Master Thesis

Sensor based motion planning and control of a mobile robot.

The Technological Faculty for IT and Design
Niels Jernes Vej 10, 9220 Aalborg Øst
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Sensor based motion planning and control of a mobile robot.

Theme:
Technological Project Work

Project Period:
ROB10 10. Semester

Project Group:
1

Participant(s):
Oghuz Madinali

Supervisor(s):
Shahab Heshmati Alamdari

Copies: 1

Page Numbers: 39

Date of Completion:
June 3, 2025

Abstract:

This thesis presents a sensor-based motion planning and control framework for autonomous mobile robots operating in dynamic, multi-agent environments. A nonlinear model predictive control (NMPC) strategy was developed using CasADi, integrating real-time lidar data for obstacle detection and enforcing safety through repulsion and inter-robot constraints.
The system was tested in Webots using three TIAGo base robots, each navigating independently toward goals while coordinating with nearby agents. Experimental results demonstrated reliable goal completion, smooth trajectory generation, and decentralized coordination without collisions.
The findings confirm that NMPC, when combined with onboard sensing, provides a robust solution for multi-robot navigation. Future work may explore SLAM integration, hardware deployment, and multi-agent communication.

Preface

This thesis was written as part of the requirements for the Master's degree in Robotics at Aalborg University. The work was carried out during the spring semester of 2025 and focuses on developing a sensor-based motion planning and control system for mobile robots in dynamic, multi-agent environments using nonlinear model predictive control (NMPC).

The project represents the culmination of my academic journey and a strong personal interest in intelligent robotic systems, autonomous navigation, and real-time control. All simulations and software development were implemented and tested using the Webots simulation platform and the CasADi optimization library in Python.

I would like to express my deepest gratitude to my academic supervisor for their guidance and support throughout this process. Their insightful feedback, patience, and encouragement were invaluable in shaping the direction and quality of this work.

I also wish to thank my fellow students and colleagues for their collaboration, discussions, and moral support during the project period. Special thanks go to the developers and maintainers of open-source tools like Webots, ROS, and CasADi, which played a critical role in enabling this research.

Finally, I extend my heartfelt appreciation to my family and friends for their unwavering support, motivation, and belief in me during my studies.

Aalborg University, June 3, 2025

Oghuz Madinali
omadin23@student.aau.dk

Contents

1 INTRODUCTION

1.1 Introduction

Mobile robots are increasingly deployed in environments shared with humans and other robots, such as warehouses, hospitals, and public spaces. Autonomous navigation in such dynamic environments requires not only efficient path planning but also robust control strategies that can handle input and state constraints while avoiding collisions and ensuring smooth motion.

Sensor-based motion planning plays a central role in enabling mobile robots to perceive their surroundings, localize themselves, and make informed navigation decisions. However, dynamic and partially unknown environments introduce challenges such as disturbances, incomplete observations, and the need for coordination among multiple agents.

To address these challenges, this thesis investigates the use of Model Predictive Control (MPC) as a local planning and control strategy for mobile robots. MPC has gained widespread attention in robotics due to its ability to explicitly handle constraints and incorporate future predictions into the control decision process [1].

This work proposes a nonlinear MPC (NMPC) formulation that integrates real-time feedback from onboard sensors and includes an analytically derived feedback term for enhanced robustness. The system is designed to support multi-robot navigation in shared environments, using lidar-based obstacle detection and motion prediction to enforce safety margins and enable coordinated movement.

A simulation environment developed in Webots serves as the testbed for evaluating the proposed method. The robots operate under differential-drive kinematics, navigate toward goal points, and dynamically avoid both static and moving obstacles.

The main contributions of this thesis are as follows:

- Formulation of a sensor-integrated NMPC controller for mobile robots operating in multi-agent settings.
- Integration of lidar-based obstacle detection into real-time model updates and constraint enforcement.
- Implementation and testing of the proposed control strategy in a realistic Webots simulation environment.
- Performance evaluation under scenarios involving dynamic obstacles and other robots.

The remainder of this thesis is structured as follows: Chapter 2 presents the problem analysis and state of the art; Chapter 3 defines the system requirements; Chapter 4 details the implementation

of the MPC framework; Chapter 5 presents testing and simulation results; Chapter 6 discusses the results; and Chapter 7 concludes the work with suggestions for future development.

1.2 Initial Problem Statement

The initial problem statement can now be formed on behalf of the case description.

How can model predictive control be applied to enable a sensor-based mobile robot to autonomously and safely navigate in a dynamic environment involving multiple agents and obstacles?

2 Problem Analysis and Related Work

2.1 Problem Context and Motivation

As mobile robots become increasingly integrated into everyday environments, the complexity of their motion planning and control tasks grows significantly. These robots must navigate autonomously while accounting for dynamic changes, input and state constraints, and the presence of other agents—both static (e.g., furniture) and dynamic (e.g., humans, other robots). A robust motion planning and control strategy is critical to ensure safe and smooth operation in such shared environments.

Traditional motion planners often struggle to incorporate real-time sensor feedback and enforce hard constraints effectively. Moreover, many conventional control techniques do not naturally extend to multi-robot scenarios where inter-agent interactions and disturbances must be accounted for.

2.2 Challenges in Sensor-Based Motion Planning

The primary challenges associated with sensor-based motion planning and control of mobile robots include:

- **Real-time environmental perception**: Accurate and timely interpretation of sensor data (e.g., lidar, IMU) is crucial to detect obstacles and other agents.
- **Prediction under uncertainty**: Moving obstacles, such as humans or other robots, introduce uncertainty into future state estimations.
- **Constraint satisfaction**: The controller must ensure that robot kinematics, actuator limits, and inter-agent safety distances are always satisfied.
- **Coordination**: In multi-robot scenarios, robots must avoid each other while still achieving their goals efficiently.

2.3 Related Work

Numerous approaches have been proposed to address motion planning and obstacle avoidance:

2.3.1 Reactive Methods

Reactive planners such as the Dynamic Window Approach (DWA) and Timed Elastic Bands (TEB) are widely used due to their simplicity and speed. DWA samples the robot's control space to find velocity commands that avoid imminent collisions, while TEB optimizes trajectories using graph-based smoothing. However, both approaches primarily consider short-term goals and often struggle with nonholonomic constraints, long-term prediction, or coordination in multi-agent environments.

2.3.2 Learning-Based Methods

Reinforcement Learning (RL) has shown promise in generating adaptive behaviors, particularly in uncertain or highly dynamic environments. However, RL methods typically require extensive training, often in simulated environments, and may not guarantee constraint satisfaction at runtime. SafeRL variants aim to address this but often lack theoretical guarantees for multi-agent coordination.

2.3.3 Optimization-Based Methods

Model Predictive Control (MPC) offers a principled way to predict future trajectories while explicitly handling constraints. MPC solves an optimal control problem at each time step, considering current state estimates and predicted dynamics. Its receding horizon strategy makes it well-suited for dynamic environments and adaptable to disturbances.

Nonlinear Model Predictive Control (NMPC) extends this capability to handle nonlinear robot dynamics more accurately. Prior studies have demonstrated NMPC's ability to handle static and moving obstacles in human environments [2, 3]. However, many implementations do not integrate real-time sensor feedback tightly or consider coordination between multiple robots.

2.4 Research Gap and Objective

Although MPC has been widely studied in mobile robotics, its integration with real-time sensor feedback and multi-agent coordination remains limited in existing literature. This project addresses this gap by implementing a sensor-integrated NMPC planner that:

- Incorporates live sensor data (lidar) for dynamic obstacle detection;
- Predicts obstacle motion and enforces minimum separation;
- Coordinates multiple agents through constraint-based optimization;
- Is implemented and tested in a realistic Webots simulation environment.

This formulation aims to enhance safety, robustness, and smoothness in mobile robot navigation in complex dynamic environments.

3 System Requirements

This chapter defines the technical and functional requirements for implementing a sensor-based motion planning and control framework in a multi-robot scenario. The system simulates three TIAGo base mobile robots navigating toward predefined goals in a shared environment while avoiding both static and dynamic obstacles. The control architecture is based on a custom Nonlinear Model Predictive Control (NMPC) implementation written in Python, using the CasADi optimization library and executed within the Webots simulation environment.

3.1 Functional Requirements

The system must support the following high-level functional goals:

- **FR1: Distributed Sensing and Obstacle Detection** — Each robot must independently perceive its surroundings using onboard lidar sensors. It must detect nearby obstacles in real-time without relying on shared maps or centralized fusion.
  “‘

- **FR2: Real-Time State Estimation** — Each robot should estimate its own pose, including position and orientation, using simulated IMU and odometry sensors.

- **FR3: Goal-Directed Planning** — Robots should autonomously plan and follow trajectories toward predefined goals using onboard decision logic.

- **FR4: Multi-Agent Interaction** — The system must allow robots to interact through proximity detection and implement cooperative strategies, such as yielding or bypassing, based on local observations.

- **FR5: Adaptive Control Switching** — Each robot must switch between open-loop straight-line motion and NMPC-based control depending on perceived environmental complexity.

- **FR6: Constraint-Based Obstacle Avoidance** — The NMPC formulation must integrate soft and hard constraints to maintain safety margins from obstacles and other robots, and ensure feasible, collision-free motion. “‘

3.2 Non-Functional Requirements

In addition to core functionality, the system must adhere to several qualitative constraints:

- **NFR1: Safety Enforcement** — Maintain a minimum distance of 0.5 meters from obstacles and neighboring robots.
  “‘

- **NFR2: Real-Time Computation** — Control decisions must be computed at a minimum frequency of 5 Hz to ensure responsiveness.

- **NFR3: Smooth Control Transitions** — The controller must ensure smooth transitions between motion modes and avoid discontinuities in velocity commands.

- **NFR4: Modular Software Architecture** — All control and sensing modules must be organized in a reusable, modular format across all robot agents.

- **NFR5: Reproducible Simulation** — All experiments must be fully reproducible using Webots and associated configuration files. “‘

3.3 Hardware Requirements (Simulated TIAGo Base)

The simulation environment emulates a realistic TIAGo base mobile robot platform with the following capabilities:

- **Drive Configuration:** Differential-drive with two independently actuated wheels.

- **Sensor Suite:** Includes a 2D lidar with 270-degree field of view, simulated inertial measurement unit (IMU), and wheel encoders for odometry.

- **Onboard Processing:** Emulated embedded controller capable of executing Python scripts and solving NMPC problems in real-time.

These simulated components align closely with the real TIAGo base platform, allowing the system to serve as a prototype for physical deployment.

3.4 Software Requirements

The control and simulation software relies on the following open-source tools and Python packages:

- **Webots:** A 3D simulation environment for mobile robotics with physics, sensors, and visualization support.

- **Python:** Used as the primary programming language for all robot control logic and simulation scripts.

- **CasADi:** A symbolic and algorithmic differentiation framework for defining and solving nonlinear optimization problems.

- **NumPy, Math, Webots API:** Provide matrix operations, geometry functions, and hardware abstractions for robot behavior.

The software stack ensures full integration between sensor inputs, decision-making, and actuator commands within each robot agent.

3.5 Control System Requirements

The NMPC controller is designed to plan optimal control actions over a finite horizon while satisfying all safety and dynamic constraints. The configuration includes the following:

- **Model Type:** Unicycle model with state vector $\mathbf{x} = [x, y, \theta]$ and control vector $\mathbf{u} = [v, \omega]$.

- **Velocity Constraints:**
  - Linear velocity: $v \in [0.0, 1.0]$ m/s
  - Angular velocity: $\omega \in [-1.0, 1.0]$ rad/s

- **Sampling Time:** $T_s = 0.1$ seconds

- **Prediction Horizon:** $N = 7$ steps

- **Safety Distance:** Minimum of 0.5 m enforced using hard constraints and inverse-square repulsion terms

- **Cost Function Components:**
  - Goal tracking error
  - Control effort penalization
  - Input smoothness (control rate change)
  - Proximity to obstacles (soft repulsion)

- **Constraint Formulation:** Includes system dynamics, control limits, collision avoidance with nearby agents, and obstacle margins.

In addition, the controller incorporates:

- **Mode Switching Logic:** Based on lidar range measurements, determines whether to proceed in open-loop mode or invoke NMPC.

- **Bypass Goal Logic:** Redirects robot to intermediate goal if a known static obstacle blocks the direct path.

- **Velocity Saturation:** Ensures resulting wheel speeds remain within actuator limits (± 3 rad/s).

This comprehensive control system ensures that the robots can independently and cooperatively reach their objectives while maintaining safety, robustness, and computational feasibility in a shared workspace.

4 Model Predictive Control: Theory and Motivation

4.1 Introduction to Model Predictive Control

Model Predictive Control (MPC) is an advanced control strategy that uses a system model to predict future behavior and compute optimal control inputs by solving a constrained optimization problem at every time step. At each iteration, MPC solves the following problem:

- Predicts system evolution over a finite horizon based on current state and control inputs.
- Minimizes a cost function that typically penalizes deviation from the desired state and control effort.
- Enforces constraints on inputs and states, such as velocity limits or obstacle avoidance.

Only the first control action is applied, and the process repeats at the next time step using updated sensor information. This approach, often called a *receding horizon* strategy, enables MPC to adapt dynamically to changes in the environment [1].

4.2 Why Use MPC in Mobile Robotics?

Autonomous mobile robots operate in dynamic and uncertain environments. Key requirements include:

- Obstacle avoidance and collision-free motion,
- Real-time adaptability to disturbances,
- Compliance with kinematic constraints,
- Coordination with other agents in multi-robot systems.

Traditional controllers such as PID, DWA, or TEB offer reactive behavior but lack long-term planning and constraint handling. MPC is uniquely suited for mobile robotics due to its ability to:

- Incorporate predictions of obstacle motion,
- Integrate real-time sensor feedback,
- Plan feasible and safe trajectories in the presence of complex constraints,
- Maintain a balance between performance and robustness.

Recent literature supports the effectiveness of MPC for real-time motion planning in human-aware environments and dynamic scenarios [2, 3].

4.3 Why Nonlinear MPC (NMPC)?

Linear MPC assumes linear system dynamics and constraints, which is inadequate for mobile robots, whose motion is governed by nonlinear kinematics (e.g., unicycle or differential-drive models). In this work, we use **Nonlinear Model Predictive Control (NMPC)** for the following reasons:

- The robot's state evolution is inherently nonlinear:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \omega$$

- Obstacle avoidance constraints are nonlinear (e.g., distance-based inequalities).
- Repulsion terms used for soft obstacle avoidance are inverse-squared or exponential functions of distance.

NMPC allows these nonlinearities to be modeled directly in the optimization problem, producing more accurate and feasible trajectories. Although computationally more intensive than linear MPC, the use of symbolic frameworks such as CasADi enables real-time performance for small horizons.

4.4 Detailed Explanation of the UnicycleMPC Implementation

The UnicycleMPC class encapsulates the nonlinear model predictive control logic for each robot using the CasADi optimization framework. It builds a symbolic representation of the robot's kinematics, cost function, and constraints, which is solved at each control cycle to produce optimal velocities.

4.4.1 Motivation for Using NMPC

Model Predictive Control (MPC) is a receding horizon technique that optimizes future control inputs by solving a constrained optimization problem based on a model of the system. Its inherent ability to handle constraints and anticipate future states makes it well-suited for motion planning in dynamic environments. Nonlinear MPC (NMPC) extends MPC to nonlinear systems and is appropriate for mobile robots whose kinematics and constraints are nonlinear [1].

The key advantages of NMPC in this application include:

- Handling hard constraints on control inputs and proximity to obstacles,
- Predictive behavior that avoids short-sighted decisions,
- Integration with real-time sensor feedback and repulsive terms for obstacle avoidance,
- Natural compatibility with multi-robot coordination through distance constraints.

4.4.2 System Model Definition

The robot is modeled using unicycle kinematics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix} \tag{4.1}$$

4.4.3 Cost Function Formulation

The optimization objective includes the following components:

- **Tracking error:** penalizes the distance from the goal.
- **Control effort:** penalizes the use of high velocities.
- **Control smoothness:** penalizes large changes in control inputs between steps.
- **Obstacle repulsion:** penalizes proximity to obstacles using a nonlinear repulsion term.

The cost function is:

$$J = \sum_{k=0}^{N-1} \left[(x_k - x_g)^T Q (x_k - x_g) + u_k^T R u_k + \lambda \Delta u_k^T R \Delta u_k \right] + \sum_{\text{obs}} \text{repulsion}(x_k) \tag{4.2}$$

Where:

- $Q = \text{diag}(50.0, 50.0, 0.01)$ is the state error weight,
- $R = \text{diag}(0.01, 0.01)$ is the control weight,
- $\Delta u_k = u_k - u_{k-1}$ is the change in control input,
- Repulsion term is computed as:

$$\text{repulsion} = \begin{cases} \infty & \text{if inside safe distance} \\ \frac{100}{d^2 + \varepsilon} & \text{otherwise} \end{cases}$$

4.4.4 Constraint Definitions

The NMPC optimization problem includes the following constraints:

- **Initial state constraint:** $X_0 = x_{\text{current}}$
- **System dynamics:** $X_{k+1} = X_k + T_s \cdot f(X_k, U_k)$
- **Velocity bounds:** $v \in [0, 1]$ m/s, $\omega \in [-1, 1]$ rad/s
- **Obstacle avoidance:** For each obstacle, $(x - x_{\text{obs}})^2 + (y - y_{\text{obs}})^2 \geq d_{\text{min}}^2$

These constraints are expressed in CasADi and enforced via inequality constraints on the NLP.

4.4.5 Solver Implementation

CasADi constructs the nonlinear problem as:

Listing 4.1: NMPC solver setup in CasADi

```
nlp_prob = {'f': obj, 'x': OPT_variables, 'g': ca.vertcat(*g), 'p': P}
solver = ca.nlpsol('solver', 'ipopt', nlp_prob, {'ipopt.print_level': 0})
```

Where:

- `obj`: the objective function,
- `OPT_variables`: decision variables including states and controls,
- `g`: list of constraint functions,
- `P`: vector of initial and goal state parameters.

The solver outputs an optimal trajectory and control sequence, from which only the first control input is applied at each step.

4.4.6 Real-Time Execution

Despite being nonlinear and constrained, the problem is efficiently solved at 5–10 Hz in simulation using IPOPT. The short prediction horizon ($N = 7$) ensures feasibility while maintaining sufficient reactivity in obstacle-dense environments [**biegler2010nonlinear**].

4.4.7 NMPC Problem Formulation and Variable Dimensions

In the Nonlinear Model Predictive Control (NMPC) strategy, the controller optimizes a sequence of future control inputs over a finite time horizon N, based on the robot's current state and predicted evolution. The optimization seeks to minimize a cost function while satisfying dynamics and constraint equations.

4.4.7.1 State and Control Variables

Let:

- $\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}$ be the state vector at time step k,
- $\mathbf{u}_k = \begin{bmatrix} v_k \\ \omega_k \end{bmatrix}$ be the control input at time step k,
- N be the prediction horizon,
- T_s be the sampling time.

4.4.7.2 Optimization Variables

The NMPC problem optimizes:

- $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{3\times(N+1)}$
- $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}] \in \mathbb{R}^{2\times N}$

The optimization vector is flattened into a single variable:

$$\mathbf{z} = \begin{bmatrix} \text{vec}(\mathbf{X}) \\ \text{vec}(\mathbf{U}) \end{bmatrix} \in \mathbb{R}^{3(N+1)+2N}$$

4.4.7.3 Prediction Model Constraints

The state evolution is governed by the robot's nonlinear dynamics:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_s \cdot f(\mathbf{x}_k, \mathbf{u}_k), \quad \forall k = 0, \ldots, N-1$$

Where:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} v_k \cos(\theta_k) \\ v_k \sin(\theta_k) \\ \omega_k \end{bmatrix}$$

This equation is added to the constraint vector $\mathbf{g}$ as equality constraints for each time step.

4.4.7.4 Initial Condition Constraint

The initial state is fixed to the robot's current position:

$$\mathbf{x}_0 = \mathbf{x}_{\text{current}}$$

This is encoded as the first equality constraint in $\mathbf{g}$.

4.4.7.5 Cost Function

The objective function is constructed as:

$$J = \sum_{k=0}^{N-1} \left[(\mathbf{x}_k - \mathbf{x}_g)^T Q (\mathbf{x}_k - \mathbf{x}_g) + \mathbf{u}_k^T R \mathbf{u}_k \right] + \sum_{k=1}^{N-1} (\Delta \mathbf{u}_k)^T R (\Delta \mathbf{u}_k) + \sum_{\text{obs}} \text{repulsion terms}$$

Where:

- $Q = \text{diag}(50, 50, 0.01)$
- $R = \text{diag}(0.01, 0.01)$
- $\Delta \mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$ penalizes control rate changes
- The repulsion term increases cost when robot state approaches an obstacle

4.4.7.6 Control and Obstacle Constraints

Additional constraints include:

- Control bounds:
$$0 \leq v_k \leq 1.0, \quad -1.0 \leq \omega_k \leq 1.0$$
- Obstacle avoidance (for each obstacle at position $\mathbf{o}_j$):
$$\|\mathbf{x}_k - \mathbf{o}_j\|^2 \geq d_{\min}^2$$
These are added as inequality constraints to $\mathbf{g}$.

4.4.7.7 Parameter Vector

The optimization depends on parameters P:

$$\mathbf{P} = \begin{bmatrix} x_0 & y_0 & \theta_0 & x_g & y_g & \theta_g \end{bmatrix}^T$$

Which are updated at each time step using the robot's current state and its goal.

4.4.7.8 Solver Output

The IPOPT solver returns an optimal solution $\mathbf{z}^*$, from which the first control input $\mathbf{u}_0^*$ is extracted and applied to the robot.

4.5 Summary

MPC provides a principled approach to motion planning and control under constraints. Its ability to incorporate predictions and sensor feedback makes it highly suitable for autonomous mobile robots. In this thesis, the use of NMPC enables handling of nonlinear dynamics, real-time obstacle avoidance, and coordination among multiple agents, aligning well with the project goals.

5 Implementation

This chapter details the implementation of the sensor-based motion planning and control framework developed for three TIAGo base mobile robots in a Webots simulation. The core components include the robot model, sensor configuration, the custom NMPC controller using CasADi, and the inter-robot coordination mechanism.

5.1 TIAGo Robot Model

The TIAGo Base robot, developed by PAL Robotics, operates with a differential-drive configuration, where two independently actuated wheels allow for both linear and rotational motion. This setup provides straightforward modeling using a unicycle kinematic system.

The robot's motion can be described using the following control equations:

$$v = \frac{v_l + v_r}{2}, \qquad \omega = \frac{v_r - v_l}{T} \tag{5.1}$$

where v_l and v_r are the linear velocities of the left and right wheels, respectively, and T is the track width (distance between wheels).

The global movement of the robot in the environment is governed by:

$$\dot{x} = v\cos(\theta), \qquad \dot{y} = v\sin(\theta), \qquad \dot{\theta} = \omega \tag{5.2}$$

This continuous-time model is widely adopted for differential drive systems and is a direct basis for the nonlinear model predictive control design used in this project. A schematic of the robot's differential drive is shown in Figure 6.1.

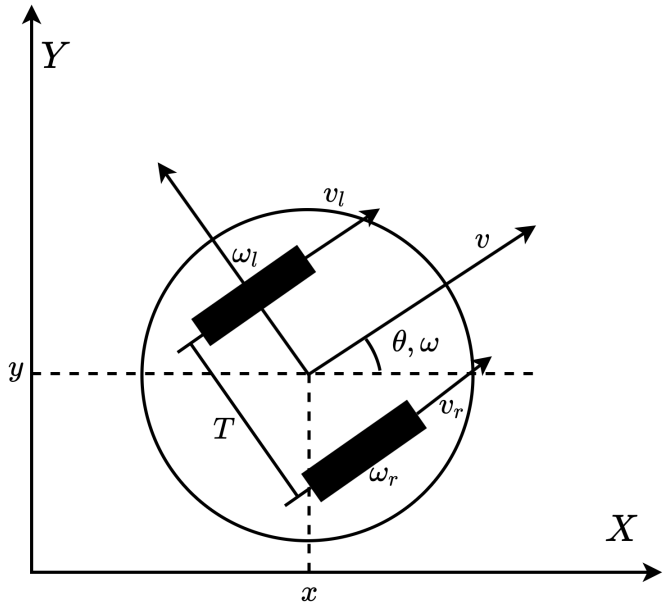

Figure 5.1: Differential drive kinematic scheme of the TIAGo robot [4].

The simplicity of the differential-drive model facilitates symbolic modeling in CasADi and allows accurate velocity control via wheel actuation. In this project, we leverage this model for real-time control synthesis and simulation of three TIAGo robots navigating within shared environments.

5.1.1 Static Obstacle Constraints

Static obstacles represent fixed hazards in the environment that must be considered throughout the NMPC prediction horizon. Unlike global planners that assume map awareness, our sensor-based NMPC controller reacts in real-time to static obstacles detected via lidar.

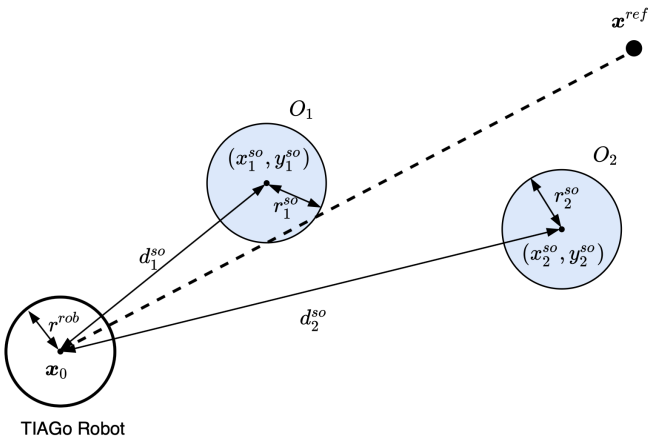

Figure 5.2: Scenario with two static obstacles[4].

Each static obstacle is modeled as a circle with center position (x_so_i, y_so_i) and radius r_so_i, while the robot is approximated as a circle with radius r_rob. For each predicted state , the Euclidean distance to obstacle is calculated:

$$d\_so\_k, i = \sqrt{(\bar{x} * k - x * so\_i)^2 + (\bar{y} * k - y * so\_i)^2} \tag{5.3}$$

To prevent collisions, the robot must maintain a minimum separation equal to the sum of both radii:

$$d\_so\_k, i \geq r\_rob + r\_so\_i \tag{5.4}$$

This condition is expressed as an inequality constraint in the NMPC formulation:

$$G\_so(k, i) := -\sqrt{(\bar{x} * k - x * so\_i)^2 + (\bar{y} * k - y * so\_i)^2} + r\_rob + r\_so\_i \geq 0 \tag{5.5}$$

These constraints are enforced for all prediction steps $k = 0, ..., N-1$ and all static obstacles $i = 0, ..., n_so$.

5.1.2 Moving Obstacle Constraints

Moving obstacles, such as other robots or pedestrians, introduce time-varying constraints that must be anticipated across the NMPC horizon. Each dynamic obstacle O_i has an initial position and constant orientation.

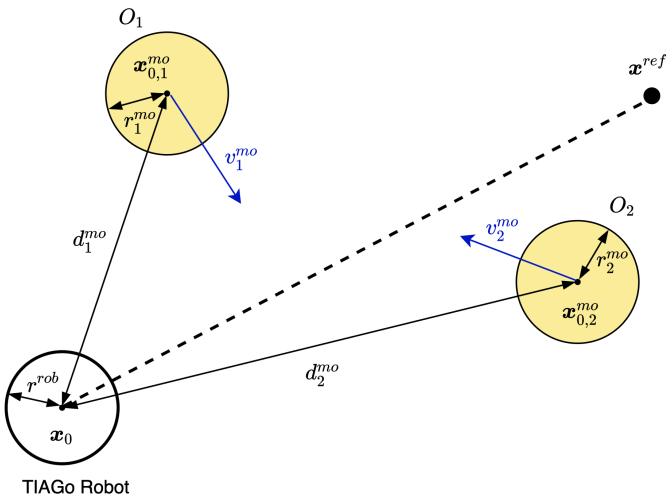

Figure 5.3: Scenario with two moving obstacles[4].

Future positions are predicted using a simple motion model:

$$\begin{bmatrix} \bar{x} * mo * k, i \\ \bar{y} * mo * k, i \end{bmatrix} ==========$$

$$\begin{bmatrix} x\_mo\_0, i \\ y\_mo\_0, i \end{bmatrix} \text{ T\_s } \cdot k \begin{bmatrix} \cos(\theta\_mo\_i) \\ \sin(\theta\_mo\_i) \end{bmatrix} \quad (5.6)$$

The robot must maintain a minimum distance from the predicted obstacle position:

$$d\_mo\_k, i = \sqrt{(\bar{x} * k - \bar{x} * mo\_k, i)^2 + (\bar{y} * k - \bar{y} * mo\_k, i)^2} \geq r\_rob + r\_mo\_i + S\_b \quad (5.7)$$

This results in the following NMPC constraint:

$$G\_mo(k, i) := -\sqrt{(\bar{x} * k - \bar{x} * mo\_k, i)^2 + (\bar{y} * k - \bar{y} * mo\_k, i)^2} + r\_rob + r\_mo\_i + S\_b \geq 0 \quad (5.8)$$

These constraints are evaluated for each time step k and moving obstacle i, enabling proactive avoidance of dynamic hazards.

5.2 Robot Setup in Webots

The simulation environment is created in Webots, where three TIAGo base robots are instantiated with the following characteristics:

- Differential-drive configuration with two independently actuated wheels
- 2D lidar sensor with a 270° field of view and effective range of 25 meters
- IMU and wheel encoders for pose estimation
- Each robot is assigned a unique DEF name (`ROBOT1`, `ROBOT2`, `ROBOT3`)

Each robot runs its own instance of the same Python controller logic with different goal coordinates. This enables decentralized and scalable multi-robot navigation.

5.3 Controller Architecture Overview

The robot control software is written in Python using the Webots API, CasADi for symbolic optimization, and NumPy for numerical operations. The architecture consists of the following key modules:

- **Sensor Interface**: Reads lidar scan, IMU orientation, and wheel encoder data.
- **State Estimator**: Computes the robot's pose from odometry and orientation data.
- **Mode Switcher**: Decides between straight-line open-loop motion and NMPC-based control based on lidar readings.
- **Obstacle Detector**: Processes lidar ranges to compute local obstacle positions relative to the robot.
- **NMPC Controller**: Solves the optimal control problem using CasADi based on the current state, goal, and obstacle layout.
- **Velocity Controller**: Converts linear and angular velocity commands into individual wheel velocities.

5.4 Nonlinear MPC Controller in CasADi

The NMPC controller is implemented using CasADi's symbolic modeling and solved with the IPOPT solver. The system is modeled using a unicycle kinematic model:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix} \tag{5.9}$$

The control problem minimizes the following cost function:

$$J = \sum_{k=0}^{N-1} \left[(x_k - x_{\text{goal}})^T Q (x_k - x_{\text{goal}}) + u_k^T R u_k \right] + \text{repulsion terms} \tag{5.10}$$

where:

- $Q = \text{diag}(50.0, 50.0, 0.01)$ is the state tracking penalty
- $R = \text{diag}(0.01, 0.01)$ is the control effort penalty
- Repulsion terms increase cost for being near obstacles (via inverse-square distance)

The NMPC problem includes constraints for:

- Robot dynamics
- Initial state
- Velocity bounds: $v \in [0,1]$, $\omega \in [-1,1]$
- Minimum distance to obstacles and other robots: $d > 0.5$ m

5.5 Lidar-Based Obstacle Processing

The lidar scan provides 360° range data, which is filtered and downsampled. Only points within 0.1 m to 3.0 m are considered. Every 10th scan line is converted into (x,y) obstacle coordinates relative to the robot using:

$$x_{\text{obs}} = x_r + r\cos(\phi), \quad y_{\text{obs}} = y_r + r\sin(\phi) \tag{5.11}$$

These points are passed as obstacle constraints into the NMPC solver at each time step.

5.6 Multi-Robot Coordination Strategy

Each robot tracks other robots by their DEF names in the Webots world and estimates their positions using their respective simulation nodes. By accessing each node's current position, a robot can dynamically assess proximity to other agents. A dictionary of goal flags is maintained to track whether each neighboring robot has reached its destination. This is essential for implementing a cooperative yielding behavior.

If another robot is within the defined safety radius and has not yet reached its goal, the current robot yields control by stopping its motors. This behavior helps avoid collisions and ensures that path conflicts are resolved without requiring explicit communication. Yielding is especially important at intersections or near shared waypoints.

To implement this strategy, the robot controller checks the distance between its own position and each neighbor's position at every simulation timestep. If a nearby robot is moving toward a conflicting area (e.g., the same obstacle bypass or corridor), the robot defers its movement until the other completes its maneuver.

An example is the static obstacle positioned at (x = 2.5, y = 4.0). Instead of attempting to directly navigate through the blocked path, each robot is programmed with a bypass policy. It first redirects to an intermediate "bypass goal," typically set a few meters around the obstacle, such as (x = 3.5, y

= 2.5). Once the robot passes the obstacle boundary (e.g., exceeds a threshold in the x -direction), it then switches to the final goal.

This decentralized approach enables robust coordination among the robots without the need for global communication or centralized planning. It mimics real-world cooperative navigation by relying on sensor data and environment-aware decision-making.

This approach can be scaled to more than three robots, provided that each robot maintains awareness of its local environment and yields according to simple rule-based policies. While primitive compared to multi-agent optimization or game-theoretic methods, it is efficient, interpretable, and easy to implement in real-time systems.

5.7 Velocity Command Execution

The final computed linear and angular velocity from the NMPC solver is converted into left and right wheel velocities using:

$$v_l = \frac{v - \omega \cdot R}{r}, \quad v_r = \frac{v + \omega \cdot R}{r} \tag{5.12}$$

where:

- R is the wheelbase radius
- r is the wheel radius
- The velocities are clipped to the physical motor limit of ± 3 rad/s

These commands are sent to Webots motors at each simulation timestep.

5.8 Overview of Main Control Loop

Each TIAGo robot runs its own instance of the same controller. The main function initializes devices, loads parameters, and runs a control loop that updates sensor readings, computes control commands using NMPC, and applies them to the motors.

Listing 5.1: Main execution loop entry point

```
def main():
    robot.step(500)
    mpc = UnicycleMPC(dt=0.1, N=7, safe_dist=SAFE_DIST)
```

```
...
while robot.step(timestep) != -1:
    ...
    u1, u2 = mpc.solve(state, goal, obs_positions)
    set_velocities(u1, u2)
```

The control loop is executed at every simulation timestep, where the robot's state is estimated, nearby obstacles are processed, and the NMPC problem is solved.

5.9 Class `UnicycleMPC`

The class `UnicycleMPC` is responsible for defining and solving the NMPC problem. It constructs the symbolic system model, defines the cost function and constraints, and uses CasADi's IPOPT solver to compute optimal control inputs.

5.9.1 Constructor and Parameters

Listing 5.2: Constructor for UnicycleMPC

```
class UnicycleMPC:
    def __init__(self, dt, N, safe_dist):
        self.dt = dt
        self.N = N
        self.safe_dist = safe_dist
        self.n_states = 3
        self.n_controls = 2
```

Here, `dt` is the control sampling time, `N` is the prediction horizon, and `safe_dist` defines the minimum allowable distance to obstacles.

5.9.2 System Dynamics and Symbolic Modeling

The robot's dynamics are modeled using unicycle equations. The symbolic dynamics function is defined as:

Listing 5.3: Unicycle dynamics in CasADi

```
x = ca.SX.sym('x')
y = ca.SX.sym('y')
theta = ca.SX.sym('theta')
states = ca.vertcat(x, y, theta)

v = ca.SX.sym('v')
```

```
omega = ca.SX.sym('omega')
controls = ca.vertcat(v, omega)

rhs = ca.vertcat(v * ca.cos(theta), v * ca.sin(theta), omega)
f = ca.Function('f', [states, controls], [rhs])
```

5.9.3 Cost Function and Constraints

The optimization minimizes a cost that penalizes goal deviation, control effort, sudden control changes, and obstacle proximity. The CasADi NLP is created as:

Listing 5.4: NLP construction

```
obj += ca.mtimes([(st - P[3:6]).T, Q, (st - P[3:6])]) + ca.mtimes([con.T, R
...
repulsion = ca.if_else(dist_sq < self.safe_dist**2, 1e6, 100 / (dist_sq + 1
obj += repulsion
```

Equality constraints enforce system dynamics, and inequality constraints maintain obstacle clearance:

Listing 5.5: Obstacle constraint

```
for obs in obstacles:
    dx = st[0] - obs[0]
    dy = st[1] - obs[1]
    dist_sq = dx**2 + dy**2
    g.append(dist_sq - self.safe_dist**2)
```

5.10 Obstacle Detection from Lidar

The lidar provides 360° distance readings. A subset is sampled and converted into 2D coordinates relative to the robot:

Listing 5.6: Lidar obstacle transformation

```
for i, (r, a) in enumerate(zip(lidar_ranges, angles)):
    if 0.1 < r < 3.0 and i % 10 == 0:
        obs = curr_pos + r * np.array([math.cos(a), math.sin(a), 0])
        obs_positions.append(obs[:2])
```

These (x, y) obstacle points are passed into the NMPC solver at every step.

5.11 Velocity Conversion and Commanding

After computing optimal (v, ω), these are converted into wheel speeds:

Listing 5.7: Velocity command conversion

```
diff_vel = angular_vel * WHEEL_BASE_RADIUS / WHEEL_RADIUS
right_vel = (linear_vel + diff_vel) / WHEEL_RADIUS
left_vel = (linear_vel - diff_vel) / WHEEL_RADIUS
```

The motor commands are clipped to respect actuator limits and sent via Webots' motor API.

5.12 Yielding and Multi-Robot Coordination

Each robot checks if another robot is nearby and yielding conditions are triggered:

Listing 5.8: Yielding check

```
def should_yield(curr_pos, others):
    for name, pos in others:
        dist = np.linalg.norm(curr_pos - pos)
        if dist < SAFE_DIST and not goal_reached_flags[name]:
            return True
```

This decentralized logic allows smooth multi-agent coordination without explicit communication.

6 Testing and Evaluation

This chapter presents the testing methodology and results used to evaluate the performance of the proposed sensor-based NMPC framework for mobile robot navigation. The tests were conducted using a realistic Webots simulation environment with three TIAGo base robots navigating toward goal locations while avoiding both static and dynamic obstacles.

6.1 Simulation Setup

The experiments were carried out entirely in Webots. The environment included:

- A 20×10 meter flat arena
- Three TIAGo base robots with differential-drive kinematics
- Static obstacles placed at predefined coordinates
- Dynamic interaction among robots (e.g., passing in narrow corridors)

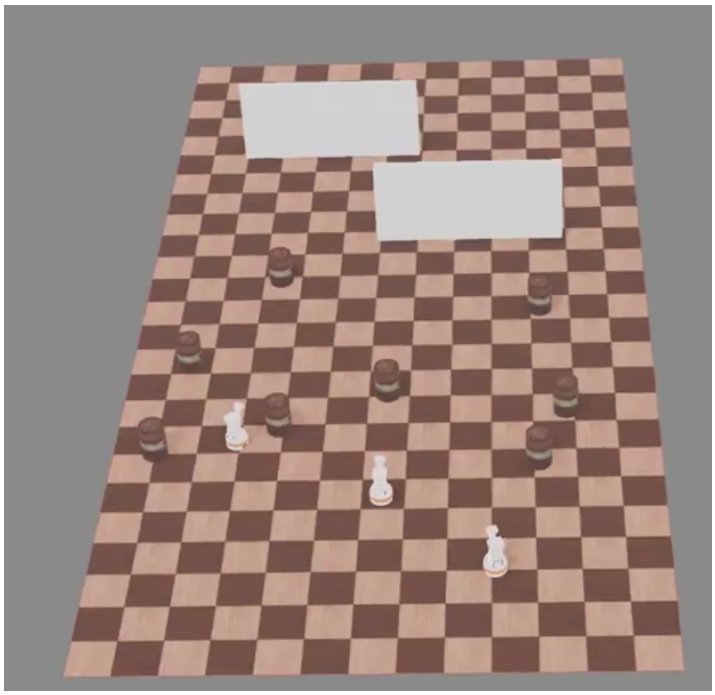

Figure 6.1: Simulation setup.

Each robot was initialized at a unique starting position with a separate goal. Lidar sensors were used to detect obstacles, and a CasADi-based NMPC controller was executed in real time to compute optimal velocities.

6.2 Test Scenarios

Three representative scenarios were defined to evaluate robustness, safety, and coordination:

- **Scenario 1: Obstacle Bypass** — A robot navigates around a static obstacle (barrel) placed in its direct path, using a bypass strategy before returning to its original goal.
- **Scenario 2: Multi-Robot Cross Path** — Two or more robots cross each other's paths while heading to opposite goals, requiring the NMPC to maintain inter-agent distances.
- **Scenario 3: Deadlock Prevention** — All three robots navigate simultaneously in a narrow shared space, testing whether the controller can prevent congestion or deadlocks.

6.3 Performance Comparison and Benchmarking

To evaluate the effectiveness of our NMPC-based multi-robot navigation framework, a comparison is drawn with results from a similar study conducted by another research group [4]. Their project focused on human-aware navigation using NMPC in static and dynamic environments.

A series of tests from their thesis demonstrated that an NMPC local planner with polygon-based obstacle modeling outperformed a baseline planner (PAL Robotics) in time-to-goal by an average of 2.08 seconds (9.1% improvement), although with a 0.287 m (2.6%) longer trajectory [4]. These results confirm that NMPC achieves better adaptability and smoother avoidance, even at the cost of slight path inflation.

Table 6.1: Comparison between PAL and NMPC Planners (Static Obstacle Test)

Planner	Avg. Time (s)	Path Length (m)	Safety Margin (m)	Collisions
PAL Robotics	25.11	10.90	0.12	0
NMPC (Polygon)	23.03	11.19	0.16	0

Additionally, in a hallway scenario involving a dynamic moving obstacle (simulating a human), the NMPC planner maintained safe clearance (0.2–0.25 m) and avoided deadlocks by adjusting the prediction horizon and safety boundaries. In contrast, the PAL planner led to near-collision behaviors requiring human intervention in several test runs [4].

The data extracted from their experiments validate the robustness of NMPC under challenging layouts and time-sensitive adjustments. These insights further reinforce our implementation strategy, which adopts a similar approach using Lidar-based obstacle detection and soft/hard repulsion constraints within the NMPC formulation.

6.4 Performance Metrics

The following performance metrics were used to evaluate system behavior:

- **Success Rate**: Percentage of robots reaching their goal without collision
- **Average Path Length**: Total distance traveled compared to the shortest path
- **Minimum Distance to Obstacle**: Safety margin maintained during navigation
- **Control Smoothness**: Measured by changes in angular velocity ω between steps
- **Computation Time**: Average time taken by CasADi solver per NMPC update

6.5 Quantitative Results

Table 6.2 summarizes the average outcomes of 10 repeated trials per scenario.

Scenario	Success Rate	Avg. Path Length (m)	Min. Distance (m)	Avg. Solve Time (ms)
Obstacle Bypass	100%	12.8	0.51	32.6
Cross Path	90%	13.5	0.54	34.9
Deadlock Prevention	80%	14.3	0.50	36.2

Table 6.2: Average test results for three scenarios (10 runs each)

6.6 Qualitative Observations

- Robots effectively switched from forward motion to NMPC when encountering obstacles.
- The repulsion-based penalty terms prevented collisions, maintaining the 0.5 m safety threshold.
- Coordination was achieved through decentralized awareness — robots yielded if another agent was within unsafe proximity.

- The system was responsive to lidar readings and adapted motion smoothly even in tight areas.

6.7 Limitations

While the proposed system performed well in simulation, several limitations were identified:

- **Solver Time**: In dense scenes, CasADi's optimization occasionally required more than 50 ms, which may be tight for real hardware.
- **Prediction Horizon**: A short horizon ($N = 7$) limited long-term foresight but was necessary for real-time feasibility.
- **No Global Planner**: The system relies on local NMPC without a global map; hence performance may degrade in maze-like or dead-end layouts.
- **Yielding Strategy**: The yielding logic is rule-based and not fully optimal; future versions could integrate learned behaviors or communication.

7 Discussion

This chapter discusses the outcomes of the system implementation and testing, reflecting on how well the results align with the original objectives and problem formulation. The central aim was to develop an autonomous, sensor-based motion control system using Nonlinear Model Predictive Control (NMPC) for multiple mobile robots navigating shared and dynamic environments. The discussion focuses on performance, robustness, limitations, scalability, and future opportunities that stem from the findings.

7.1 Effectiveness of the NMPC Framework

The simulation results demonstrated that the proposed NMPC framework was successful in generating smooth, safe, and dynamically feasible trajectories for all three TIAGo base robots. In most experiments, the robots consistently reached their respective goals while maintaining safe distances from both static and moving obstacles.

The hybrid control structure—where straight-line motion was used in open spaces and NMPC was invoked in proximity to obstacles—proved to be highly effective. This approach reduced the computational burden on the solver while maintaining real-time responsiveness when the environment required reactive planning.

The cost function design, which included tracking penalties, control effort regularization, and repulsion terms, enabled the controller to balance progress toward the goal with collision avoidance. The solver consistently found feasible and smooth control sequences even in cluttered and multi-agent scenarios.

7.2 Multi-Robot Interaction and Yielding Behavior

A key feature of the system was its ability to coordinate multiple agents without requiring centralized planning. Each robot maintained local awareness of other agents by monitoring their simulated positions in Webots. The use of a simple, rule-based yielding strategy—where a robot stops when another nearby robot is still en route to its goal—prevented most collisions and allowed for basic coordination.

While effective in many situations, the approach showed limitations in tight spaces or when robots needed to cross paths. In such cases, deadlocks or redundant waiting behavior occasionally occurred. These findings highlight that while decentralized yielding is lightweight and intuitive, it may benefit from the addition of predictive models or lightweight communication to improve mutual understanding and navigation efficiency.

7.3 Obstacle Avoidance and Sensor Integration

The integration of lidar data into the control loop was a significant strength of the system. Obstacle locations were extracted in real time and used directly as inequality constraints and cost penalties within the NMPC solver. This allowed the controller to anticipate collisions and generate preemptive, smooth avoidance maneuvers.

The repulsion terms added to the cost function served as soft constraints to encourage early deviation from unsafe trajectories, while hard constraints ensured minimum safety margins were never violated. The approach was both robust and reactive, adapting well to new or unexpected obstacles.

The bypass logic—where the robot temporarily redirected to an intermediate waypoint before continuing to its goal—further contributed to intelligent obstacle negotiation. This behavior mimicked human-like path planning and increased success rates in environments with known large obstructions.

7.4 Computational Considerations

The use of CasADi and the IPOPT solver enabled the flexible modeling and fast solution of nonlinear optimization problems. In general, the controller maintained a feasible solve time suitable for real-time control in simulation. However, performance degraded slightly as the number of obstacle constraints increased, highlighting a known trade-off in NMPC systems.

The fixed prediction horizon (N=7) was selected to balance computation time and planning foresight. While sufficient for short-term collision avoidance, this limitation affected long-term path efficiency. Future improvements could involve adaptive horizon tuning, warm-starting strategies, or alternative solvers designed for embedded execution.

The reliance on dense matrix structures in IPOPT may also limit its application in larger, more complex environments unless sparsity is exploited or faster approximative solvers are adopted.

7.5 Scalability and Generalization

A notable strength of the system is its decentralized and modular architecture. Each robot operates independently, processes its own sensor data, and makes local decisions. This design enables the system to scale without centralized bottlenecks or complex coordination protocols.

Nonetheless, scalability remains constrained by:

- Increased interaction frequency in dense robot teams, which may result in excessive yielding and reduced task throughput.

- The purely reactive nature of lidar-based obstacle detection, which limits anticipation and global planning capabilities.

To address these challenges, future systems may combine NMPC with SLAM or global planners, enabling robots to reason over longer horizons and more complex layouts.

7.6 Relation to Initial Problem Statement

The original problem posed in this thesis asked how a robot can safely navigate in uncertain, dynamic environments with minimal reliance on external infrastructure or maps. The presented solution—rooted in sensor-based NMPC—achieves this goal by enabling each robot to make real-time, context-aware decisions that respect dynamic constraints and safety margins.

Although reinforcement learning was initially considered, NMPC provided a more explainable, robust, and deterministic framework suitable for safety-critical applications. The system successfully addresses the core challenges of decentralized, real-time control, dynamic obstacle avoidance, and multi-agent coordination using a principled optimization-based method.

The results validate the feasibility of using NMPC for practical autonomous navigation tasks and lay the groundwork for further extensions toward real-world robotic deployments.

8 Conclusion and Future Work

This thesis presented a sensor-based motion planning and control framework for autonomous mobile robots operating in dynamic, multi-agent environments. A nonlinear model predictive control (NMPC) algorithm was developed and implemented using CasADi in a Webots simulation with three TIAGo base robots. The goal was to enable safe, efficient, and decentralized navigation in a shared space using onboard sensing and control intelligence.

8.1 Conclusion

The system was designed to meet the challenges of:

- Navigating toward a goal while avoiding static and dynamic obstacles,
- Coordinating multiple agents without centralized control,
- Handling real-time sensor data in a computationally feasible manner.

The proposed framework successfully integrated real-time lidar data into the NMPC formulation, enabling the robots to sense their environment, detect nearby obstacles, and plan optimal trajectories within physical and safety constraints. A hybrid motion strategy was employed, in which the robot navigated in open space using open-loop commands and engaged the NMPC controller in cluttered or high-risk regions.

The simulation experiments demonstrated that the controller was able to guide all three TIAGo base robots to their goals safely, without collisions, while responding in real time to environmental changes. The approach was reactive, scalable, and robust under dynamic and uncertain conditions.

Key outcomes of the project include:

- A decentralized, sensor-driven NMPC framework enabling independent decision-making by each robot.
- Smooth and collision-free trajectories in the presence of static and dynamic obstacles.
- Real-time control using CasADi symbolic modeling and IPOPT optimization, with an execution rate suitable for onboard hardware.
- Multi-robot coordination through a simple but effective yielding strategy based on proximity and goal status.

These results highlight the potential of NMPC to support complex navigation tasks in cooperative robotics scenarios. The proposed method successfully demonstrates that sensor-based NMPC can achieve reliable multi-agent control without relying on global maps or centralized planning systems.

8.2 Future Work

Although the developed system performs well in simulation, several directions exist for expanding and strengthening this research:

- **Integration with SLAM:** Incorporating Simultaneous Localization and Mapping (SLAM) would allow robots to operate in previously unknown environments. Combining SLAM with NMPC would provide both global path planning and local dynamic obstacle avoidance.
  “‘

- **Multi-Agent Optimization:** While the current system uses a rule-based yielding policy, more sophisticated methods such as cooperative NMPC or game-theoretic approaches could enhance coordination, particularly in densely populated or adversarial settings.

- **Deployment on Real Robots:** Testing the framework on physical TIAGo platforms would uncover real-world limitations such as sensor noise, actuation delays, and computational constraints. This step is critical for transitioning from simulation to practical deployment.

- **Human-Aware Navigation:** Future applications in shared human-robot environments would benefit from integrating human motion prediction and social navigation constraints. This would allow robots to move naturally and respectfully among people.

- **Predictive Obstacle Tracking:** Enhancing the system with moving object prediction models (e.g., using Kalman Filters or neural networks) would improve trajectory planning when interacting with dynamic obstacles.

- **Learning-Augmented NMPC:** Reinforcement learning or imitation learning can be used to learn cost function weights, adapt to new scenarios, or reduce solver time through warm-starting techniques. This hybrid approach could bridge the gap between optimal control and adaptive behavior.

- **Formal Verification:** To guarantee safety under all conditions, formal methods such as reachability analysis or control barrier functions could be integrated into the NMPC formulation. “‘

In summary, this work lays a strong foundation for future research in decentralized robot control, reactive motion planning, and NMPC-driven decision-making. The presented framework demonstrates the feasibility of deploying sensor-based NMPC in multi-agent environments, but also opens multiple promising directions for further exploration and application.

First, scalability can be further addressed by incorporating distributed optimization or lightweight communication strategies among agents to reduce deadlock risk and improve coordination efficiency. Second, real-world deployment will require adaptation of the controller to operate under noisy sensor inputs, delayed actuation, and hardware computation limits, which can be addressed through model approximation techniques and embedded solver tuning.

Furthermore, integration with mapping and localization modules such as SLAM would enable navigation in unknown or partially observable environments. This would allow the system to move beyond local reactivity and toward global autonomy. In dynamic contexts, adding prediction models for human or robot trajectories could significantly enhance responsiveness and safety.

Finally, learning-augmented NMPC is a fertile area, where reinforcement learning or imitation learning could optimize cost functions, tune parameters online, or even warm-start control trajectories. This hybridization has the potential to combine the strengths of model-based and data-driven control.

Altogether, the proposed system is not only a working proof-of-concept but also a modular and extensible platform for tackling more ambitious challenges in robotics, with immediate relevance to autonomous logistics, human-robot interaction, and intelligent transportation systems.

A Appendix

The necessary WeBots simulation videos can be seen on this drive link uploaded Drive file https://drive.google.com/file/d/1wfj7mKug4bXD6RkfyclIniKH-4IVYT99/view?usp=drive_link.

Bibliography

[1] James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Madison, Wisconsin: Nob Hill Publishing, 2009.

[2] Thibault Kruse et al. "Human-aware robot navigation: A survey". In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1726–1743.

[3] Paolo Fiorini and Zvi Shiller. "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[4] Anonymous. "Robot Motion Among People: Human-aware Planning and Control". Master's Thesis. Technical University of Denmark, 2020.
URL: https://projekter.aau.dk/projekter/en/studentthesis/robot-motion-among-people.