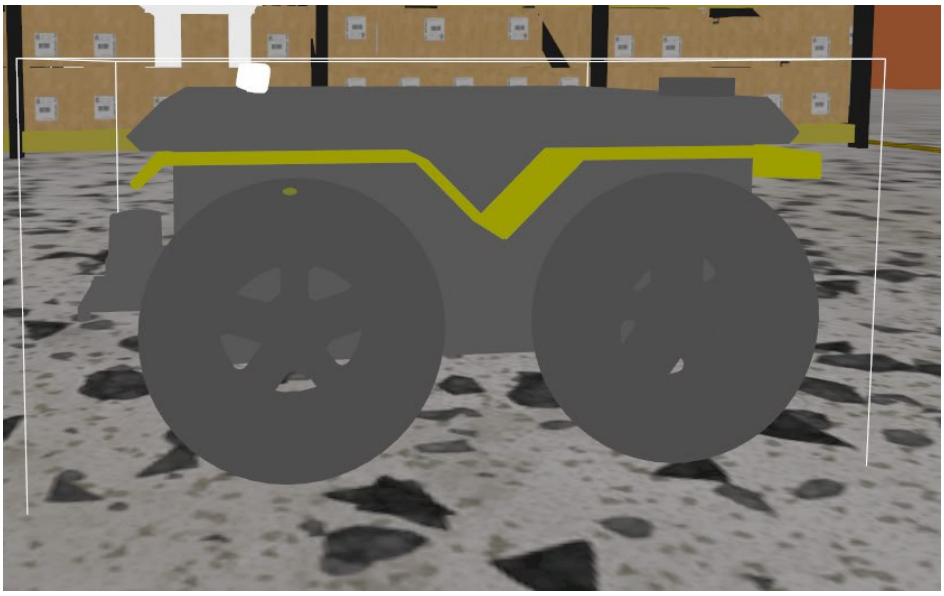


Figure 4.3: Jackal robot with LiDAR and Inter Realsense installed

This milestone established core functionality for perception tasks and laid the foundation for further development involving advanced sensor fusion, obstacle avoidance, and mapping in future phases of the project.

4.6 Integration of Human Detection

We were specifically looking for an open-source package that would support efficient human detection without consuming significant computational resources, as we need to reserve GPU load for additional tasks, including navigation (Nav2 package) and MPC computations. YOLOv8 was selected because it offers high accuracy in real-time detection while being flexible in terms of model size.

We successfully integrated the open-source YOLOv8 package into our system, enabling real-time human detection. We tested different versions of the YOLOv8 model—specifically, the medium and nano versions—to determine the computational load and efficiency. Our findings showed that the nano version provided excellent performance while occupying only 3% of the total GPU load, making it highly efficient for our application.

This model allows us to receive 3D bounding boxes of human obstacles through 2D segmentation of obstacle from RGB camera and depth information gained from Depth camera. Information contains size of a 3D bounding box (x, y, z) and distance to the box itself, which

we will be using for the NMPC implementation.

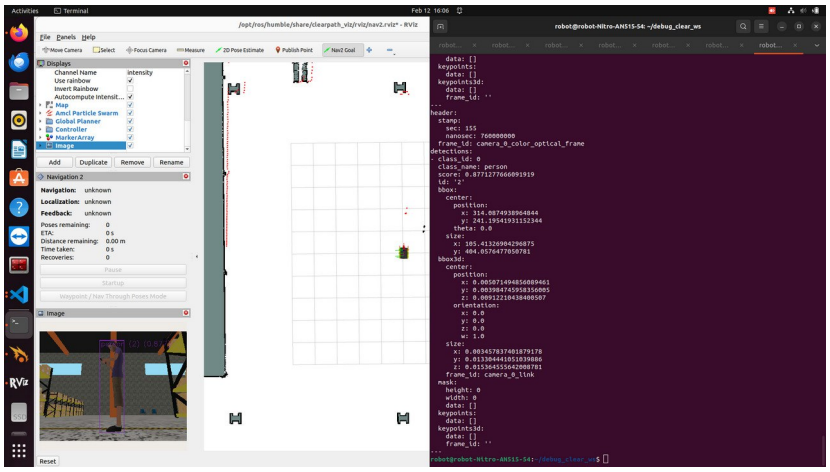


Figure 4.4: 3D bounding box topic in simulation

In addition to the simulation, we were able to test the real Intel Realsense D435 RGB-D camera that we will be using in the practical part. Results showed the correct representation of 3D bounding boxes during the testing.

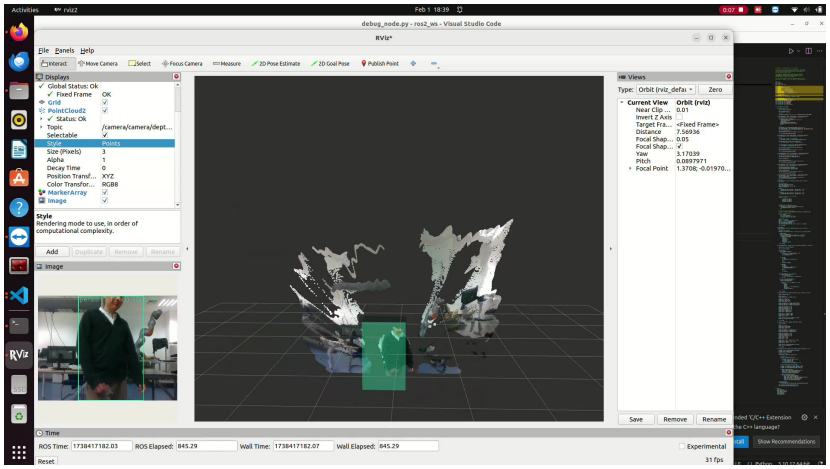


Figure 4.5: 3D bounding box visualization in real camera

4.7 Navigation and Localization

We successfully ran the Nav2 package with localization and conducted initial tests integrating it with the human detection package. This integration allowed us to retrieve human coordinates in the environment and measure the distance to the detected humans, demonstrating the system's capability to perceive and interact with its surroundings effectively.

We started by using the Clearpath simulator that we configured earlier with the necessary robot specifications, including a Realsense camera and a Hokuyo UST10 LiDAR. The process began with launching the Clearpath simulator, where we ensured that the robot.yaml configuration file was correctly placed in the setup folder. Once the simulation was running, we opened RViz to visualize the robot's position and surroundings. Using the localization

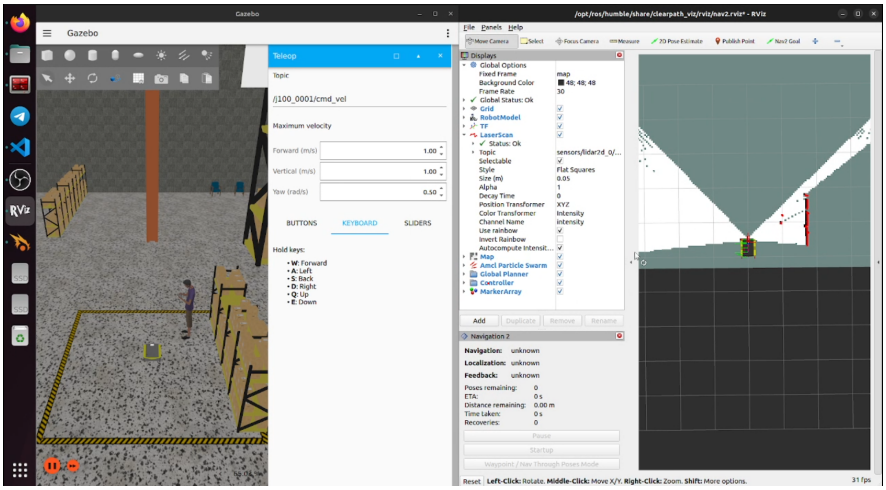


Figure 4.6: Gazebo simulation with localization

package, we enabled the robot to determine its position within the simulated map, ensuring that it could accurately track its movements.

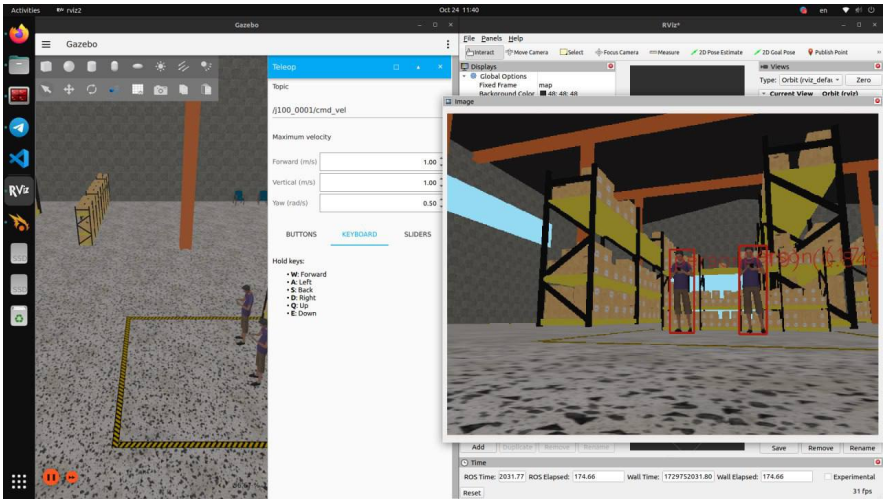


Figure 4.7: Human identification with RViz.

Figure 4.8 shows the robot's capability for human identification using RViz. It visualizes detected human figures, highlighted within the RViz interface. This is achieved by combining data from a Hokuyo LiDAR and a RealSense camera. Human detection plays a critical role in ensuring safe navigation, as it enables the robot to identify and avoid humans while interacting with its environment.

Figure 4.9 highlights the robot's simulation with integrated localization, mapping, and object detection using Nav2 and Yolo. Localization is crucial for determining the robot's position within the environment and for constructing an accurate occupancy map. The occupancy map is built by processing LiDAR data, which represents the environment as a grid of occupied and free spaces. This map, combined with object detection from Yolo, allows the robot to perceive and plan its paths effectively while avoiding obstacles and interacting with the surroundings in real time. This simulation demonstrates a robust navigation system that combines perception, mapping, and planning. To test the navigation capabilities, we launched the Nav2 stack, which provided essential components such as path planning and motion control. After setting the robot's initial pose in RViz, we assigned navigation goals, and the robot successfully planned and followed paths to reach those targets.

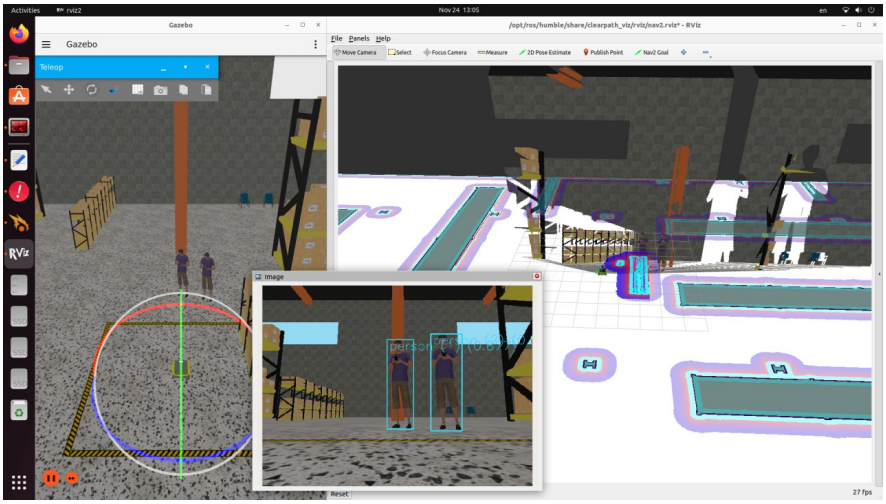


Figure 4.8: Simulation with localization, Nav2 and YOLOv8 Nano.

These tests confirmed that navigation, localization, and human detection were working together effectively, laying the groundwork for the next stages of our project.

4.8 MPC Implementation

The approach we are implementing is as follows: instead of using the local planner from the Nav2 package, we will replace it with our customized NMPC implementation. This implementation will output the velocities and angles for the right and left wheels of the robot. The positions and velocities received from the Nav2 global planner will be fed into the MPC, which will compute the robot's trajectory. This way, the trajectory will account for obstacles, specifically humans, by incorporating them as constraints and adjusting the robot's velocity accordingly.

The computations for the NMPC will be implemented using the ACADOS solver. The C-generated code files made by Jackal_map will be used in our goal trajectory computation.

The Use of NMPC:

For computational efficiency, obstacles are approximated as spheres of different radius. Walls will be represented as a series of spheres, while a human will be modeled as a single sphere. The robot itself will also be approximated as a single sphere. The costmap will be analyzed to identify obstacles. The map (approximately 3×3 meters in size) will probabilistically contain around 250 spherical obstacles which will be used as constraints for the solver. The NMPC solver will account for these obstacles during its optimization process.

Goals will be tracked dynamically: the next waypoint is selected based on the obstacle's location. An NMPC optimization problem is then solved to compute control inputs, producing velocity commands for the left and right wheels. These wheel commands are then published to execute the robot's movement. A 5 cm safety offset is maintained around static obstacles to prevent the robot from scratching them upon contact.

To account for the robot and human occupancy, we will apply the constraints from [28] within the NMPC framework. The following constraints will ensure the safe navigation around dynamic obstacles:

1)

$$(x_r + x_s^i)^2 + (y_r + y_s^i)^2 \geq (r_s^i + r_r + 0.05)^2 \tag{4.1}$$

where (x_s^i, y_s^i) is obstacle's position, (x_r^i, y_r^i) is robot's position, and r_s^i r_r^i are radius of obstacle and robot respectively, 0.05 meters is the additional safety margin.

2)

$$v_{rx}^2 + v_{ry}^2 < f(D_{rh}) \tag{4.2}$$

where v_{rx} and v_{ry} are robot's velocity in x and y directions, D_{rh} is the distance between robot and obstacle. $f(D_{rh})$ is a function that determines the maximum allowable speed as a function of the robot-obstacle distance. The function is defined as:

$$f(D_{rh}) = \alpha^2 \left(D_{rh}^2 - (R_r + R_h + \bar{d})^2\right) \tag{4.3}$$

where:

- D_{rh} - distance between the robot and the human
- R_r - radius of the robot
- R_h - radius of the human.
- $\bar{d}$ - safety margin distance (a tuning parameter ensuring extra buffer)
- α - scaling factor for velocity constraint (a non-negative tuning parameter)

The Jackal UGV is a skid-steer wheeled mobile robot (SSWM) with a kinematic model described in [29], where the relationship between the wheel velocities and the robot's velocity is expressed as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{v_r + v_l}{2} \cos\theta \\ \frac{v_r + v_l}{2} \sin\theta \\ \frac{v_r - v_l}{\alpha b} \end{pmatrix} \tag{4.4}$$

where b is the track width (the distance between the left and right wheels), and α is a dimensionless factor that accounts for the skid-steering dynamics.

Since skid-steering introduces nonlinearities due to wheel slippage, an empirical correction factor α can be introduced. If applied, the angular velocity equation becomes:

$$\dot{\theta} = \frac{v_r - v_l}{\alpha b} \tag{4.5}$$

where α is a parameter calibrated from real-world experiments to better capture the Jackal's actual turning behavior.

We decided to implement NMPC as a node that receives the global path from the navigation stack Nav2, processes the costmap to detect obstacles, and dynamically tracks goals by selecting the next waypoint 1 meter ahead. It then solves an NMPC optimization problem, publishes wheel commands for movement, and visualizes the robot and goal poses in Rviz. To simultaneously receive information from the Nav2 node while preventing it and the 'twist' topic from sending messages to the robot's wheels, we will remap them to a nonexistent

topic. Meanwhile, our NMPC node will handle wheel commands, effectively implementing the low-level controller.

The goal arrow and robot arrow functions indicate the trajectory and the robot's odometry, while the MPC solver is managed using the holder file that refers to the c-generated code from ACADOS.

The node subscribes to global path, AMCL localization, local costmap, and Gazebo clock topics to update the robot's goal position and obstacle constraints.

In the main loop, it continuously tracks the goal, computes control commands, and adjusts the trajectory based on real-time localization and obstacles. If the solver fails, it is reinitialized. Finally, the computed trajectory and control commands are published to the robot's motor controller and RViz visualization topics.

For deployment on the real Jackal robot, the NMPC node will receive localization data from the onboard sensors (wheel encoders, IMU, and LiDAR-based AMCL). The global path and costmap will be generated using the Nav2 stack, while real-time obstacle detection will be handled via the robot's onboard perception system. The computed control commands will then be sent directly to the Jackal's motor controller via the ROS `cmd_vel` topic, ensuring smooth and adaptive navigation in dynamic environments.

4.8.1 NMPC as a Nav2 Plugin

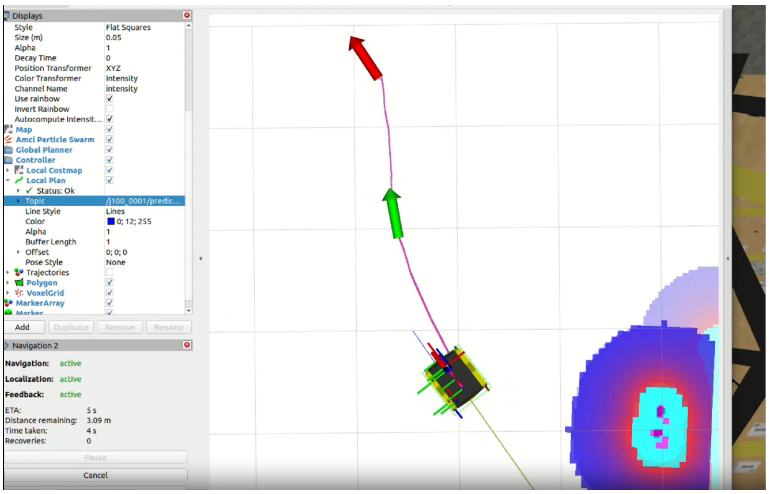


Figure 4.9: Implementation of the NMPC node as a Nav2 Plugin visualised in RViz.

To enable trajectory tracking using the ACADOS NMPC solver, we implemented a custom controller plugin within the Nav2 framework by extending the *nav2_core* :: *Controller* interface. This MPCController computes optimal linear and angular velocity commands that allow the robot to follow a global path while avoiding obstacles in real time.

The plugin transforms the global plan into the robot's local frame, processes local costmap data, and uses current pose and velocity information to solve an NMPC optimization problem. It handles large orientation differences, smooths velocity commands to reduce oscillations, and publishes predicted trajectories for visualization. The controller is compatible with ROS 2 lifecycle nodes, uses tf2 for frame transformations, and integrates seamlessly with Nav2's costmap system.

The controller dynamically selects intermediate goal poses, allowing an adaptive local goal

with varying distance based on how far its final goal is. It was used to improve tracking performance, while reducing oscillations and to ensure stable motion in complex global paths.

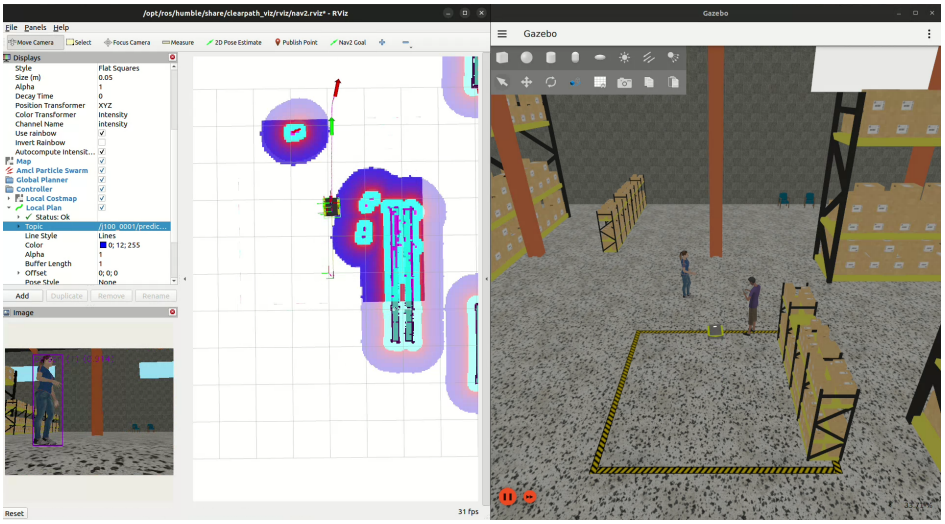


Figure 4.10: Local (green arrow) and Final (red arrow) goals for NMPC controller.

Additionally, the plugin subscribes to a 3D detection topic published by YOLOv8-Nano node, enabling real-time detection and segmentation of dynamic obstacles, in our case humans. Detected obstacle positions are transformed into the robot's base frame and incorporated into the NMPC optimization problem. This approach allows the controller to proactively adjust not only its predicted trajectory, but also linear and angular velocities when a person is detected near the robot's safety region. This allows to maintain safe speed and trajectory when robot passes by the human. This approach provides smooth and adaptive control while respecting the robot's kinematic constraints, enabling robust navigation in dynamically changing environments.

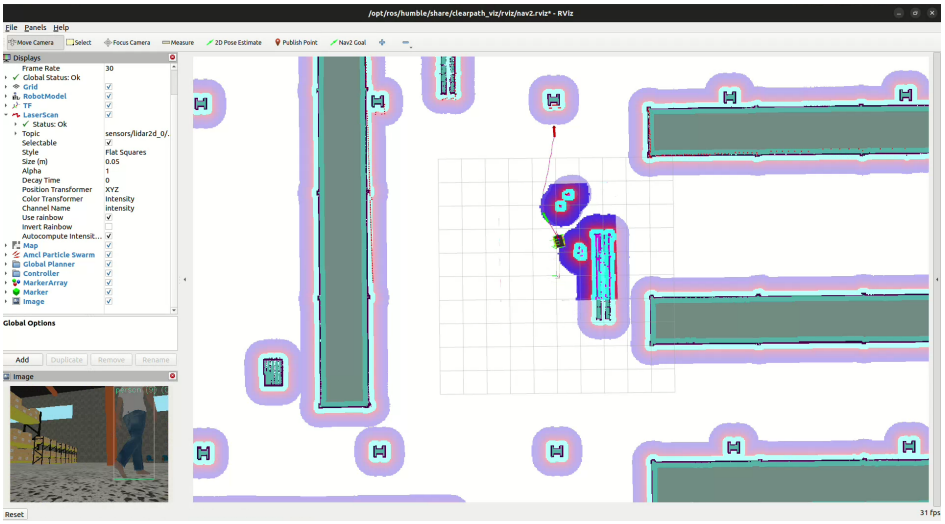


Figure 4.11: RViz visualization of trajectory change due to human detection by YOLO model.

Chapter 5: **Results and Discussion**

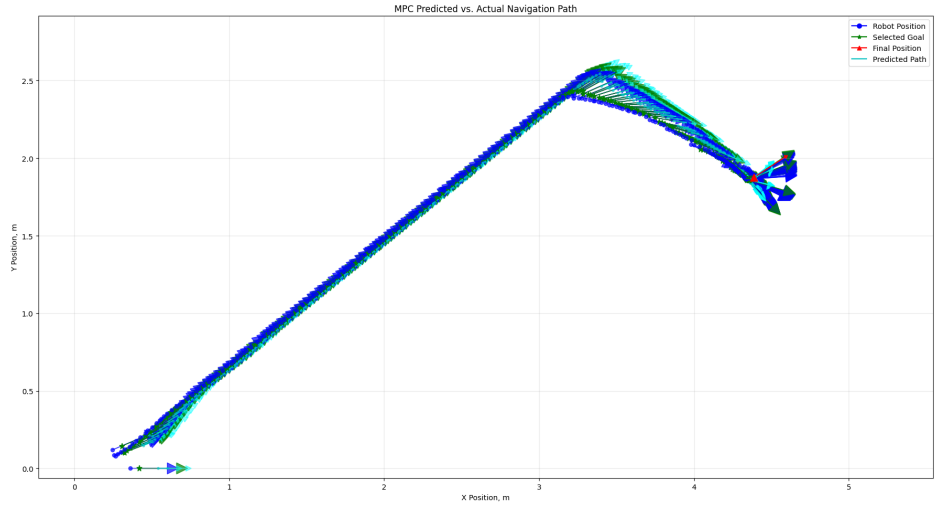


Figure 5.1: NMPC Path Tracking Performance.

Figure 5.1 provides a comparative analysis between the trajectory predicted by the MPC algorithm and the actual path followed by the robot, which executes the velocities computed as control outputs of the MPC. The positions are shown relative to the robot frame (i.e., in a robot-centric coordinate system).

The close alignment observed between predicted and actual trajectories demonstrates the MPC's modeling capabilities and robust control performance. This successful path tracking is achieved through an optimization process that employs specific weight parameters: weight_x=1.0 and weight_y=1.0 maintain balanced path following, weight_yaw=15.0 ensures proper orientation control, while the significantly higher terminal weights (weight_terminal_x =5000.0, weight_terminal_y=5000.0, weight_terminal=100.0) enforce precise goal convergence. These weight settings explain the robot's strong adherence to the target position visible in the plot, particularly the aggressive corrective maneuvers near the goal where the terminal weights dominate the optimization.

The MPC operates within well-defined kinematic and dynamic constraints, including maximum velocity/acceleration limits and steering angle restrictions, which contribute to the smooth and feasible motion characteristics visible in the actual path. The system's real-time performance capability is demonstrated by its ability to compute control inputs within 58-65 ms per iteration while maintaining accurate trajectory tracking.

The robot's behavior near the goal, particularly, the sharp rotations to achieve the desired final orientation, may stem from the current weight settings in the MPC cost function. These weights appear to prioritize terminal position and orientation accuracy over smooth path tracking throughout the trajectory. As a result, the controller may tolerate larger intermediate deviations in favor of minimizing final pose error, which could explain the discrepancies observed between the predicted and actual paths. This trade-off likely contributes to sudden corrections near the goal, as well as occasional overshooting during turns or final adjustments.

The velocity plots in Figure 5.2 show the robot's movement patterns - the speed changes gradually while steering adjusts more frequently. This reveals the controller is constantly

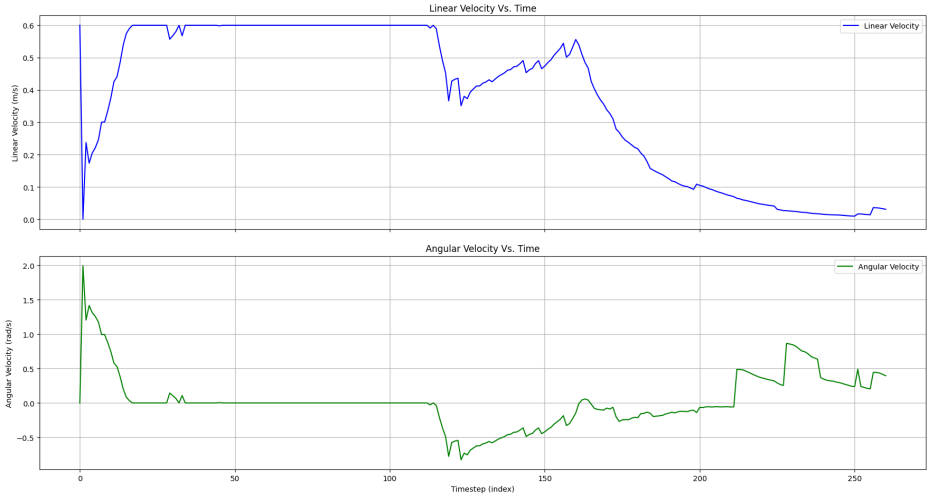


Figure 5.2: Linear and angular velocities over time.

correcting direction while being more careful with acceleration. The jerky steering suggests the system may be overreacting to small orientation errors, likely due to high yaw weights in the MPC settings. The speed profile's sawtooth pattern indicates either conservative speed limits or the controller struggling to maintain steady velocity while making steering corrections. These patterns point to needed tuning of the motion control parameters, particularly finding a better balance between straight-line speed and turning precision to achieve smoother navigation.

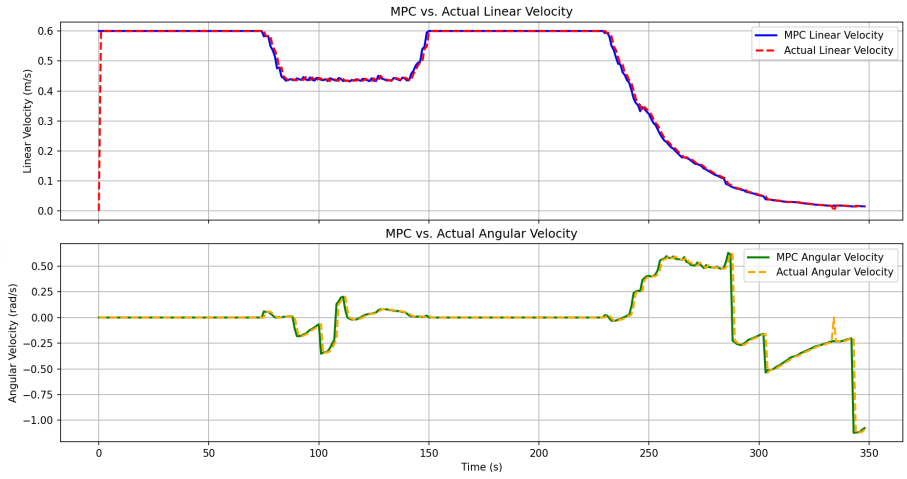


Figure 5.3: Change in linear and angular velocities without human presence.

Figure 5.3 illustrates the reference velocities generated by the MPC alongside the actual velocities measured on the robot. Figure 5.4 presents the corresponding velocity errors. The observed discrepancies are primarily due to differences between the internal model used by the MPC and the robot's actual dynamics. Even with precise wheel speed commands, some deviations are expected as a result of model inaccuracies and unmodeled physical effects. Additionally, the robot tends to perform small rotational corrections near the goal to better align its final orientation. This behavior could potentially be reduced by fine-tuning the weighting parameters in the MPC cost function, particularly those related to orientation and terminal pose accuracy.

The velocity profiles in Figure 5.3 demonstrate clear speed adaptation when operating near humans, showing responsive safety behaviors. This is the draft implementation of the SSM (Speed Separation and Monitoring). The linear velocity maintains conservative speeds be-

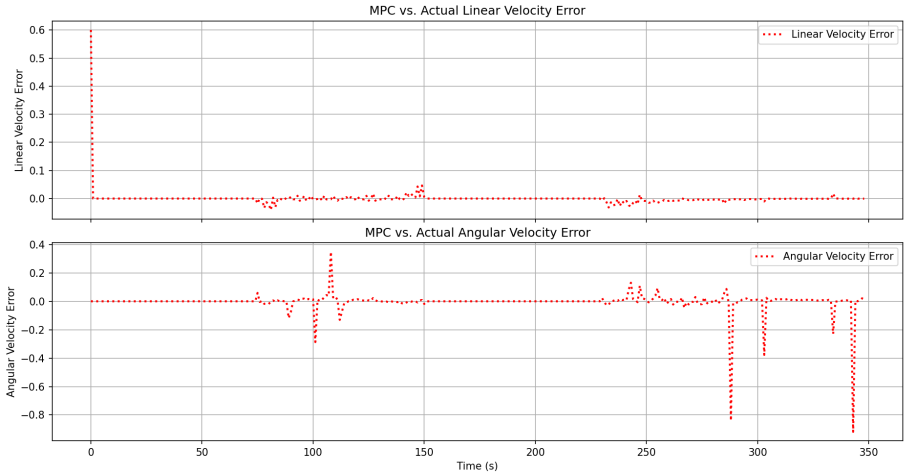


Figure 5.4: Error between actual controller velocities and MPC velocities without human presence.

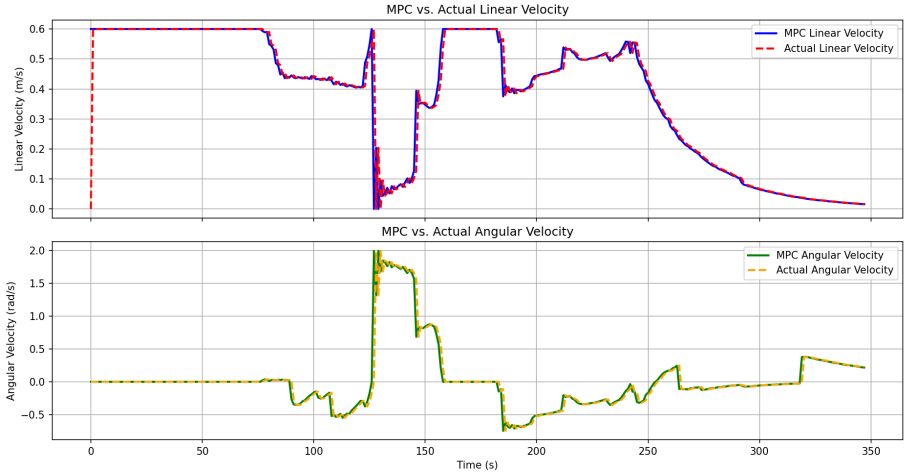


Figure 5.5: Change in linear and angular velocities near human presence.

low 0.5 m/s, with frequent deceleration phases indicating deliberate motion moderation in human presence. The angular velocity reveals particularly cautious steering, constrained within a tight ± 0.3 rad/s range, which are significantly reduced from typical operational limits, suggesting prioritized stability over maneuverability when humans are nearby. Both profiles exhibit smooth, gradual transitions rather than abrupt changes, reflecting careful motion planning for human proximity. The controller successfully implements speed-dependent safety constraints. These patterns confirm the system's human-aware navigation strategy successfully modulates both translational and rotational dynamics for safe co-existence.

Chapter 6: **Challenges and Solutions**

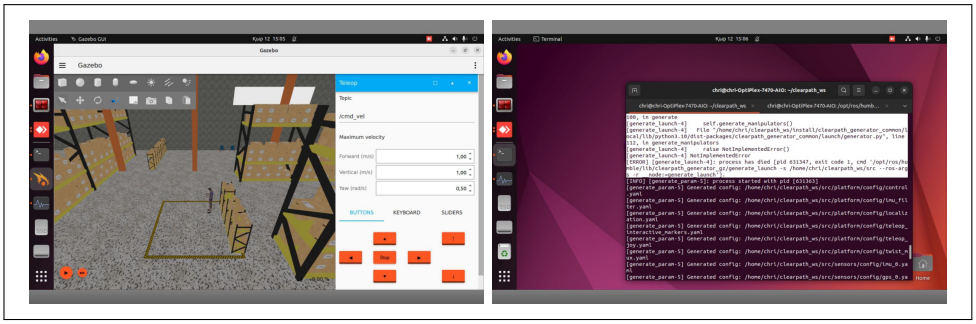


Figure 6.1: Challenges with the Clearpath Gazebo.

While installing Clearpath Gazebo for simulating Clearpath Robotics platforms, an issue arose due to two conflicting overlaying workspaces, which caused a problem with generating the robot in Gazebo. In ROS2, overlaying workspaces occur when multiple workspaces are sourced incorrectly, leading to conflicts in dependencies or package paths. This resulted in the simulation environment being unable to correctly load the Clearpath robot models, as certain files or configurations were inaccessible. Despite attempts to resolve the conflict by adjusting the workspace paths and environment settings, the issue persisted. The only viable solution was to properly reinstall Clearpath Gazebo. By carefully ensuring that no conflicting workspaces were sourced and performing a clean installation, the necessary files and configurations were correctly set up, allowing the robot models to generate and function as expected in the Gazebo simulation.

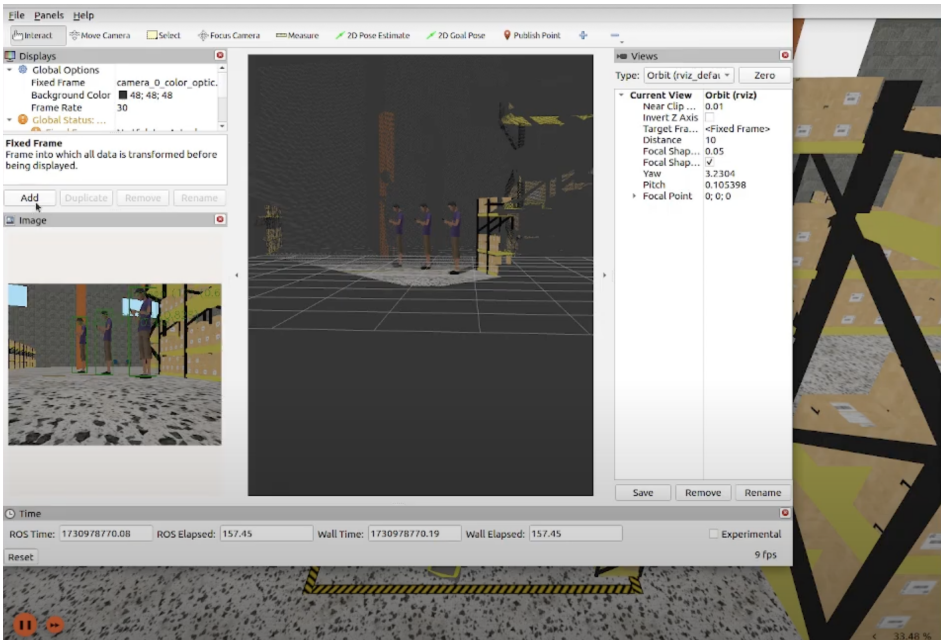


Figure 6.2: Point Cloud received from the depth camera.

We encountered an issue with inactive controllers while creating a node for robot motion, which initially prevented the robot from responding to velocity commands in the simulation. We attempted to manually activate the controllers by sending commands to enable them, but

this did not resolve the issue and it needed to run the command every time the simulation was launched. After further investigation, we discovered that the root cause was a missing controller configuration in the simulation launch file. The problem was resolved by adjusting the launch file to ensure the joint state broadcaster and platform velocity controller were properly loaded during simulation startup. This modification ensured that the controllers were active as soon as the simulation began, allowing for smooth robot control.

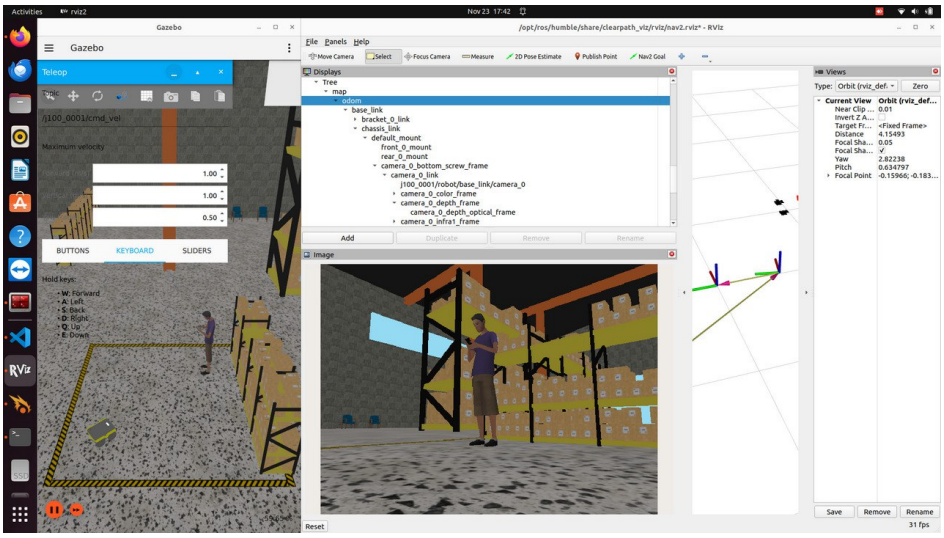


Figure 6.3: RGB image with YOLOv8 nano and point cloud from depth camera.

During our integration of YOLOv8 Nano for human detection with segmentation and point cloud generation, we encountered significant challenges when using the RealSense D435 camera in combination with our depth processing requirements. Our first setup involved reading only RGB data from the RealSense camera, which allowed YOLOv8 Nano to detect and segment humans effectively. However, when we attempted to extend the functionality by incorporating the depth camera to generate a point cloud and obtain distance data, the detection system encountered a critical issue: the YOLOv8 model stopped displaying the RGB image feed from the camera, and the expected point cloud output was also missing. Currently, we are using the depth image directly, but this is not an optimal solution. In an attempt to address the issue with the RGB image feed, we experimented with displacing objects of the robot in the RViz simulation. This adjustment successfully restored the RGB image display. However, the challenge of human identification persists due to errors in the transformation of frames. These frame transformation errors are disrupting the proper alignment between RGB and depth data, which is critical for accurate human detection and point cloud generation. Currently we are manually adding 2 additional transformation frames in order to activate 3D detection node to get 3D bounding boxes for our obstacles(human). We are actively working on resolving this issue without manually adding these 2 frames to enable full integration of RGB and depth data, which will allow us to proceed with enhanced detection and control capabilities. Further investigation showed, that the issue only occurs in simulated environment, while the real RGB-D camera does not have issues with implementing the YOLOv8 Nano directly.

Chapter 7: **Conclusion**

Our NMPC-based controller demonstrates efficient real-time performance by solving the optimization problem within 50–70 ms per control cycle using the ACADOS solver, making it suitable for dynamic navigation tasks. The spherical obstacle approximation ($\sim$ 260 obstacles in a 3 $\times$ 3 m area) and skid-steer kinematic model with an empirical correction factor (α) maintain computational tractability while preserving motion accuracy. The controller achieves smooth execution with only minor discrepancies between the reference velocities (generated by the MPC) and the actual velocities observed on the robot. These discrepancies can be attributed to differences between the model used by the MPC and the real-world dynamics of the robot. Even if we were able to command exact wheel speeds, some deviation would still be expected due to model inaccuracies and unmodeled dynamics. Additionally, we observe that the robot often performs small rotational adjustments near the goal to better align its orientation. This behavior could be further refined by tuning the weight parameters in the MPC cost function, particularly those related to orientation and terminal constraints.

In our current implementation of safe navigation around humans, the custom MPC-based controller leverages local costmaps updated at 5 Hz and operates with a prediction horizon of five steps at a 20 Hz controller frequency. For comparison, [21] uses a sampling time of 50 Hz (0.02 s) with a 10-step horizon, while [20] adopts a 20 Hz (0.05 s) sampling time with an 80-step horizon. This places our controller update rate within a robust and responsive range suitable for real-time navigation.

While this setup enables timely decision-making with a planning-execution cycle of 50 ms, human motion is currently treated in a simplified manner as an obstacle in a local costmap.

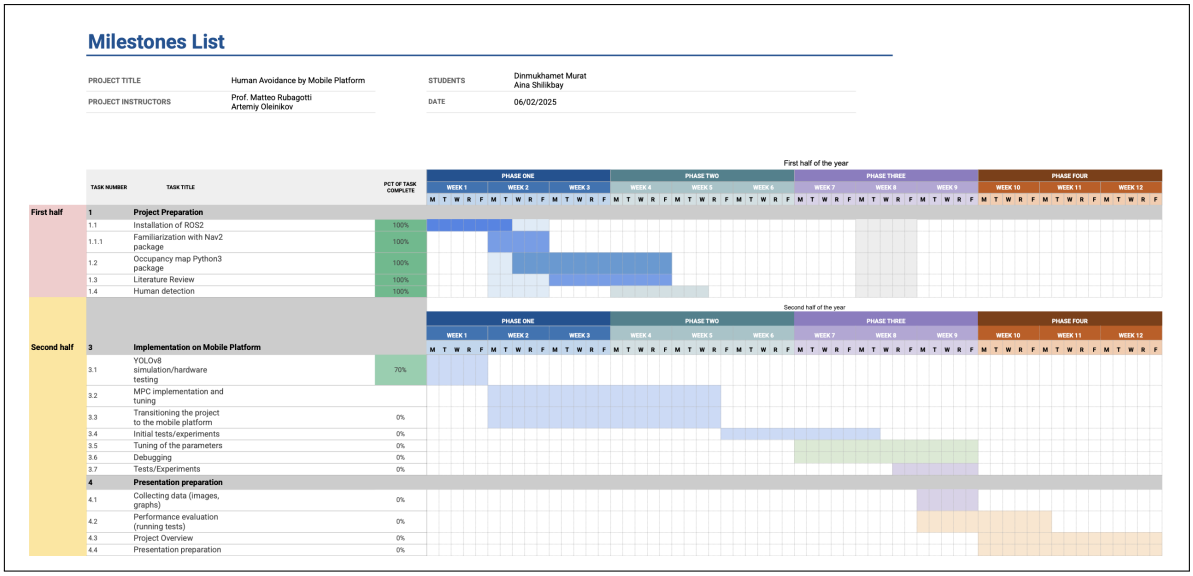
We plan to utilize the past few samples of human positions—collected over approximately one second at our controller's 20 Hz update rate—as the basis for predicting future motion. One initial approach is to perform linear extrapolation using the observed velocity and direction to estimate the human's short-term path. This could be visualized as projecting a straight-line trajectory from the human's most recent motion pattern.

To improve upon this, we are considering the use of learning-based models, particularly neural networks, to capture more complex and realistic human behaviors. These models would take recent trajectory data as input and generate more accurate predictions of likely future positions. This predictive capability would then be integrated into the MPC's cost function or constraints to proactively plan around expected human motion.

Human-aware velocity adaptation (linear $<$ 0.5 m/s, angular $<$ $\pm$0.3 rad/s) is executed without computational overhead through embedded distance-speed constraints (Eq. 4.3). The ROS 2 implementation shows efficient integration with Nav2, processing costmap updates and YOLOv8 detections at 10 Hz while maintaining the control cycle rate.

In conclusion, we successfully achieved the goal of developing and integrating a custom NMPC-based local controller within the Nav2 framework. While the tuning is not yet fully ideal, particularly, in handling sharp turns and complex dynamic obstacles, it demonstrates strong real-time performance and control accuracy. The controller's computational efficiency, combined with its predictive capabilities, makes it a promising foundation for future improvements. Ongoing work will focus on refining constraint handling and enhancing trajectory smoothness, with plans to deploy the system in extended real-world scenarios.

Chapter 8: **Future Work**



Milestones List

PROJECT TITLE	Human Avoidance by Mobile Platform
PROJECT INSTRUCTORS	Prof. Matteo Rubagotti Artemiy Oleinikov
STUDENTS	Dinmukhamet Murat Aina Shilikbay
DATE	06/02/2025

First half of the year

	TASK NUMBER	TASK TITLE	PCT OF TASK COMPLETE
First half	1	**Project Preparation**	
	1.1	Installation of ROS2	100%
	1.1.1	Familiarization with Nav2 package	100%
	1.2	Occupancy map Python3 package	100%
	1.3	Literature Review	100%
	1.4	Human detection	100%
Second half	3	**Implementation on Mobile Platform**	
	3.1	YOLOv8 simulation/hardware testing	70%
	3.2	MPC implementation and tuning	
	3.3	Transitioning the project to the mobile platform	0%
	3.4	Initial tests/experiments	0%
	3.5	Tuning of the parameters	0%
	3.6	Debugging	0%
	3.7	Tests/Experiments	0%
	4	**Presentation preparation**	
	4.1	Collecting data (images, graphs)	0%
	4.2	Performance evaluation (running tests)	0%
	4.3	Project Overview	0%
	4.4	Presentation preparation	0%

Second half of the year

Figure 8.1: Gantt Chart with Future Work.

According to our Gantt chart, we are pleased to report that we have successfully completed all the tasks scheduled for the first half of the year. This includes achieving critical milestones such as developing a fully functional simulation environment in Gazebo. Within this simulation, we have integrated and validated key features including navigation, localization, and human detection packages, ensuring they operate seamlessly together without occupying too much computational capacity necessary for further MPC implementation.

By the end of the academic year, we successfully implemented an NMPC-based controller for a mobile robot in a simulated environment, achieving promising results in trajectory tracking and obstacle avoidance. Ongoing work focuses on fine-tuning the cost function weights, enhancing the logic for dynamic obstacle handling, and exploring the integration of Speed and Separation Monitoring (SSM) constraints for safer velocity control.

After completing the fine-tuning of MPC implementation in simulated environment, we will be transitioning from the simulation environment to real-world testing. During this phase, we will rigorously evaluate the system's performance in practical scenarios, validating the reliability of navigation, localization, and human detection under real-world conditions. This step is crucial for ensuring that the robot performs as expected outside the controlled simulation environment and provides valuable insights for further improvement of the system.

Overall, these steps mark a pivotal advancement in our project, and we look forward to tackling these challenges in the upcoming months.

Bibliography

[1] M. Cognominal, K. Patronymic, and A. Wańkowicz, "Evolving field of autonomous mobile robotics: Technological advances and applications," Fusion of Multidisciplinary Research, An International Journal, vol. 2, no. 2, pp. 189-200, 2021.

[2] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," Tsinghua Science and Technology, vol. 26, no. 5, pp. 674-691, 2021.

[3] D. Bozhinoski *et al.*, "Safety for mobile robotic systems: A systematic mapping study from a software engineering perspective," *Journal of Systems and Software*, vol. 151, pp. 150-179, 2019.

[4] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in AAAI, 2005, pp. 1324-1329.

[5] R. Biswas et al., "Towards object mapping in non-stationary environments with mobile robots," in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, IEEE, 2002, pp. 1014-1019.

[6] D. Hahnel et al., "Map building with mobile robots in dynamic environments," in 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), vol. 2, IEEE, 2003, pp. 1557-1563.

[7] J. Almeida, A. Almeida, and R. Araújo, "Tracking multiple moving objects for mobile robotics navigation," in 2005 IEEE Conference on Emerging Technologies and Factory Automation, vol. 1, IEEE, 2005, pp. 8-210.

[8] A. AlAttar, D. Chappell, and P. Kormushev, "Kinematic-Model-Free Predictive Control for Robotic Manipulator Target Reaching With Obstacle Avoidance," *Frontiers in Robotics and AI*, vol. 9, Feb. 2022, doi: https://doi.org/10.3389/frobt.2022.809114

[9] P. Salvini, D. Paez-Granados, and A. Billard, "Safety concerns emerging from robots navigating in crowded pedestrian areas," International Journal of Social Robotics, vol. 14, no. 2, pp. 441-462, 2022.

[10] G. A. Kebede et al., "Review of the characteristics of mobile robots for health care application," International Journal of Intelligent Robotics and Applications, pp. 1-23, 2024.

[11] C. Hofner and G. Schmidt, "Path planning and guidance techniques for an autonomous mobile cleaning robot," Robotics and Autonomous Systems, vol. 14, no. 2-3, pp. 199-212, 1995.

[12] D. Rodriguez-Losada et al., "Urbano, an interactive mobile tour-guide robot," in Advances in Service Robotics, IntechOpen, 2008.

[13] K. Eder, C. Harper, and U. Leonards, "Towards the safety of human-in-the-loop robotics: Challenges and opportunities for safety assurance of robotic co-workers," in The 23rd IEEE International Symposium on Robot and Human Interactive Communication, IEEE, 2014, pp. 660-665.

[14] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology," AIChE Symposium Series, vol. 93, no. 316, pp. 232-256, 1997. New York, NY: American Institute of Chemical Engineers.

[15] F. Kuhne, W. F. Lages, and J. G. da Silva Jr., "Model predictive control of a mobile robot using linearization," in Proceedings of Mechatronics and Robotics, vol. 4, no. 4, pp. 525-530, 2004.

[16] M. Ghandour, H. Liu, N. Stoll and K. Thurow, "A hybrid collision avoidance system for indoor mobile robots based on human-robot interaction," 2016 17th International Conference on Mechatronics - Mechatronika (ME), Prague, Czech Republic, 2016, pp. 1-7.

[17] A. Pandey, S. Pandey, and D. R. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review," Int. Rob. Auto. J., vol. 2, no. 3, pp. 00022, 2017.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," The International Journal of Robotics Research, vol. 5, no. 1, pp. 90-98, 1986, doi: 10.1177/027836498600500106.

[19] A. Sgorbissa and R. Zaccaria, "Planning and obstacle avoidance in mobile robotics," Robotics and Autonomous Systems, vol. 60, no. 4, pp. 628-638, 2012.

[20] Z. Sun, Y. Xia, L. Dai, K. Liu, and D. Ma, "Disturbance Rejection MPC for Tracking of Wheeled Mobile Robot," in *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2576-2587, Dec. 2017, doi: 10.1109/TMECH.2017.2758603.

[21] S. Yu, Y. Guo, L. Meng, T. Qu, and H. Chen, "MPC for Path Following Problems of Wheeled Mobile Robots," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 247-252, 2018, doi: 10.1016/j.ifacol.2018.11.021.

[22] G. Droge, P. Kingston, and M. Egerstedt, "Behavior-Based Switch-Time MPC for Mobile Robots," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 408-413.

[23] N. Eslami and R. Amiadifard, "Moving Target Tracking and Obstacle Avoidance for a Mobile Robot Using MPC," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, IEEE, 2019, pp. 1163-1169.

[24] P. Li, S. Wang, H. Yang and H. Zhao, "Trajectory Tracking and Obstacle Avoidance for Wheeled Mobile Robots Based on EMPC With an Adaptive Prediction Horizon," in IEEE Transactions on Cybernetics, vol. 52, no. 12, pp. 13536-13545, Dec. 2022, doi: 10.1109/TCYB.2021.3125333.

[25] L. Pacheco and N. Luo, "Testing PID and MPC Performance for Mobile Robot Local Path-Following," International Journal of Advanced Robotic Systems, vol. 12, no. 11, 2015, doi: 10.5772/61312.

[26] F. Künhe, J. Gomes, and W. Fetter, "Mobile robot trajectory tracking using model predictive control," in *Proceedings of the IEEE Latin-American Robotics Symposium*, vol. 51, pp. 5-10, 2005.

[27] T. P. Nascimento, C. E. T. Dórea, and L. M. G. Gonçalves, "Nonholonomic mobile robots' trajectory tracking model predictive control: A survey," Robotica, vol. 36, pp. 676–696, 2018, doi: 10.1017/S0263574717000637.

[28] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. I. Mann, R. G. Gosine, and M. Diehl, "Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs," IEEE Trans. Control Syst. Technol., vol. 24, no. 4, pp. 1394–1406, Jul. 2016.

[29] A. Tahirovic and G. Magnani, "General framework for mobile robot navigation using passivity-based MPC," IEEE Trans. Autom. Control, vol. 56, no. 1, pp. 184–190, Jan. 2011.

[30] A. S. Lafmejani and S. Berman, “Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots,” Robotics Auton. Syst., vol. 141, p. 103774, 2021.

[31] A. Brooks, T. Kaupp, and A. Makarenko, “Randomised MPC-based motion-planning for mobile robot obstacle avoidance,” in Proc. 2009 IEEE Int. Conf. Robot. Autom., May 2009, pp. 3962–3967.

[32] I. Maurović, M. Baotić, and I. Petrović, “Explicit model predictive control for trajectory tracking with mobile robots,” in Proc. 2011 IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM), Jul. 2011, pp. 712–717.

[33] F. Bertoncelli, F. Ruggiero, and L. Sabattini, “Linear time-varying MPC for nonprehensile object manipulation with a nonholonomic mobile robot,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2020, pp. 11032–11038.

[34] A. J. Prado, D. Chávez, O. Camacho, M. Torres-Torriti, and F. A. Cheein, “Adaptive nonlinear MPC for efficient trajectory tracking applied to autonomous mining skid-steer mobile robots,” in 2020 IEEE ANDESCON, 2020, pp. 1–6.

[35] Y. Song, Q. Zhang, Z. Hu, and J. Liu, ”Safe and Robust Human Following for Mobile Robots Based on Self-Avoidance MPC in Crowded Corridor Scenarios,” in Proc. 2023 IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), Shenzhen, China, Dec. 2023, pp. 1–6.

[36] Hu, Y., Su, H., Fu, J., Karimi, H. R., Ferrigno, G., De Momi, E., and Knoll, A. (2020). Nonlinear model predictive control for mobile medical robot using neural optimization. IEEE Transactions on Industrial Electronics, 68(12), 12636-12645.

[37] J. G. Storms and D. M. Tilbury, ”Blending of human and obstacle avoidance control for a high speed mobile robot,” in Proc. 2014 American Control Conference (ACC), Portland, OR, USA, Jun. 2014, pp. 3488–3493. IEEE.

[38] T. Hielscher, L. Heuer, F. Wulle, and L. Palmieri, ”Towards using fast embedded model predictive control for human-aware predictive robot navigation,” arXiv preprint arXiv:2405.12616, 2024.

[39] A. J. Mahmud, A. H. Raj, D. M. Nguyen, X. Xiao, and X. Wang, ”Human-Robot Co-Transportation with Human Uncertainty-Aware MPC and Pose Optimization,” arXiv preprint arXiv:2404.00514, 2024.

[40] A. Oleinikov, S. Kusdavletov, A. Shintemirov, and M. Rubagotti, ”Safety-Aware Nonlinear Model Predictive Control for Physical Human-Robot Interaction,” IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 5665-5672, July 2021, doi: 10.1109/LRA.2021.3083581.

[41] S. Rabiee and J. Biswas, ”A Friction-Based Kinematic Model for Skid-Steer Wheeled Mobile Robots,” in Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, May 2019, pp. 8563-8569, doi: 10.1109/ICRA.2019.8794373.

Chapter 9: **Appendix**

The code repository on GitHub:

`https://github.com/mdinmukhamet9/CapstoneProject/tree/d_temp`

```
geometry_msgs::msg::PoseStamped MPCController::selectGoalPose(
    const geometry_msgs::msg::PoseStamped& current_pose,
    const nav_msgs::msg::Path& global_plan,
    rclcpp::Logger logger,
    rclcpp::Clock::SharedPtr clock)
{
    geometry_msgs::msg::PoseStamped goal;
    goal.header = global_plan.header;
    // 1. Handle empty plan case
    if (global_plan.poses.empty()) {
        RCLCPP_WARN_THROTTLE(logger, *clock, 1000, "Empty path received");
        return goal;
    }
    // 2. Get current position
    const double current_x = current_pose.pose.position.x;
    const double current_y = current_pose.pose.position.y;
    // 3. Calculate distances
    const auto& final_pose = global_plan.poses.back().pose.position;
    const double dist_to_final = std::hypot(
        final_pose.x - current_x,
        final_pose.y - current_y);
    // 4. Determine lookahead distance
    double lookahead = min_lookahead_distance_;
    if (dist_to_final > max_lookahead_distance_) {
        lookahead = max_lookahead_distance_;
    } else if (dist_to_final > min_lookahead_distance_) {
        lookahead = dist_to_final;
    }
    // 5. Find goal point
    size_t goal_index = global_plan.poses.size() - 1;
    double accumulated_dist = 0.0;
    for (size_t i = 1; i < global_plan.poses.size(); ++i) {
        const double dx = global_plan.poses[i].pose.position.x -
                          global_plan.poses[i-1].pose.position.x;
        const double dy = global_plan.poses[i].pose.position.y -
                          global_plan.poses[i-1].pose.position.y;
        accumulated_dist += std::hypot(dx, dy);

        if (accumulated_dist >= lookahead) {
            goal_index = i;
            break;
        }
    }
    // 6. Ensure minimum distance requirement
```

```
    double actual_dist = std::hypot(
        global_plan.poses[goal_index].pose.position.x - current_x,
        global_plan.poses[goal_index].pose.position.y - current_y);
    if (actual_dist < min_lookahead_distance_ && goal_index < global_plan.poses.size() - 1) {
        for (size_t i = goal_index + 1; i < global_plan.poses.size(); ++i) {
            actual_dist = std::hypot(
                global_plan.poses[i].pose.position.x - current_x,
                global_plan.poses[i].pose.position.y - current_y);

            if (actual_dist >= min_lookahead_distance_ || i == global_plan.poses.size() - 1)
    {
                goal_index = i;
                break;
            }
        }
    }
    // 7. Handle orientation
    goal = global_plan.poses[goal_index];
    if (goal_index < global_plan.poses.size() - 1) {
        // For intermediate goals, use next point
        const auto& next_pose = global_plan.poses[goal_index + 1];
        const double yaw = std::atan2(
            next_pose.pose.position.y - goal.pose.position.y,
            next_pose.pose.position.x - goal.pose.position.x);
        tf2::Quaternion q;
        q.setRPY(0, 0, yaw);
        goal.pose.orientation = tf2::toMsg(q);
    } else if (goal_index > 0) {
        // For final goal, use previous point
        const auto& prev_pose = global_plan.poses[goal_index - 1];
        const double yaw = std::atan2(
            goal.pose.position.y - prev_pose.pose.position.y,
            goal.pose.position.x - prev_pose.pose.position.x);

        tf2::Quaternion q;
        q.setRPY(0, 0, yaw);
        goal.pose.orientation = tf2::toMsg(q);
    }

    // 8. Debug output
    RCLCPP_INFO(logger_,"Selected goal %zu/%zu: (%.3f, %.3f) @ %.3frad | Lookahead: %.3fm |
    Dist to final: %.3fm | Actual dist: %.3fm \n",
        goal_index, global_plan.poses.size() - 1,
        goal.pose.position.x, goal.pose.position.y,
        tf2::getYaw(goal.pose.orientation),
        lookahead, dist_to_final, actual_dist);

    return goal;
}
```

Listing 9.1: Excerpt from `mpc_controller.cpp`