SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER, CONTROL AND MANAGEMENT ENGINEERING

Safe Robot Navigation in a Crowd Combining NMPC and CBF

AUTONOMOUS AND MOBILE ROBOTICS - GIUSEPPE ORIOLO

Supervisors:
Tommaso Belvedere
Michele Cipriano
Francesco D'Orazio

Students:
Davide De Santis
Davide Esposito Pelella
Paolo Ferretti

Academic Year 2023/2024

Contents

1 Introduction

In the rapidly evolving landscape of technology, mobile robots have emerged as versatile and indispensable tools, revolutionizing various industries and daily life. Equipped with sensors, processors and actuators, they possess the ability to move and perform tasks in any type of environment. From warehouses and manufacturing floors to healthcare facilities and offices. The evolution of mobile robots is deeply related with advancements in robotics, artificial intelligence, and sensor technologies. In this project a safe navigation algorithm is developed based on a Nonlinear Model Predictive Control (NMPC) solving at each control cycle an Optimal Control Problem (OCP) throughout a finite horizon. The OCP takes a desired trajectory as reference and considers at the same time different type of constraints: linear constraints to bound the speed and acceleration both of the wheels and the robot itself; linear constraints, functions of the position of the robot, to keep it inside convex regions free from fixed obstacles of the known structured environments such as walls and furniture; nonlinear constraints formulated via discrete-time Control Barrier Functions (CBFs) to perform collision avoidance of moving obstacles. Specifically, we assume that the exact positions of these obstacles are known, and using this information, we predict human movement over the prediction horizon. This safe navigation approach is inspired by the work on Vulcano [1]. The whole project is implemented in python using ROS and the simulations are performed on Gazebo environments using the TIAGo robot. The report is organized as follow: in chapter 2 the problem is formally defined; in chapter 3 and 4 the proposed approach is presented, respectively the crowd prediction module and the motion generation module are described; chapter 5 gives an overview on the practical implementation; in chapter 6 and 7 simulation environments, settings and results are shown; in the end in chapter 8 the conclusions are drawn up.

2 Problem Formulation

Let's consider the scenario where a TIAGo mobile robot with a differential drive base, (depicted in Figure 1) is tasked with navigating to a specified target point, G, located within an environment, E, filled with moving people.

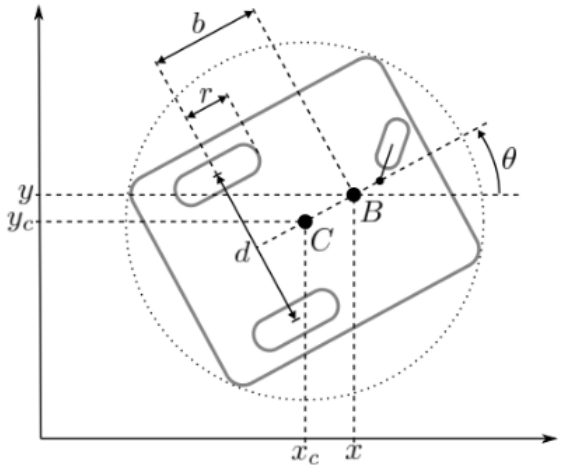

Figure 1: The differential drive mobile robot. Image taken from [1]

In this situation, the robot's configuration is represented by a vector $q = (x, y, \theta)$, where x and y denote the Cartesian coordinates of a reference point B, located along the sagittal axis at a distance $b > 0$ from the midpoint between the drive wheels, with θ indicating their shared orientation. The region occupied by the robot at any given configuration q is denoted as $R(q) \subset E$, and $H(t) \subset E$ represents the area occupied by humans at time t.
Assuming the robot's wheels move without slipping and that the robot's movements are controlled by adjusting the angular accelerations $\dot{\omega}_R$ and $\dot{\omega}_L$ of the right and left wheels respectively, the kinematics of the robot are described by the following model:

$$\dot{\xi} = f(\xi, u) = \begin{pmatrix} v\cos\theta - \omega b \sin\theta \\ v\sin\theta + \omega b\cos\theta \\ \omega \\ \frac{r}{2}(\dot{\omega}_R + \dot{\omega}_L) \\ \frac{r}{d}(\dot{\omega}_R - \dot{\omega}_L) \end{pmatrix}$$

where $\xi = (x, y, \theta, v, \omega)$ represents the state vector, encompassing the robot's configuration q and its driving and steering velocities v and ω, with $u = (\dot{\omega}_R, \dot{\omega}_L)$ as the control input vector, r as the wheel radius, and d as the distance between the wheel centers.
The goal of safely navigating through a crowd involves generating a real-time motion that suits the robot's design, avoids collisions with humans at all time instants, and leads to the target area G. Although the robot is equipped with onboard sensors for continuous data collection, our approach relies on the ground truth about the crowd's

position and movement instead of sensor-derived information and it is built around two key steps: predicting where the crowd will move (with the Crowd Prediction Module) and deciding how the robot should move (with the Motion Generation Module). These steps are carried out one after the other during each sampling interval to make sure the robot moves safely and effectively. We will delve more into the details in the next sections.

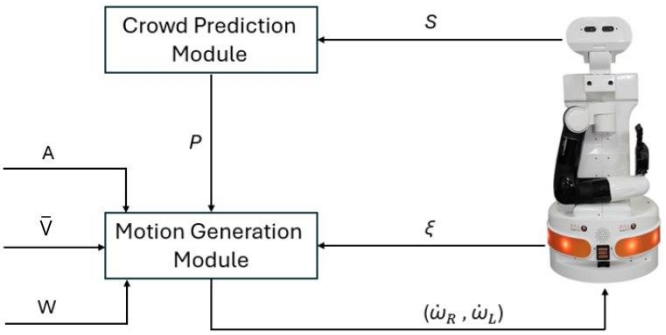

Figure 2: Block scheme of the proposed safe navigation stategy

The proposed strategy to obtain the desired behaviour is constituted by two modules, i.e., the crowd prediction and motion generation module, interconnected as shown in Figure 2. In particular at each sampling instant, the crowd prediction module uses the ground truth data S containing the current position of the humans and it predicts their position in the whole prediction horizon via discrete time integration.
The motion generation module initially receives the velocity references $\bar{V}$, the convex area A and the waypoints W. Using these waypoints, it constructs the complete reference path. At each sampling interval, this module also receives the current state of the robot ξ and the predicted crowd motion, encapsulated within set P. Based on the robot's current position, the module selects the appropriate segment of the reference path. The motion generation module runs a real time NMPC algorithm to produce wheel acceleration commands $(\dot{\omega}_R, \dot{\omega}_L)$ the robot must perform to accomplish the task as good as possible according to CBF constraints and convex regions constraints for collision avoidance of both moving and fixed obstacles. To select the correct convex region along the prediction horizon different heuristics can be chosen. In particular in this study four heuristics based on predicted robot position, last reached waypoint, nearest waypoint and time are tested.

3 Crowd Prediction

To address the challenges outlined in the previous section, our work, inspired by the original safe navigation framework, keeps the structure composed of two principal modules: crowd prediction and motion generation. However, we substitute the crowd sensor position information with precise ground truth data coming from the environment

and employ a differential numerical method to generate the crowd predicted motion on the considered horizon.
Then, at each sampling instant $t_k = k\delta$, we directly utilize this ground truth data as input to the crowd prediction module. This module is designed to predict the movements of M humans over the interval $[t_k, t_k + T_p]$, where $T_p = N\delta$ denotes the duration and N the number of sampling intervals within it. Specifically, the prediction for each person's movement is defined as:

$$P_k^j = (p_{0|k}^j, \ldots, p_{N|k}^j)$$

where $p_{i|k}^j$ represents the predicted absolute position of the j-th individual at the time $t_{k+i} = (k + i)\delta$. Once the first position of the human $p_{0|k}^j$ is extracted from the ground truth, the following predicted position are simply obtained via discrete time integration, considering that the human moves at constant speed $\dot{p}^j$, i.e.

$$p_{i+1|k}^j = p_{i|k}^j + i\delta\dot{p}^j \quad , \quad i = 0, \ldots, M$$

In this context, M denotes the total number of people within the environment considered for collision avoidance. Contrary to approaches that may limit M based on sensor range or other detection capabilities, our method assumes the capability to track and predict the movements of all individuals. This assumption is possible within our framework due to the availability of ground truth data, simplifying the model by eliminating the need for sensor-based information.

4 Motion Generation

4.1 Waypoints, Path Generation, and Reference Update

Waypoints are strategically selected to navigate through critical areas, such as narrow passages and regions that overlap multiple zones. It's important to note that the selection of waypoints depends on the layout of the environment. Figure 3 illustrates an example of a world with waypoints and the generated path, highlighting how waypoints are placed in key locations to guide the robot's movement efficiently.
The path between each pair of consecutive waypoints is generated using interpolation, which determines intermediate points based on the time it takes to travel between them at a preferred speed. The calculation starts by determining the distance between consecutive waypoints, $d_i = \|\mathbf{w}_{i+1} - \mathbf{w}_i\|$, where $\mathbf{w}_i$ represents the i^{th} waypoint. Given a reference velocity, $\bar{v}$, the travel time between waypoints is $t_i = \frac{d_i}{\bar{v}}$. The number of points for each path segment is then computed as $n_i = \lceil \frac{t_i}{\Delta t} \rceil$, where Δt is the controller's sampling time. The preferred velocity was set to half the robot's maximum speed, ensuring a balanced approach between speed and safety.
To minimize errors, when the robot needs to halt or significantly alter its course to

avoid moving obstacles, the path segment nearest to the robot at any given moment is used as the reference for the NMPC for the upcoming prediction horizon. This method helps to maintain accuracy in path following, despite any necessary deviation or stop.

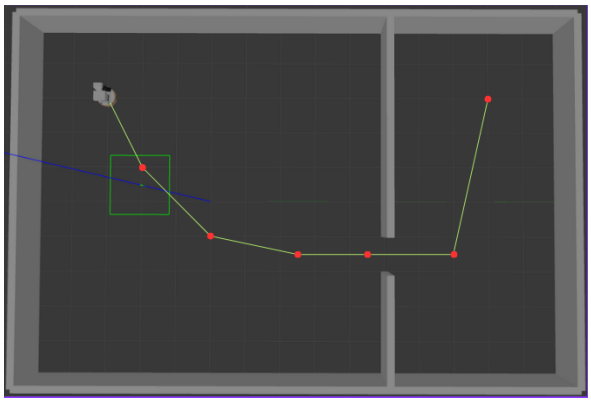

Figure 3: Illustration of a robot's world with strategically placed waypoints and the interpolated path connecting them.

4.2 Areas and Switching Heuristics

Obstacle-free zones are designed with an arbitrary number of vertices (up to six) but must maintain convexity to be effectively represented as inequality constraints within the NMPC framework provided by Acados. Figure 4 illustrates an example of an office environment with the defined areas.

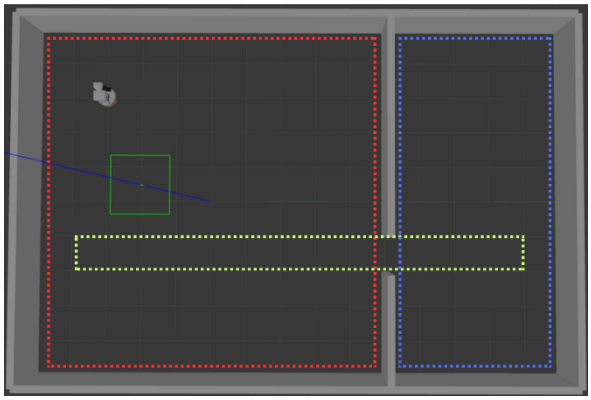

Figure 4: Simplified representation of the simple office layout highlighting designated maneuverable areas.

The strategy for transitioning between defined areas is determined by one of several predefined heuristics. Each heuristic is designed for different navigation purposes and scenarios within this project, as described below:

- **Prediction heuristic:** Focusing on the area currently occupied by the robot, this heuristic gives precedence to the area declared later in the system setup in

case of overlapping zones. Utilizing the NMPC's prediction horizon, a positional constraint index array is generated to determine the active area constraints for the NMPC update, as detailed in Algorithm 1.

- **Next waypoint heuristic:** Each area is linked to a specific waypoint. Upon reaching this waypoint, the corresponding area is activated. Similar to the prediction heuristic, the NMPC's prediction horizon is employed to assign a positional constraint index array for the next NMPC update, activating the appropriate area constraints, as detailed in Algorithm 2

- **Nearest waypoint heuristic:** This heuristic associates each waypoint with an area, with the possibility of multiple waypoints belonging to a single area. The area corresponding to the waypoint nearest to the robot is selected. Again, the NMPC's prediction horizon is used to generate a positional constraint index array that influences the selection of the active area for the next instant. See Algorithm 3.

- **Time heuristic:** Differently from the first three, the transition between areas is dictated by fixed time intervals, independent of the robot's position or actions, as detailed in Algorithm 4.

Algorithm 1 Get Area Constraint Index Array Based on Robot Predicted Motion

Require: Robot predicted motion: pred_motion, Previous position constraint index array: position_constraint_index_array, Areas: areas, Size of the prediction horizon N

Ensure: Updated position constraint index array

```
if pred_motion is None then
    return position_constraint_index_array
end if
for i, A in enumerate(areas) do
    if all_p in pred_motion are in A then
        position_constraint_index_array ← [i] × N
    end if
end for
return position_constraint_index_array
```

Algorithm 2 Get Area Constraint Index Array Based on Next Waypoint

Require: Robot predicted motion: pred_motion, Previous position constraint index array: position_constraint_index_array, Areas: areas, Switch Waypoints defined based on the world: W, Size of the prediction horizon N

Ensure: Updated position constraint index array

```
if pred_motion is None then
  return position_constraint_index_array
end if
threshold ← 0.2
A_num ← size(areas)
for i, p_i in enumerate(pred_motion) do
  for j from 0 to A_num − 1 do
    w_{x,j} ← W[j][0]
    w_{y,j} ← W[j][1]
    d ← sqrt((w_{x,j} − p_i_x)^2 + (w_{y,j} − p_i_y)^2)
    if position_constraint_index_array[i] = j AND d < threshold then
      position_constraint_index_array ← [j] × N
      position_constraint_index_array[i:] ← j + 1
    end if
  end for
end for
return position_constraint_index
```

Algorithm 3 Get Area Constraint Index Array Based on Nearest Waypoint

Require: Robot predicted motion: pred_motion, Previous position constraint index array: position_constraint_index_array, Waypoints W

Ensure: Updated position constraint index array

if pred_motion is None **then**
  return position_constraint_index_array
end if
for p_i in pred_motion **do**
  for j in range(size(W)) **do**
    $\mathrm{w}_{x,\mathrm{j}} \leftarrow W[\mathrm{j}][0]$
    $\mathrm{w}_{y,\mathrm{j}} \leftarrow W[\mathrm{j}][1]$
    distances$[j] \leftarrow \sqrt{(\mathrm{w}_{x,j} - \mathrm{p_i}_x)^2 + (\mathrm{w}_{y,j} - \mathrm{p_i}_y)^2}$
  end for
  closest_index $\leftarrow \arg\min_j$ (distances)
  Update position_constraint_index_array based on the world and closest_index
end for
return position_constraint_index_array

Algorithm 4 Update Area Constraint Index Based on Elapsed Time

Require: Elapsed time: elapsed_time, Predicted motion: pred_motion

Ensure: Updated position constraint index array

if pred_motion is None **then**
  return position_constraint_index
end if
Determine area_index based on elapsed_time and scenario-specific thresholds
position_constraint_index[:] $\leftarrow$ area_index
return position_constraint_index

It is important to emphasize that these heuristics are not universally applicable but are specifically designed for each environment, taking into account of the unique layout of waypoints and areas.

4.3 Nonlinear Model Predictive Control

4.3.1 NMPC Algorithm

At the core of the NMPC algorithm is the task of determining the optimal angular velocities for the robot, ensuring it follows the desired trajectory while considering dynamic constraints such as crowd movement. The NMPC algorithm takes as input the robot's current state ξ, the desired future states over the prediction horizon, and control inputs, along with predictions of external factors like crowd motion.
Operationally, the NMPC algorithm executes within each control cycle by solving a finite horizon Optimal Control Problem (OCP), optimized for computational efficiency through formulation as a Nonlinear Program (NLP). To this end, the NLP uses the discrete-time version

$$\xi_{k+1} = F(\xi_k, u_k)$$

of the robot kinematic model, obtained by numerically integration of the continuous time kinematic model, assuming piecewise constant control inputs.
The variables $\xi_{i|k}$ and $u_{i|k}$ denote the predicted robot state and control input at time t_{k+i}, calculated at t_k. These predictions are aggregated into vectors:

$$\Xi_k = (\xi_{0|k}, \ldots, \xi_{N|k})$$
$$U_k = (u_{0|k}, \ldots, u_{N-1|k})$$

where Ξ_k and U_k represent the NLP decision variables at time t_k. The NMPC's objective is to minimize control effort while ensuring the robot follows the desired path at specified velocities. Let $\eta_{i|k}$ and $v_{i|k}$ be the predicted position and velocity of the robot at t_{k+i}, with η_d and v_d representing the desired position and velocity. The position and velocity errors, $e_{i|k} = \eta_d - \eta_{i|k}$ and $z_{i|k} = v_d - v_{i|k}$, are then used to define the running and terminal costs as follows:

$$V_{i|k}(\xi_{i|k}, u_{i|k}) = e_{i|k}^T Q e_{i|k} + z_{i|k}^T R z_{i|k} + u_{i|k}^T S u_{i|k}$$
$$V_{N|k}(\xi_{N|k}) = e_{N|k}^T Q_N e_{N|k} + z_{N|k}^T R_N z_{N|k}$$

Here, Q, R, and S are the weighting matrices for predicted position error, velocity error, and control effort, respectively. Similarly, Q_N and R_N weight the final position and velocity errors in the prediction horizon. The NLP is thus formulated to minimize the sum of running and terminal costs subject to the initial condition, system dynamics, input and state constraints, and collision avoidance constraints:

$$min_{\Xi_k, U_k} \sum_{i=0}^{N-1} V_{i|k}(\xi_{i|k}, u_{i|k}) + V_{N|k}(\xi_{N|k})$$

subject to

$$\xi_{0|k} - \xi_k = 0 \quad (a)$$

$$\xi_{i+1|k} - F(\xi_{i|k}, u_{i|k}) = 0 \quad i = 0, \dots, N-1 \quad (b)$$

$$\xi_{min} \leq \xi_{i|k} \leq \xi_{max} \quad i = 0, \dots, N \quad (c)$$

$$u_{min} \leq u_{i|k} \leq u_{max} \quad i = 0, \dots, N-1 \quad (d)$$

$$collision\ avoidance\ constraints \quad (e)$$

where (a) enforces the initial condition, (b) the discrete-time system dynamics, (c) and (d) the state and input bounds and (e) collision avoidance throughout the prediction horizon. In particular, for the latter we have two different kind of constraints: linear inequality constraints based on areas definition for collision avoidance of fixed obstacles and nonlinear constraints based on CBF for the obstacle avoidance of moving obstacles. Once the NLP is solved, the first control samples $u_{0|k}$ are extracted from the obtained U_k and applied to the robot.

4.3.2 Areas-based Collision Avoidance

To ensure safety around fixed obstacles, the NMPC utilizes constraints modeled by the inequalities based on the decision variables, which are dynamically updated within each prediction horizon. The constraints ensure the robot remains within defined navigable areas, avoiding collisions with fixed obstacles:

$$a_i \cdot q_{x,k} + b_i \cdot q_{y,k} \leq c_i \quad \forall k \in \{1, \dots, N\},$$

where N is the number of prediction steps in the horizon, $m = 6$ represents the number of vertices defining each navigable area, (a_i, b_i) is the normal vector to the boundary line of the i-th constraint, and c_i is a scalar that determines the boundary line's distance from the origin along the normal vector.
The coefficients a_i, b_i, and c_i are derived as follows:

$$a_i = y_{i+1} - y_i,$$

$$b_i = x_i - x_{i+1},$$

$$c_i = x_i \cdot y_{i+1} - x_{i+1} \cdot y_i,$$

for each vertex (x_i, y_i) of the area, ensuring the vertices are ordered counterclockwise to correctly define the constraints.

4.3.3 CBF-based Collision Avoidance

For dynamic obstacles, such as humans, the NMPC incorporates Control Barrier Function (CBF) based constraints. These constraints are defined using the smallest bounding circle that encompasses the robot's planar projection, with radius ρ and center C. The center's coordinates, $c(\xi)$, vary with the robot configuration q within the state vector ξ.

A state ξ is considered safe regarding a human if the distance between the robot's bounding circle and the human's center of mass p exceeds the sum of ρ and a safety margin d_s, i.e.,

$$\|c(\xi) - p\| \geq \rho + d_s.$$

The safe set of states, C, relative to a human is thus defined as

$$C = \{\xi \in D : h(\xi) \geq 0\},$$

where

$$h(\xi) = \|c(\xi) - p\|^2 - (\rho + d_s)^2.$$

Function $h(\xi)$ serves as a CBF in discrete time, satisfying

$$\Delta h(\xi_k) \geq -\gamma h(\xi_k),$$

with $\Delta h(\xi_k) = h(\xi_{k+1}) - h(\xi_k)$ and $0 < \gamma \leq 1$.

To enforce safety across the prediction horizon for all perceived humans, the constraints are formulated as

$$\Delta h^j(\xi_{i|k}) \geq -\gamma h^j(\xi_{i|k}),$$

where

$$h^j(\xi_{i|k}) = \|c(\xi_{i|k}) - p^j_{i|k}\|^2 - (\rho + d_s)^2,$$

with $p^j_{i|k}$ being the predicted position of the j-th human at the i-th step of the prediction horizon, computed at time k.

For accurate prediction and collision avoidance, the fourth-order Runge-Kutta method (RK4) is employed. This numerical technique, aimed at solving differential equations, computes $c(\xi_{k+1})$ to subsequently determine $h(\xi_{k+1})$ with improved precision.

Given the system of differential equations $\frac{dx}{dt} = f(x, u)$, where x and u represent the state and control input vectors, and t denotes time, the RK4 approach for the next state prediction, x_{n+1}, from the current state x_n over a time step dt is outlined as:

$$
\begin{aligned}
k_1 &= f(x_n, u), \\
k_2 &= f\left(x_n + \frac{1}{2}k_1 dt, u\right), \\
k_3 &= f\left(x_n + \frac{1}{2}k_2 dt, u\right), \\
k_4 &= f\left(x_n + k_3 dt, u\right), \\
x_{n+1} &= x_n + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

This method calculates intermediate steps (k_1, k_2, k_3, and k_4) that estimate the system's evolution over the time step from different points, effectively capturing the dynamics with a higher degree of accuracy than simpler methods. The final state x_{n+1} is computed as a weighted average of these four approximations, significantly enhancing the prediction's precision.

Specifically, in the context of NMPC for collision avoidance, this precise state prediction allows for the accurate computation of the center of the robot's bounding circle in the next time step. Once this is determined, the function $h(\xi_{k+1})$, which is critical for evaluating safety distances from humans, can be accurately computed.

5 Implementation

The navigation strategy for the TIAGo mobile robot was implemented using Python, with the acados library to handle the Nonlinear Model Predictive Control (NMPC) aspects. For simulation and testing, the system was integrated with Gazebo and the Robot Operating System (ROS).

5.1 Implementation Framework

Gazebo, thanks to the physics engine, offers the capability to simulate dynamic different environments with fidelity. It serves as the core of the simulation setup, enabling the creation of detailed virtual environments that include both static (walls, fixed obstacles) and dynamic (moving agents) elements. ROS, on the other hand, works as the bridge for robot control and the deployment of navigation algorithms. It facilitates communication between the simulated robot in Gazebo and the control algorithms, ensuring that the robot behaves and responds as it would in a real-world scenario.

5.2 Environments setup

Three different environments were developed, each characterized by different configuration of walls and static obstacles. Furthermore the environments were enhanced by the introduction of moving agents, simulating human pedestrians to serve as moving obstacles to avoid while following the trajectory.
The different combinations between the environments boundaries, fixed obstacles and moving agents allowed to stress the navigation capabilities of the robot and to show the different behaviour of the four heuristics together with the NMPC, underling their advantages and disadvantages.
Furthermore, the different environments explore different convex areas, from simpler rectangles, up to more complex 6 vertex figures in order to take full advantage of the environment disposition and allowing as much as possible mobility to the robot in order to correctly avoid not only the fixed obstacles but also the moving agents.

5.2.1 Trajectory Tracking Strategy

The NMPC algorithm optimizes the robot's control inputs to follow a desired trajectory, which is defined by a series of waypoints. The trajectory tracking process involves several key steps:

Path Interpolation This process aims to create a series of points that the robot can follow more accurately than by simply navigating from waypoint to waypoint. The interpolation is achieved by linearly interpolating between each pair of consecutive waypoints, generating connecting segment between the waypoints:

Given a series of waypoints $W = \{w_1, w_2, \ldots, w_n\}$, where each waypoint w_i is a 2D point (x_i, y_i), the path interpolation involves calculating intermediate points between each pair of waypoints. The number of points n to interpolate between each pair of waypoints is determined based on the robot's average velocity $\bar{v}$ and the control time step Δt as:

$$n = \frac{t}{\Delta t},$$

where t is the time required to travel between two consecutive waypoints, calculated from the distance d between them over the average velocity $\bar{v}$ of the TIAGo:

$$t = \frac{d}{\bar{v}},$$

with d being the Euclidean distance between waypoints w_i and w_{i+1}:

$$d = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}.$$

At this point, a linear segment connecting the two waypoints is computed with the evenly spaced number of points.

Reference Trajectory The path interpolated from the waypoints outlines the desired positions of the robot over time, guiding it towards the goal while avoiding obstacles. For each control interval, the reference trajectory is dynamically updated based on the robot's current position, the remaining interpolated path and the computed predicted trajectory of the robot in the horizon $N = 50$.

Area Constraints The navigation strategy incorporates area constraints to ensure that the robot remains within permissible areas and allow the switch between them. These constraints are defined by polygons that represent either safe areas or obstacles. The navigation algorithm dynamically adjusts the robot's trajectory to comply with these constraints, ensuring adherence to safe navigation requirement.
The constraints for each area are represented as a series of linear inequalities:

$$a_i \cdot q_{x,k} + b_i \cdot q_{y,k} \leq c_i \quad \forall k \in \{1, \ldots, N\},$$

where N is the number of prediction steps in the horizon. These inequalities ensure the robot's path remains within allowed areas and avoids obstacles by satisfying all area constraints.

6 Simulation Settings

The proposed safe navigation method was tested in the Gazebo which provided a realistic simulated environment for the physics and the interaction of the robot with an environment with fixed and moving obstacles. As targeted platform was used the TIAGo robot. According to the robot physical characteristics, its driving and steering velocities are bounded as, respectively, $-0.25\,\frac{m}{s} \leq v \leq 1.05\,\frac{m}{s}$ and $|\omega| \leq 1.05\,\frac{rad}{s}$, while the angular accelerations of the wheels as $|\dot{\omega}_{R,L}| \leq 5.07\,\frac{rad}{s^2}$.
For the kinematic parameters we have that wheel radius is set at $0.0985\,m$, while the wheel separation is setted at $0.4044\,m$ and the distance from the center of the robot considered for position tracking $b = 0.25\,m$. Across all the simulations the sampling time was set to $dt = 0.05\,s$, the prediction horizon $T = 1\,s$, the number of control intervals into the horizon were $N = 50$, while regarding the CBF parameters for the collision avoidance, they have been chosen to be: $\rho = 0.265\,m$, $d_s = 0.4\,m$ and $\gamma = 0.3\,m$.
Whenever the NLP solved by the motion generation module proved to be infeasible, it is imposed to the robot to stay still until the moment in which it becomes feasible again.
To test the performances of the proposed method, simulations were made in three different environments of increasing difficulty with variable number of humans (5, 10 or 15). In each of them all the switching heuristics previously presented are tested.
All human entities in the simulation move back and forth on pre-fixed linear paths with constant speed. They are not able to see the robot and to move consequently, so obstacle avoidance is performed by the robot agent only. Furthermore, the velocity of the humans is constant up to the direction change, this lead to a more difficult handling of the trajectory prediction performed by the TIAGo, because can lead to more unpredictability in the human entities behaviour.
The simulation tests was associated with a weight tuning for the NMPC which found out its best resulting parameters in the following weights: $p\_weight = 1e0$, $v\_weight = 1e4$, $omega\_weight = 1e0$, $u\_weight = 1e0$, $terminal\_factor = 5e3$, as the best weights all around the different simulations.
As said before, TIAGo has no possibility to rely on the sensors, so, in order to safely navigate in the environment it relies on the obstacle-free Convex Areas, which determine the feasible area for the robot navigation together with the CBF. In particular, the Convex area are designed in order to allow as much motion possibility as possible in the environment and at the same time to allow the most effective navigation through the waypoints. Because the convex area determines the available area in which the robot can navigate, great attention was paid to assure the possibility of the robot to travel from an area to another, assuring the presence of superposition between convex areas.

6.1 Simple Office

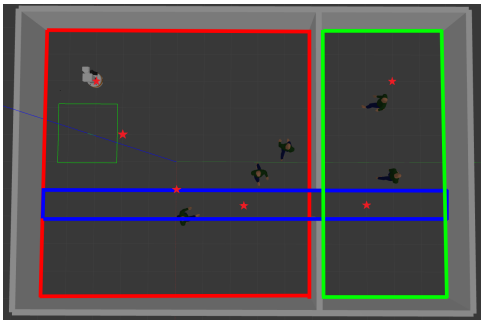

Figure 5: Two rooms office

The first simulated environment was composed of 2 rooms, connected through a door. TIAGo start in position $[-3, -3]$, in the upper left corner of the left room, and it's target is set to $[-3, 8]$, in the upper right corner of the right room and the connection door is centered in $[1.5, 5]$. For this simulation environment were designed 3 simple convex area:

- **First area (red):** The first area is designed to allow the robot to move freely in the first room, allowing to avoid collisions with the walls of the room itself, avoid the humans and approaching the door.
- **Second area (blue):** The second area aim to correclty handle the robot passage through the door narrow space. It superpose to both the first and third areas allowing the robot to travel from one to another. The 3rd and 4th waypoint defined in this area are used in particular to handle the approach of the robot to the passage, avoiding wall collision.
- **Third area (green):** The last area allow the robot to switch from the door handling to the last room and proceed up to the target.

Furthermore, were conducted 3 simulation variations, first with 5 people in the environment, then with 10 and with 15 people, stressing the robot's navigation capabilities.

6.1.1 Prediction

The Prediction Heuristic takes full advantage of the areas definition by using the superposition of the second area with the others 2 in particular, allowing the smooth passage of TIAGo from on area to another.

6.1.2 Next Waypoint

The Next waypoint heuristic determines a waypoint-based switch for the actual configuration. The switch from the first area to the second is performed when the distance of the

TIAGo from the 3rd waypoint (the door approaching waypoint) becomes lower than a certain treshold of 0.2. While the switch from the second area to the third is performed with the fifth.

6.1.3 Nearest Waypoint

Similarly to the Next Waypoint the Nearest heuristic performs an area switch based on the distance of each point of the prediction to each waypoint.

6.1.4 Time

The Time heuristic provide a reasonable time for the robot to travel safely the environment, taking into account not only the actual needed time to travel the space, but also to avoid possible collisions. For this reason, the times needed to travel the trajectory were used, computed by the Interpolator module in the MotionGeneratorManager, plus extra time for the collision avoidance feasibility, leading to the 2 switches at the following time elapsed: $20s$ and $25s$.

6.2 Double Office

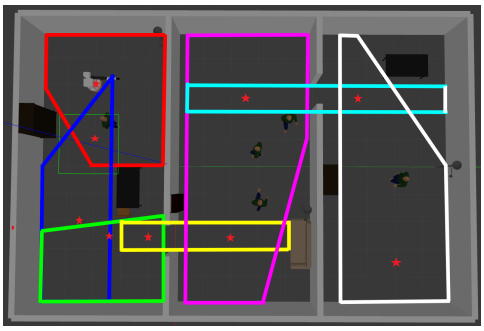

Figure 6: Three rooms office

The Double Office environment simulates a more complex layout with multiple narrower rooms and more obstacles inside of the rooms. The environment has a total of 7 convex areas:

- **First area (red):** The first area is designed to allow the robot to move in the first part of the first room, avoiding collisions with fixed obstacles like the table in the center of the room and the shelves.
- **Second area (blue):** The second area aims to handle the narrow passage of the robot between the wall and the table, superposing with first and third area.
- **Third area (yellow):** The third area covers the bottom area of the room, allowing the switch from the second area to the fourth area.

- **Fourth area (green):** The fourth area deal with the approach of the first door connecting the first two rooms.
- **Fifth area (orange):** This area allows to safely navigate the second room avoiding collisions with large fixed obstacles in the bottom right area.
- **Sixth area (violet):** Again this area allows the passage through the second door, connecting the second room with the third one.
- **Seventh area (light blue):** In the end the last area covers the navigation of the last room.

6.2.1 Prediction

The Prediction heuristic needed a careful implementation of the superposition area between the first and second area, which appears quite narrow and is furthermore made complex by the presence of an human in the very first part of the trajectory. Also the switch from the third to the fourth area appears quite narrower, while the others resembles the others simulation environments.

6.2.2 Next Waypoint

The next waypoint threshold is set again to 0.2, and we have the following switches: waypoint 2 (area 1 to area 2); waypoint 4 (area 2 to area 3); waypoint 5 (area 3 to area 4); waypoint 6 (area 4 to area 5), waypoint 7 (area 5 to area 6) and waypoint 8 (area 6 to area 7). Allowing a precise switch.

6.2.3 Nearest Waypoint

The nearest waypoint, also in this case, allows the switch without the threshold reference and we have the following switches: waypoint 2 or 3 for the area 2; waypoint 4 for area 3; waypoint 5 for area 4; waypoint 6 for area 5; waypoint 7 for area 6 and waypoint 8 for area 7.

6.2.4 Time

The times used for the double office environment were: $8s$ to switch from starting area 1 to area 2; $18s$ from area 2 to area 3; $22s$ from area 3 to area 4; $28s$ from area 4 to area 5; $40s$ from area 5 to area 6 and $50s$ to area 7.

Furthermore, 2 simulation variations were conducted, first with 5 people in the environment, then with 10 people stressing the robot's navigation capabilities.

6.3 Corridor

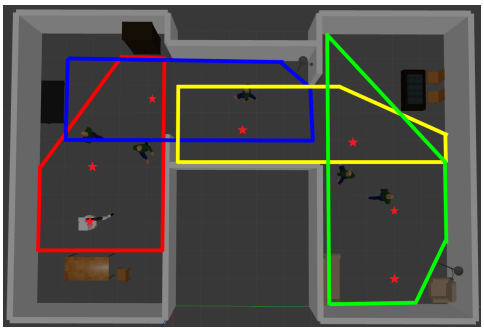

Figure 7: Office with corridor

The Corridor environment is designed to simulate a longer narrow passage scenario, testing the robot's capability to navigate through constrained and irregular spaces efficiently. This environment challenges the robot's path planning and obstacle avoidance algorithms also by introducing more complex convex areas to avoid fixed obstacles and take advantage of all the space available. The robot starts in position $[-3, -3]$ and the target is setted in $[-1, 8]$, making the robot travelling from the bottom left to the bottom right area of the map.

- **First area (red):** The first area is designed to allow the robot to move freely in the first room, in order to avoid collisions with the walls of the room itself, to prevent collision with the humans and to approach the corridor.
- **Second area (blue):** The second area aims to handle the first half of the passage through the corridor. It superposes to both the first and third areas allowing the robot to travel from one to another.
- **Third area (yellow):** The third area handles the second part of the navigation of the corridor thanks also to a waypoint positioned in-between them and the corridor and superpose with the last area to allow the robot to switch.
- **Fourth area (green):** The last area allow the robot to reach the desired target avoiding the fixed obstacles in the area, in particular the table in the top right and the sofa in the bottom right.

6.3.1 Prediction

The prediction again allows a waypoint-independent switch for the robot, based on the superposition of the areas, observable in the disposition of the second and third area. The double area was preferred to a single one through the whole corridor, because, due to the irregularities of the wall it would have provided a narrower area for the robot, and so to an harder navigation.

6.3.2 Next Waypoint

The Next Waypoint allows the switch under a threshold of 0.2 at the waypoint 3 from the first to the second area. The waypoint 4 allows the switch between the 2 convex areas in the corridor and the waypoint 5 allows the switch to the last area.

6.3.3 Nearest Waypoint

The Nearest waypoint heuristic shows a similar behaviour to the next, but, because it is based on the 'nearest' rather than on a certain threshold, the switch happens way before in the path compared to its counterpart.

6.3.4 Time

The time approach also in this case takes into account of the needed traveling time plus a safe obstacle avoiding time, leading to the following time switches: $12s$ from the first to the second, 20 from the second to the third, 36 from the third to the fourth.

7 Results

In this section, we outline the outcomes of our simulations across different environments, showcasing the performance of our approach.

7.1 Simple Office

First of all, the easiest scenario, described in section 6.1, was tested. Only two rooms divided by a wall with a narrow door and three rectangular convex areas. The robot has to reach the goal passing through four intermediate waypoints while avoiding 5, 10 or 15 people. Due to the simplicity of the scenario no big troubles were addressed by the controller with the exception of the time heuristic, the least robust out of the presented four. In this environment the initial orientation of the robot is not favorable with the desired path, so at the beginning of all the simulations there is an important maneuver and consequently a small increase in the tracking error and a pick in the steering velocity (negative since it turns clock-wise) ending some instants before the driving velocity has settled to the desired value.

7.1.1 Simple Office 5 people

As shown in the snapshots the path is perfectly followed and the tracking error is almost zero after the initial maneuver except in the case in which a person is encountered. It is in fact visible in Figure 9 and Figure 10 that at 23 seconds i.e. at the end of the curve of the path and so at the pick of the steering velocity there is a sudden variation of the driving and steering accelerations to keep the safe distance from one of the moving obstacle in the upper room and a small deviation from the desired path is performed. Here all the heuristic successfully switch among the areas and the NMPC never fail. The execution time is reasonably around 0.02 seconds with the exception of some rare cases.

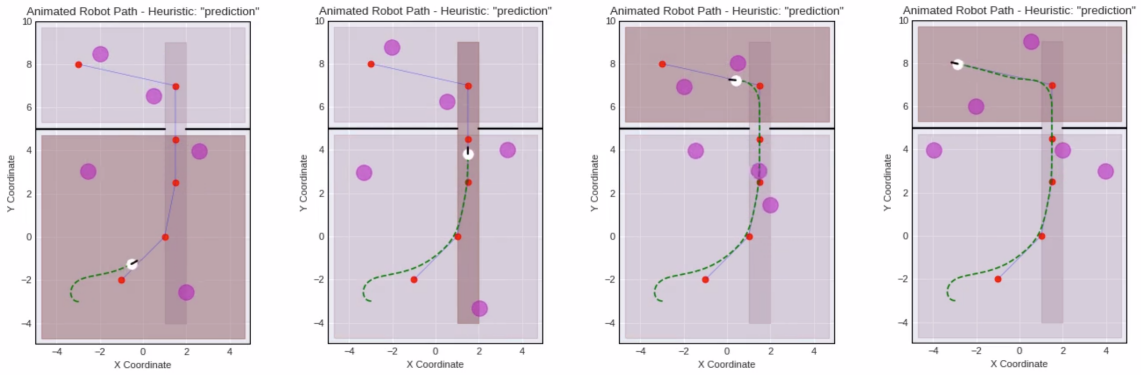

Figure 8: Snapshots from the animated plot **prediction heuristic**

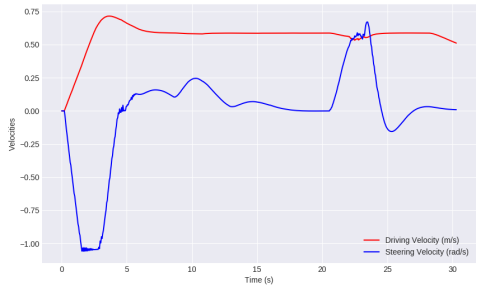

Figure 9: Robot driving and steering velocities **prediction heuristic**

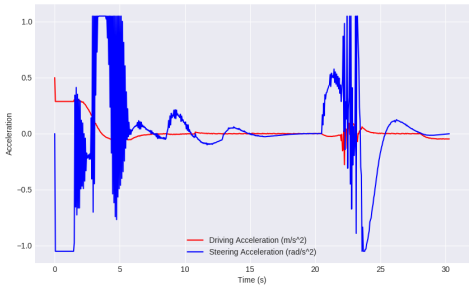

Figure 10: Robot driving and steering accelerations **prediction heuristic**

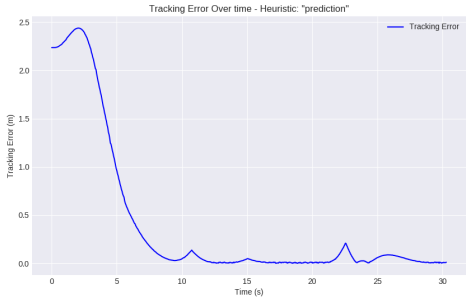

Figure 11: Tracking error **prediction heuristic**

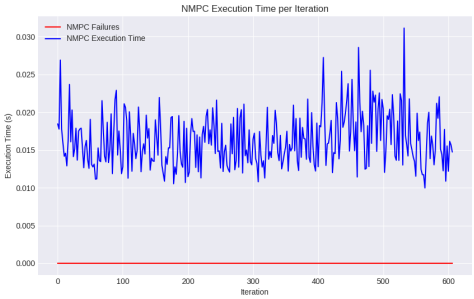

Figure 12: NMPC execution time and failures **prediction heuristic**

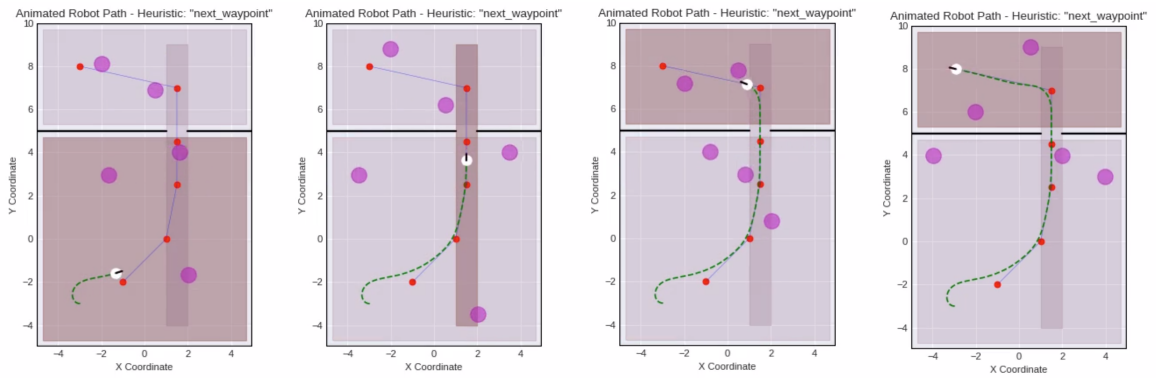

Figure 13: Snapshots from the animated plot **next waypoint heuristic**

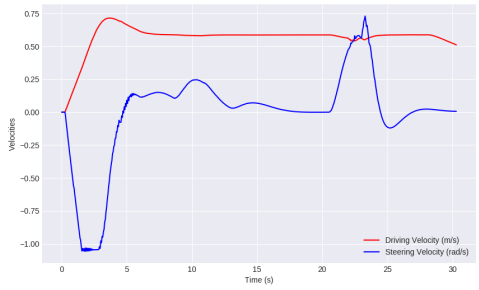

Figure 14: Robot driving and steering velocities **next waypoint heuristic**

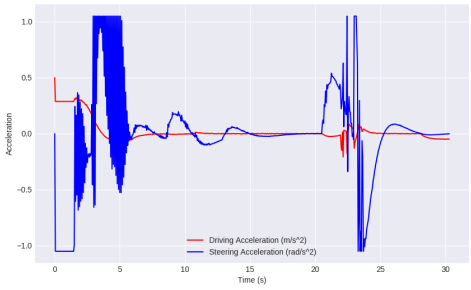

Figure 15: Robot driving and steering accelerations **next waypoint heuristic**

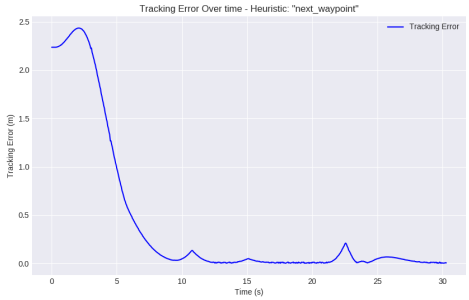

Figure 16: Tracking error **next waypoint heuristic**

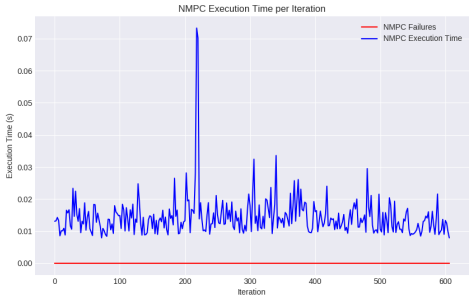

Figure 17: NMPC execution time and failures **next waypoint heuristic**

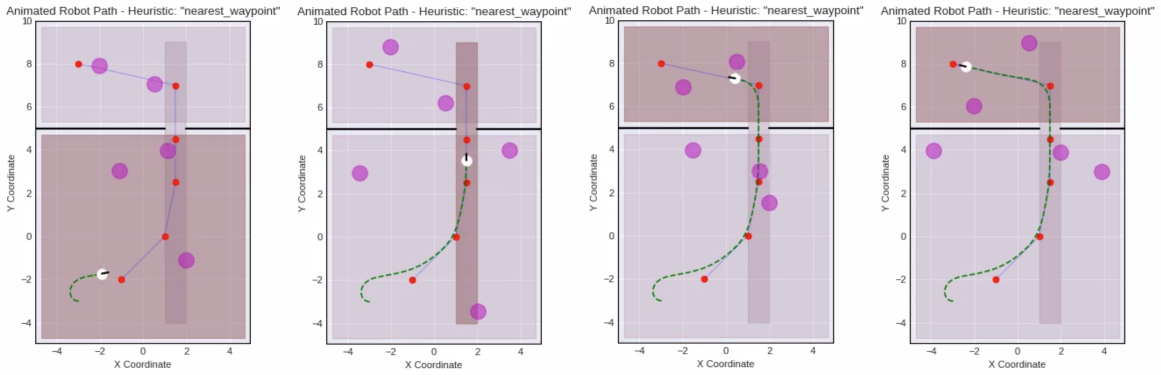

Figure 18: Snapshots from the animated plot **nearest waypoint heuristic**

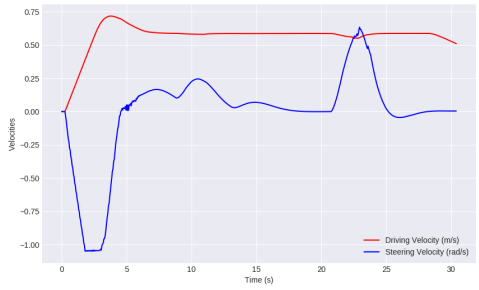

Figure 19: Robot driving and steering velocities **nearest waypoint heuristic**

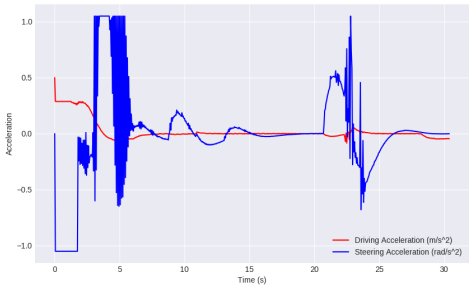

Figure 20: Robot driving and steering accelerations **nearest waypoint heuristic**

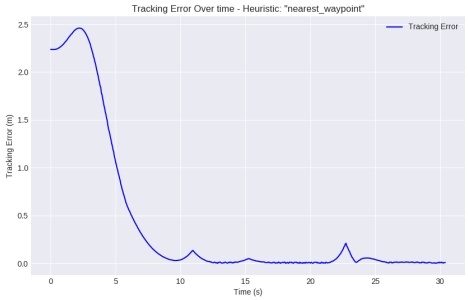

Figure 21: Tracking error **nearest waypoint heuristic**

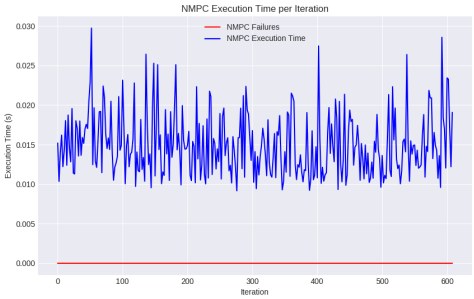

Figure 22: NMPC execution time and failures **nearest waypoint heuristic**

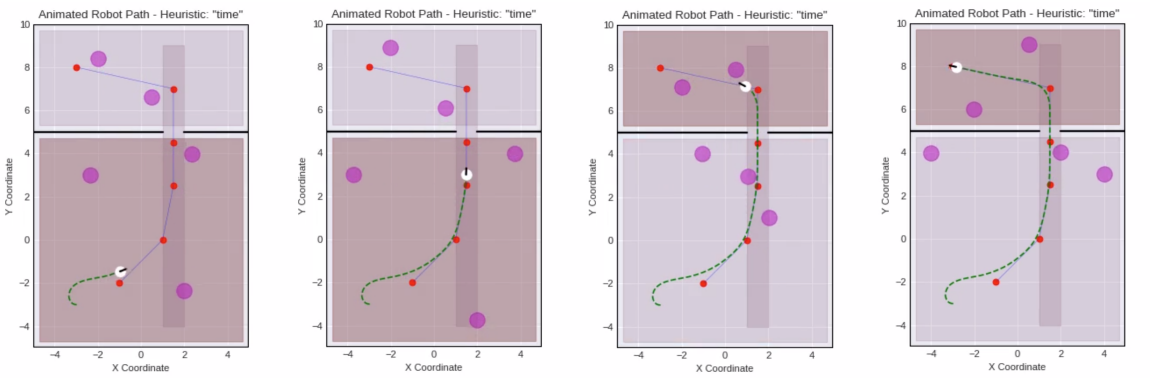

Figure 23: Snapshots from the animated plot **time heuristic**

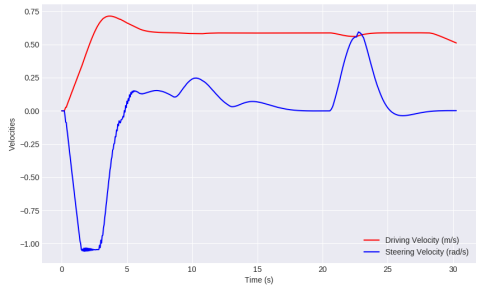

Figure 24: Robot driving and steering velocities **time heuristic**

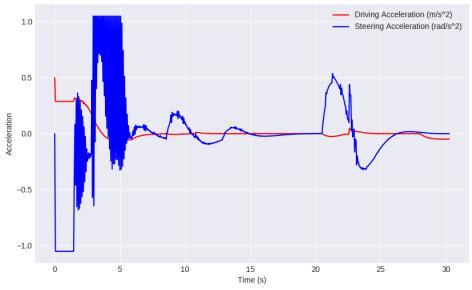

Figure 25: Robot driving and steering accelerations **time heuristic**

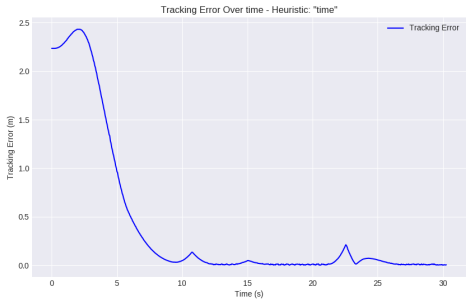

Figure 26: Tracking error **time heuristic**

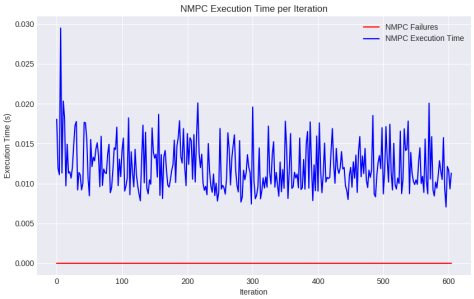

Figure 27: NMPC execution time and failures **time heuristic**

7.1.2 Simple Office 10 people

In this case with the addition of 5 people the NMPC has to face some more tricky situations. In fact, as can be seen in the third snapshot of Figure 28, at one point the robot is located between to people in the moment in which it is crossing the door. Due to the hard constraints of the second convex region it is unable to find a way to avoid the collision and the NMPC fails (around 20 seconds). With the first three heuristics,

that are position-based the failure causes no problems; once the person above moves from the door, the robot starts again and recovers the tracking error. With the time heuristic instead, the fact that the robot has to stop, makes the whole task fail since the third area is activated before the robot effectively enters it.

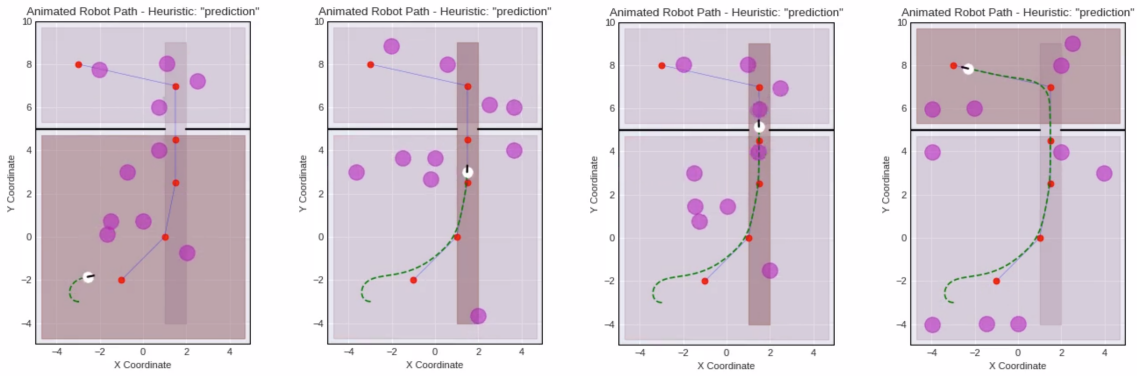

Figure 28: Snapshots from the animated plot **prediction heuristic**

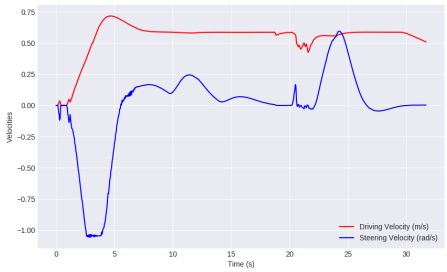

Figure 29: Robot driving and steering velocities **prediction heuristic**

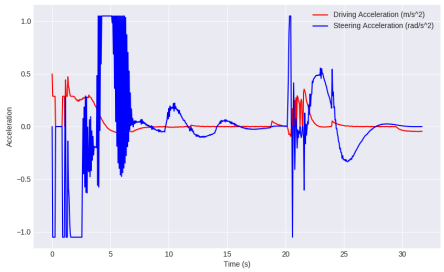

Figure 30: Robot driving and steering accelerations **prediction heuristic**

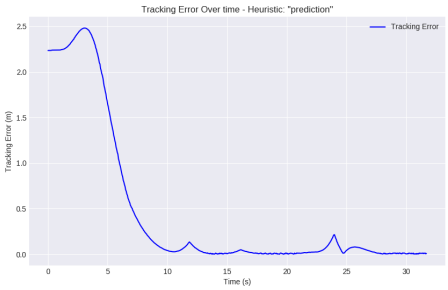

Figure 31: Tracking error **prediction heuristic**

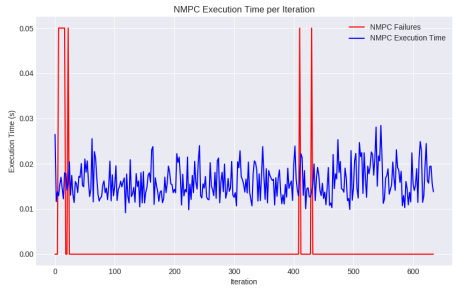

Figure 32: NMPC execution time and failures **prediction heuristic**

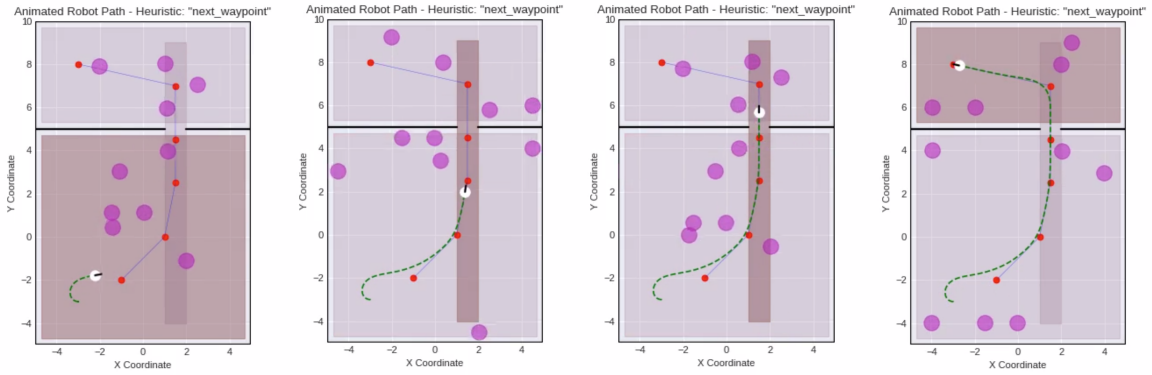

Figure 33: Snapshots from the animated plot **next waypoint heuristic**

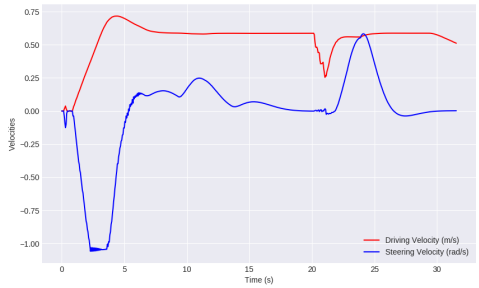

Figure 34: Robot driving and steering velocities **next waypoint heuristic**

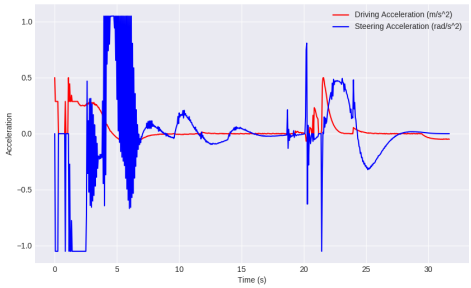

Figure 35: Robot driving and steering accelerations **next waypoint heuristic**

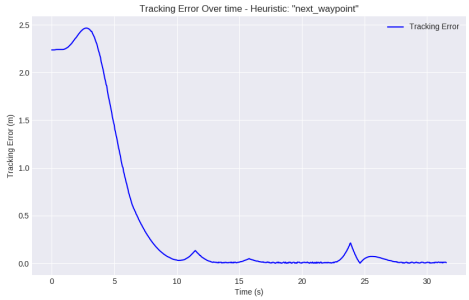

Figure 36: Tracking error **next waypoint heuristic**

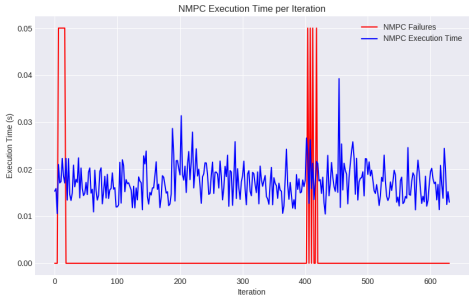

Figure 37: NMPC execution time and failures **next waypoint heuristic**

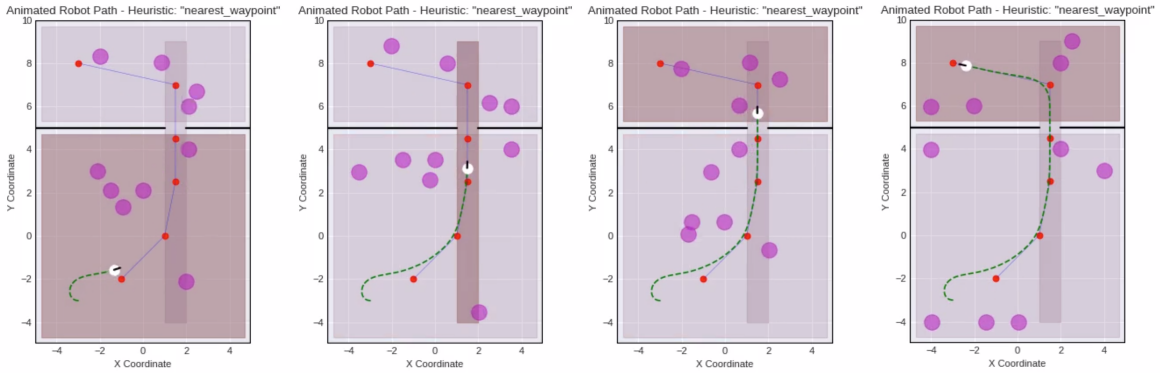

Figure 38: Snapshots from the animated plot **nearest waypoint heuristic**

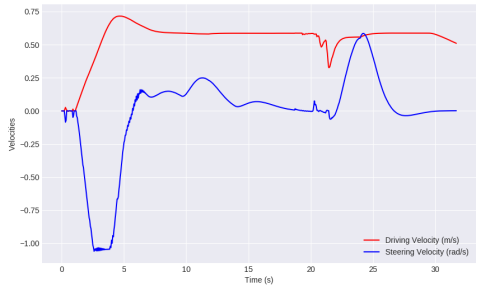

Figure 39: Robot driving and steering velocities **nearest waypoint heuristic**

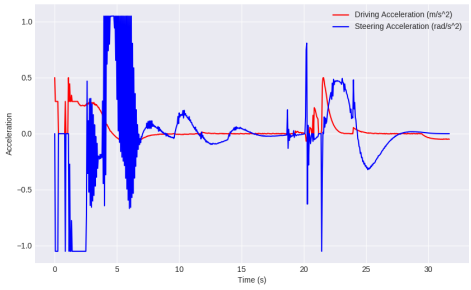

Figure 40: Robot driving and steering accelerations **nearest waypoint heuristic**

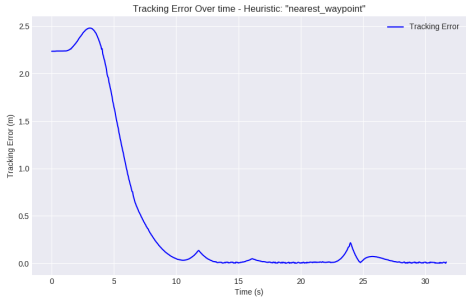

Figure 41: Tracking error **nearest waypoint heuristic**

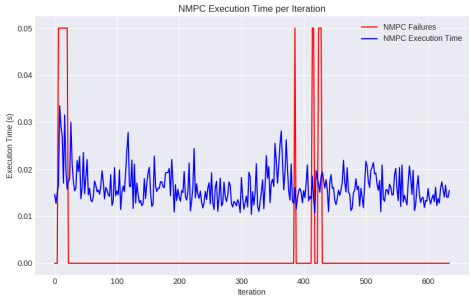

Figure 42: NMPC execution time and failures **nearest waypoint heuristic**

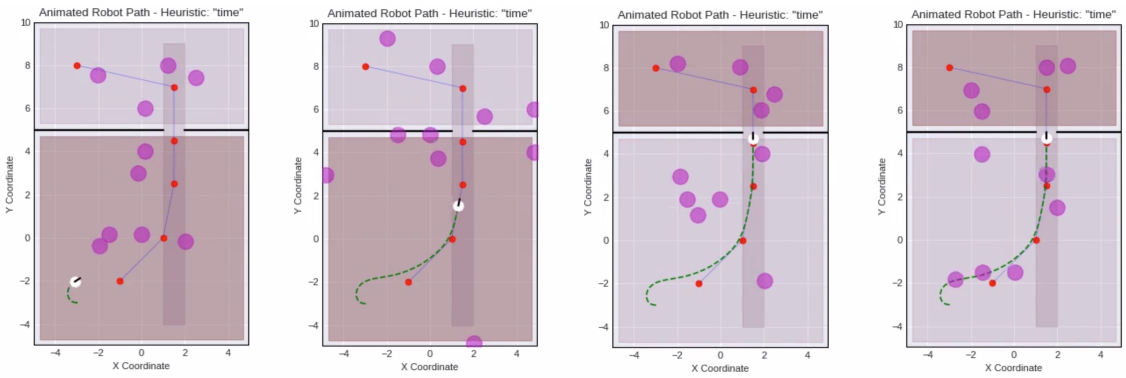

Figure 43: Snapshots from the animated plot **time heuristic**

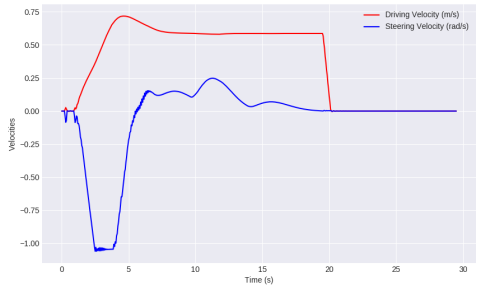

Figure 44: Robot driving and steering velocities **time heuristic**

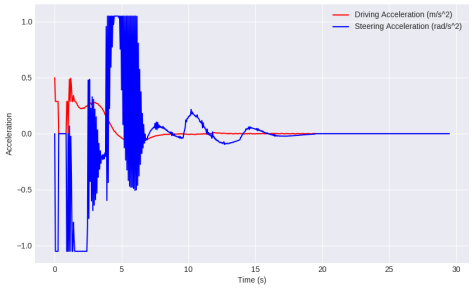

Figure 45: Robot driving and steering accelerations **time heuristic**

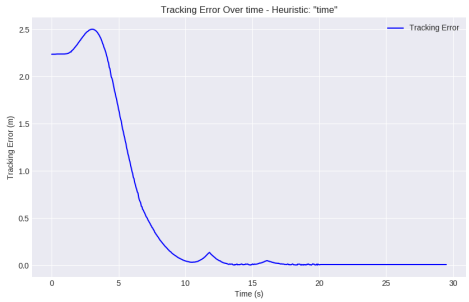

Figure 46: Tracking error **time heuristic**

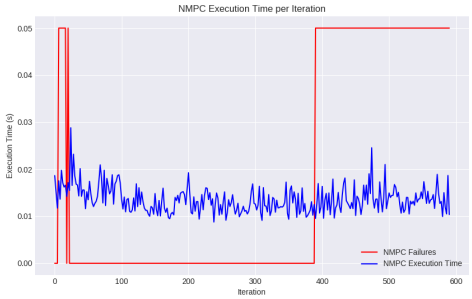

Figure 47: NMPC execution time and failures **time heuristic**

7.1.3 Simple Office 15 people

In this section will follow the results of the last variation of the simple office environment with 15 people, presenting a significantly challenging environment for the robot's navigation. The presence of additional people greatly affects the robot's maneuverability due to the increased density and unpredictability of pedestrian movement. By using the Prediction Heuristic the robot is able to successfully navigate the environment up

to the goal (Figure 48), respecting the bounds in speed and acceleration (Figure 49, Figure 50) and satisfying the constraints over the trajectory error (Figure 51), recovered rapidly, according to the planned path. The NMPC encountered many, sequential short failures into the first part (Figure 52) due to the initial superposition with 2 humans while the robot is trying to recover the initial tracking error as fast as possible.

Also the Next Waypoint Heuristic performed well up to the goal (Figure 53). The speed bound assigned to the robot is correctly satisfied, with some saturation of the steering accelerations (Figure 50). Also in this case we have only initially some failures of the NMPC (Figure 57) as for the Prediction case and some failures in the second room where the model is dangerously near to 2 humans.

In the case of the Nearest Heuristic we can observe that, also in this approach the robot successfully reach the end (Figure 58), the behaviour is quite similar to the previous case, the velocity required is satisfied (Figure 59) but the switch from one area to another is "lazier", providing more room to the robot to perform maneuvers before changing convex area and then constraints.

In the end we have the Time Heuristic, less general compared to all the others. In particular we can observe that, even if a manual tuning could lead to the possibility for the robot to navigate from one room to another reaching the goal, by keeping the same time switch from the previous experiments with less people, the robot spend more times into the trajectory, losing the "sweet spot" for the switch, and than failing into the navigation (Figure 63).

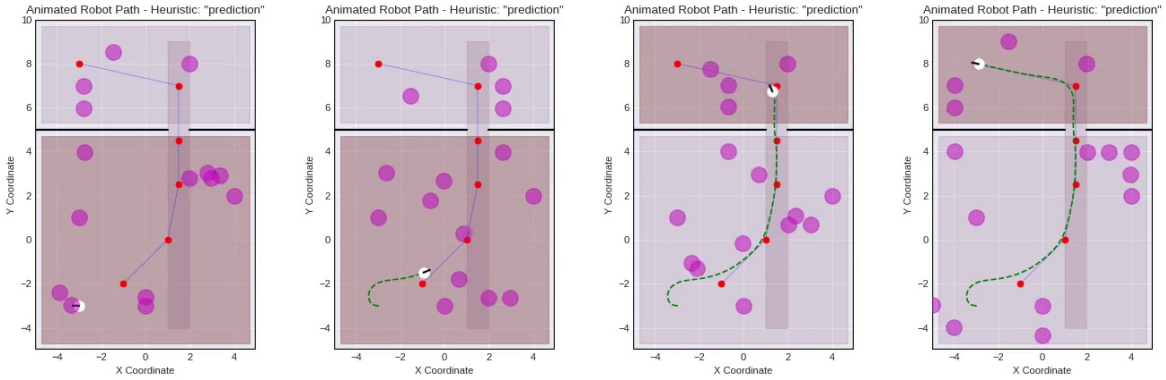

Figure 48: The robot path for the **Prediction Heuristic**

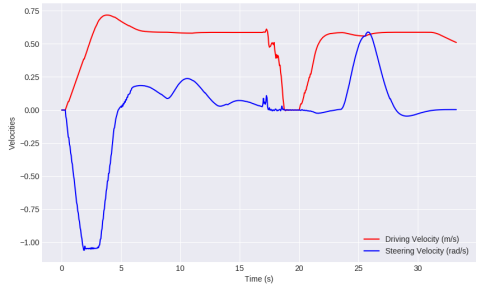

Figure 49: Driving and Steering Velocities - **Prediction Heuristic**

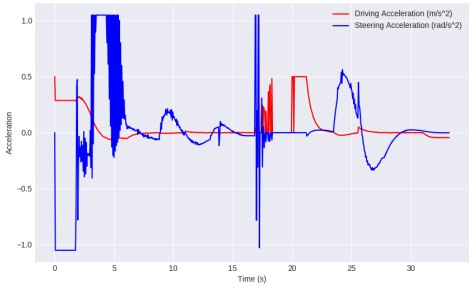

Figure 50: Driving and Steering Accelerations - **Prediction Heuristic**

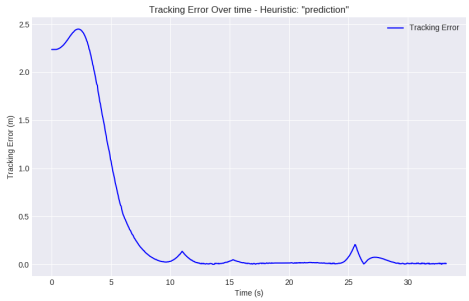

Figure 51: Absolute Tracking Error - **Prediction Heuristic**

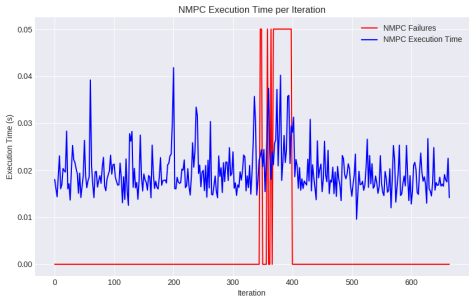

Figure 52: NMPC Execution Time and Failures - **Prediction Heuristic**

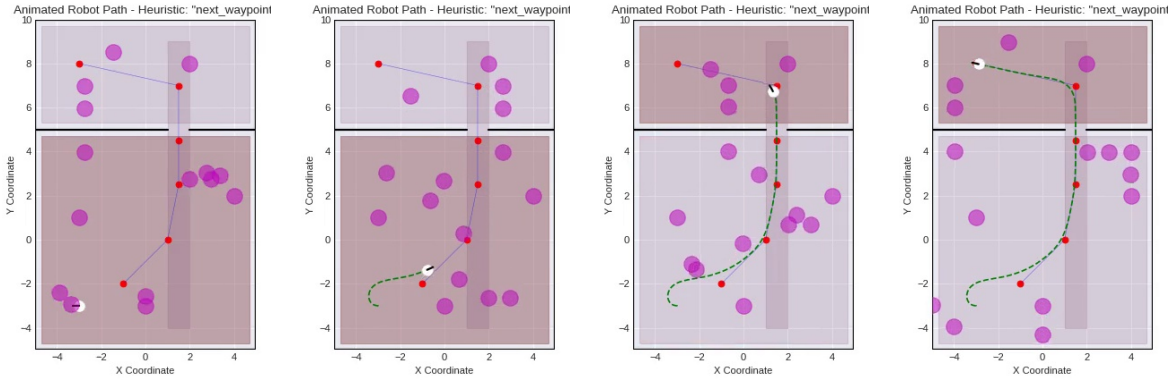

Figure 53: The robot path for the **Next Waypoint Heuristic**

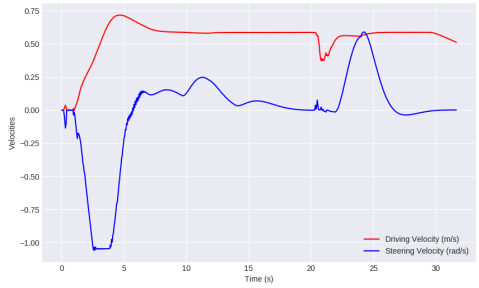

Figure 54: Driving and Steering Velocities - **Next Waypoint Heuristic**

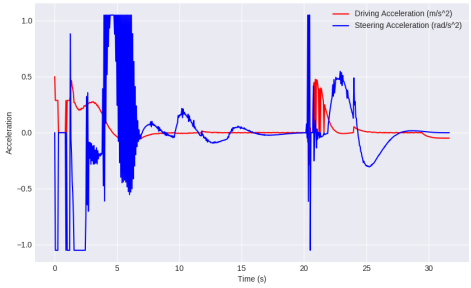

Figure 55: Driving and Steering Accelerations - **Next Waypoint Heuristic**

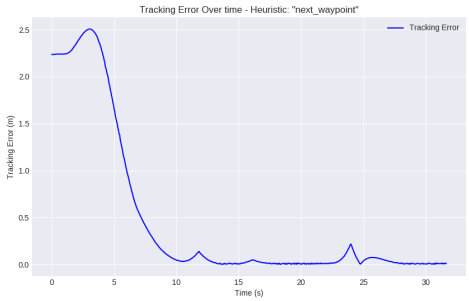

Figure 56: Absolute Tracking Error - **Next Waypoint Heuristic**

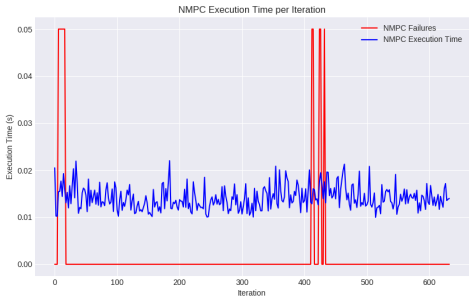

Figure 57: NMPC Execution Time and Failures - **Next Waypoint Heuristic**

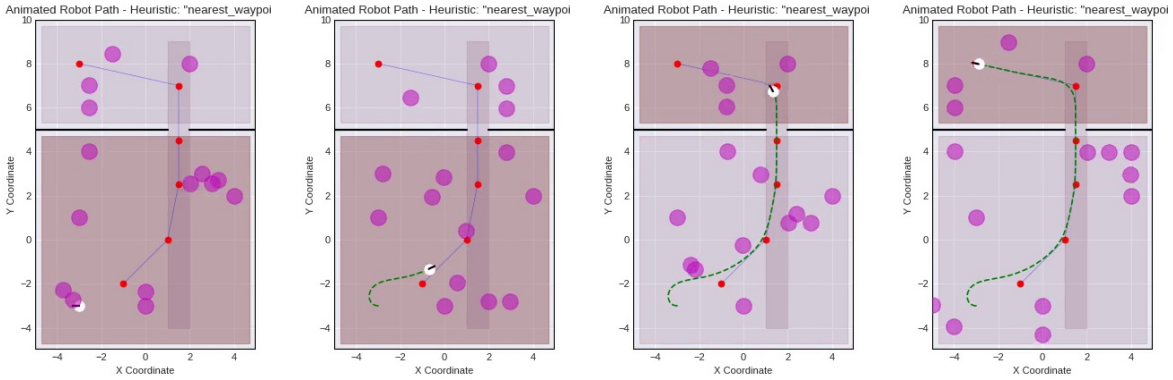

Figure 58: The robot path for the **Nearest Waypoint Heuristic**

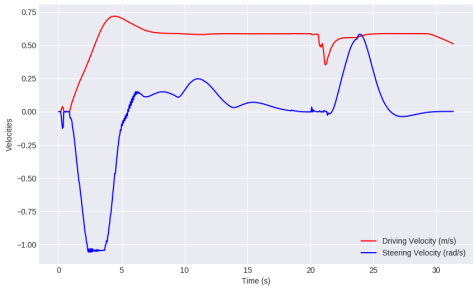

Figure 59: Driving and Steering Velocities - **Nearest Waypoint Heuristic**

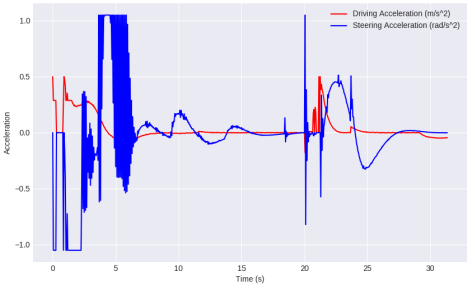

Figure 60: Driving and Steering Accelerations - **Nearest Waypoint Heuristic**

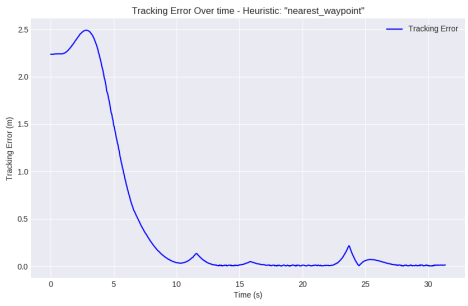

Figure 61: Absolute Tracking Error - **Nearest Waypoint Heuristic**

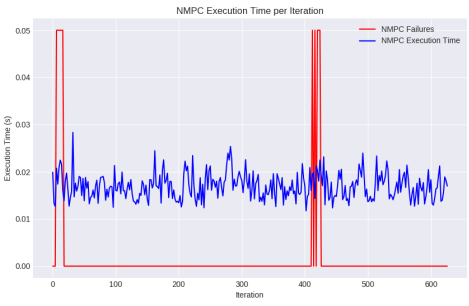

Figure 62: NMPC Execution Time and Failures - **Nearest Waypoint Heuristic**

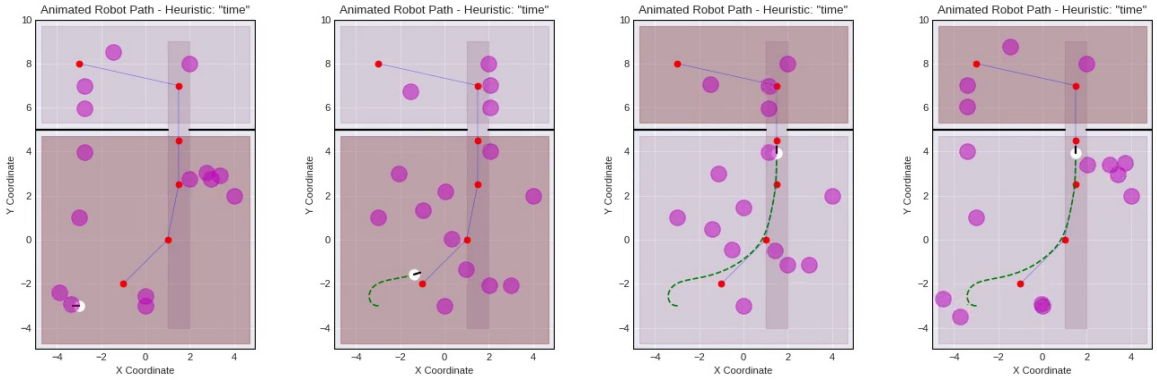

Figure 63: The robot path for the **Time Heuristic**

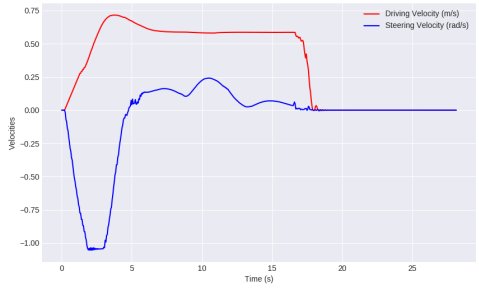

Figure 64: Driving and Steering Velocities - **Time Heuristic**

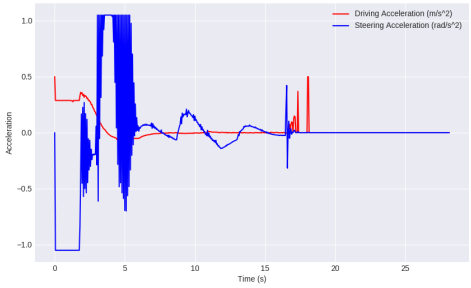

Figure 65: Driving and Steering Accelerations - **Time Heuristic**

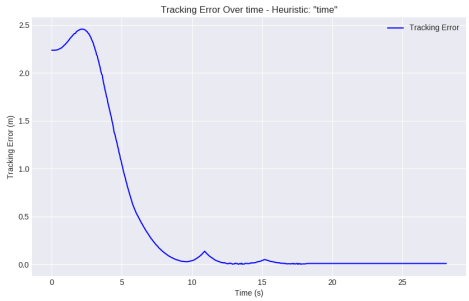

Figure 66: Absolute Tracking Error - **Time Heuristic**

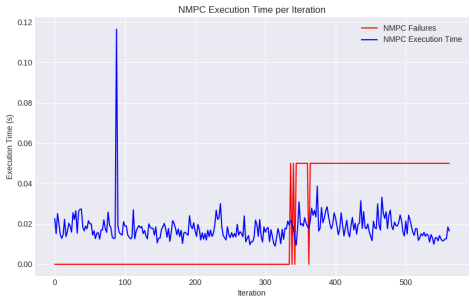

Figure 67: NMPC Execution Time and Failures - **Time Heuristic**

7.2 Double Office

In this section are showed the simulations performed on the second environment, the Double Office. The Environment represent a more complex version of the Office simulation. There are 3 different sequential rooms connected by a door in over the

long edge of each. Furthermore the robot needs to deal both with the moving people and with static obstacles in rooms which limit the space in which the robot can freely move, like the table in the center of the first room, or the table in the last room. The simulations were performed with 5 and 10 peoples. Here follows a more detailed approach to the results.

7.2.1 Double office 5 people

The First simulations were performed with a total of 5 people moving in the environment, and the starting position of the robot is quite challenging, due to the orientation, opposite to the trajectory path to follow up to the goal. Unfortunately, also due to the higher complexity of this environment, not all the heuristics were able to correctly complete the task, leading to failure cases.
The Prediction Heuristic also in this case appear to be a successful and versatile approach to the problem (Figure 68), being able to correctly lead the robot from the starting position into the first room, to the target goal into the third. Unfortunately there is a semi-collision in the second room, near to the door, due to the fact that the walking person in front of the door change direction (unexpectedly for the horizon of prediction of the robot) while the robot is almost behind him, not providing much time for a new planning and leading to a minor failure in the NMPC (Figure 72). This lead consequentially to a drop in the Driving Velocity (Figure 79) and later a spike in acceleration to regain velocity (Figure 85).

The Next Waypoint Heuristic successfully guide the robot to the target goal, similarly experiencing the problem of the Prediction Heuristic, but leading to a slightly wider trajectory, which better handle the situation. (Figure 73) Compared to the Prediction Heuristic it accumulates a little more of error on the absolute tracking error (Figure 76), velocity (Figure 79) and acceleration (Figure 80) profiles provides similar results, a little less smooth compared to the previous.

Due to the intrinsic nature of the Nearest Waypoint Heuristic itself, in this environment the robot fails in the task, stopping himself in the center of the second room (Figure 78). This happens because the nearest approach forces a switch in the area in the center of the line between the sixth and seventh waypoints when the seventh becomes closer compared to the sixth. Actually before the failure switch the behaviour is quite good, providing a constant velocity as required (Figure 79), with low tracking error (Figure 81), and no failures in the NMPC (Figure 82).

In the Time Heuristic we have a similar situation (Figure 83), the robot gets stuck in the second room because of a variation in the trajectory path avoiding a collision which leads to an increased time of traveling, and provoking a faster switch in the area,

leading to a failure of the NMPC (Figure 87). Furthermore this approach leads to an higher variability in the driving and steering accelerations (Figure 80), while obviously, due to the failure there is an increase in the absolute tracking error (Figure 81).

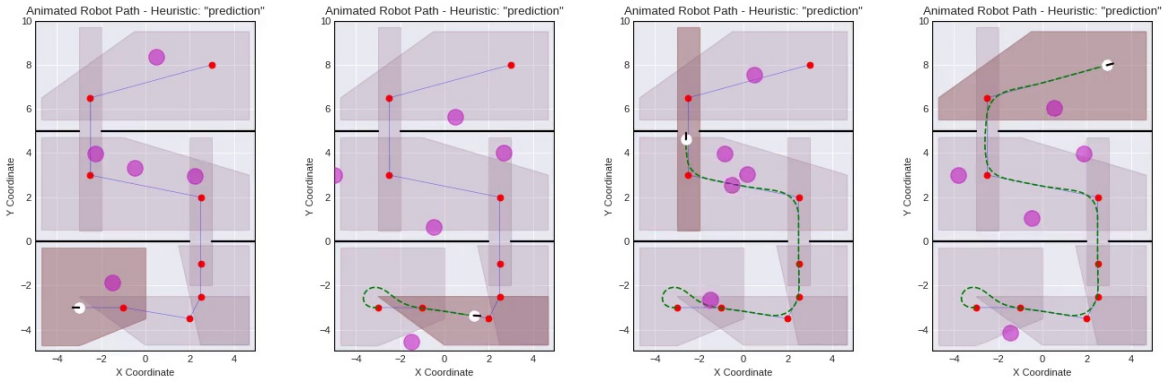

Figure 68: The robot path for the **Prediction Heuristic**

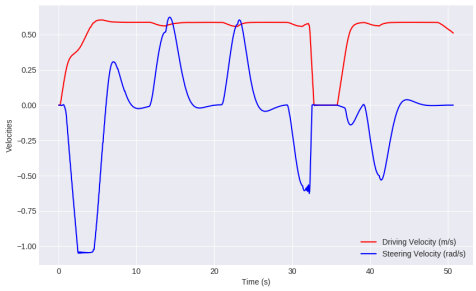

Figure 69: Driving and Steering Velocities - **Prediction Heuristic**

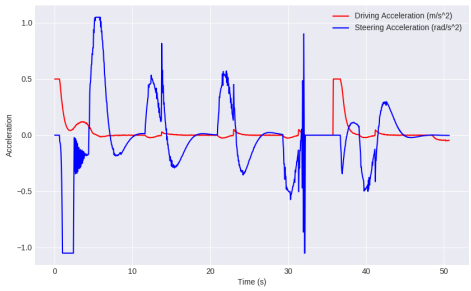

Figure 70: Driving and Steering Accelerations - **Prediction Heuristic**

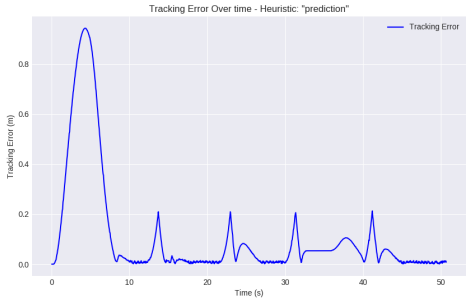

Figure 71: Absolute Tracking Error - **Prediction Heuristic**

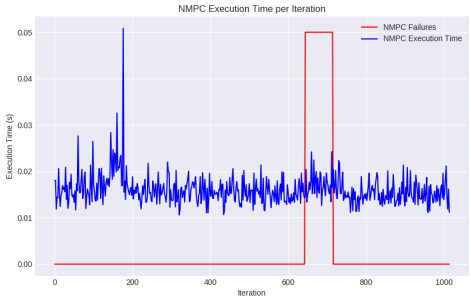

Figure 72: NMPC Execution Time and Failures - **Prediction Heuristic**

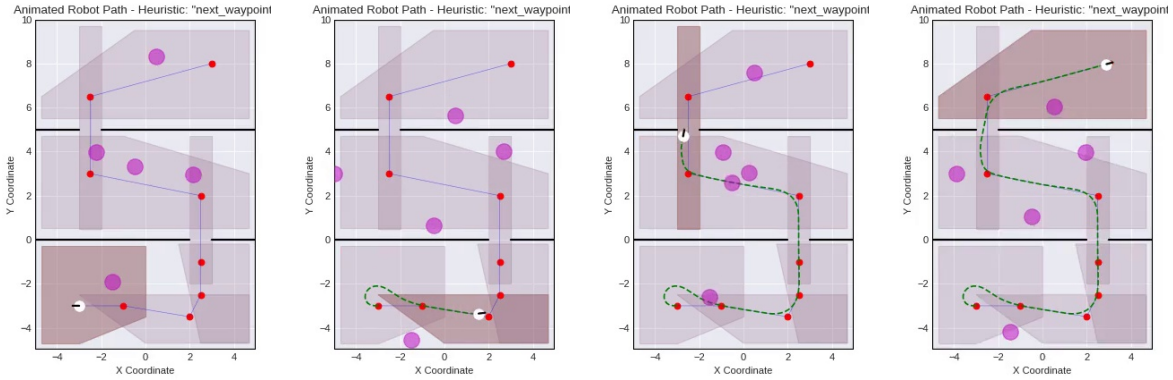

Figure 73: The robot path for the **Next Waypoint Heuristic**

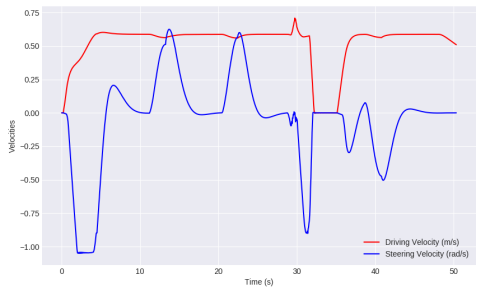

Figure 74: Driving and Steering Velocities - **Next Waypoint Heuristic**

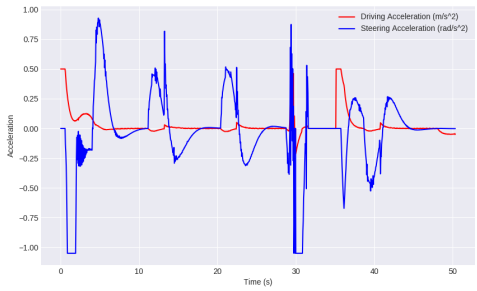

Figure 75: Driving and Steering Accelerations - **Next Waypoint Heuristic**

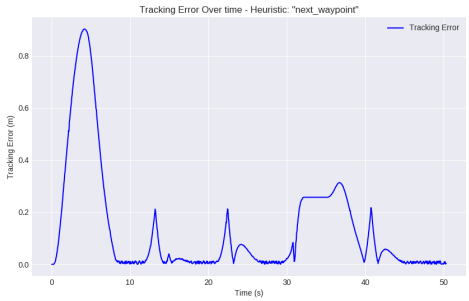

Figure 76: Absolute Tracking Error - **Next Waypoint Heuristic**

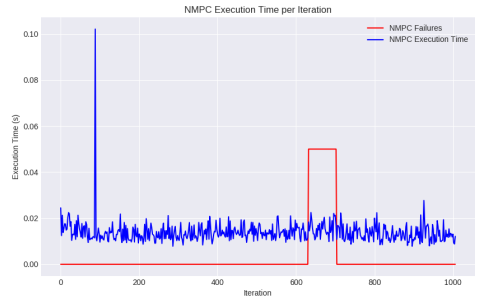

Figure 77: NMPC Execution Time and Failures - **Next Waypoint Heuristic**

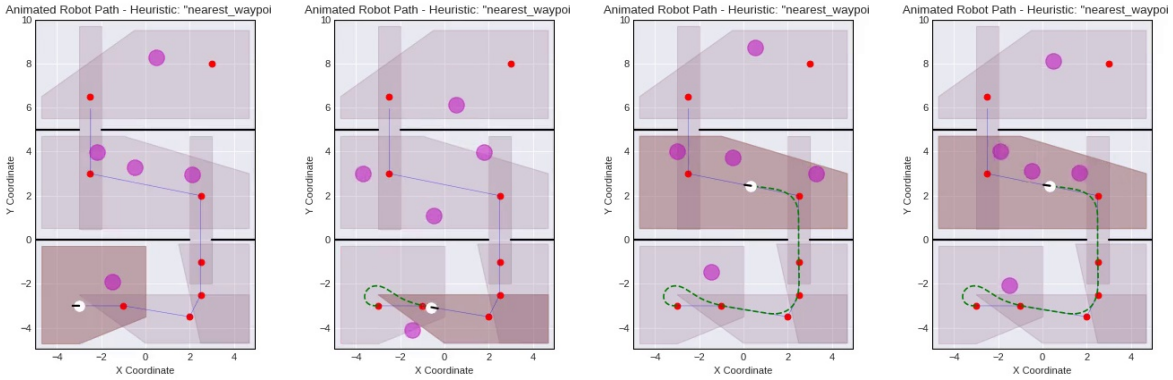

Figure 78: The robot path for the **Nearest Waypoint Heuristic**

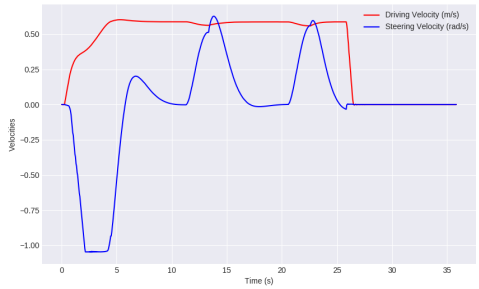

Figure 79: Driving and Steering Velocities - **Nearest Waypoint Heuristic**

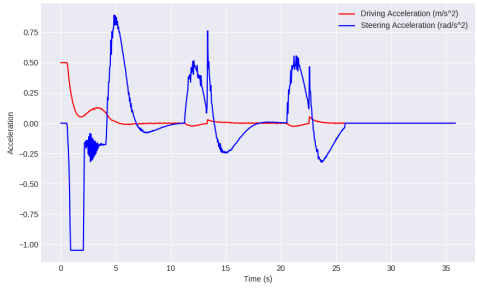

Figure 80: Driving and Steering Accelerations - **Nearest Waypoint Heuristic**

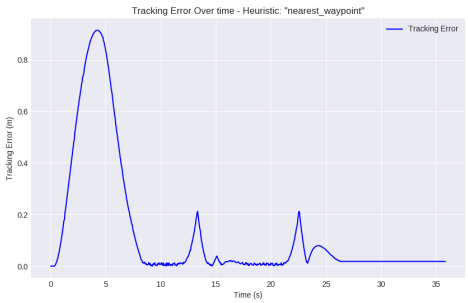

Figure 81: Absolute Tracking Error - **Nearest Waypoint Heuristic**

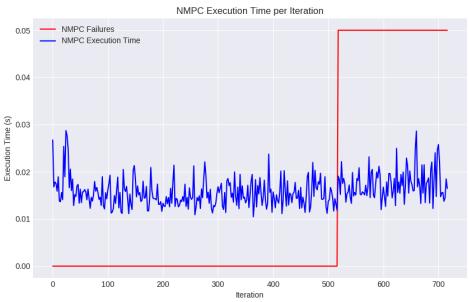

Figure 82: NMPC Execution Time and Failures - **Nearest Waypoint Heuristic**

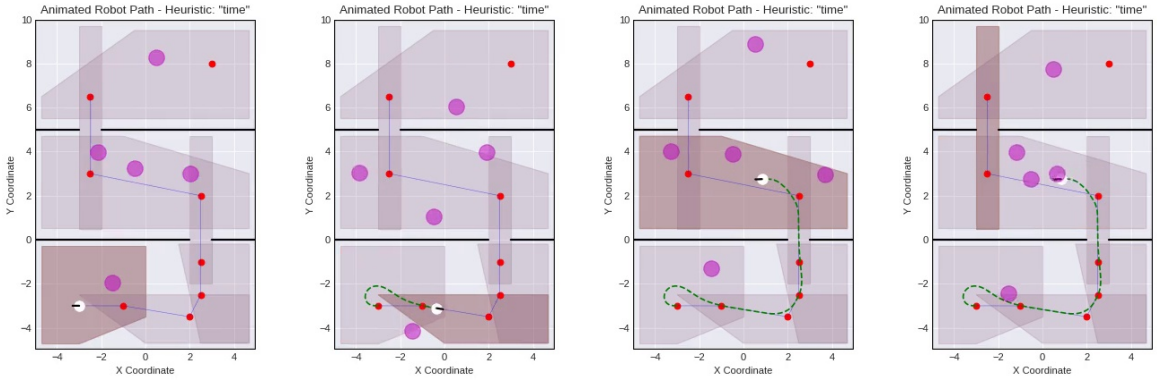

Figure 83: The robot path for the **Time Heuristic**

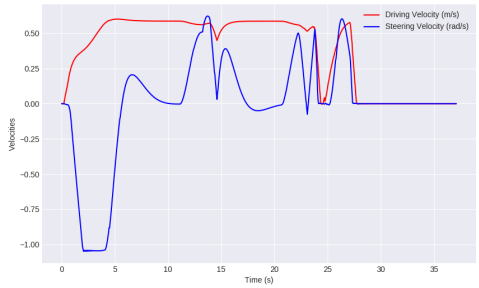

Figure 84: Driving and Steering Velocities - **Time Heuristic**

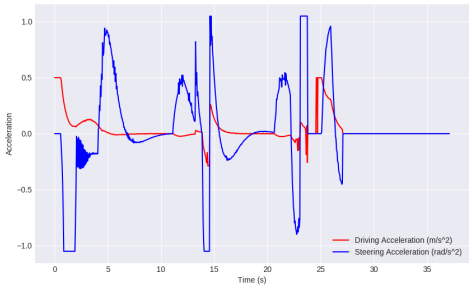

Figure 85: Driving and Steering Accelerations - **Time Heuristic**

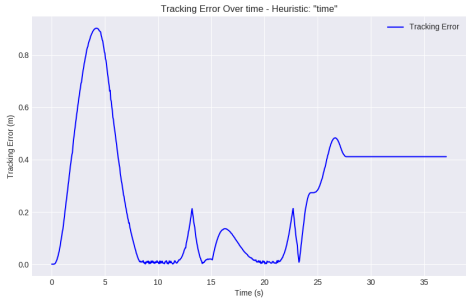

Figure 86: Absolute Tracking Error - **Time Heuristic**

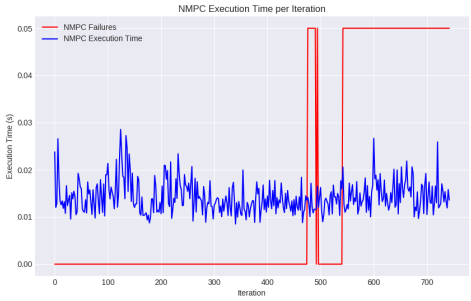

Figure 87: NMPC Execution Time and Failures - **Time Heuristic**

7.2.2 Double office 10 people

In the scenario described in this section, we've added five more people to our simulation to test the robot's navigation skills in a busier environment.
Using the Prediction Heuristic the robot successfully navigated to the goal, adhering to the bounds on both speed (Figure 89) and acceleration (Figure 90), while also

meeting the necessary constraints and maintaining a trajectory closely aligned with the planned path (Figure 88) (Figure 91), even in a denser and more complex environment. The execution time of the NMPC was within acceptable limits. Notably, the NMPC algorithm did not encounter failures except if the humans directly went against TIAGo.

As in the case of the Prediction Heuristic, the Next Waypoint Heuristic did well in guiding the robot to the goal, while staying within the speed (Figure 94) and acceleration (Figure 95) limits, and sticking closely to the intended path (Figure 93). However, there were a few spots along the way where the robot have a higher tracking error (Figure 96). Also, because we assigned the constaint area index to the waypoints manually before running the simulation, this approach isn't as flexible or strong compared to the Prediction Heuristic.

Contrary to the successful navigation for the Prediction and Next Waypoint Heuristics, the robot utilizing the Nearest Waypoint approach encountered a critical failure in completing its path. This happened due to its inability to accurately retrieve the nearest waypoint for defining the constraint region. As a result, the robot stopped its movement in the second room, caught in a state of indecision between the preceding and upcoming waypoints. This result highlights one of the primary limitations of the Nearest Waypoint Heuristic (Figure 98).

Finally, the Time Heuristic didn't manage to get the robot to the goal (Figure 103). It tried to switch the convex area constraint index based on how long it would take the robot to get close to the next area. The big issue here is that the robot acts differently every time because of how people move around and other changes in the simulation. Since the robot's behavior varies within each run, sticking to a predetermined time schedule to switch areas just didn't work.

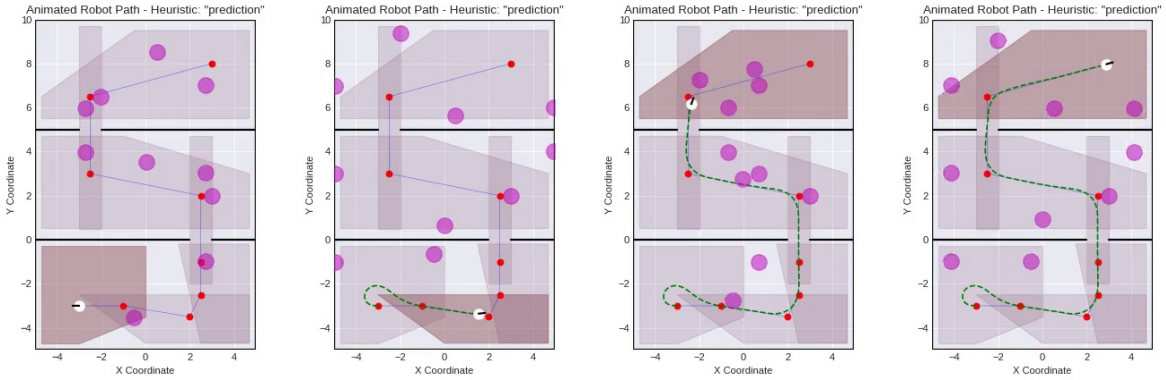

Figure 88: The robot path for the Prediction Heuristic

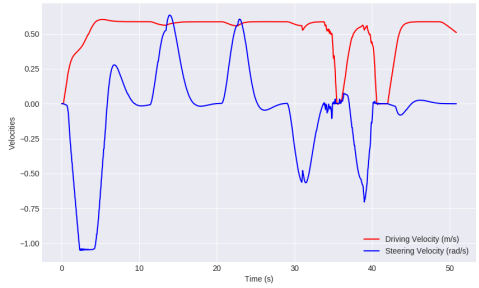

Figure 89: Driving and Steering Velocities - Prediction Heuristic

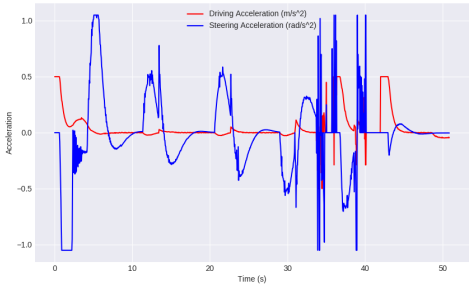

Figure 90: Driving and Steering Accelerations - Prediction Heuristic

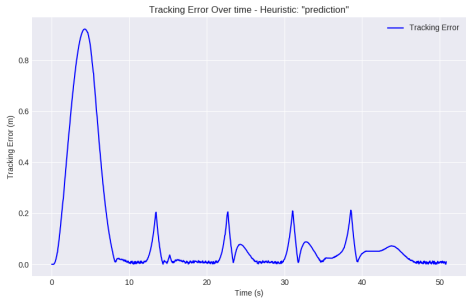

Figure 91: Absolute Tracking Error - Prediction Heuristic

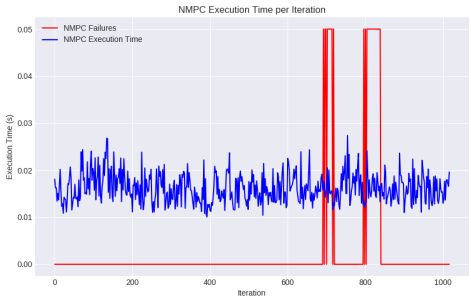

Figure 92: NMPC Execution Time and Failures - Prediction Heuristic

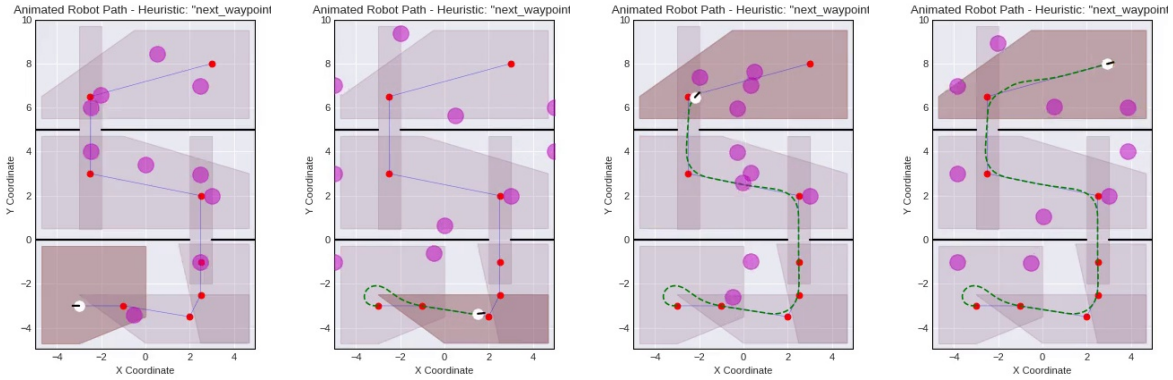

Figure 93: The robot path for the Next Waypoint Heuristic

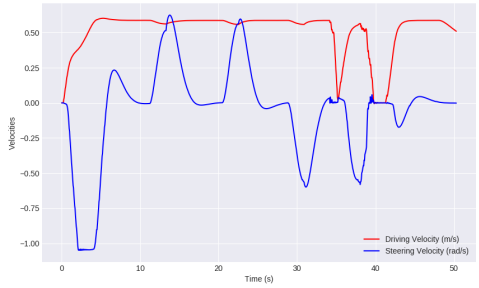

Figure 94: Driving and Steering Velocities - Next Waypoint Heuristic

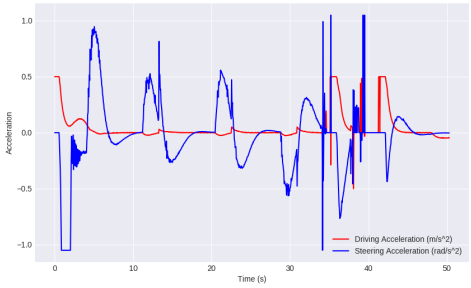

Figure 95: Driving and Steering Accelerations - Next Waypoint Heuristic

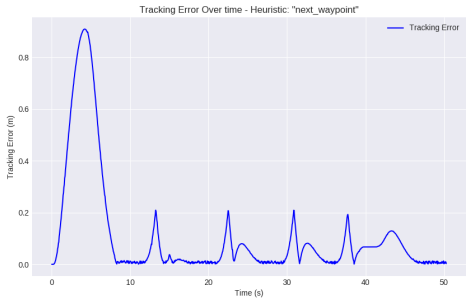

Figure 96: Absolute Tracking Error - Next Waypoint Heuristic

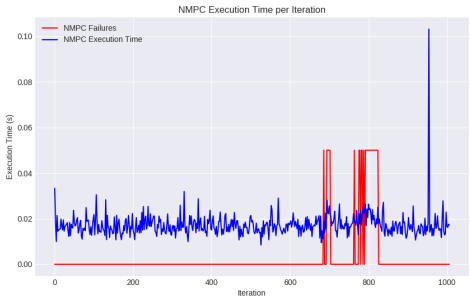

Figure 97: NMPC Execution Time and Failures - Next Waypoint Heuristic

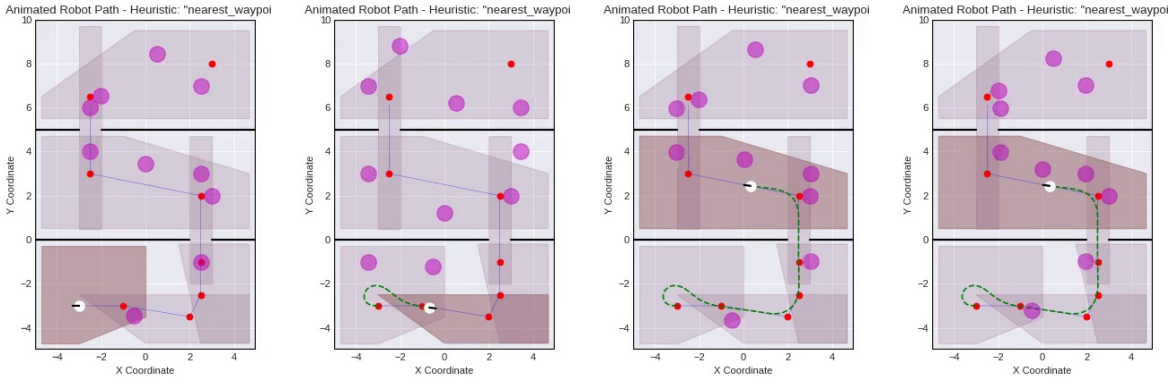

Figure 98: The robot path for the Nearest Waypoint Heuristic

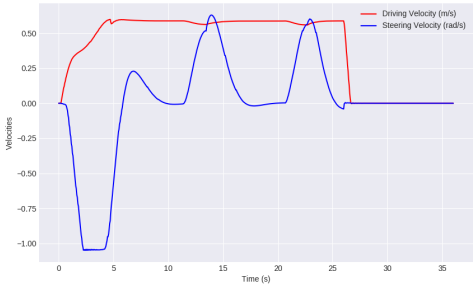

Figure 99: Driving and Steering Velocities - Nearest Waypoint Heuristic

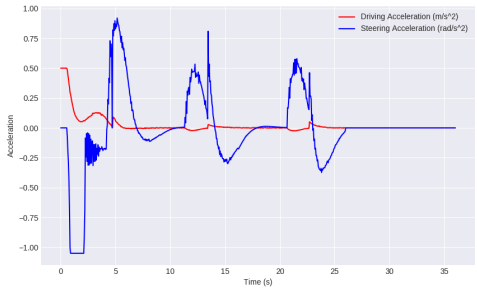

Figure 100: Driving and Steering Accelerations - Nearest Waypoint Heuristic

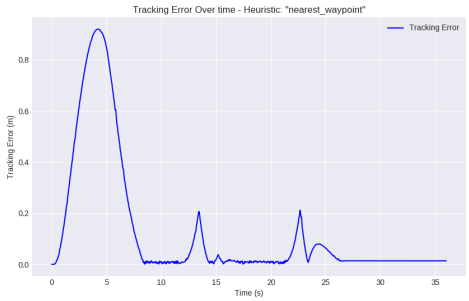

Figure 101: Absolute Tracking Error - Nearest Waypoint Heuristic

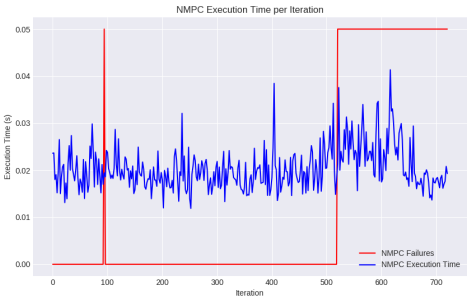

Figure 102: NMPC Execution Time and Failures - Nearest Waypoint Heuristic

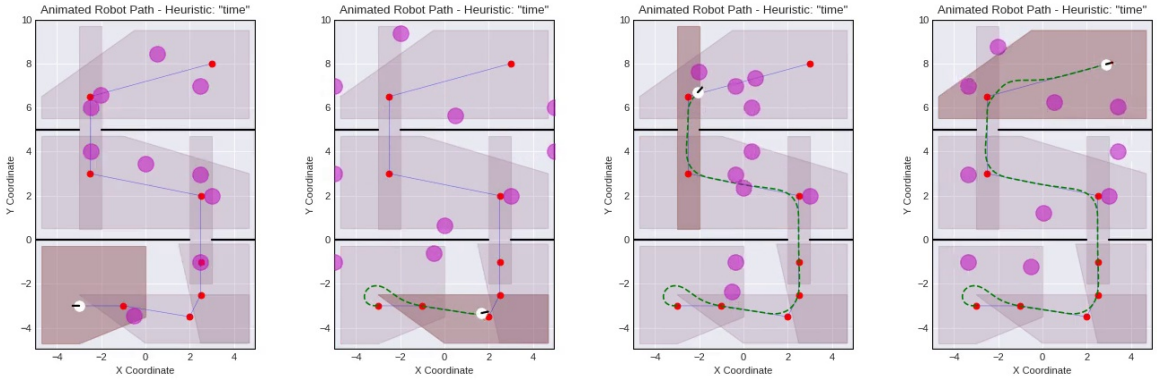

Figure 103: The robot path for the Time Heuristic

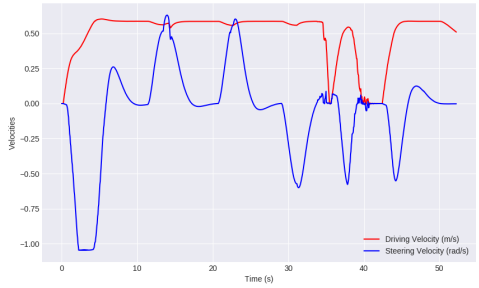

Figure 104: Driving and Steering Velocities - Time Heuristic

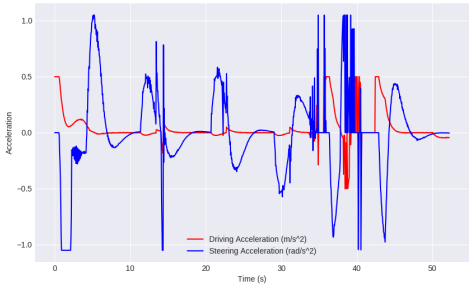

Figure 105: Driving and Steering Accelerations - Time Heuristic

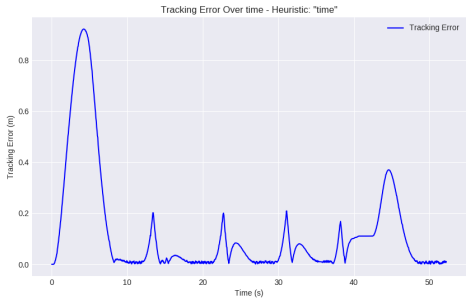

Figure 106: Absolute Tracking Error - Time Heuristic

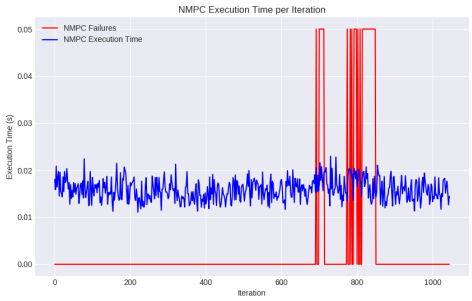

Figure 107: NMPC Execution Time and Failures - Time Heuristic

7.3 Corridor

In this final section, we present the results for the last environment in which we conducted simulations. As previously explained, this environment consists of two rooms and a corridor connecting them. It was tested with five people inside since,

being smaller than previous environments, adding more would have prevented the robot from moving within these narrow spaces. TIAGo reached the goal in every simulation using all the heuristics, demonstrating almost identical behaviors. Below, we report the results and the plots.

7.3.1 Corridor 5 people

Using the Prediction Heuristic the robot successfully navigated to the goal, adhering to the bounds on both speed (Figure 89) and acceleration (Figure 90), while also meeting the necessary constraints and maintaining a trajectory closely aligned with the planned path (Figure 88) (Figure 91). The execution time of the NMPC was within acceptable limits. Notably, the NMPC algorithm did not encounter failure except if the humans directly went against TIAGo.

As in the case of the Prediction Heuristic, the Next Waypoint Heuristic did well in guiding the robot to the goal, while staying within the speed (Figure 94) and acceleration (Figure 95) limits, and sticking closely to the intended path (Figure 93). Also, because we assigned the constaint area index to the waypoints manually before running the simulation, this approach isn't as flexible or strong compared to the Prediction Heuristic.

Contrary to the unsuccessful navigation shown in the Double Office world, this time the robot utilizing the Nearest Waypoint approach successfully completed its path. The overall results and plots are similar to the previous cases.

Finally, in this run, the Time Heuristic did manage to get the robot to the goal (Figure 103). Despite this, we can't consider this behaviour as robust as the robot acts differently every time because of how people move around. As we concluded in the previous simulations, sticking to a predetermined time schedule to switch areas does not provide a robust result.

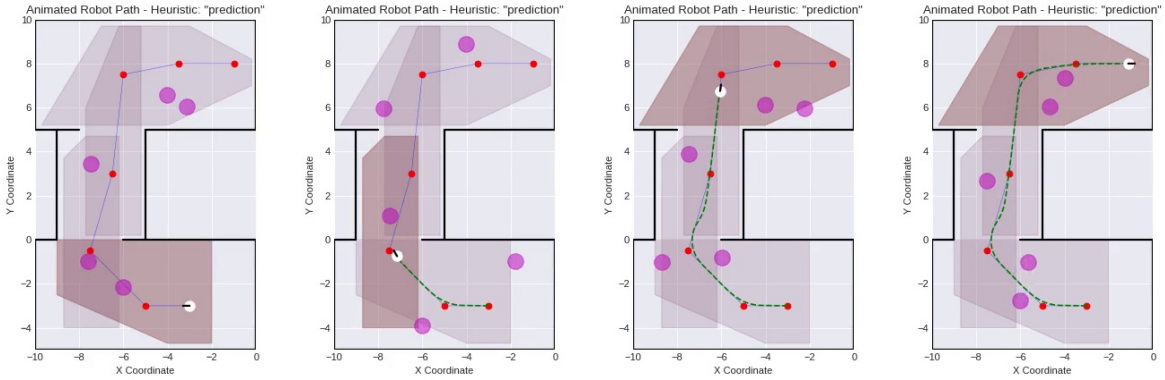

Figure 108: The robot path for the Prediction Heuristic

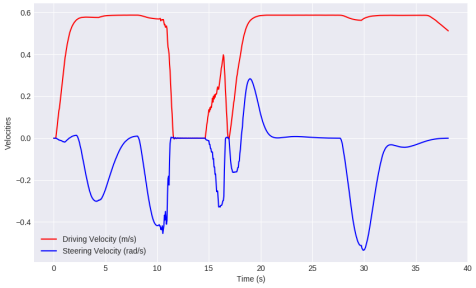

Figure 109: Driving and Steering Velocities - Prediction Heuristic

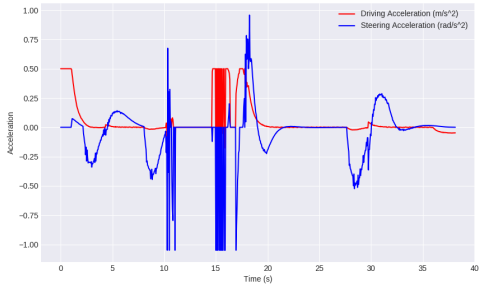

Figure 110: Driving and Steering Accelerations - Prediction Heuristic

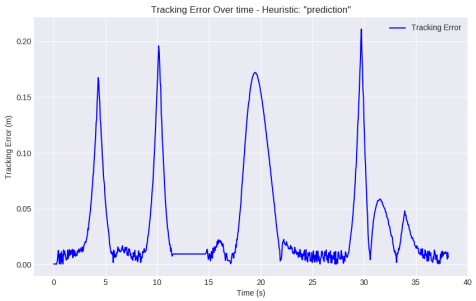

Figure 111: Absolute Tracking Error - Prediction Heuristic

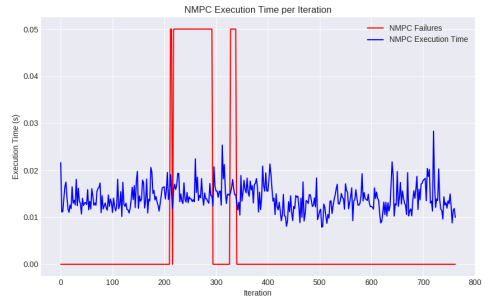

Figure 112: NMPC Execution Time and Failures - Prediction Heuristic

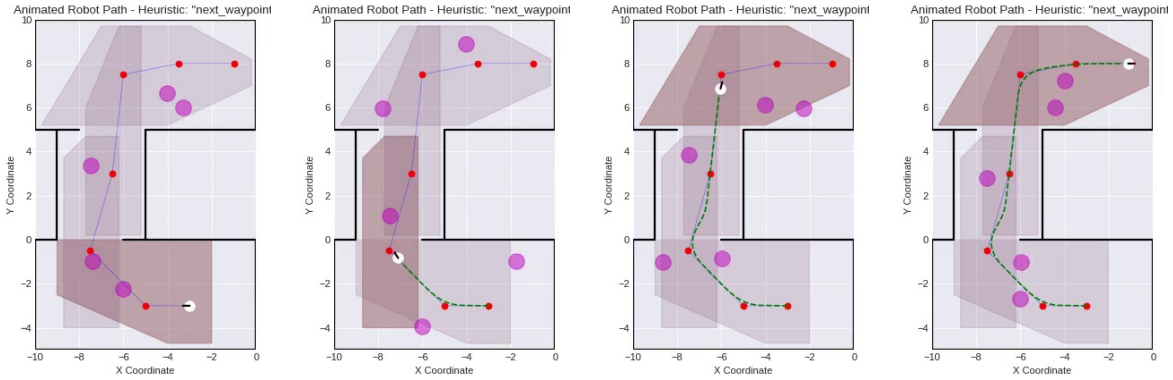

Figure 113: The robot path for the Next Waypoint Heuristic

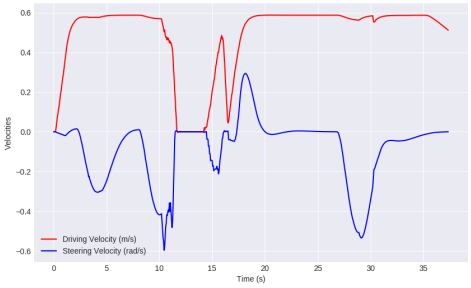

Figure 114: Driving and Steering Velocities - Next Waypoint Heuristic

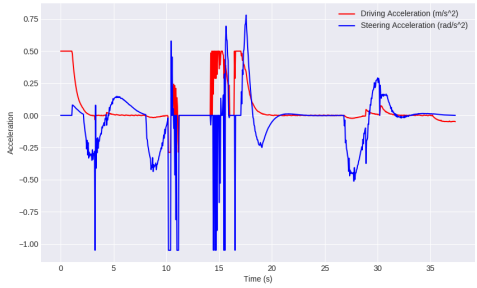

Figure 115: Driving and Steering Accelerations - Next Waypoint Heuristic

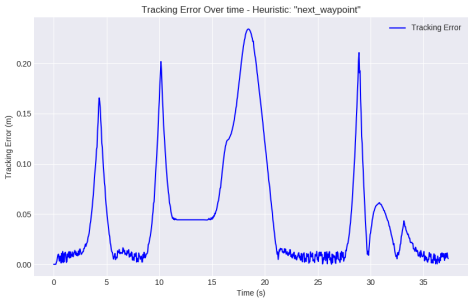

Figure 116: Absolute Tracking Error - Next Waypoint Heuristic

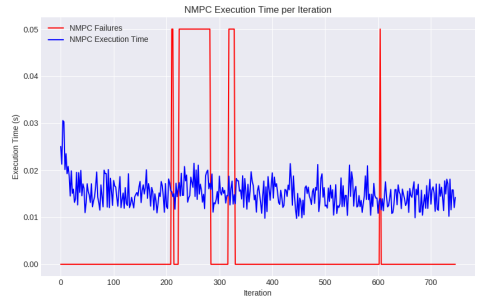

Figure 117: NMPC Execution Time and Failures - Next Waypoint Heuristic

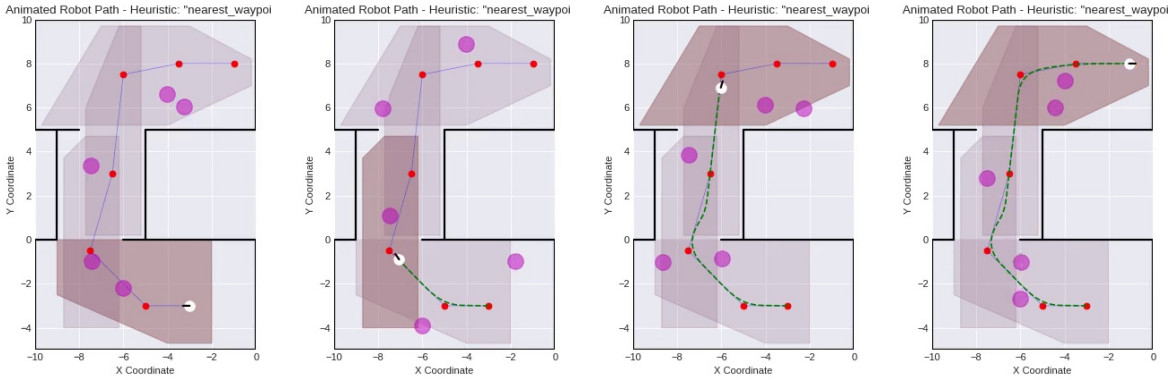

Figure 118: The robot path for the Nearest Waypoint Heuristic

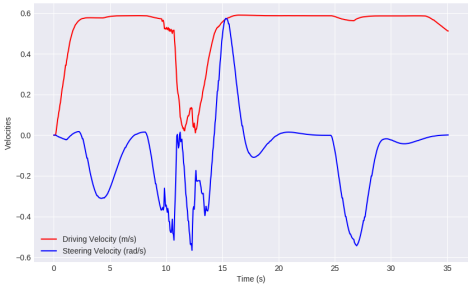

Figure 119: Driving and Steering Velocities - Nearest Waypoint Heuristic

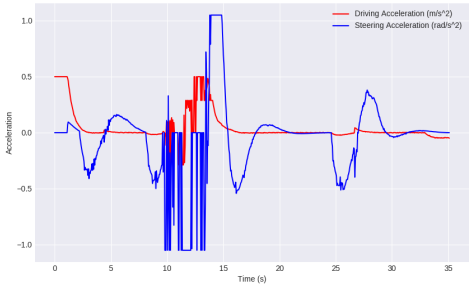

Figure 120: Driving and Steering Accelerations - Nearest Waypoint Heuristic

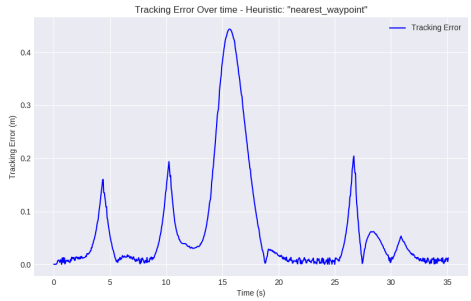

Figure 121: Absolute Tracking Error - Nearest Waypoint Heuristic

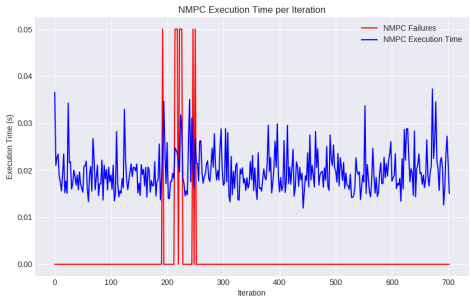

Figure 122: NMPC Execution Time and Failures - Nearest Waypoint Heuristic

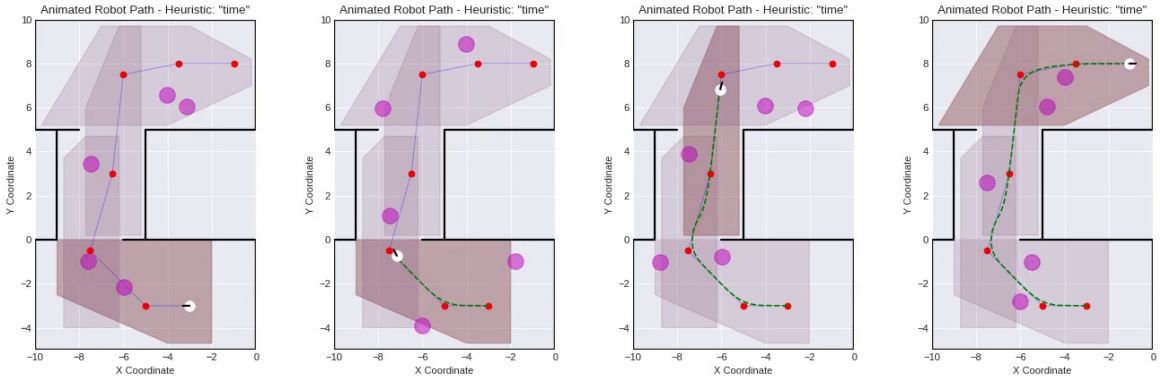

Figure 123: The robot path for the Time Heuristic

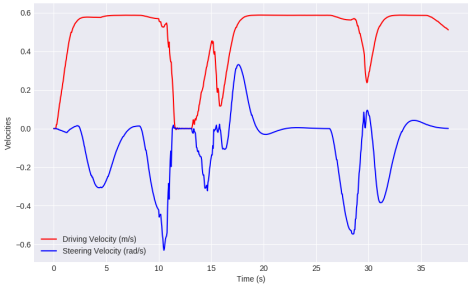

Figure 124: Driving and Steering Velocities - Time Heuristic

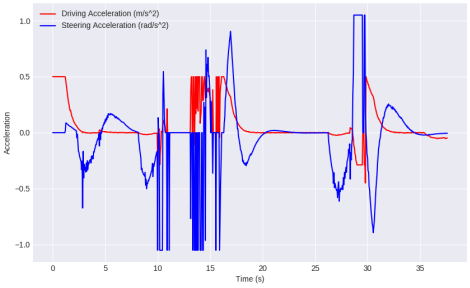

Figure 125: Driving and Steering Accelerations - Time Heuristic

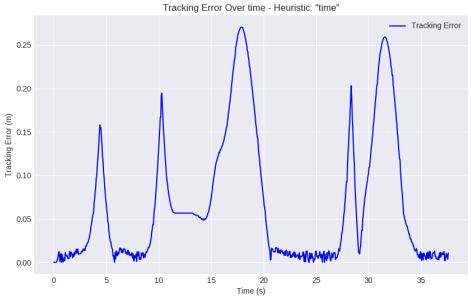

Figure 126: Absolute Tracking Error - Time Heuristic

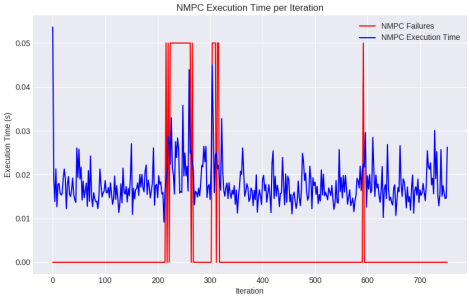

Figure 127: NMPC Execution Time and Failures - Time Heuristic

8 Conclusions

This project successfully demonstrated the development and implementation of a Nonlinear Model Predictive Control (NMPC) based navigation strategy for the TIAGo mobile robot in simulated environments using Gazebo and the Robot Operating System (ROS), imposing convex area constraints.

The foundational work utilized a sensor-based navigation scheme, leveraging Kalman filters for crowd prediction and a combination of Nonlinear Model Predictive Control (NMPC) and discrete-time Control Barrier Functions (CBFs) for motion generation. This effectively allowed for real-time adaptation to human movements, enhancing robot navigation safety and efficiency in crowded environments.

In our implementation, we integrate NMPC with nonlinear CBF and convex area constraints, which are updated at each control cycle. The defining of convex areas was guided by four distinct heuristics: prediction outputs from NMPC, proximity to the next or nearest waypoint, and simulation time.

Our testing environments varied in complexity, from a simple office space to a more intricate setting featuring multiple rooms and corridors, populated by varying numbers of people (5 to 15 individuals). These simulations exposed several critical insights and limitations. Notably, the heuristic based on the nearest waypoint and simulation time showed less robustness, occasionally failing to adapt to rapid changes in the environment or predict human movements accurately. This led to several instances where the NMPC could not prevent collisions, especially in scenarios where humans were not aware of the robot's presence.

It is evident that while our approach extends the foundational work's capabilities, there remain areas for further exploration. Incorporating a broader variety of settings, including those with static obstacles, could provide deeper insights into the flexibility and scalability of our approach. Conducting tests where humans are aware of the robot's presence might yield different behavioral dynamics, potentially improving the predictability of human movements and the effectiveness of the NMPC. Investigating alternative heuristics could reduce reliance on handcrafted rules and increase the robustness of the system.

References

[1] Veronica Vulcano, Spyridon G. Tarantos, Paolo Ferrari, and Giuseppe Oriolo. Safe robot navigation in a crowd combining nmpc and control barrier functions. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3321–3328, 2022.