**POLITECNICO DI MILANO**
**Corso di Laurea Magistrale in Ingegneria dell'Automazione**
**Dipartimento di Elettronica Informazione e Bioingegneria**

POLITECNICO
MILANO 1863

# Autonomous navigation in human crowded environments: integrating model predictive control within global planning

**Relatore: Prof. Luca Bascetta**
**Correlatore: Prof. Maria Prandini**

**Tesi di Laurea di:**
**Silvia Tellarini, matricola 892261**

**Anno Accademico 2018-2019**

# Abstract

The work developed in this thesis is in the field of mobile robotics and focuses on autonomous navigation in human crowded environments, with specific reference to autonomous personal mobility vehicles. The goal is to develop an innovative solution to the problem of autonomous navigation in human crowded environments, resting on the integration of two layers: a Global Planner and a Local Planner. The Global Planner is the layer specialized in planning, which computes a feasible trajectory, in a given environment, by choosing a feasible geometric path and endowing it with the time information. The geometric path is determined exploiting a Probabilistic RoadMap algorithm. The Local Planner solves a trajectory tracking problem while detecting obstacles from sensor data and ensuring collision avoidance, pedestrian safety and comfort. A novel approach is adopted to enable socially compliant human-robot interactions, based on a socially aware navigation model. This is achieved through a Model Predictive Control scheme, where the distance from the reference trajectory provided by the Global Planner is minimized, subject to operational, collision avoidance, safety and human comfort constraints. The two layers are intertwined, in order to allow the Local Planner to correct the trajectory designed by the Global Planner according to the detected obstacles, and the Global Planner to intervene and replan a new trajectory whenever the Local Planner is not able to find a feasible solution to the trajectory tracking problem. Simulation results show the effectiveness of the proposed solution.

# Sommario

Il lavoro sviluppato in questa tesi appartiene al campo della robotica mobile e si concentra sulla navigazione autonoma in ambienti affollati, con specifico riferimento ai veicoli autonomi per la mobilità personale. L'obiettivo è sviluppare una soluzione innovativa al problema della navigazione autonoma in ambienti affollati, basata sull'integrazione di due livelli: un Pianificatore Globale e un Pianificatore Locale. Il Pianificatore Globale è il livello specializzato in pianificazione, che calcola una traiettoria fattibile, in un dato ambiente, scegliendo un percorso geometrico fattibile e dotandolo delle informazioni temporali. Il percorso geometrico viene determinato sfruttando un algoritmo del tipo Probabilistic RoadMap. Il Pianificatore Locale si occupa dell'inseguimento della traiettoria, rilevando al contempo gli ostacoli dai dati dei sensori al fine di prevenire le collisioni e garantire la sicurezza e il comfort dei pedoni. È stato adottato un nuovo approccio per consentire la coesistenza nello stesso ambiente di umani e robot, basato su un modello di navigazione consapevole dei vincoli sociali. Ciò è ottenuto attraverso uno schema di tipo Model Predictive Control, soggetto a vincoli operativi, di prevenzione delle collisioni, di sicurezza e comfort, nel quale la distanza dalla traiettoria di riferimento fornita dal Pianificatore globale viene minimizzata. I due livelli sono interconnessi, per consentire al Pianificatore Locale di correggere la traiettoria progettata dal Pianificatore Globale in base agli ostacoli rilevati e al Pianificatore Globale di intervenire e ripianificare una nuova traiettoria ogni volta che il Pianificatore Locale non è in grado di trovare una soluzione al problema di inseguimento della traiettoria. I risultati delle simulazioni mostrano l'efficacia della soluzione proposta.

# Contents

# List of Figures

# Chapter 1

# Introduction

The latest two decades have witnessed an explosion in the research activities in the field of mobile robotics. In fact, mobile robotics related applications are emerging in the market, including aerial, ground and underwater variants, and their presence in households, offices and public places is increasing each day. Methods for safe and efficient autonomous navigation in human crowded environments are needed, which should take into account the constraints of human comfort as well as social rules. Indeed, a great deal of attention has been directed towards mobile robot autonomy, efficiency, reliability, and safety. In particular, planning, control, and perception play an important role in terms of safety and performance of an autonomous vehicle performing challenging tasks in complex environments.

## 1.1 Context and Motivations

The work developed in this thesis belongs to the field of mobile robotics and focuses on autonomous navigation in human crowded environments, with specific reference to Autonomous personal mobility vehicles (APMV). APMV are a promising technology that will facilitate social participation, enabling active and healthy aging and providing services for super-aging societies.
According to [7], in 2010 the number of people in the U.S.A. who used a wheelchair to assist with mobility was of 3.6 million. Among those aged 65 and older, roughly 2.0 million people used a wheelchair and 7.0 million used a cane, crutches, or a walker. As the population ages, the percentage of people with disabilities increases. The use of autonomous

wheelchairs for elderly people with deteriorated physical and cognitive functions could expand their range of activity and socialization.

## 1.2 Problem formulation

The main goal of this thesis is to develop a solution that allows an autonomous vehicle to move safely in scenarios with unknown and dynamic obstacles, in particular people. Indeed, the analysis takes into account a crowded environment, in which the robot will navigate with the purpose of reaching a desired location on a map while, at the same time, performing collision avoidance in order to guarantee human safety and comfort.

In the literature, many works like [5],[8],[12] and [22] introduce two different levels of control to solve the autonomous navigation problem. In this thesis this concept is further developed, integrating the two levels. The level specialized in planning, also known as Global Planner, computes a feasible trajectory within the environment by choosing a geometric path and endowing it with the time information. Plenty of algorithms in the robotic literature solve the path planning problem. Some widely used methods of remarkable efficiency are Probabilistic Roadmaps [26], Rapidly-exploring Random Tree [27], A* [31], Potential Field [22]. Furthermore, there are examples about many new approaches based on Neural Networks [40] and Genetic Algorithms [39].
The level specialized in Control, also known as Local Planner, solves a trajectory tracking problem while reactively acting in order to detect obstacles from sensor data and ensure collision avoidance. Many approaches for trajectory tracking and collision avoidance in the literature are based on the Dynamic Window Approach [16] and on the Timed Elastic Band [38], [37], [36] algorithms. Model Predictive Control (MPC) [34] has gained interest, in the past decade, in the field of autonomous vehicle control. This advanced control method is particularly suitable for a large number of applications, thanks to its flexible constraint handling capabilities and the possibility of formulating the control problem as an optimization program, as highlighted in [28] and [17]. In particular, the MPC control method allows to specify constraints for obstacle avoidance, actuator limitations, pedestrian safety and comfort in the optimization problem, using a model of the system to predict its state at future time instants. In order to allow autonomous vehicles to navigate in human-

crowded environments safely and to take into account human comfort, it is necessary to model the human behaviour, as in [9], [3] and [19].

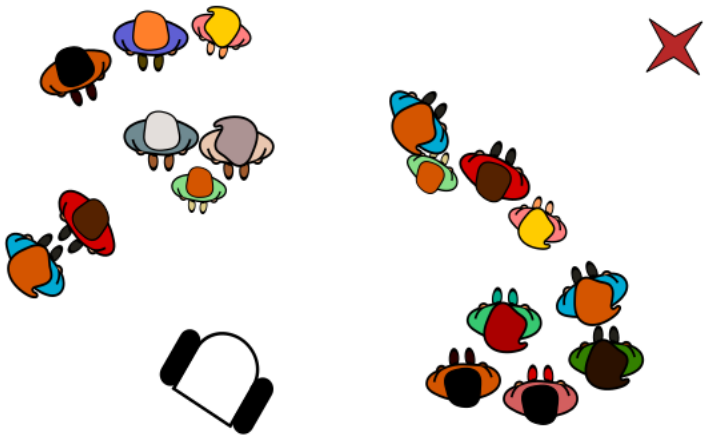


*Figure 1.1: Autonomous navigation in human crowded environment*

## 1.3 Thesis Outline

This thesis is composed of five main chapters, an introduction and a conclusion.

The first chapter, entitled Introduction and problem formulation, illustrates the context, the motivations and provides the problem formulation.

In the second chapter, the proposed solution strategy, elaborated integrating model predictive control within global planning, is presented.

In the third chapter, a global planner is developed to generate the reference trajectory based on the prior knowledge of the environment in two steps: choosing a geometric path and endowing it with the time information.

In Chapter 4, a collision free region surrounding the wheelchair is defined, which takes into account pedestrians and obstacles, as well as human comfort, with the introduction of the concept of proxemics. To

allow the use of convex optimization tools in the control problem, the region needs to be represented as a convex obstacle free region.

Chapter 5 deals with the MPC problem formulation for trajectory tracking and obstacle avoidance, in order to achieve a safe and socially-aware navigation.

Chapter 6 presents the results obtained through simulations within a crowded environment, motivating the implementation choices related to the control parameters.

In Chapter 7, some conclusions are drawn and future developments are proposed.

# Chapter 2

# Proposed strategy for autonomous navigation in human crowded environments

Moving towards the target destination while simultaneously avoiding obstacles and reacting to environmental changes is a natural task for humans, but may not be so for an autonomous vehicle. Indeed, autonomous navigation can be a challenging task in human crowded environments such as schools, shopping malls, or sidewalks. It requires efficient motion planning, safe and human-comfortable navigation in dynamic environments, obstacle detection and tracking, world state estimation and motion prediction.

In general, the vehicle autonomy can be achieved in three main steps: perception and localization, trajectory planning and vehicle control.
The perception of the surrounding environment is accomplished by exploiting exteroceptive sensors such as cameras, sonars, ToF laser sensors, 3D depth sensors etc., in order to detect pedestrians and other unknown objects, whereas the vehicle is localized within an unknown environment with proprioceptive sensors, such as IMU and GPS.
Trajectory planning governs the actual vehicle transition from one feasible configuration to the following one, influencing both the kinematic and dynamic properties of the motion. It makes use of the known environmental data such as satellite imagery, street and indoor maps to plan a safe and smooth trajectory, complying with the vehicle dynamic limitations. However, the vehicle may encounter previously unknown obstacles and pedestrians on its way to the final destination. As a rule, it

is not possible to assume a collaborative external behaviour, since there are contexts in which humans are not used to coexist and cooperate with autonomous vehicles. Therefore, once a suitable trajectory is generated, the next step is to make use of the data acquired with perception to control the vehicle in a safe and human-friendly way, while tracking the reference trajectory with the desired speed profile defined by the planning module. Indeed, an optimal control manoeuvres the vehicle actuators in order to obtain a safe, human-friendly and collision-free vehicle motion.

## 2.1 A two-layer approach

This thesis proposes a complete solution for the problem of autonomous navigation in human crowded environments, based on an innovative method that integrates two different layers, in order to guarantee both planning and control, exploiting the sensory data from perception. As already mentioned, the motion strategy must rely on exteroceptive sensor information to move safely in environments with unknown and dynamic obstacles. The layer specialized in planning, also known as Global Planner, computes a feasible trajectory, considering the given environment. On the other hand, the controller, also known as Local Planner, solves a tracking problem while reactively acting in order to detect obstacles from sensor data and ensure collision avoidance, pedestrian safety and comfort. The two levels are intertwined, in order to allow both the Local Planner to correct the trajectory designed by the Global Planner according to the obstacles detected, and the Global Planner to intervene and replan a new trajectory whenever the Local Planner is not able to solve the trajectory tracking problem. In the following, the two layers will be analysed more in detail.

In order to reach the desired goal, autonomous vehicles need to plan a collision-free trajectory. In fact, the Global Planner determines the vehicle reference trajectory within the considered environment in two steps: choosing a geometric path and endowing it with the time information. Path planning is the task of computing a continuous path that drives the vehicle from the start to the goal configuration, while satisfying motion constraints and guaranteeing feasibility. In this work and for testing purposes, the Probabilistic Roadmaps (PRM) [26] algorithm was selected to solve the path planning problem, because of its effectiveness and ease of implementation. PRM consists in a network

graph of possible collision-free paths in a given map. As already specified, only known fixed obstacles are considered in this first analysis, e.g. walls of the selected environment. The reference trajectory between the waypoints is then calculated by determining a timing law through a Trapezoidal Velocity Profile based on the approach in [35], imposing constraints on maximum velocity, maximum acceleration, desired travelling distance, Jerk time and Snap time. The desired trajectory tracking and collision avoidance are then performed by the Local sensor-based Planner, a Model Predictive Control (MPC) scheme based on a quadratically constrained quadratic programming model. This advanced control method is particularly suitable for a large number of applications, thanks to its flexible constraint handling capabilities and the possibility to formulate the control problem as an optimization one, where operational constraints can be enforced and predictions can be included. In the context of MPC-based autonomous vehicle control, with specific reference to our case (control of the unicycle-type model, as will be further explained in Chapter 5), we rely on linear models thanks to a standard feedback linearization procedure, similar to the one adopted in [29] and [13]. This approach allows to simplify the MPC optimization problem, allowing for a real-time efficient implementation, and to formulate operational, safety, human-comfort, and collision avoidance requirements as constraints. To take into account actuator limitations and comfort requirements, both quadratic and linear maximum velocity constraints are enforced on the vehicle longitudinal speed, as well as linear constraints on the velocity variation. To perform collision avoidance, linear state constraints are enforced, based on a novel approach developed in this thesis that makes use of the concept of Velocity Obstacle ([15], [14]), and of the introduction of a sensor-based circular Virtual Box. Finally, to further fulfill pedestrian comfort requirements, the concept of Proxemics (introduced in [21]) is presented and applied with the introduction of Slack variables. In fact, in order to allow autonomous vehicles to navigate in human-crowded environments safely and taking into account human comfort, it is necessary to model human navigation behaviours. The method presented in this thesis relies on the introduction of soft position constraints ([23]) that consider the navigation behavior of interacting pedestrians, modeling their personal space (as in [25]), and exploiting the concept of Proxemics in order to avoid it.

Whenever the Local Planner optimization fails, i.e. the MPC does not

find any feasible solution, it means that the robot encountered previously unknown obstacles that obstruct the planned reference trajectory. If the optimization fails for more than a specified number of subsequent iterations, the trajectory tracking stops and the Global Planner is called again to plan a different reference trajectory, as shown in Figure 2.1.

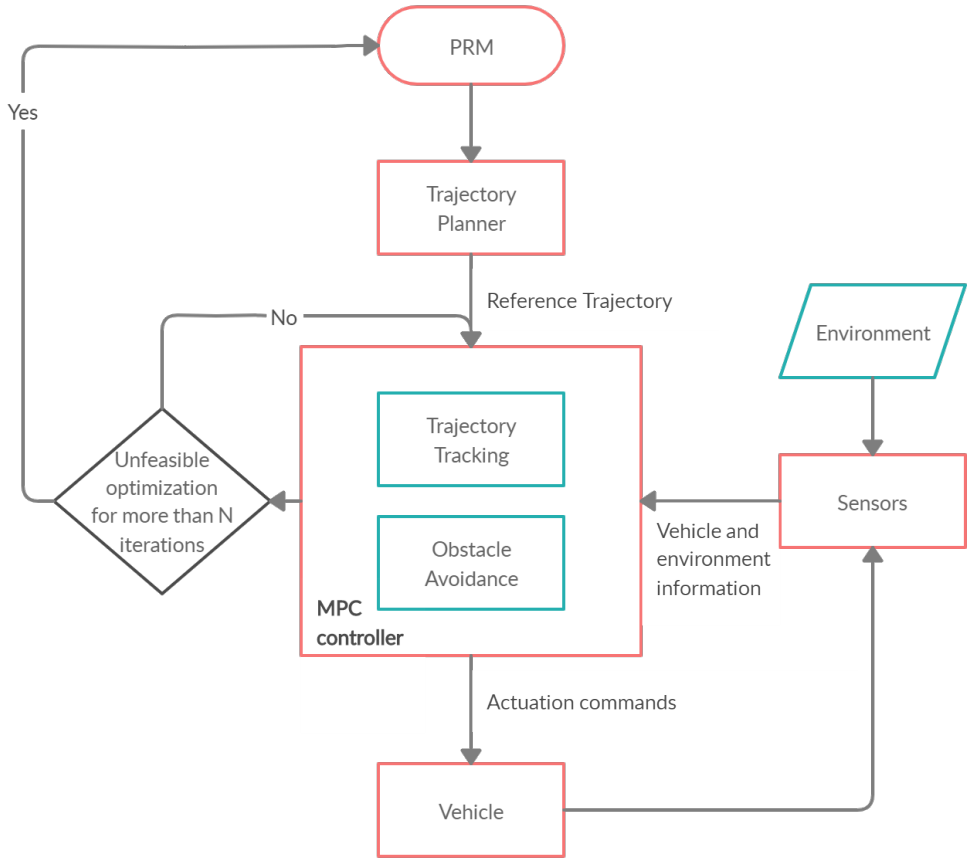



*Figure 2.1: Proposed solution strategy*

# Chapter 3

# Global planning for trajectory generation

Trajectory planning is a crucial part in the field of autonomous navigation. It governs the actual vehicle transition from one feasible configuration to the following one, influencing both the kinematic and dynamic properties of the motion. A trajectory is preferable rather than a simple reference path or a constant speed reference, since it allows to prevent undesired strong acceleration and jerk, and provides a more precise time varying reference value to be fed to the trajectory tracking controller.

The trajectory generation problem can be formulated as in [4]. Given a starting configuration $q(t_i) = q_s$ and a goal configuration $q(t_f) = q_g$ within a specified environment, the collision-free trajectory planning problem is the one of finding $q(t)$ for $t \in [t_i, t_f]$ subject to the initial and final configuration constraints, and can be broken down into

1. finding a collision-free geometric path $q(s)$, with $dq(s)/ds \neq 0$

2. determining a timing law $s = s(t)$, with $s \in [s(t_i), s(t_f)]$.

Indeed, in this work, the global planner determines the reference trajectory with a priori knowledge of the environment, e.g. given the map of the environment, in two steps: choosing a feasible geometric path and endowing it with the time information.

## 3.1 PRM

Path planning is the task of computing a feasible path that will drive the vehicle from the start to the goal configuration, while satisfying motion constraints and guaranteeing feasibility. A path can be described as a sequence of waypoints defining the trajectory coarsely.

Before analyzing the state of the art motion planning algorithms, the notion of Configuration Space, as well as the feasible path planning problem, are introduced. To this aim, by defining a single vehicle configuration $q(t) \in R^n$ as the vector of generalized coordinates describing the robot pose inside a $n$-dimensional space, the Configuration Space $Q \subset R^n$ is the set representing all the possible system configurations. Moreover, it is possible to define the obstacle region $Q_{obs} \subset Q$ as the set including the configurations that lead to collisions in the given environment, while the free space $Q_{free} := Q \backslash Q_{obs}$ is the set of collision free configurations. Considering a feasible path planning problem $(Q_{free}, q_{init}, Q_{goal})$ as in Figure 3.1, where the initial condition $q_{init} \in Q_{free}$ is the initial robot configuration and the goal region $Q_{goal} \subset Q_{free}$ is the set of desired configurations to be reached, the aim is to find a feasible (continuous and collision-free) path $\sigma : [0, 1] \rightarrow Q_{free}$, such that $\sigma(0) = q_{init}$ and $\sigma(1) \in Q_{goal}$, if one exists. If no such path exists, the planning algorithm must report failure.

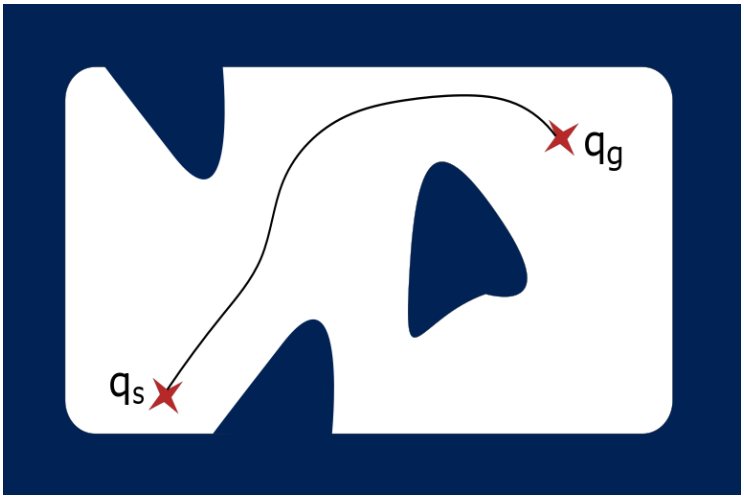



*Figure 3.1: Path Planning Problem example, $q_s = q_{init}$ and $q_g \in Q_{goal}$. In blue the obstacle region $Q_{obs}$, in white the free space $Q_{free}$.*

By exploiting a path planning algorithm, a feasible geometric path is generated from the initial configuration $q_i$ to the final goal $q_f$, passing through appropriate waypoints.
Plenty of algorithms in the robotic literature solve the path planning problem. Considerable attention has been directed towards the creation of efficient heuristic planners, since complete path planning methods, applied to robots performing challenging tasks autonomously in complex environments, may have overwhelming complexity. Indeed, exact motion planning for high-dimensional systems is too computationally intensive, thus hardly applicable in practice. On the other hand, there are other widely used heuristic methods of remarkable efficiency like Probabilistic Roadmaps (PRM) [26], Rapidly-exploring Random Tree (RRT) [27], A* [31], Potential Field [22]. Furthermore, there are examples about many new approaches based on Neural Networks [40] and Genetic Algorithms (GAs) [39].

In this work and for testing purposes, the selected path planning algorithm is the Probabilistic Roadmaps method, because of its ease of implementation. As already mentioned, this thesis focuses on autonomous navigation in human crowded environments, with a great attention to obtain a safe, human-friendly and collision-free vehicle motion. However, the chosen path planning algorithm does not include in the computation the presence of pedestrians and other moving obstacles, even though asymptotically optimal sampling-based motion planning algorithm for real-time navigation in dynamic environments exist, such as RRTX [30]. The reason for the choice of the PRM path planner is that this thesis is mainly focused on the development of an efficient, reactive and human-aware Local Planner, and on its integration within the Global Planner, which was implemented only to obtain a feasible trajectory to be followed.

Given a path planning problem, the Probabilistic Roadmaps method proceeds in two phases: a learning phase and a query phase, generating a feasible path as in Figure 3.2. During the learning phase a roadmap is constructed in a probabilistic way for the considered environment, as summarized in Algorithm 1. As described in [26], the roadmap is an undirected graph $R(N, E)$ whose nodes $N$ are a set of randomly sampled robot configurations suitably chosen over the free space. The edges in $E$ correspond to feasible paths connecting the respective nodes. In the query phase, the roadmap is searched for a path joining the start and

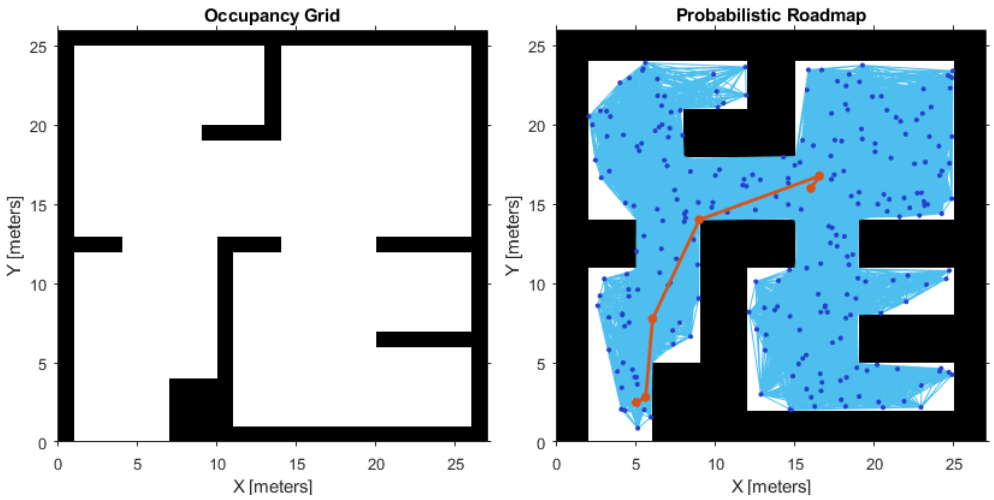
goal configurations by solving individual path planning problems in the input scene. It consists in a search of the shortest path in the graph, such as Dijkstra's algorithm [11].

**Algorithm 1** PRM: Roadmap Construction

```
1: V ← 0; E ← 0;
2: for i = 0, ..., N do
3:    q_rand ← Samplefree_i;
4:    U ← Near(G, q_rand, r);
5:    V ← V ∪ {q_rand};
6:    for each u ∈ U in order of increasing ||u − q_rand|| do
7:       if q_rand and u are not in the same connected component of G
   then
8:          if CollisionFree(q_rand, u) then
9:             E ← E ∪ {(q_rand, u)};
10:         end if
11:      end if
12:   end for
13: end for
14: return G = (V, E);
```

In Algorithm 1, the function $Samplefree_i$ returns a random free configuration sampled from the free space, the function $Near(G, q_{rand}, r)$ returns the vertices of the graph $G$ that are contained in a ball or radius $r$ centered at $q_{rand}$, while the boolean function $CollisionFree(q_{rand}, u)$ returns true if the segment connecting $q_{rand}$ and $u$ is collision free, false otherwise.

As already mentioned, whenever the Local Planner does not find a solution to the trajectory tracking problem, i.e. the Model Predictive Control optimization fails repeatedly, the path needs to be replanned. The number of nodes constituting the Probabilistic Roadmap is then increased, in order to determine more feasible paths and boost the efficiency of the final path. Finally, the PRM algorithm recalculates the node placement and generates a new network of nodes and a new feasible path.




*(a) A priori knowledge of the environment* *(b) Probabilistic RoadMap planning*

*Figure 3.2: Example of the Probabilistic Roadmap alghorithm results, given the map of the environment.*

## 3.2 Time law

Path planning involves the computation of a feasible path from start to goal configuration, whereas trajectory generation requires the consideration of robot dynamics and actuator constraints as well.

Once a feasible path, represented by waypoints (a sequence of collision-free points along the path), has been provided by the path planning algorithm, the trajectory is determined by introducing a suitable timing law. Different timing laws can be found in literature, such as Polynomial, Spline, Harmonic, Cycloidal, Trapezoidal in the velocity, etc. [2].

In this work and for testing purposes, a Trapezoidal Velocity Profile (TVP) was chosen. A Trapezoidal move profile is characterized by three different sectors, as can be seen from Figure 3.3: a constant acceleration (up to the desired maximum velocity) region, a constant velocity region, and a constant deceleration region (up to zero velocity). This approach generates the reference trajectory as a dense sequence of intermediate points in the free space scheduled at a fixed rate (depending on the controller sampling time), taking into account constraints related to maximum velocity, maximum acceleration, desired travelling distance, Jerk time and Snap time. The resulting trajectory thus consists in a linear position profile adjusted with parabolic bends. However, since Trajectory generation aims at creating a trajectory interpolating two or

more waypoints, in order to avoid a motion that stops in each waypoint, it may be useful to start the planning of a trajectory before the end of the preceding one.

For testing purposes, a Trapezioidal Velocity Profile implementation based on the approach in [35] was used to return the position, velocity and acceleration profiles for a snap controlled law from the specified constraints on maximum velocity, maximum acceleration, desired traveling distance, Jerk time and Snap time.

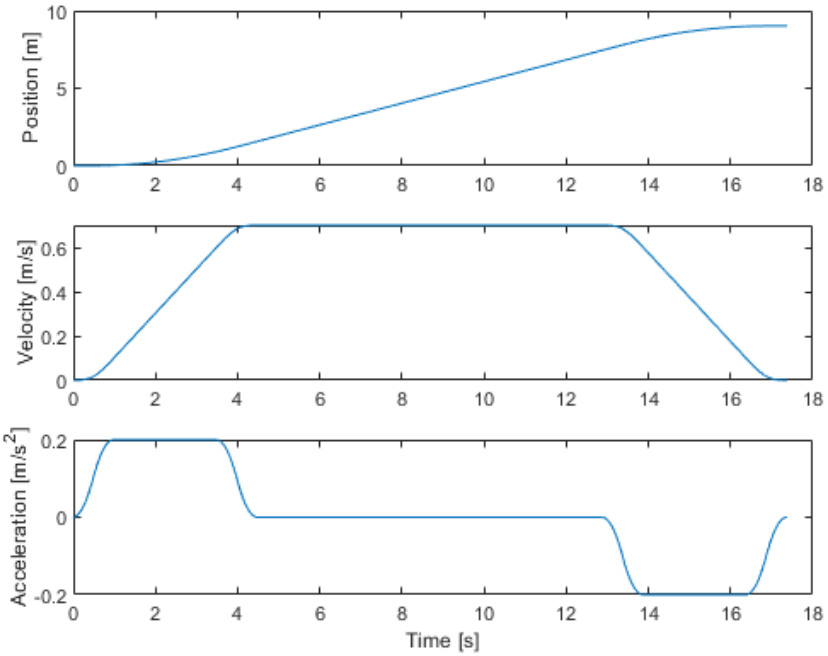



*Figure 3.3: Timing law determination*

# Chapter 4

# Avoiding static and moving obstacles

Autonomous navigation can be a challenging task in dynamic and human crowded environments, such as schools, shopping malls, or sidewalks, since it involves being able to react quickly and safely when the relationship with the surrounding objects changes. It is necessary to guarantee safety for all the agents (people, vehicles and the wheelchair itself) moving in the space. The obstacles in the environment can be divided into two general classes: a priori known obstacles (already considered in the trajectory generation by the Global Planner) and unexpected obstacles encountered during the navigation. The on-line motion planning in a real context must then depend on the detection of unexpected obstacles, the tracking of the moving ones, e.g. pedestrians, and the prediction of the future states of the world. Indeed, pedestrians and other unforeseen obstacles, detected in real-time, need to be considered in the trajectory tracking problem.

To allow the use of convex optimization tools in the control problem, the obstacle-free space surrounding the wheelchair needs to be represented as a convex obstacle-free region.

An algorithm generating a convex obstacle-free region, which can be integrated within the Local Planner computation, is here proposed. This approach allows to compute a polytopic region of obstacle-free space within sensor range, by generating a set of hyperplanes which separate the convex region of space containing the wheelchair from the detected obstacles. The algorithm presented assumes that the obstacles themselves are convex. This assumption is fulfilled with the introduction of a

circular sensor-based virtual box, as it will be seen in the following sections. The iteration through the obstacles generates, for each obstacle in potential collision, a hyperplane which is tangent to the obstacle virtual box, and separates it from the convex region surrounding the vehicle. It is necessary to implement a similar approach, since it is not always possible to guarantee the convexity of the detected trust region (the free space of the environment in which the vehicle is able to move freely), due to the morphology of the environment and potential errors in the data acquisition.

Another approach was considered to determine a convex obstacle free region in human crowded environments. This method is based on the definition of an ellipsoidal region of obstacle-free space surrounding the wheelchair. This method guarantees the convexity of the trust region, and thus the admissibility of the optimization problem. However, only the first presented algorithm was effectively implemented (and it is here reported), while the approach (reported in Appendix A) resulted to be too computationally intensive for the considered hardware.

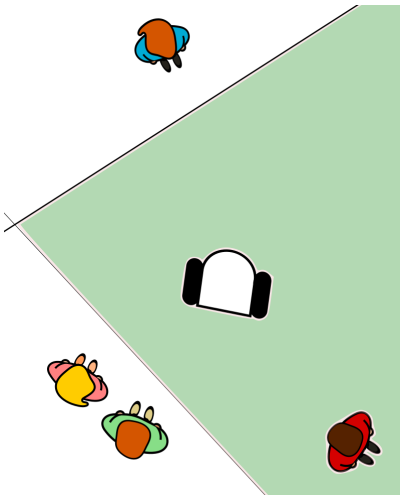


*Figure 4.1: The Trust Region, in green, is defined as the intersection of half planes.*

The obstacle-free space surrounding the wheelchair needs to be represented as a convex obstacle-free region that takes into account a priori known obstacles, pedestrians, and other unforeseen obstacles detected in real-time. Considering the vehicle Configuration space, the hyperplanes, and thus the constraints, delimiting the trust region (as it can be seen from Figure 4.1), take the form of:

$$h_x\ x + h_y\ y \leq l \tag{4.1}$$

where $h_x$ and $h_y$ are the coefficients related to the variables $x$ and $y$ identifying the slope of the 1-dimensional line, while $l$ is the constant term defining the distance from the origin. Each equation of the form (4.1) divides the state space into two half planes, one representing the admissible free space, while the other the forbidden space.
In this work, the trust region is reduced only if there is an obstacle in potential collision with the wheelchair. In fact, if there is no danger of potential collision, it is not necessary to limit the obstacle-free convex region surrounding the vehicle. In order to determine if an obstacle is in potential collision, the Velocity Obstacle method is introduced. Then the concept of virtual box is defined, allowing to delineate the linear limits dividing the trust region from the pedestrian space. Finally, the concept of proxemics is presented, in order to model a human-aware obstacle-free region definition that takes into account human comfort.

## 4.1 Velocity Obstacles

Differently from the approach presented in [12], in this work the constraints delimiting the trust region are selected based on the concept of Velocity Obstacles [14], which maps the dynamic environment into the robot velocity space. This method, similarly to the one presented in [15], allows to determine potential collisions, using velocity and position information about the robot moving in a time-varying environment. In fact, by mapping the dynamic environment into the robot velocity space, it is possible to determine the vehicle Collision Cone (Figure 4.2). This allows to retrieve and avoid those input velocities which would cause a collision with an obstacle at some future time, within a given time horizon.
The concept of Velocity Obstacle (VO) is here presented for a single obstacle, as it can be easily extended to multiple obstacles. In the analysis, both the vehicle and the obstacle are inscribed in a circular virtual box (as it will be explained in the next subsection), thus considering a planar problem with no rotations. The obstacles are assumed to move along arbitrary trajectories, while their instantaneous state is either known or measurable.
The analysis considers the two circular objects, $A$ representing the vehi-

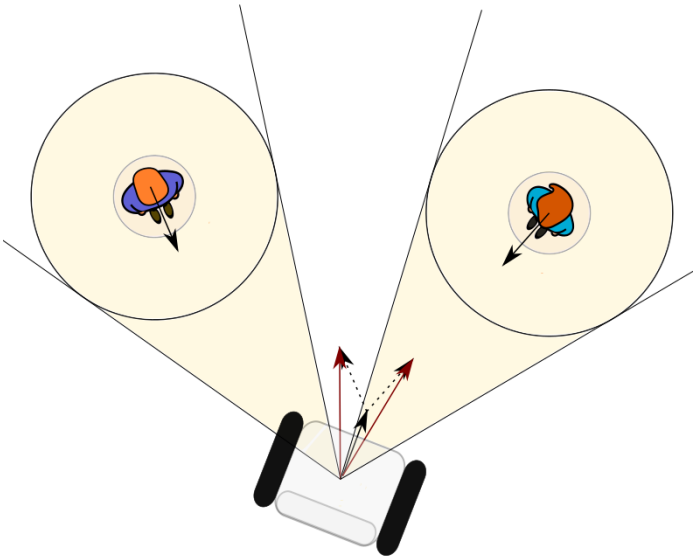


*Figure 4.2: Vehicle Collision Cones. Of the two approaching pedestrians, only one is in potential collision with the vehicle.*

cle and $B$ representing the obstacle, with velocities $v_A$ and $v_B$ at time $t$, as shown in Figure 4.3.
To compute the Collision Cone it is necessary to map the obstacle $B$ into the Configuration Space of the vehicle $A$. This is done by reducing $A$ to the point $\widehat{A}$ and by enlarging $B$ of the radius of circle $A$, becoming $\widehat{B}$. The state of each object can then be represented by its position and velocity vector attached to its center. The Collision Cone $CC_{A,B}$ is then defined as the set of colliding relative velocities between $\widehat{A}$ and $\widehat{B}$:

$$CC_{A,B} = \{v_{A,B} \mid \lambda_{A,B} \cap \widehat{B} \neq 0\} \tag{4.2}$$

where $\lambda_{A,B}$ is the direction of $v_{A,B}$, and $v_{A,B}$ is the relative velocity

$$v_{A,B} = v_A - v_B \tag{4.3}$$

The Collision Cone is the planar sector with apex in $\widehat{A}$, bounded by the two tangents $\lambda_f$ and $\lambda_r$ from $\widehat{A}$ to $\widehat{B}$, as shown in Figure 4.4.
The Collision Cone is defined as the the set of colliding relative velocities lying between the two tangents to the obstacle $\widehat{B}$ and connecting the vehicle center $\widehat{A}$. If the intersection between the relative velocity and the Collision Cone is different from zero, then a collision will happen. In-

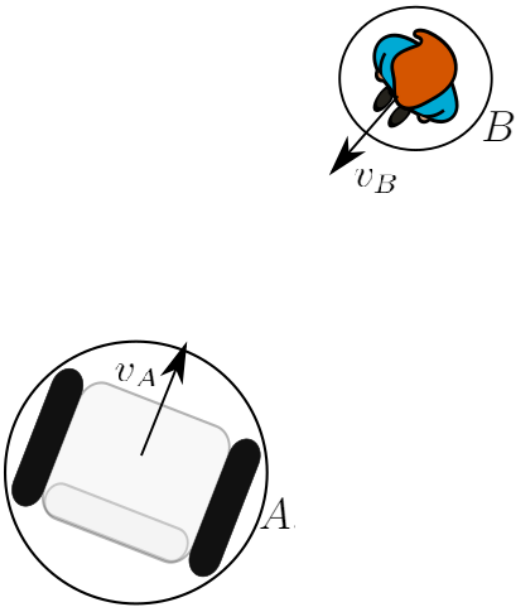



*Figure 4.3: Vehicle and pedestrian representation*

stead, any relative velocity outside the Collision Cone is then guaranteed to be collision-free, since the analysis is restricted to the case of fixed obstacles or moving at constant velocity (within the controller sampling time interval), that do not change in shape. Clearly, the collision cone is specific to a particular robot/obstacle pair.

## 4.2 Virtual Box

Having introduced the notion of Velocity Obstacle, it is then possible to delineate the concept of Virtual Box. In fact, the obstacles detected by the sensors are inscribed in circular Virtual Boxes (which can be described as obstacle containers), in order to ensure the admissibility of the optimization problem, which is related to the convexity of the state space. As already mentioned, the perception of the surrounding environment, and thus of the obstacles, is accomplished by two ToF laser sensors mounted on the wheelchair. Time of Flight (ToF) is a highly accurate distance mapping technology. It measures the distance between the sensor and an eventual object, by detecting the time difference between the emission of a signal and its return to the sensor, after being reflected

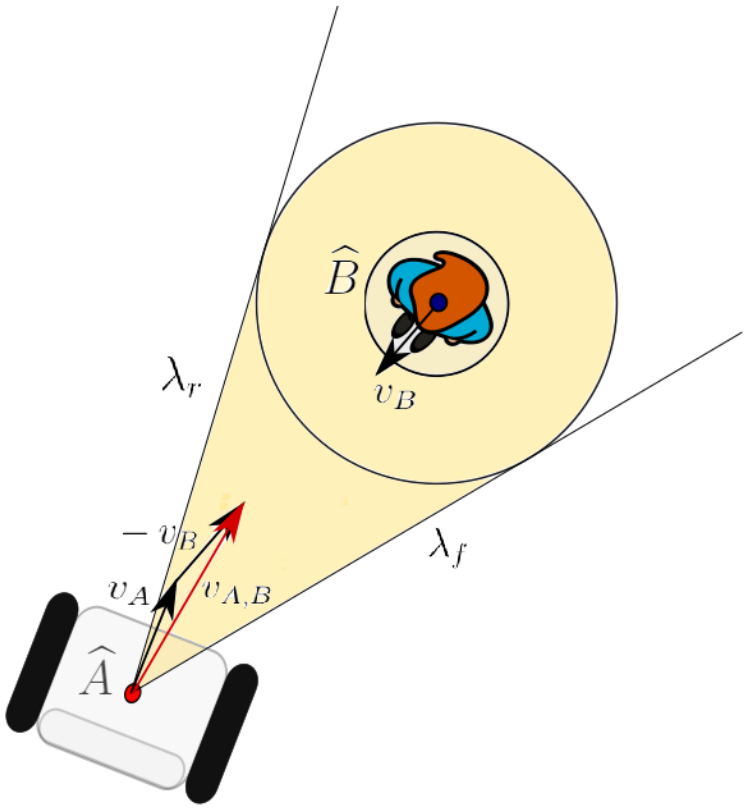



*Figure 4.4: The Collision Cone, in yellow, is the set of relative velocities that may lead to collision, and is defined as the planar sector with apex in $\widehat{A}$, bounded by the two tangents $\lambda_f$ and $\lambda_r$ from $\widehat{A}$ to $\widehat{B}$.*

by the object. The result is a dense set of measurements, consisting in angular positions associated to the respective detected distances.
First of all, it is necessary to transform the raw measurement retrieved from the sensors into data suitable for the control purposes. An example is the case of [8], transforming scans into PointCloud data in the global reference system. These data represent the view from the wheelchair perspective, detecting the nearest obstacles according to a radial geometry, as shown in Figure 4.5.

The amount of data collected may need to be refined, for a precise trust region definition. It can be reasonably stated that, if two consecutive points of the data sequence are farther (in terms of euclidean distance) than a selected threshold, the two points can be considered as belonging to two distinct obstacles.
For each subset of points representing a detected obstacle, the approxi-

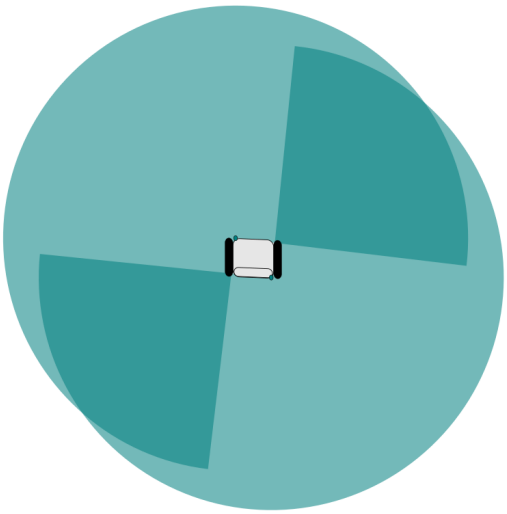
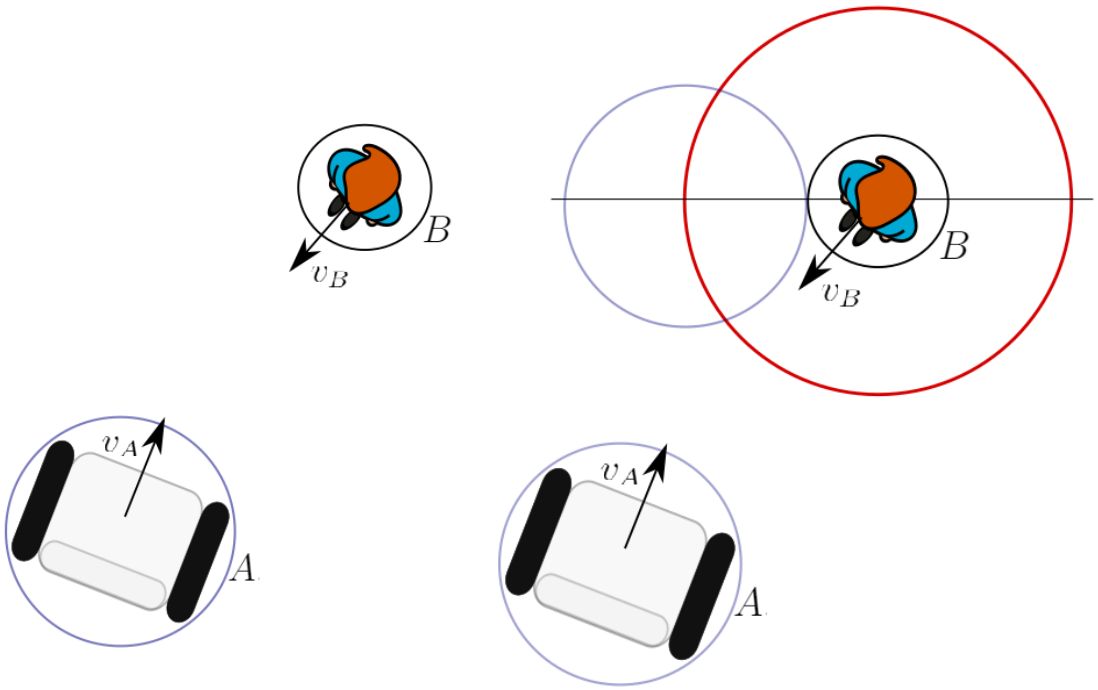


*Figure 4.5: Sensors field of view.*

mate center is calculated, as well as the radius necessary to inscribe all the subset points into a circumference (the obstacle virtual box). The position data detected at preceding time instants are used to compute the pedestrian velocity as the average velocity between each specific position at two following time instants. In literature, many algorithms allow people detection from laser range data, such as the ones reported in [24]. Therefore, it is possible to track subsequent pedestrian positions. Finally, to determine the inaccessible space surrounding the obstacles, each obstacle is inscribed in a circumference of suitable dimensions, in order to take into account the obstacle, i.e. the approaching person, as well as the vehicle dimensions. The wheelchair dimensions are taken into account by enlarging the obstacle virtual box of the circumference radius in which the vehicle can be inscribed, as shown in Figure 4.6, in order to map the obstacle into the Configuration Space of the vehicle.

To ensure the convexity of the state space, an appropriate linear position constraint is considered if the obstacle is dangerously approaching the vehicle. These constraints will be in the form of a linear inequality dividing the state space into two half-planes, only one of which defining the eligible region. The contemporary imposition of all constraints results in the definition of a convex admissible region.

Each constraint will be determined as the tangent to the circumference in the intersection point with the relative velocity vector and the circumference itself.




(a) (b)

*Figure 4.6: Obstacle virtual box enlarged by the circumference radius in which the wheelchair can be inscribed*

### 4.2.1 Walls and Fixed Obstacles

The Virtual Box model introduced in the previous sections can also be applied to the case of fixed obstacles characterizing the environment. However, the analysis may not be convenient for the large elements in the environment, e.g. walls. In that case, it is generally more advantageous to keep the geometry of the obstacle unchanged, by simply reconstructing its shape through the processing of sensor data, as in [8]. In this work, fixed obstacles detected by the sensors are included in the optimization problem after the delineation and simplification of their profile. Moreover, in order to allow the solvability of the optimization problem, each detected obstacle is segmented, if needed, into a set of convex-shaped obstacles. A further and more detailed analysis of this approach can be found in [8] and [5].

## 4.3 Trust region definition

The Velocity Obstacle concept was applied considering the obstacle Virtual Box, in order to determine the set of lines delimiting the convex obstacle-free region surrounding the wheelchair. As already introduced, any relative velocity lying between the two lines tangent to the enlarged obstacle $\widehat{B}$ and connecting the vehicle center $\widehat{A}$ will potentially cause a collision; to avoid that, the state space must be limited. The tangent lines are calculated as in Figure 4.7. To consider multiple obstacles, the same analysis is repeated for each obstacle.

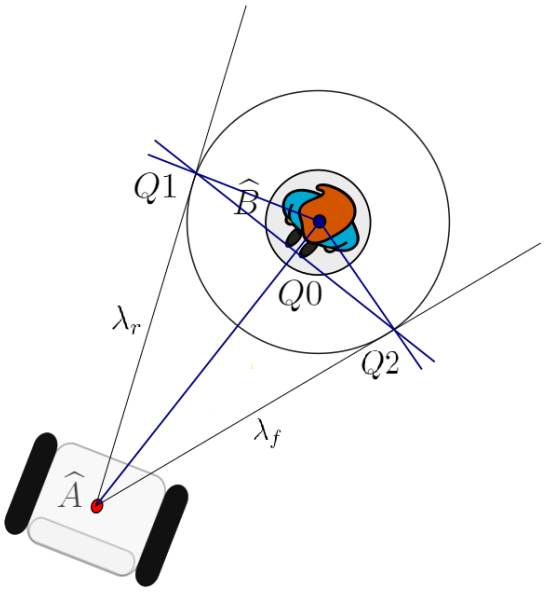



*Figure 4.7: The points of tangency to the virtual box circumference starting from the vehicle's center $\widehat{A}$, $Q1$ and $Q2$, can be retrieved exploiting similar triangles.*

To discriminate whether the relative velocity $v_{\vec{A,B}}$ between the vehicle and the pedestrian lies within the Collision Cone, the following variables are introduced:

$$\nu_1 = \begin{bmatrix} Q_1x - P_rx \\ Q_1y - P_ry \end{bmatrix} \tag{4.4}$$

$$\nu_2 = \begin{bmatrix} Q_2x - P_rx \\ Q_2y - P_ry \end{bmatrix} \tag{4.5}$$

$$\nu_R = \nu_r - \nu_{obs} = \begin{bmatrix} \nu_r x - \nu_{obs} x \\ \nu_r y - \nu_{obs} y \end{bmatrix} \tag{4.6}$$

Where $\nu_r$ and $\nu_{obs}$ are the vectors defining respectively the direction of the velocities of the robot and the obstacle, $P_r$ is the robot current position, while $Q_1$ and $Q_2$ are the tangent points to the virtual box, as shown in Figure 4.7. The vector $\nu_R$ can be written as a linear combination of the two vectors defining the Collision Cone:

$$\nu_R = \alpha_1 \nu_1 + \alpha_2 \nu_2 \tag{4.7}$$

where $\alpha_1$ and $\alpha_2$ are two constants.
With this notation, it is then possible to analyse the direction of the vector $\nu_R$ with respect to the Collision Cone by simply looking at the values of $\alpha_1$ and $\alpha_2$, in particular by looking at their signs. By rewriting with respect to the global reference axes:

$$\begin{bmatrix} \nu_{Rx} \\ \nu_{Ry} \end{bmatrix} = \begin{bmatrix} \nu_{1x} & \nu_{2x} \\ \nu_{1y} & \nu_{2y} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{4.8}$$

$$\alpha_{12} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} V_{12} \end{bmatrix}^{-1} \nu_R \tag{4.9}$$

it is possible to retrieve the values of $\alpha_1$ and $\alpha_2$. If both $\alpha_1$ and $\alpha_2$ are positive numbers, then the relative velocity lies within $\nu_1$ and $\nu_2$ and a collision may happen; it is thus necessary to introduce a hyperplane constraining the state variable.

The constraint will be determined as the tangent to the intersection point of the relative velocity and the circular virtual box enclosing the pedestrian, as shown in Figure 4.8. If two intersection points exist, the one closest to the vehicle will be considered, in order to impose the most meaningful and conservative constraint, that is the one in which the vehicle current position and the pedestrian one belong to different half planes. Considering the relative velocity line, in the form of $y = mx + c$, and the circular virtual box, in the form of $(x - p)^2 + (y - q)^2 = r^2$, with simple geometrical considerations, the intersection points can be retrieved. First, substituting $y = mx + c$ (or $x = k$ for the vertical lines case) into $(x - p)^2 + (y - q)^2 = r^2$ to obtain:

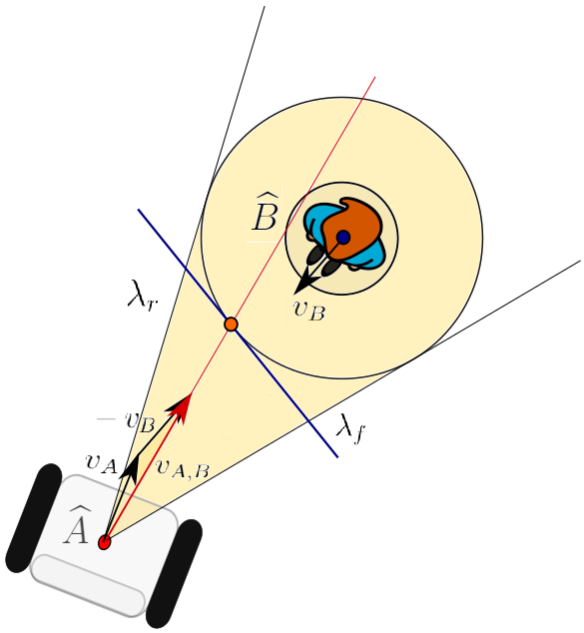



*Figure 4.8: Collision Cone implementation. The red circle represents the robot current position, while the blue circle represents the obstacle's current one. The bigger black circle represents the obstacle virtual box, i.e. the inaccessible space. The black lines are the tangents to the virtual box, highlighting the Collision Cone, while the red line represents the relative velocity ($\vec{\nu_R}$) between the robot and the obstacle. Since $\vec{\nu_R}$ lies in the Collision Cone, the tangent to the intersection point (represented by the orange circle) of the relative velocity and the circular virtual box is selected as constraint (blue line).*

$$(x-p)^2 + (mx+c-q)^2 = r^2 \tag{4.10}$$

Next, expanding out both brackets, and suitably collecting terms:

$$(m^2+1)x^2 + 2(mc-mq-p)x + (q^2-r^2+p^2-2cq+c^2) = 0 \tag{4.11}$$

Eq. 4.11 is quadratic in $x$ and can thus be solved using the quadratic formula, relabeling the coefficients to obtain the form $Ax^2+Bx+C=0$. Then the tangent is retrieved as the line passing through the intersection point with slope perpendicular to the line connecting the interception and the centre of the circumference $\widehat{B}$.

## 4.4 Socially compliant navigation

In the context of human crowded environments, where autonomous vehicles co-exist and cooperate with people, the ability to perceive and understand the human behavior is of crucial importance, in order to adjust the wheelchair trajectory accordingly. Socially compliant navigation in human environments emphasizes the need to respect human comfort while executing the selected task, i.e. reaching the final destination. It is not enough to guarantee a safe trajectory. In fact, a human observer may perceive a motion as not safe even if perfectly planned in order to avoid obstacles, since perceived proximity (influenced by human comfort) can be in contrast to actual proximity. Pedestrian comfort involves the safety, smoothness and naturalness of the trajectory, as well as the compliance with cultural conventions. In order to allow autonomous vehicles to navigate in human-crowded environments safely and taking into account human comfort, it is therefore necessary to model subtle human behaviours and navigation rules. Generally speaking, human social conventions are tendencies. While these result instinctive to humans, despite several dissimilitudes appear between different cultures, socially compliant navigation is still difficult to quantify and implement in an autonomous navigation algorithm, due to the stochasticity in people's behaviors. Many attempts in this field were made as in [9], [3] and [19], to cite some. In this work, a novel approach was implemented to enable socially compliant human-robot interaction that does not disturb nearby humans. This method analyzes the navigation behavior of interacting pedestrians, modeling their personal space (exploiting the concept of Proxemics as in [25] and [21]) in order to avoid it, unless the wheelchair is in over-crowded or critical conditions.

### 4.4.1 Proxemics

> *"Proxemics is the study of personal space and the degree of separation that individuals maintain between each other in social situations" [20]*

The concept of Proxemics was first introduced by Edward T. Hall in [21]. He described each person personal space as the psychological area surrounding them and the physical distances they try to keep from other people, according to subtle cultural rules [25]. These rules differ from

country to country, and depend also on personal experiences. Most people value their personal space and may find physical proximity to be uncomfortable. However, in the crowded urban environments that characterize our modern society, it can be difficult to preserve each person personal space. That is why the personal space (and the corresponding physical comfort zone) can be variable and difficult to measure.
Anyway, Hall in his study divided the interpersonal distances of man, as shown in Figure 4.9, in four distinct circular zones, each one related to a different kind of interaction:

1. Intimate space
2. Personal space
3. Social space
4. Public space

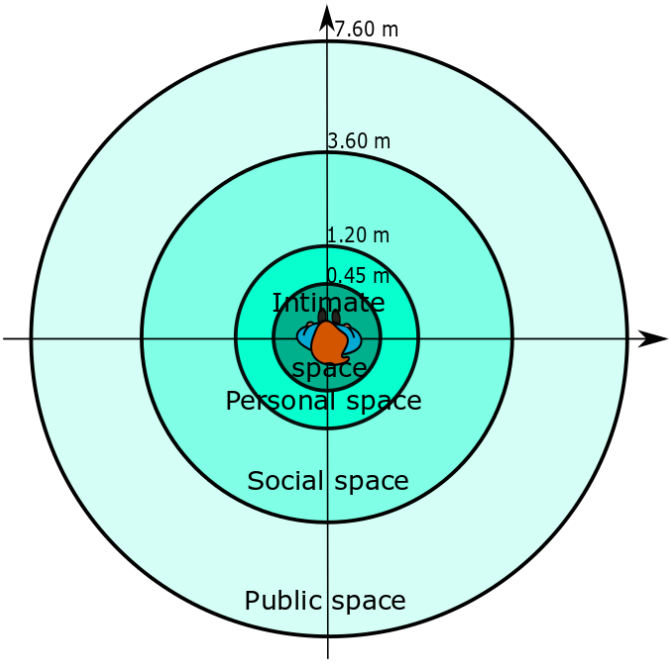



*Figure 4.9: Proxemics, The bubble of Personal Space [21]*

However, this result proved to be true only in static situations. More recent works extended Hall's personal space definition to include the effects of motion. The study [18] determined a correlation between the

subject's walking speed and the personal space dimensions. Moreover, this work, with the experiments [32] and [41] analyzing cases of non-interacting walking people, showed that the shape of personal space is not symmetrical from all sides. In fact, these three studies found that in motion, the distance from others maintained in front of a person is bigger than the distance kept on the lateral side.
To define the personal space shape of a moving person, in mathematical terms, Kirby in [25] modeled this area as two halves of 2D Gaussian functions. An interesting aspect about this study is that the size of the Gaussian functions is related to the walking speed.

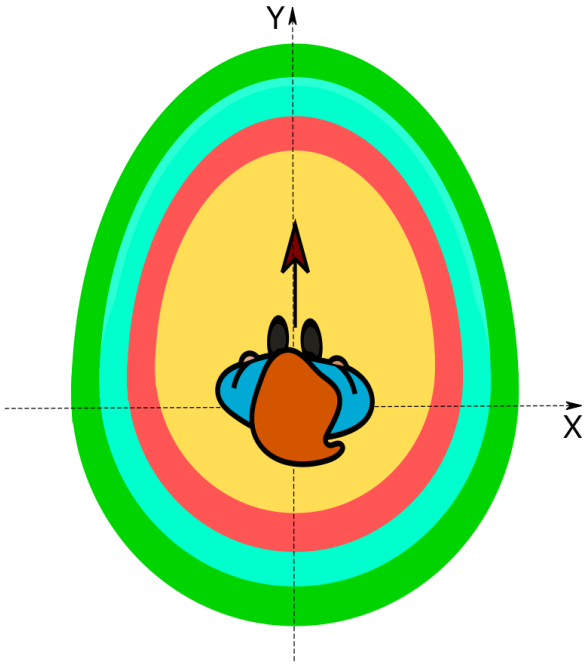



*Figure 4.10: Personal space cost function for a person moving along the positive Y-axis, with a relative velocity of 1 m/s ([25])*

More specifically, considering an autonomous vehicle and an approaching person, the size of the human personal space depends on the relative velocity between the person and the robot.
As it can be seen in Figure 4.10, the Personal Space function is composed by two Gaussian functions: below the X-axis, the function is a symmetric Gaussian with

$$\sigma_x = \sigma_y = 3v \tag{4.12}$$

where $v$ is the relative velocity between the person and robot; above the

X-axis, it is an elliptical function with

$$\sigma_y = 1.5\sigma_x \tag{4.13}$$

### 4.4.2 Trust region modification

The interacting pedestrians' personal space is thus modeled in mathematical terms as presented in the previous section. In this Chapter the obstacle-free space surrounding the wheelchair was represented as a convex obstacle-free region that takes into account a priori known obstacles, pedestrians, and other unforeseen obstacles detected in real-time. The hyperplanes, and thus the constraints, delimiting the trust region, take the form of:

$$h_x\ x + h_y\ y \leq l \tag{4.14}$$

where $h_x$ and $h_y$ are the coefficients related to the variables $x$ and $y$ identifying the slope of the 1-dimensional line, while $l$ is the constant term defining the distance from the origin.
The approach developed in this thesis exploits the Personal Space model in order to modify the constraints (in the form of 4.14) delimiting the trust region, with the aim to fulfill the human-comfort navigation requirements. In fact, a constraint will be imposed not only if an obstacle is dangerously approaching the vehicle, but also, in the case of a pedestrian, if his/her personal space may not be respected. The constraint can be rewritten as:

$$h_x\ x + h_y\ y \leq l - \Delta l \tag{4.15}$$

where the coefficient where $\Delta l = \Delta l_{sl}\sqrt{(h_x)^2 + (h_y)}$. The coefficient $\Delta l_{sl}$ represents the safety distance to be imposed in order to correctly include the interacting pedestrian personal space, measured according to the mathematical model presented in the previous section. The coefficient $\Delta l_{sl}$ is determined as shown in Figure 4.11 and summarized in Algorithm **??**, recalling that the Personal Space function is composed by two Gaussian functions.

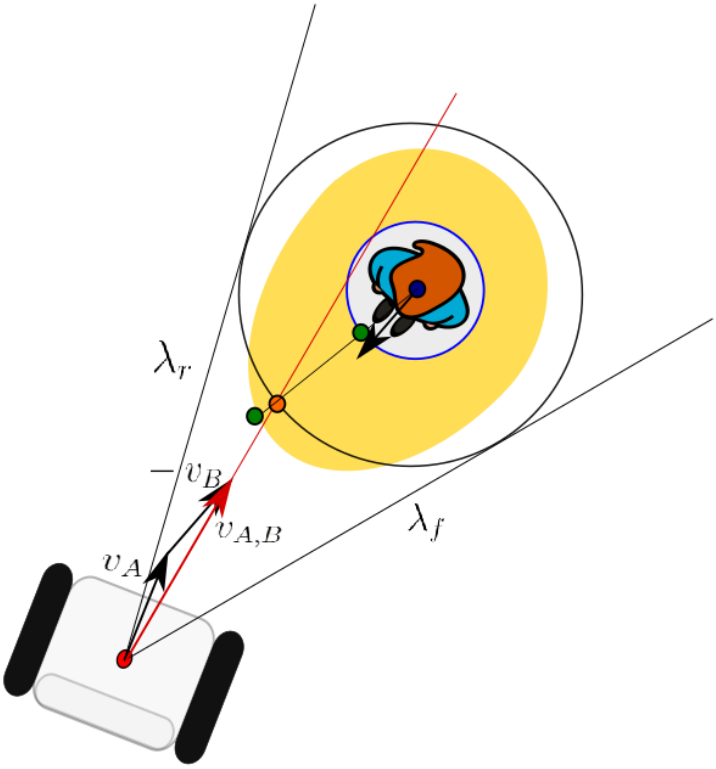



*Figure 4.11: The coefficient $\Delta l_{sl}$ is determined as the euclidean distance between the points highlighted by the two green circles, representing the intersections between the pedestrian sensor virtual box (blue circle) and the Pedestrian personal space function (in yellow), respectively, with the black line defined joining the pedestrian centre and the intersection between the relative velocity $v_{A,B}$ and the pedestrian virtual box enlarged to take into account the vehicle's dimensions.*

In Algorithm 2, $v_r$ is the relative velocity between the wheelchair and the pedestrian, $VB$ is the obstacle virtual box enlarged considering the vehicle dimensions, while $reducedVB$ is the sensor-detected virtual box; $P_p$ is the pedetrian current position, while $PSf$ is the pedestrian personal space function.

However, it is important to remark that humans make exceptions to and modify their space requirements. In fact, under circumstances where standard space requirements cannot be met, such as in crowded trains, elevators or streets, personal space requirements are modified accordingly. The constraints can then be modified to take this behavior into account, as:

**Algorithm 2** $\Delta l_{sl}$ determination

1: $P_1 \leftarrow GetIntersectionPoint(v_r, VB)$;
2: $l \leftarrow GetLineFrom2Points(P_1, P_p)$;
3: $P_2 \leftarrow GetIntersectionPoint(l, reducedVB)$;
4: $P_3 \leftarrow GetIntersectionPoint(l, PSf)$;
5: $\Delta l_{sl} \leftarrow EuclideanDistance(P_2, P_3)$;
6: **return** $\Delta l_{sl}$;

$$h_x\ x + h_y\ y \leq l - \Delta l(1 - u) \tag{4.16}$$

where $u$ is an optimization variable that weighs the compliance with human social conventions and the smoothness and executability of the wheelchair trajectory.

# Chapter 5

# Local Planning using Model Predictive Control

Model Predictive Control (MPC) is a family of algorithms that has had an enormous industrial impact in the last forty years. This advanced control method is particularly suitable for a large number of applications, thanks to its flexible constraint handling capabilities and the possibility to formulate the control problem as an optimization one, as highlighted in [28] and [17]. In fact, this method exploits an explicit dynamic model of the system in order to predict the effect of the manipulated variables on the output (thus the name "Model Predictive Control").

An MPC algorithm is based on: the process model, input, output, and state constraints, a cost function defined over a finite time horizon $[k, k+N]$, an optimization algorithm to compute the optimal control sequence and the Receding Horizon principle. The Receding Horizon (RH) principle was introduced in order to obtain a more robust, closed-loop like behaviour. At any time instant $k$ the optimization problem is solved with respect to the future control sequence $[u(k), ..., u(k+N-1)]$, by taking into account the actual system information in the time window $[k, k+N]$ and making sure that the predicted response has certain desirable characteristics. Then, only the first element $u(k)$ of the sequence is considered, as shown in Figure 5.1. At the following time instant $k+1$, a new optimization problem is solved, considering the time window $[k+1, k+N+1]$, by taking into account the updated system measurements. Thanks to the recursive nature of this algorithm, a time invariant feedback control strategy is implemented.

Regarding the control problem developed in this thesis, the MPC control

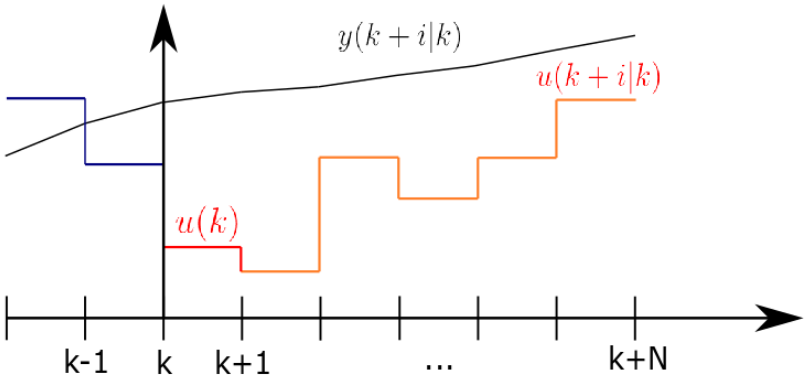



*Figure 5.1: At any time instant $k$ the optimization problem is solved with respect to the future control sequence $[u(k), ..., u(k+N-1)]$ and the predicted outputs $y(k+i|k)$ $\forall i \in \{0, ..., N\}$ within the Prediction Horizon $[k, k+N]$. Only the first element $u(k)$ of the sequence is actuated.*

method is particularly interesting, since it allows to specify constraints for obstacle avoidance and actuator limitations in the optimization problem, explicitly using a model of the system to predict its state at future time instants.

In the following section, general concepts about the MPC algorithms are introduced. The next sections will focus on its formalization with regard to the case of study of this thesis, the wheelchair.

## 5.1 MPC formulation

Considering a generic linear discrete-time system in the form:

$$\begin{cases} x(k+1) = A\ x(k) + B\ u(k) \\ y(k) = C\ x(k) \end{cases} \tag{5.1}$$

where $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$ are the state and the control variables, respectively. The state is assumed to be measurable. The optimization problem aims at computing, at each time instant $k$ the control variable sequence $\mathcal{U}(k) = [u(k), ..., u(k+N-1)]^T$ that minimizes the cost function $J$ defined over the selected prediction horizon $N$:

$$J = \sum_{i=0}^{N-1} l(x(k+i), u(k+i)) + V^f(x(k+N)) \tag{5.2}$$

where $l(x, u)$ is a suitable positive definite function that takes the name of stage cost, while $V^f(x(k+N))$ is defined as terminal cost.

The main goals of the Model Predictive Control include the fulfillment of the requirements on the system variables, the optimal convergence of the state variables of interest to the desired reference value, and the avoidance of excessive fluctuations in the control variables.
For the case studied in this work, these requirements can be formalized by expressing $J$ as a finite horizon quadratic cost function:

$$J(x(k), u(k)) = \sum_{i=0}^{N-1} (||x(k+i)||_Q^2 + ||u(k+i)||_R^2) + ||x(k+N)||_S^2 \tag{5.3}$$

where $Q = Q' \geq 0$, $R = R' > 0$ and $S = S' \geq 0$ are weight matrices of suitable dimension, and the expression

$$||x||_Z^2 = x^T Z\ x \tag{5.4}$$

As already mentioned, in the Model Predictive Control method it is possible to directly include constraints on the state and control variables, in the form:

$$x(k) \in \mathbb{X} \subset \mathbb{R}^n \tag{5.5}$$

$$u(k) \in \mathbb{U} \subset \mathbb{R}^m \tag{5.6}$$

The constraints must be expressed in a way which is compatible with the optimization algorithm, as it will be seen in the following sections.

As described in [28], in the context of the MPC, the idea of stability is related to the Recursive Feasibility principle. First, it is important to introduce the concept of terminal set. Given a state variable $x(k)$ such that:

$$x(k) \in \mathbb{X} \subset \mathbb{R}^n \tag{5.7}$$

where $\mathbb{X}$ is defined as trust region of the state, the terminal set can be defined as:

$$\mathbb{X}^f \subset \mathbb{X} \tag{5.8}$$

In order to ensure the solvability of the optimization problem, it is required that the final state, at the end of the prediction horizon, belongs to this particular subset of the trust region of the state:

$$x(k+N) \in \mathbb{X}^f \tag{5.9}$$

The terminal set is designed with reference to the auxiliary time-varying optimal control law:

$$u^o(k+i) = -K(i)x(k+i) \tag{5.10}$$

$$\forall i \in \{0, ..., N-1\}$$

Considering the generic linear discrete-time system (5.1), the values of $K$ of the time-varying optimal control law (5.10) must be chosen in order to ensure that the eigenvalues of the matrix $A-BK$ guarantee the stability of the closed-loop system

$$x(k+1) = (A - BK)\, x(k) \tag{5.11}$$

Then, by taking into account the Discrete Riccati Equation:

$$(A-BK)^T S(A-BK) - S = -(Q + K^T R K) \tag{5.12}$$

where penalty matrices Q and R are the ones introduced for the MPC problem, the positive definite matrix S can be determined.

The terminal set $\mathbb{X}^f$ is said to be positive invariant with respect to the closed loop system (5.11) if

$$x(\bar{k}) \in \mathbb{X}^f \Rightarrow x(k) \in \mathbb{X}^f, \quad \forall k \geq \bar{k} \tag{5.13}$$

Finally, it is required that

$$u(k) = Kx(k) \subseteq \mathbb{U} \quad \forall x(k) \in \mathbb{X}^f \tag{5.14}$$

In this way, starting from and initial state $x(\bar{k}) \in \mathbb{X}^f$ and eventually applying the auxiliary control law, the state will belong to the terminal set $\mathbb{X}^f$ and guarantee the control requirements.

## 5.2 The autonomous wheelchair model

The approach proposed in this thesis can be applied to many different autonomous vehicle. However, in order to make reference to a real case, the electric wheelchair already considered in [5],[8] and [12], will be taken into account.
The Degonda Twist t4 2x2 (shown in Figure 5.2) is a wheelchair for indoor and outdoor use produced by Degonda Rehab SA. This wheelchair is equipped with two rear driving wheels, whose motors are characterized by a maximum power of $0.35[kW]$, and three caster wheels with stabilization function. It is also endowed with two SICK TiM561 time-of-flight laser sensors able to identify the relative distance of the surrounding objects. The two sensors are positioned opposite to each other so as to guarantee a complete 360° vision of the surrounding environment within $10[m]$ from the wheelchair.

*Figure 5.2: Degonda Twist t4 2x2 wheelchair.*

From a kinematic point of view, the motion of the considered autonomous robot can be represented using a differential-drive vehicle model. A differential-drive vehicle (illustrated in Figure 5.3) generally has two separately controlled fixed wheels with a common axis of rotation, and

one or more passive caster wheels to keep the robot statically balanced. The two fixed wheels are actuated by independent motors, having a rotational velocity $\omega_R$ and $\omega_L$ and linear velocity $v_R = \omega_R R$ and $v_L = \omega_L R$, where $R$ is the wheels radius.

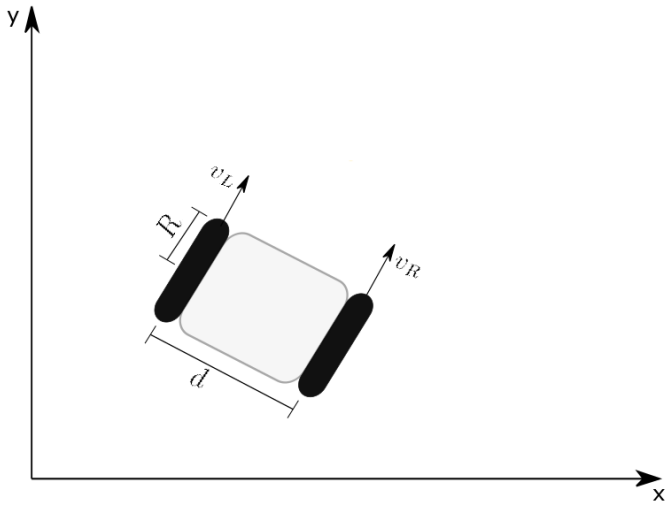



*Figure 5.3: Differential drive vehicle*

The vehicle control variables are its linear velocity $v$ and angular velocity $\omega$. By simple kinematic considerations, we can relate them to the angular speed of the two separately controlled wheels. In fact, the linear velocity $v$ is the average value between the two wheel velocities

$$v = R\frac{\omega_R + \omega_L}{2} \tag{5.15}$$

while the rate of rotation $\omega$ of the vehicle around the Instantaneous Center of Curvature, is generated by a difference in the velocity of the two wheels

$$\omega = R\frac{\omega_R - \omega_L}{d} \tag{5.16}$$

From a practical point of view a differential drive vehicle is thus kinematically equivalent to a unicycle (depicted in Figure 5.4), and can be represented using the unicycle nonlinear model, i.e.,

$$\begin{cases} \dot{x}(t) = v(t)cos\theta(t) \\ \dot{y}(t) = v(t)sin\theta(t) \\ \dot{\theta}(t) = \omega(t) \end{cases} \tag{5.17}$$

where $x(t)$, $y(t)$, $\theta(t)$ are the state variables representing its center position and orientation in the global reference system, and the longitudinal and angular velocities ($v(t)$ and $\omega(t)$ respectively) are the input variables.

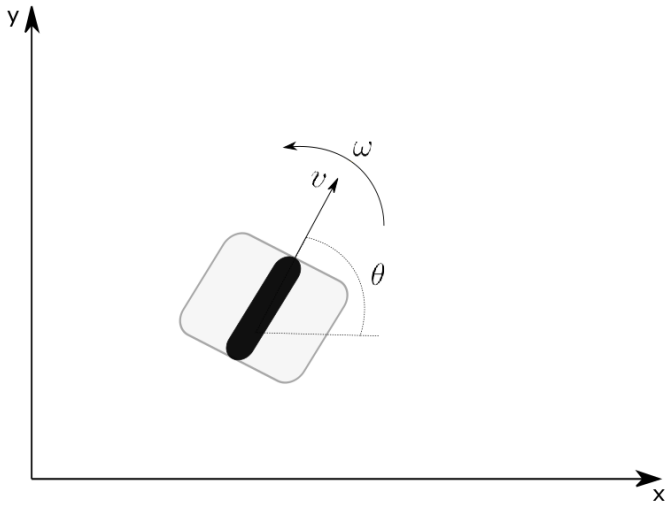



*Figure 5.4: Unicycle*

The model is characterized by a non-linear relationship between the state variables ($x(t)$, $y(t)$, $\theta(t)$) and the control variables ($v(t)$, $\omega(t)$) and this is not compatible with the development of a linear controller. However, thanks to an inner feedback-linearizing loop [29], the system can be approximated as a particle, displaying a linear dynamics under a suitable change of variables. The formulation of the trajectory tracking problem is thus simplified. To this aim, with reference to [5], [8] and [12], a point $P$ placed at distance $\varepsilon$ from the unicycle wheel axle center in the direction of the longitudinal velocity (see Figure 5.5) is defined, whose coordinates with respect to the global reference frame are:

$$\begin{cases} x_P(t) = x(t) + \varepsilon \cos\theta(t) \\ y_P(t) = y(t) + \varepsilon \sin\theta(t) \end{cases} \tag{5.18}$$

By deriving with respect to time:

$$\begin{cases} \dot{x}_P(t) = \dot{x}(t) + \varepsilon \sin\theta(t)\dot{\theta}(t) \\ \dot{y}_P(t) = \dot{y}(t) + \varepsilon \cos\theta(t)\dot{\theta}(t) \end{cases} \tag{5.19}$$

and recalling that $\dot{x}(t) = v(t)\cos\theta(t)$, $\dot{y}(t) = v(t)\sin\theta(t)$ and $\omega(t) = \dot{\theta}(t)$,

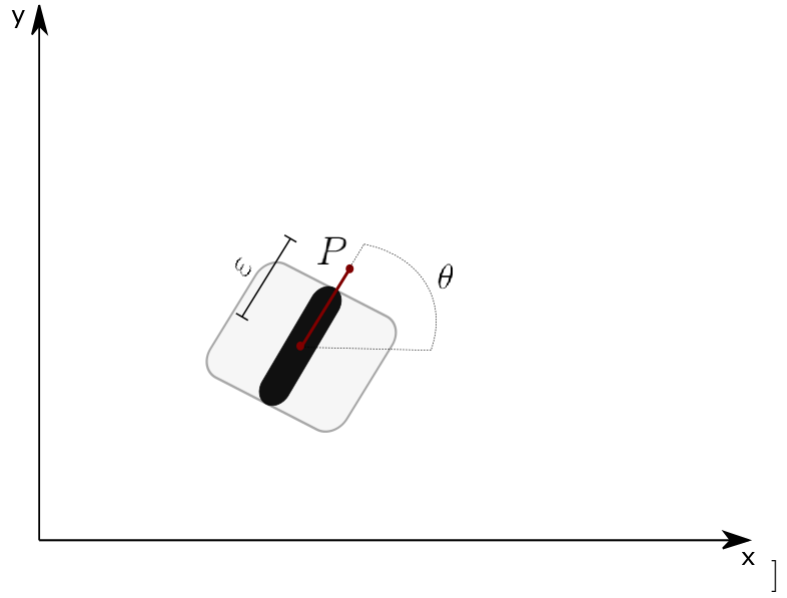



*Figure 5.5: Feedback Linearization unicycle model*

it is possible to rewrite Eq. 5.19 in matrix form:

$$\begin{bmatrix} \dot{x}_P(t) \\ \dot{y}_P(t) \end{bmatrix} = \begin{bmatrix} v_{Px}(t) \\ v_{Py}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & -\varepsilon\sin\theta(t) \\ \sin\theta(t) & \varepsilon\cos\theta(t) \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{5.20}$$

From which it is possible to retrieve the Feedback Linearization transformation matrix:

$$T(\theta,\varepsilon) = \begin{bmatrix} \cos\theta(t) & -\varepsilon\sin\theta(t) \\ \sin\theta(t) & \varepsilon\cos\theta(t) \end{bmatrix} \tag{5.21}$$

Matrix $T(\theta,\varepsilon)$ is non-singular, and thus invertible, $\forall\theta$ and $\forall\varepsilon \neq 0$. It is then possible to relate the variables of the linearized system ($v_{Px}(t)$, $v_{Py}(t)$) and the real one ($v(t)$, $\omega(t)$).

The Feedback Linearization procedure starts from a 3D configuration space (the vehicle pose), and ends with a reduced configuration space (the vehicle position). In fact, the change of coordinates induces a loss in the observability of the system, transforming it from a third order system into a second order one. Therefore, the heading is not observable anymore from the output. For this reason, the trajectory tracking consists in a position control. If also a goal on orientation is required, i.e. if the desired pose differs from the one tangent to the trajectory, a

further orientation control that cannot exploit the Feeback Linearization law needs to be introduced.

The final model obtained from the Feedback linearization procedure is then described by a linear decoupled system, characterized by two integrators:

$$\begin{cases} \dot{x}_P(t) = v_{Px}(t) \\ \dot{y}_P(t) = v_{Py}(t) \end{cases} \tag{5.22}$$

Eq. 5.22 is a continuous-time model. In order to make it compatible with the discrete nature of the control approach used, a further transformation is thus needed. The Forward Euler method is then introduced. This discretization approach is based on a truncated Taylor series expansion, so that imposing:

$$s = \frac{z-1}{\tau} \tag{5.23}$$

where $\tau$ is the sampling time set for the controller, it is then possible to obtain the discrete model of the system:

$$\begin{cases} x_P(k+1) = x_P(k) + \tau \ v_{Px}(k) \\ y_P(k+1) = y_P(k) + \tau \ v_{Py}(k) \end{cases} \tag{5.24}$$

which can be rewritten in compact form:

$$\xi(k+1) = A \ \xi(k) + B \ u(k) \tag{5.25}$$

where:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix} \quad \xi(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \quad u(k) = \begin{bmatrix} v_{Px}(k) \\ v_{Py}(k) \end{bmatrix} \tag{5.26}$$

## 5.3 Cost function

As already introduced, the cost function is designed to fulfill the requirements on the system variables, to guarantee the optimal convergence of the state variables of interest to the desired reference value, and to avoid excessive fluctuations in the control variables. In this section, the finite horizon quadratic cost function is determined specifically for the wheelchair under consideration, in order to perform the tracking of the trajectory calculated by the Global Planner while meeting the control requirements:

$$J(k) = \sum_{i=0}^{N-1} (\|\xi(k+i) - \xi_{ref}(k+i)\|_Q^2 + \|u(k+i)\|_R^2) + \\ + \|\xi(k+N) - \xi_{ref}(k+N)\|_S^2 \tag{5.27}$$

where $\xi(k+i)$ is the ith-step ahead state prediction computed based on the current state $\xi(k)$, on the input sequence $u(k), ..., u(k+i-1)$ and on the previously introduced wheelchair model:

$$\xi(k+1) = A\ \xi(k) + B\ u(k) \tag{5.28}$$

where:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix} \quad \xi(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \quad u(k) = \begin{bmatrix} v_{Px}(k) \\ v_{Py}(k) \end{bmatrix} \tag{5.29}$$

The weight matrices $Q$, $R$ and $S$, must be tuned to ensure the fulfillment of the requirements on the state and control variables, as well as the system stability. This will be done in Chapter 6. The position reference $\xi_{ref}(k+i)$ is the trajectory planned by the Global Planner at the considered time instant

$$\xi_{ref}(k+i) = \begin{bmatrix} x_{ref}(k+i) \\ y_{ref}(k+i) \end{bmatrix} \tag{5.30}$$

In order to allow the optimization solver to find a solution to the optimization problem, the cost function is formalized as an Open-Loop Solution, derived by computing the prediction of the future states based on the value of the current state $\xi(k)$. In fact, in view of the Lagrange equation:

$$\xi(k+i) = A^i \xi(k) + \sum_{j=0}^{i-1} A^{i-j-1} B u(k+j) \quad , \quad i > 0 \tag{5.31}$$

Letting:

$$\mathcal{A} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}_{(2(N+1),2)} \qquad \mathcal{B} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}_{(2(N+1),2N)} \tag{5.32}$$

$$\Xi(k) = \begin{bmatrix} \xi(k) \\ \vdots \\ \xi(k+N) \end{bmatrix}_{(2(N+1),1)} \qquad \mathcal{U}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix}_{(2N,1)} \tag{5.33}$$

it follows that:

$$\Xi(k) = \mathcal{A}\ \xi(k) + \mathcal{B}\ \mathcal{U}(k) \tag{5.34}$$

The cost function $J(\xi(k), u(k), k)$ can then be equivalently rewritten as:

$$\begin{aligned} \tilde{J}(\xi(k), \mathcal{U}(k)) = ||\Xi(k) - \Xi_{ref}||^2_{\mathcal{Q}}\ +\ ||\mathcal{U}(k)||^2_{\mathcal{R}} = \\ = (\Xi(k) - \Xi_{ref})^T\ \mathcal{Q}\ (\Xi(k) - \Xi_{ref})\ +\ \mathcal{U}^T(k)\ \mathcal{R}\ \mathcal{U}(k) \end{aligned} \tag{5.35}$$

where the matrices $\mathcal{Q}$ and $\mathcal{R}$ are defined as:

$$\mathcal{Q} = \begin{bmatrix} Q & & & \\ & \ddots & & \\ & & Q & \\ & & & S \end{bmatrix}_{(2(N+1),2(N+1))} \qquad \mathcal{R} = \begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix}_{(2N,2N)} \tag{5.36}$$

By recalling Equation (5.32):

$$\begin{aligned} \tilde{J}(k) = (\mathcal{A}\ \xi(k) + \mathcal{B}\ \mathcal{U}(k) - \Xi_{ref})^T\ \mathcal{Q}\ (\mathcal{A}\ \xi(k) + \mathcal{B}\ \mathcal{U}(k) - \Xi_{ref}) + \\ +\ \mathcal{U}^T(k)\ \mathcal{R}\ \mathcal{U}(k) \end{aligned} \tag{5.37}$$

and rearranging:

$$\tilde{J} = \mathcal{U}^T(k) H \mathcal{U}(k) + 2 f^T \mathcal{U}(k) + cost \tag{5.38}$$

a cost function formalization suitable to solve the optimization problem is obtained, where $H = \mathcal{B}^T \mathcal{Q} \mathcal{B} + \mathcal{R}$ is called Hessian matrix and $f = (\mathcal{A}\xi(k) - \Xi_{ref})^T \mathcal{Q}\mathcal{B}$ is the gradient vector, while *cost* represents all the terms not depending from $\mathcal{U}(k)$.

## 5.4 Control variable constraints

The main advantage in the formulation of the Model Predictive Control is based on the fact that it is possible to directly include the constraints related to the state and the control variables. This allows to take into account the intrinsic limitations of the system, i.e. actuator limitations, comfort requirements and the presence of obstacles. In this section, the constraints related to the control variables will be discussed.
To take into account actuator limitations and comfort requirements, maximum velocity constraints are enforced on the vehicle longitudinal speed $v$, as well as constraints on the velocity variation.

### 5.4.1 Velocity constraints

From the Feedback Linearization procedure, and in particular from Eq. 5.19, it is possible to relate the linearized system velocity $v_P$ and the real system one $v$ as

$$v_P = \sqrt{v_{Px}^2 + v_{Py}^2} = \sqrt{v^2 + \varepsilon^2 \omega^2} \tag{5.39}$$

Therefore, by enforcing an upper bound to the speed $v_P$ of point $P$, it is possible to impose constraints on the vehicle velocity $v$ and thus on the velocities of the right and left wheels of the vehicle.
In fact, from the Differential Drive vehicle model, in Eq. 5.15 and Eq. 5.16, it follows that

$$|v| = \frac{|v_R + v_L|}{2} \leq v_{max} \tag{5.40}$$

$$\varepsilon\omega = \frac{|v_R - v_L|\varepsilon}{d} \leq v_{max} \tag{5.41}$$

it can be observed that, by imposing a maximum over $v$, $|v_R|$ and $|v_L|$ are guaranteed to remain constrained, as required.
Hence, $\forall i \in \{0, ..., N-1\}$ it is necessary that:

$$0 \leq \sqrt{v_{Px}^2(k+i) + v_{Py}^2(k+i)} \leq v_{max} \tag{5.42}$$

for a suitable value of $v_{max}$ depending on the vehicle limitations.

Since the constraint in Eq. 5.42 is nonlinear, the optimization problem to be solved at the discrete-time instant becomes a quadratically constrained quadratic programming problem. Differently from [8] and [5], this constraint will not be linearized with a first-order truncated Taylor series. In fact, as it will be proven in simulations, the choice of implementing the non-linear constraint will provide a more accurate solution without affecting significantly the solver performances.

In quadratic programming, the quadratic constraints on the control variables can be expressed as:

$$\mathcal{L}\,\mathcal{U}(k) + \mathcal{U}(k)^T \mathcal{Q}\,\mathcal{U}(k) \leq \mathcal{R} \tag{5.43}$$

where

$$\mathcal{U}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix}_{(2N,1)} \qquad u(k+i) = \begin{bmatrix} v_{Px}(k+i) \\ v_{Py}(k+i) \end{bmatrix} \tag{5.44}$$

and $\mathcal{L}$ is the $(2, N)$ dimensional matrix reporting the linear parts of the $N$ quadratic constraints; $\mathcal{Q}$ is the $(1, N)$ dimensional matrix, containing $N$ $(N, N)$ dimensional matrices, related to the quadratic part of the constraints, while $\mathcal{R}$ is the $(1, N)$ dimensional row vector for the scalar inequality terms of the constraints.
This formulation is thus equivalent to $N$ quadratic constraints of the kind:

$$\mathcal{L}_i\; u(k+i) + u(k+i)^T\; \mathcal{Q}_i\; u(k+i) \leq \mathcal{R}_i \tag{5.45}$$

$$\forall i \in \{0, ..., N-1\}$$

where $\mathcal{L}_i$ is the $(1,2)$ dimensional row vector reporting the linear part of the $i-th$ quadratic constraint, $\mathcal{Q}_i$ is the $(2,2)$ dimensional matrix related to the quadratic part of the constraint, while $\mathcal{R}_i$ is the scalar term of the quadratic inequality constraint.
In this particular case, the conditions imposed on the linearized system control variables $\forall i \in \{0, ..., N-1\}$ are then expressed as:

$$\begin{bmatrix} v_{Px}(k+i) & v_{Py}(k+i) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{Px}(k+i) \\ v_{Py}(k+i) \end{bmatrix} \leq v_{max}^2 \tag{5.46}$$

since $\forall i \in \{0, ..., N-1\}$:

$$\mathcal{L}_i = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad \mathcal{Q}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathcal{R}_i = \begin{bmatrix} v_{max}^2 \end{bmatrix} \tag{5.47}$$

These non-linear constraints proved in simulation to be effective. However, another formulation of the problem allowed to write an equivalent set of linear constraints limiting the vehicle velocity, suitable to be included in a quadratic and numerically simpler optimization problem. The implementation of these constraints proved to be equivalently effective in simulation.

Exploiting the relationships obtained through the Feedback Linearization procedure:

$$\begin{cases} v = v_{Px}\ cos\theta + v_{Py}\ sin\theta \\ \omega = \frac{1}{\varepsilon}(v_{Py}\ cos\theta - v_{Px}\ sin\theta) \end{cases} \tag{5.48}$$

As well as the Differential Drive kinematic relationship:

$$v = \frac{\omega_R + \omega_L}{2} R \qquad \omega = \frac{\omega_R - \omega_L}{d} R \tag{5.49}$$

$$\omega_R = \frac{2v + d\omega}{2R} \qquad \omega_L = \frac{2v - d\omega}{2R} \tag{5.50}$$

The following relationships can be obtained by substitution:

$$\omega_R = \frac{1}{2R}(2cos\theta - \frac{d}{\varepsilon}sin\theta)v_{Px} + \frac{1}{2R}(2sin\theta + \frac{d}{\varepsilon}cos\theta)v_{Py} \tag{5.51}$$

$$\omega_L = \frac{1}{2R}(2cos\theta + \frac{d}{\varepsilon}sin\theta)v_{Px} + \frac{1}{2R}(2sin\theta - \frac{d}{\varepsilon}cos\theta)v_{Py} \tag{5.52}$$

It is therefore possible to impose constraints on the linearized system control variables $v_{Px}$ and $v_{Py}$ in the form of:

$$\bar{\omega}_m \leq \omega_R \leq \bar{\omega}_M \qquad \bar{\omega}_m \leq \omega_L \leq \bar{\omega}_M \tag{5.53}$$

where $\bar{\omega}_m$ and $\bar{\omega}_M$ are the minimum and maximum limits, respectively, of the wheelchair angular velocity.
In fact, assuming $\theta$ (the state variable representing the wheelchair orientation) to be known, the contraints 5.53 can be written as:

$$\frac{1}{2R}\begin{bmatrix} 2cos\theta - \frac{d}{\varepsilon}sin\theta & 2sin\theta + \frac{d}{\varepsilon}cos\theta \\ 2cos\theta + \frac{d}{\varepsilon}sin\theta & 2sin\theta - \frac{d}{\varepsilon}cos\theta \end{bmatrix}\begin{bmatrix} v_{Px} \\ v_{Py} \end{bmatrix} \leq \begin{bmatrix} \bar{\omega}_M \\ \bar{\omega}_M \end{bmatrix} \tag{5.54}$$

$$-\frac{1}{2R}\begin{bmatrix} 2cos\theta - \frac{d}{\varepsilon}sin\theta & 2sin\theta + \frac{d}{\varepsilon}cos\theta \\ 2cos\theta + \frac{d}{\varepsilon}sin\theta & 2sin\theta - \frac{d}{\varepsilon}cos\theta \end{bmatrix}\begin{bmatrix} v_{Px} \\ v_{Py} \end{bmatrix} \leq -\begin{bmatrix} \bar{\omega}_m \\ \bar{\omega}_m \end{bmatrix} \tag{5.55}$$

or, in matrix form as:

$$\begin{bmatrix} \bar{A} \\ -\bar{A} \end{bmatrix}_{(4,2)} \begin{bmatrix} v_{Px} \\ v_{Py} \end{bmatrix}_{(2,1)} \leq \bar{b}_{(4,1)} \tag{5.56}$$

where

$$\bar{A} = \frac{1}{2R}\begin{bmatrix} 2cos\theta - \frac{d}{\varepsilon}sin\theta & 2sin\theta + \frac{d}{\varepsilon}cos\theta \\ 2cos\theta + \frac{d}{\varepsilon}sin\theta & 2sin\theta - \frac{d}{\varepsilon}cos\theta \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} \bar{\omega}_M \\ \bar{\omega}_M \\ -\bar{\omega}_m \\ -\bar{\omega}_m \end{bmatrix} \tag{5.57}$$

The linear velocity constraints along the entire prediction horizon ($\forall i \in \{0, ..., N-1\}$) can then be written in the form of:

$$\bar{A}_{vel}\, \mathcal{U}(k) \leq \bar{b}_{vel} \tag{5.58}$$

with

$$\bar{A}_{vel} = \begin{bmatrix} \bar{A}_1 & 0 & \cdots & 0 \\ 0 & \bar{A}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{A}_N \\ -\bar{A}_1 & 0 & \cdots & 0 \\ 0 & -\bar{A}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\bar{A}_N \end{bmatrix}_{(4N,2N)} \qquad \bar{b}_{vel} = \begin{bmatrix} \bar{\omega}_M \\ \vdots \\ \vdots \\ \bar{\omega}_M \\ -\bar{\omega}_m \\ \vdots \\ \vdots \\ -\bar{\omega}_m \end{bmatrix}_{(4N,1)} \tag{5.59}$$

The values of $\theta(i)$ can be reasonably estimated considering the values of the control sequence determined at the previous time instant, $\mathcal{U}(k-1)$.

### 5.4.2 Velocity variation constraints

In order to make the Model Predictive Control solution consistent with the system dynamics (with reference to the maximum allowed acceleration), and to satisfy passenger comfort and trajectory smoothness needs, it is important to limit the velocity variation between two consecutive time instants. To take into account the discrete-time nature of the Model Predictive Control, the velocity variation at time instant $k$ can be expressed in as:

$$\Delta v(k) = v(k) - v(k-1) \tag{5.60}$$

The constraint can be then imposed by limiting the velocity variation between two following time instants:

$$-\Delta v_{max} \leq \Delta v(k) \leq \Delta v_{max} \tag{5.61}$$

The threshold value $\Delta v_{max}$ can be evaluated taking into account the vehicle limitations. In fact, the maximum velocity variation can be retrieved by considering the product between the controller sampling time $\tau$ and maximum value of the average acceleration $\bar{a}_{max}$:

$$\Delta v_{max} = \bar{a}_{max}\ \tau \tag{5.62}$$

In this particular case, the constraints at time instant $k+i$ can be imposed directly on the decoupled control variables $v_{Px}(k+i), v_{Py}(k+i)$:

$$-\Delta v_{max} \leq v_{Px}(k+i) - v_{Px}(k+i-1) \leq \Delta v_{max} \tag{5.63}$$

$$-\Delta v_{max} \leq v_{Py}(k+i) - v_{Py}(k+i-1) \leq \Delta v_{max} \tag{5.64}$$

By considering the velocity variation constraints along the entire prediction horizon ($\forall i \in \{0, ..., N-1\}$), the following condition is thus obtained:

$$A_{var}(V\mathcal{U}(k) - v_0) \leq \Delta V \tag{5.65}$$

with:

$$A_{var} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix}_{(4N,2N)} \qquad \Delta V = \begin{bmatrix} \Delta v_{max} \\ \vdots \\ \Delta v_{max} \end{bmatrix}_{(4N,1)}$$

$$v_0 = \begin{bmatrix} v_{Px}(k-1) \\ v_{Py}(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{(2N,1)} \qquad V = \begin{bmatrix} I_{(2)} & & \\ -I_{(2)} & \ddots & \\ & \ddots & \ddots \\ & & -I_{(2)} \quad I_{(2)} \end{bmatrix}_{(2N,2N)} \tag{5.66}$$

where $I_{(2)}$ is the identity matrix of order 2.

Recalling that, in quadratic programming, linear constraints along the entire prediction horizon, can be expressed in the form:

$$\begin{cases} A_{ineq_u}\,\mathcal{U}(k) \leq b_{ineq_u} \\ A_{eq_u}\,\mathcal{U}(k) = b_{eq_u} \\ \mathcal{U}_{min} \leq \mathcal{U}(k) \leq \mathcal{U}_{max} \end{cases} \tag{5.67}$$

where matrices $A_{ineq}$ and $A_{eq}$ highlight the number of conditions to be imposed, $b_{ineq}$ and $b_{eq}$ are vectors of constant terms, while $\mathcal{U}_{min}$ and $\mathcal{U}_{max}$ are the lower and the upper bounds respectively for every instant of the chosen prediction horizon (sees more in [28]).
The condition to be imposed can thus be expressed as:

$$A_{\Delta v}\,\mathcal{U}(k) \leq b_{\Delta v} \tag{5.68}$$

By rearranging Eq. 5.65, matrices $A_{\Delta v}$ and $b_{\Delta v}$ can be obtained as:

$$A_{\Delta v} = [A_{var}V]_{(4N,2N)} \qquad b_{\Delta v} = [\Delta V + A_{var}v_0]_{(4N,1)} \tag{5.69}$$

## 5.5 Position Constraints

In the previous section, constraints related to the control variables were introduced, in order to handle actuator limitations, comfort and safety requirements. To take into account all the boundaries that characterize a real context, especially in a crowded environment, the presence of obstacles and people must be considered. Therefore, it is necessary to enforce constraints related to the state variables:

$$x(k) \in \mathbb{X} \tag{5.70}$$

In fact, since the state of the system coincides with the position of the vehicle, the trust region described trough the constraints is defined as the free space of the environment in which the robot is able to move freely. The approach presented in the previous chapter allows to compute a polytopic region of obstacle-free space within sensor range, by generating a set of hyperplanes which separate the convex region of space containing the wheelchair from the detected obstacles. Considering the vehicle

Configuration space, the hyperplanes, delimiting the trust region take the form of:

$$h_x \; x + h_y \; y \leq l \tag{5.71}$$

where $h_x$ and $h_y$ are the coefficients related to the variables $x$ and $y$ identifying the slope of the 1-dimensional line, while $l$ is the constant term defining the distance from the origin. Each equation of the form (5.71) can be implemented within the MPC formulation as linear state constraints dividing the state space into two half planes, one representing the admissible free space, while the other the forbidden space.
If a pedestrian is not in potential collision with the vehicle, there is no need to reduce the trust region with the introduction of a constraint.
To ensure that the half plane denoting the free space is the one in which the estimated position of the vehicle is contained, the following relationship must hold:

$$h_x \; \hat{x}_c + h_y \; \hat{y}_c \leq l \tag{5.72}$$

where $\hat{x}_c$ and $\hat{y}_c$ are the coordinates of the vehicle estimated position. If that is not the case, in order to obtain an inequality with the lower or equal sign, suitable for the MPC implementation, it is necessary to invert the signs:

$$(-h_x) \, \hat{x}_c + (-h_y) \, \hat{y}_c \leq (-l) \tag{5.73}$$

### 5.5.1 State Constraint Implementation

The constraint related to the position can thus be determined as illustrated in the previous chapter, and defined, for each time instant $k + i$ within the selected prediction horizon $N$, as linear constraints of the kind:

$$h_x^{(i)} x(k+i) + h_y^{(i)} y(k+i) \leq l^{(i)} \qquad i = 0, ..., N \tag{5.74}$$

In fact, inside a prediction horizon, different constraints for a given pedestrian can be defined, according to the position and relative speed at each

time instant.
As for the control variables case, in quadratic programming, linear system state constraints along the entire prediction horizon can be expressed in the form of a linear inequality:

$$A_{ineq_x}\Xi(k) \leq b_{ineq_x} \tag{5.75}$$

where $A_{ineq}$ is a matrix whose rows correspond to the condition imposed at a given time instant in the prediction horizon, while $b_{ineq}$ is the vector of constant terms. To include the state constraints in the Model Predictive control formulation, it is necessary to express them with respect to the vector of control variables $\mathcal{U}(k)$.
By recalling the Lagrange equation:

$$A_{ineq_x}(\underbrace{\mathcal{A}\xi(k) + \mathcal{B}\mathcal{U}(k)}_{\Xi(k)}) \leq b_{ineq_x} \tag{5.76}$$

it follows that

$$A_{ineq}\,\mathcal{U}(k) \leq b_{ineq} \tag{5.77}$$

with

$$A_{ineq} = A_{ineq_x}\mathcal{B} \qquad b_{ineq} = b_{ineq_x} - A_{ineq_x}\mathcal{A}\xi(k) \tag{5.78}$$

Constraints in the form (5.74), $\forall i \in (0, ..., N)$, can then be rewritten in a more suitable MPC formulation as:

$$\begin{bmatrix} h_k & & \\ & \ddots & \\ & & h_{k+N} \end{bmatrix}_{(N+1,2(N+1))} \begin{bmatrix} \xi(k) \\ \vdots \\ \xi(k+N) \end{bmatrix}_{(2(N+1),1)} \leq \begin{bmatrix} l_k \\ \vdots \\ l_{k+N} \end{bmatrix}_{(N+1,1)} \tag{5.79}$$

extending the position constraints to the overall prediction horizon $[k, k+N]$.
At every time instant, the robot can access only its actual position, expressed as:

$$\xi(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \tag{5.80}$$

However, the constraints are computed not only with respect to the actual position of the vehicle, but also for the future ones in the whole prediction horizon $k, ..., k+N$. This problem can be solved by recalling the Lagrange Equation:

$$\Xi(k) = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}_{(2(N+1),2)} \xi(k) + \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}_{(2(N+1),2N)} \mathcal{U}(k) \tag{5.81}$$

In which the vector:

$$\Xi(k) = \begin{bmatrix} \xi(k) \\ \vdots \\ \xi(k+N) \end{bmatrix}_{(2(N+1),1)} \tag{5.82}$$

represents all the predicted future positions of the vehicle based on the control variables $\mathcal{U}(k)$ computed at the corresponding time instant. It is then possible to rewrite (5.79) by taking into account this aspect:

$$\underbrace{\begin{bmatrix} h_{k|k-1} & & \\ & \ddots & \\ & & h_{k+N|k-1} \end{bmatrix}}_{H_{obs}} \underbrace{\begin{bmatrix} \xi(k) \\ \vdots \\ \xi(k+N) \end{bmatrix}}_{\Xi(k)} \leq \underbrace{\begin{bmatrix} l_{k|k-1} \\ \vdots \\ l_{k+N|k-1} \end{bmatrix}}_{L_{obs}} \tag{5.83}$$

In this case, the subscripts $k|k-1, ..., k+N|k-1$ express the fact that the constraint will be computed for the whole prediction horizon, by taking into account the information retrieved in the previous execution. In particular, by assuming a particle-like pedestrian kinematic model, the pedestrian positions $(x_i, y_i)$ can be predicted as:

$$\begin{cases} x_{i+1} = x_i + \tau \ v_x \\ y_{i+1} = y_i + \tau \ v_y \end{cases} \qquad \forall i = 0, ..., N-1 \tag{5.84}$$

with $(x_0, y_0)$, $v_x$ and $v_y$ correspond to the sensors initially detected position and velocity.

In conclusion, Eq. 5.83 can thus be expressed in compact form as:

$$[H_{obs}]_{(N+1,2(N+1))}\ [\Xi(k)]_{(2(N+1),1)} \leq [L_{obs}]_{(N+1,1)} \tag{5.85}$$

As already mentioned, it is necessary to express the constraint in a formulation suitable for the MPC:

$$H_{obs}\ \underbrace{(\mathcal{A}\xi(k) + \mathcal{B}\mathcal{U}(k))}_{\Xi(k)} \leq L_{obs} \tag{5.86}$$

from which

$$H_{obs}\mathcal{B}\mathcal{U}(k) \leq L_{obs} - H_{obs}\mathcal{A}\xi(k) \tag{5.87}$$

That can be rewritten as:

$$A_{obs}\ \mathcal{U}(k) \leq b_{obs} \tag{5.88}$$

where

$$A_{obs} = [H_{obs}\mathcal{B}]_{(N+1,2N)} \qquad b_{obs} = [L_{obs} - H_{obs}\mathcal{A}\xi(k)]_{(N+1,1)} \tag{5.89}$$

This constraint is related to a single obstacle. Notably, by defining $A_{obs}^{(j)}$ and $b_{obs}^{(j)}$ as the constraint related to the $j-th$ obstacle, it is possible to represent all the position constraints by defining:

$$\begin{bmatrix} A_{obs}^{(1)} \\ \vdots \\ A_{obs}^{(n_{obs})} \end{bmatrix}_{(n_{obs}(N+1),2N)} \qquad \begin{bmatrix} b_{obs}^{(1)} \\ \vdots \\ b_{obs}^{(n_{obs})} \end{bmatrix}_{(n_{obs}(N+1),1)} \tag{5.90}$$

## 5.6 Slack variables

In order to allow autonomous vehicles to navigate in human-crowded environments safely and taking into account human comfort, it is necessary to model subtle human behaviours and navigation rules. Generally speaking, human social conventions are tendencies. While these

result instinctive to humans, despite several dissimilitudes appear between different cultures, socially compliant navigation is still difficult to quantify and implement in an autonomous navigation algorithm, due to the stochasticity in people's behaviors. In this work, a novel approach was implemented to enable socially compliant human-robot interactions, based on the socially aware navigation model presented in the previous chapter. This method relies on the introduction of soft position constraints that consider the navigation behavior of interacting pedestrians, modeling their personal space (as in [25]) in order to avoid it, unless in over-crowded or critical conditions. In fact, constraints may be hard or soft; hard constraints provide an absolute limit, while soft constraints allow to slightly exceed them, but at a high associated cost in the performance index. So, if the problem results unfeasible, the constraints are relaxed in order to allow the solvability of the problem. In fact, by slightly enlarging the trust region the vehicle can get closer than the standard social distance to humans in a significantly crowded environment. The relaxation is allowed up to a certain limit, of course. In fact, the autonomous vehicle is forbidden in any case to cause collisions.
This introduction was intended to produce sociable, human-like paths; however, it helped to solve another intrinsic issue in the nature of the Model Predictive Control constraint definition. In fact, it may happen that the hard position constraints previously defined, despite guaranteeing safety, could provide an excessively strict solution, which may result in the unfeasibility of the control problem. This behaviour can be caused by sensors failures, uncertainties, noise, etc.

### 5.6.1 Soft Position Constraints

In order to formalize the model presented in Chapter 4, in a context related to an autonomous wheelchair, the work developed in [5] will be taken into account. As already mentioned, the slack variables related to the position are introduced to respect the human-like tendencies to preserve the approaching pedestrians' personal space, thus reducing the trust region of the vehicle state space, as shown in Figure 5.6.
This choice was done due to the fact that, if the problem admits a feasible solution, this is related to the position of point $P$ defined by the aforementioned Feedback linearization procedure. If the vehicle can find a feasible solution, it will maintain a bigger distance from the pedestrian; otherwise, it will get slightly closer to the personal space, with the hard

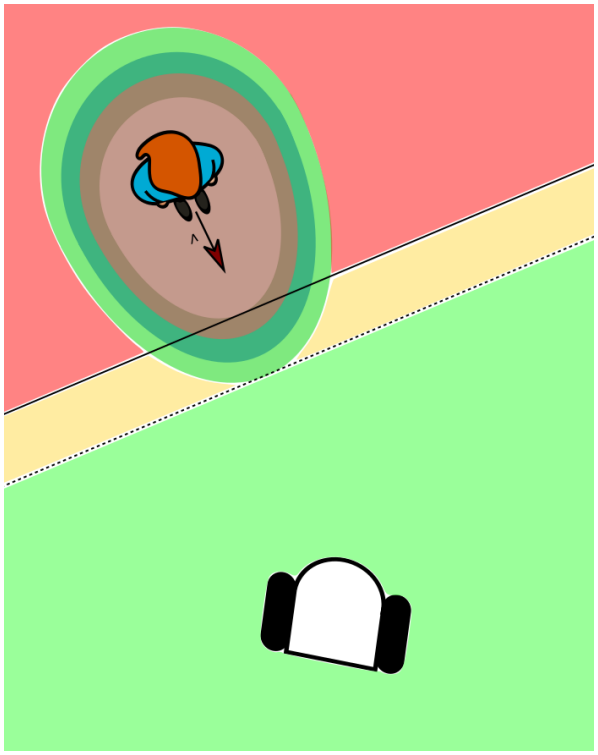


*Figure 5.6: Trust Region with Hard and Soft Constrains*

bound of avoiding to enter the close phase of intimate distance.
The position constraints will be then defined as:

$$h_x^{(i)(j)} x(k+i) + h_y^{(i)(j)} y(k+i) \leq l^{(i)(j)} - \Delta l^{(i)(j)}(1 - u_{sl\_p}^{(j)}(k)) \qquad (5.91)$$

$$\forall i \in \{0, ..., N\},\ \forall j \in \{0, ..., n_{obs}\}$$

where $\Delta l^{(i)(j)} = \Delta l_{sl}^{(i)(j)} \sqrt{(h_x^{(i)(j)})^2 + (h_y^{(i)(j)})^2}$. The coefficient $\Delta l_{sl}^{(i)(j)}$ represents the safety distance to be imposed in order to correctly include the pedestrian personal space. In fact, the size of the vehicle and of the pedestrian are already taken into account in the standard position constraints, as previously reported. The addition of slack variables aims to fulfill the human-comfort navigation requirements. The coefficient $\Delta l_{sl}^{(i)(j)}$ is determined as introduced in Chapter 4.
The slack variables introduced in Eq. 5.91 must assume values between 0 and 1:

$$0 \leq u_{sl\_p}^{(j)}(k) \leq 1 \qquad (5.92)$$

$$\forall j \in \{0, ..., n_{obs}\}$$

It is then obtained that:

$$\begin{cases} h_x^{(i)(j)} x(k+i) + h_y^{(i)(j)} y(k+i) \leq l^{(i)(j)}, & if\ u_{sl\_p}^{(j)}(k) = 1 \\ \\ h_x^{(i)(j)} x(k+i) + h_y^{(i)(j)} y(k+i) \leq l^{(i)(j)} - \Delta l^{(i)(j)}, & if\ u_{sl\_p}^{(j)}(k) = 0 \end{cases} \tag{5.93}$$

In order to extend the modified position constraints to the whole prediction horizon, it is necessary to define:

$$E^{(j)} = \begin{bmatrix} 0 & \dots & 0 & \dots & -\Delta l^{(i)(j)} & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \\ 0 & \dots & 0 & \dots & \underbrace{-\Delta l^{(i)(j)}}_{\text{j-th position}} & 0 & \dots & 0 \end{bmatrix}_{(N+1, n_{obs})} \tag{5.94}$$

where the vector different from zero occupies the $j-th$ position.
The extraction matrix (5.94) allows to formulate:

$$u_{sl\_p}^{(j)}(k) = E^{(j)}\ u_{sl\_p}(k) \tag{5.95}$$

and considering the following variant of the position constraints:

$$[H_{obs}^{(j)}]_{(N+1,2(N+1))}\ \Xi(k) \leq [\bar{L}_{obs}^{(j)}]_{(N+1,1)} \tag{5.96}$$

where

$$\bar{L}_{obs}^{(j)} = L_{obs}^{(j)} - \begin{bmatrix} \Delta l^{(j)} \\ \vdots \\ \Delta l_{N+1,1}^{(j)} \end{bmatrix} \tag{5.97}$$

it is possible to express condition (5.91) with respect to the set of enlarged control variables $\bar{\mathcal{U}}(k)$:

$$\bar{\mathcal{U}}(k) = \begin{bmatrix} \mathcal{U}(k) \\ u_{sl\_p}(k) \end{bmatrix}_{(2N+n_{obs},1)} \tag{5.98}$$

as the enlarged set of constraints:

$$\bar{A}_{obs}^{(j)} \bar{\mathcal{U}}(k) \leq \bar{b}_{obs}^{(j)} \tag{5.99}$$

with:

$$\bar{A}_{obs}^{(j)} = \begin{bmatrix} H_{obs}^{(j)}\mathcal{B} & E^{(j)} \end{bmatrix}_{(N+1,2N+n_{obs})} \qquad \bar{b}_{obs}^{(j)} = \bar{L}_{obs}^{(j)} - H_{obs}^{(j)}\mathcal{A}\xi(k) \quad (5.100)$$

However, in order to consider the enlarged optimization problem, it is fundamental to make the appropriate modifications to the cost function formulation, compatibly with the new requirements. To this aim, the new cost function becomes:

$$\bar{J}(k) = J(k) + ||u_{sl\_p}(k)||_{S_p}^2 \quad (5.101)$$

where $S_p$ is the additional weight matrix:

$$S_p = \begin{bmatrix} s_p & & \\ & \ddots & \\ & & s_p \end{bmatrix}_{(n_{obs},n_{obs})} \quad (5.102)$$

while $u_{sl\_p}(k)$ is the $n_{obs}$-dimensional vector ( where $n_{obs}$ is the number of obstacles) representing the set of slack variables related to the state variables.
The value of the parameters $s_p$ will be chosen by selecting a very high weight with respect to $q$ and $r$, in order to allow the controller to impose slack variables values different from zero only if the problem feasibility is compromised.
Finally, it is possible to formulate the new cost function in matrix form:

$$\bar{J}(k) = \frac{1}{2}\bar{\mathcal{U}}(k)^T\bar{H}\bar{\mathcal{U}}(k) + 2\bar{f}^T\bar{\mathcal{U}}(k) + cost \quad (5.103)$$

where

$$\bar{H} = \begin{bmatrix} H & \\ & S_p \end{bmatrix}_{(2N+n_{obs}+2,2N+n_{obs})} \quad (5.104)$$

$$\bar{f}^T = \begin{bmatrix} f^T & 0_{(1,n_{obs})} \end{bmatrix}_{(1,2N+n_{obs})} \quad (5.105)$$

# Chapter 6

# Simulation results

In order to test the effectiveness of the integrated strategy for autonomous navigation in human-crowded environments, simulations were run in the MATLAB® environment, exploiting the ILOG CPLEX Optimization Studio 12.5.0 IBM®. All the simulations were performed on an IntelCore i5-6200U @2.40 GHz personal computer with 12GB RAM. The vehicle taken into account for the analysis is an electric wheelchair, endowed with two driving wheels and equipped with two SICK TiM561 laser sensors characterized by a maximum range of $10[m]$ and a scanning frequency of $15[Hz]$. The wheelchair discrete-time model is:

$$\xi(k+1) = A\ \xi(k) + B\ u(k) \tag{6.1}$$

where:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix} \quad \xi(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \quad u(k) = \begin{bmatrix} v_{Px}(k) \\ v_{Py}(k) \end{bmatrix} \tag{6.2}$$

The following subsections will introduce the software used, the tuning of the PRM and MPC algorithm parameters, and the results obtained in simulation. In particular, three different pedestrian avoidance simulation cases will be analyzed: a single pedestrian scenario, a simulation in a crowded room, and a real environment based on the ETH Walking Pedestrians (EWAP) Dataset [33].

## 6.1 Software

The control algorithm was implemented in the MATLAB® environment, developed by MathWorks. MATLAB® is a programming platform that

combines a desktop environment tuned for iterative analysis and design processes with a matrix-based programming language. The Probabilistic RoadMap (PRM) path planning algorithm from MATLAB® Robotics System Toolbox™ was exploited to create the collision-free waypoints to reach the destination goal. Robotics System Toolbox™ provides tools and algorithms for designing, simulating, and testing manipulators, mobile robots, and humanoid robots.

The ILOG CPLEX Optimization Studio 12.5.0 from IBM® was exploited as the control algorithm solver. CPLEX, a feature of IBM ILOG Optimization Studio, offers state of the art performance and robustness in an optimization engine for solving problems expressed as mathematical programming models. In particular, the optimization function *cplexqcp* allows to solve quadratically constrained linear/quadratic programming problems, specified as follows:

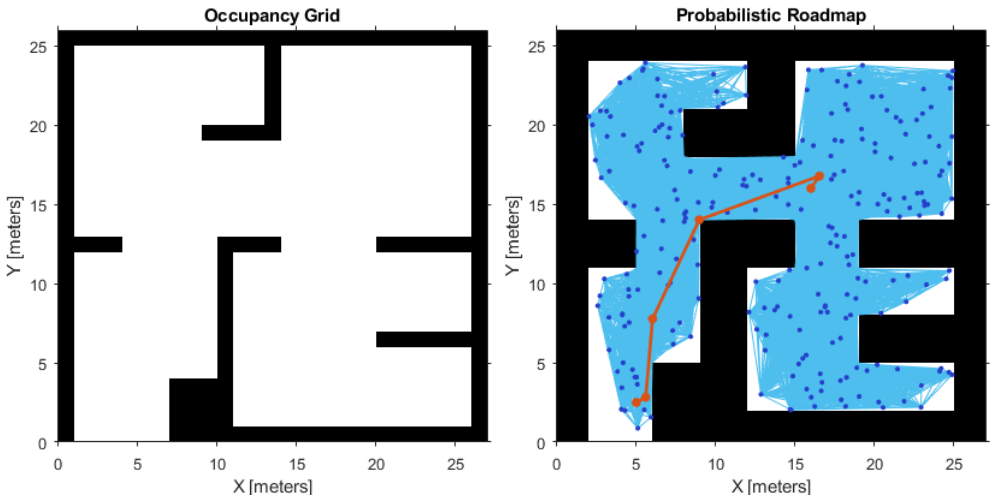
$$\begin{aligned} \min_{x} \quad & 0.5\ x^T H x + f x \\ \text{subject to} \quad & A_{ineq}\ x \leq b_{ineq}, \\ & A_{eq}\ x = b_{eq}, \\ & l x + x^T Q x \leq r, \\ & lb \leq x \leq ub. \end{aligned} \tag{6.3}$$

where the matrix $H$ and the row vector $f$ constitute the objective function to minimize, while $A_{ineq}$, $b_{ineq}$, $A_{eq}$, $b_{eq}$,$l$, $Q$, $r$, $lb$ and $ub$ are matrices and vectors of suitable dimensions expressing the linear inequality constraints, the linear equality contraints, the quadratic constraints, lower and upper bounds.

## 6.2 Parameter Tuning

### 6.2.1 PRM

The Probabilistic RoadMap (PRM) path planning algorithm from MATLAB® Robotics System Toolbox™ was exploited to create the collision-free waypoints to reach the destination goal. As already specified, only the static part of the environment is considered in the Global Planning analysis, e.g., the fixed obstacles of the selected map. Before performing the search, the environment map is inflated, in order to take into account the vehicle dimensions (as shown in Figures 6.1a and 6.1b).

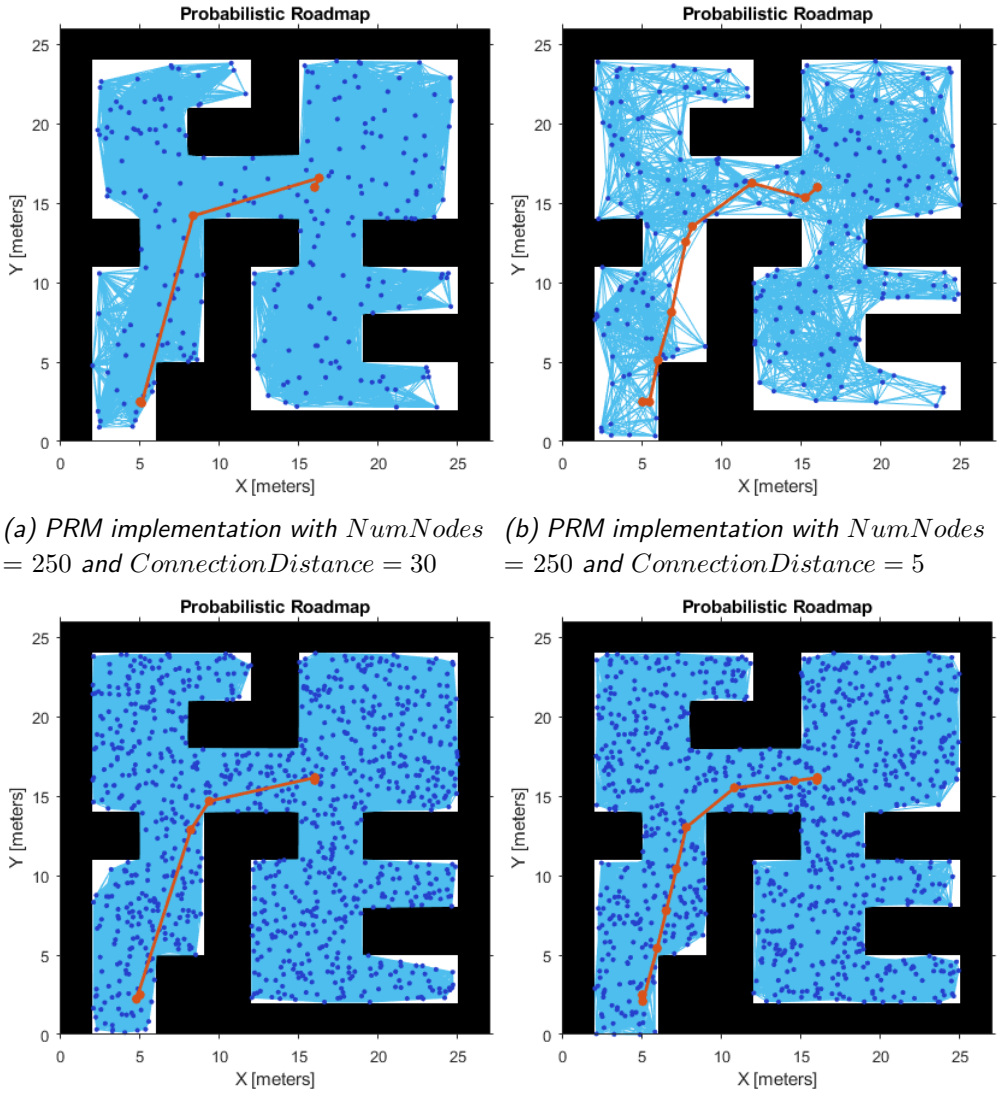



*(a) A priori knowledge of the environment* *(b) Probabilistic RoadMap planning*

*Figure 6.1: PRM algorithm*

The PRM algorithm introduces the mobileRobotPRM object to randomly generate nodes and to create connections between them, based on the PRM algorithm parameters and on the obstacle locations specified in the considered map of the environment (Figure 6.1a). In fact, the PRM algorithm uses the network of connected nodes to find an obstacle-free path from a start to an end location [1].
Among the PRM parameters, the number of nodes, NumNodes, as well as the ConnectionDistance, must be tuned properly, in order to fit the complexity of the map and to achieve the desired path efficiency. The NumNodes property specifies the number of points, or nodes, placed on the map, which the algorithm uses to generate a roadmap. Using the ConnectionDistance property as an upper threshold for distance, the algorithm connects each node to all nodes reachable with a collision-free path within the connection distance. As shown in Figure 6.2, increasing the number of nodes can increase the number of feasible paths, thus boosting the efficiency of the final path. However, the increased complexity increases computation time. On the other hand, by lowering the connection distance, the map can be simplified, since the number of connections is limited and the computation time reduced. However, a lowered distance limits the number of available paths.
As already mentioned, whenever the Local Planner does not find a solution to the trajectory tracking problem, i.e., the Model Predictive Control optimization fails repeatedly, the path needs to be replanned. The num-

ber of nodes constituting the Probabilistic Roadmap is then increased, augmenting the NumNodes property by 50, in order to determine more feasible paths and to boost the efficiency of the final path.
The starting NumNodes and ConnectionDistance property values are, for the environment considered in the following simulations: $NumNodes = 300$ and $ConnectionDistance = 20$.




*(a) PRM implementation with* $NumNodes = 250$ *and* $ConnectionDistance = 30$

*(b) PRM implementation with* $NumNodes = 250$ *and* $ConnectionDistance = 5$

*(c) PRM implementation with* $NumNodes = 1000$ *and* $ConnectionDistance = 30$

*(d) PRM implementation with* $NumNodes = 1000$ *and* $ConnectionDistance = 5$

*Figure 6.2: NumNodes and ConnectionDistance parameters tuning.*

### 6.2.2 MPC

To verify the effectiveness of the reference trajectory generation and tracking, and to determine a suitable value for the MPC parameters, the MPC control algorithm was tested in simulation considering a free environment, without the presence of people. The MPC parameters were tuned to take into account the uncertainties and delays (of measurement and actuation) of the control over the system, and to meet the computational constraints of the computer.

#### Choice of weights in the MPC cost function

Recalling the MPC cost function

$$J(k) = \sum_{i=0}^{N-1} (\|\xi(k+i) - \xi_{ref}(k+i)\|_Q^2 + \|u(k+i)\|_R^2) + \\ + \|\xi(k+N) - \xi_{ref}(k+N)\|_S^2 \tag{6.4}$$

and the discrete system model

$$\xi(k+1) = A\ \xi(k) + B\ u(k) \tag{6.5}$$

where:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix} \quad \xi(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \quad u(k) = \begin{bmatrix} v_{Px}(k) \\ v_{Py}(k) \end{bmatrix} \tag{6.6}$$

the matrices $Q$, $R$ and $S$ are design parameters expressing the penalty given to the error on the state, to the control variables and to the final state variables, respectively. Since the wheelchair model (6.5) is a symmetric decoupled system, the weight matrices can be designed as:

$$Q = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \qquad R = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \qquad S = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \tag{6.7}$$

where $q$ and $r$ are selected in order to guarantee the desired trajectory tracking and control performance, while the parameter $s$ is determined to ensure the asymptotic stability of the system.
A good trade-off between the parameters $r$ and $q$ ensures a satisfactory tracking performance, complying with actuation and comfort needs. The calibration of the $r$ and $q$ weights for the control algorithm must be

validated experimentally. The values reported in this work were chosen taking into account the experimental results reported in [8] and [12].

As it can be seen from the simulation reported in Figure 6.3, Figure 6.4 and Figure 6.5, the reference trajectory planned with constraints on the acceleration ($a \leq 0.2[m/s^2]$), velocity ($v \leq 0.55[m/s^2]$), jerk ($j \leq 0.2[m/s^3]$) and snap ($s \leq 0.2[m/s^4]$), in human-free conditions, can be easily tracked with satisfactory results. In fact, the controlled system converges towards the time-varying reference, complying with actuation and comfort constraints (in the form of velocity and velocity variation limitations). The velocity constraints were implemented in both methods presented in the previous Chapter, i.e., in form of linear inequalities and as quadratic constraints. Both implementations produced the same simulation results reported in Figure 6.3, Figure 6.4 and Figure 6.5. However, the implementation with linear constraints converged faster to a solution, as expected. As mentioned in [8], due to the intrinsic actuation delays and the uncertainties that affect the real system, the system closed loop behaviour would tend to be oscillatory. Therefore, it is necessary to use suitably calibrated thresholds for detecting the condition of convergence to the final reference that force the speed references to zero if the position is within a certain threshold distance from the goal. This threshold distance needs to be suitably calibrated in order to ensure a stop that complies with actuators limitations and comfort needs.

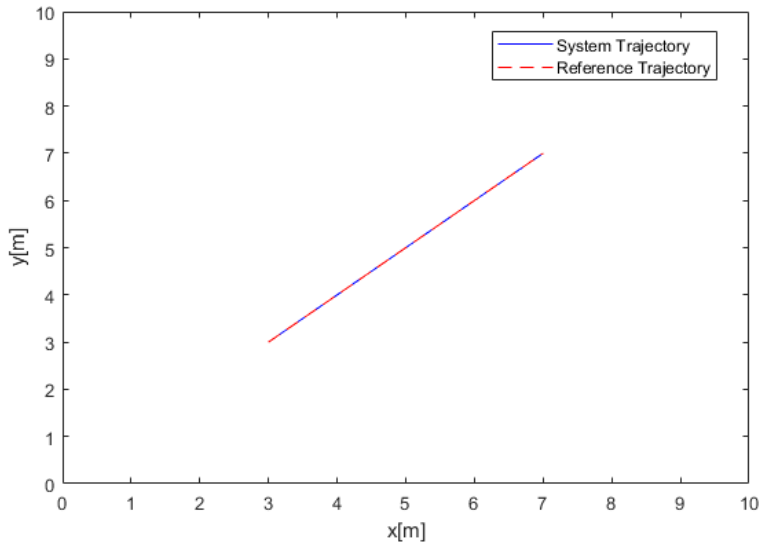



*Figure 6.3: The trajectory can be easily followed by the vehicle in simulation. The selected parameter values were $r = 1$ and $q = 1$.*

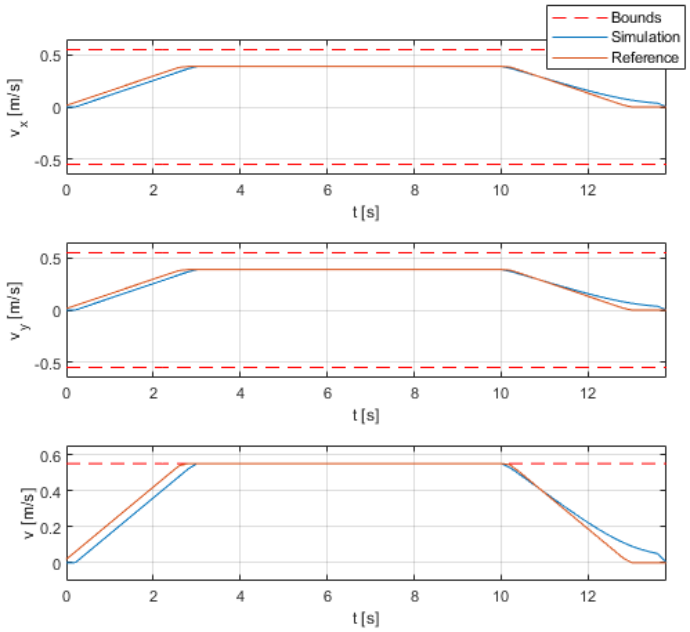



*Figure 6.4: Velocity profiles related to Figure 6.3. The orange line represents the planned reference velocity profile, while the blue line reports the simulated one. The red dashed lines highlight the set lower and upper velocity limitations.*

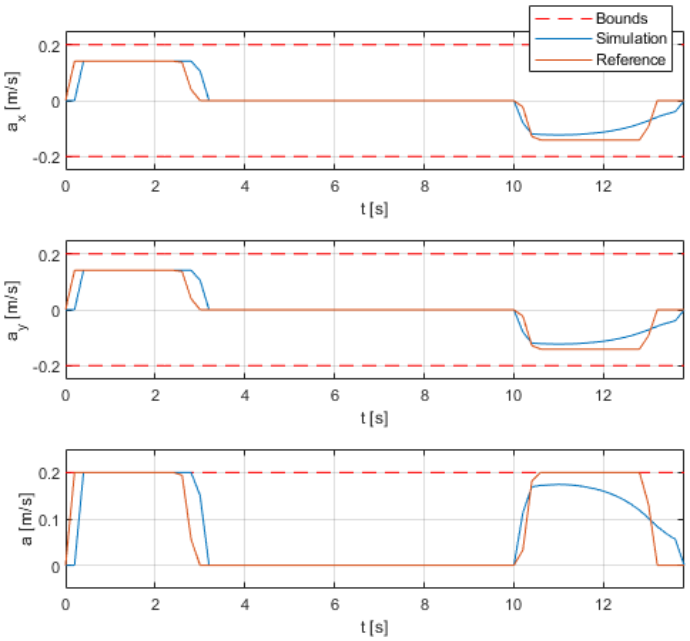



*Figure 6.5: Acceleration profile related to Figure 6.3. The orange line represents the planned reference acceleration profile, while the blue line reports the simulated one. The red dashed lines highlight the lower and upper acceleration limitations.*

A good trade-off for the MPC parameters was found by selecting:

$$q = 1 \rightarrow Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.8}$$

$$r = 1 \rightarrow R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.9}$$

By recalling Eq: 5.10, the time-varying auxiliary control law can be defined as:

$$u(k) = K\xi(k) \tag{6.10}$$

where $K$ is the symmetric diagonal matrix:

$$K = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \tag{6.11}$$

By solving the Discrete Riccati Equation (5.12), the relationship between the gain $k$ and the terminal cost parameter $s$ can be retrieved:

$$s = \frac{(q + k^2 r)}{1 - (1 + \tau k)^2} \tag{6.12}$$

The symmetric diagonal gain matrix $K$ must be selected in order to guarantee the asymptotic stability of the system. The solution reported in [5], retrieved trough the Pole Placement method, is exploited. In fact, by imposing:

$$k > -\frac{2}{\tau} \tag{6.13}$$

the position of the eigenvalues of the closed-loop system

$$\xi(k+1) = (A + BK)\xi(k) \tag{6.14}$$

guarantee the asymptotic stability of the system.
To avoid excessive oscillatory behaviour, a more conservative requirement can be imposed:

$$k > -\frac{1}{\tau} \tag{6.15}$$

A fraction of the limit value is then selected:

$$k = -\frac{1}{n\,\tau}, \quad \text{with } n > 1 \tag{6.16}$$

In particular, in this work, the selected value was $n = 2$:

$$k = -\frac{1}{2\tau} \tag{6.17}$$

and thus:

$$s = \frac{(q + k^2 r)}{1 - (1 + \tau k)^2} = 9.67 \tag{6.18}$$

$$S = \begin{bmatrix} 9.67 & 0 \\ 0 & 9.67 \end{bmatrix} \tag{6.19}$$

**Choice of the prediction horizon and the controller sampling time**

The choice of the prediction horizon $N$, as well as the controller sampling time $\tau$, depend on computational and control requirements. For example, an excessively short sampling time, although positive for the controller reactivity, may preclude the resolution in real time of the optimization problem. Moreover, it is necessary to recall that the prediction horizon $T_{ph}$, defined as the time window within which the controller predicts the system evolution, is related to $N$ and $\tau$ as:

$$N\ \tau = T_{ph} \tag{6.20}$$

The predictive ability of the controller increases in a longer prediction horizon; on the other hand, the computational load, which is related to the execution time, increases as well. The controller has a faster reactivity with a smaller sampling time $\tau$. However, a high value of $N$ implies a greater computational load. Furthermore, a small value of $\tau$ is required to satisfy the hypothesis that, during the controller sampling time $\tau$, the velocity of each moving obstacle, in particular of pedestrians, remains constant.
A trade-off between problem solvability and control performance needs to be achieved within the time interval $\tau$. A good compromise was obtained by setting $T_{ph}$ equal to 4 $s$ and:

$$\tau = 0.2\ s \qquad N = 20 \tag{6.21}$$

This choice is consistent with respect to the implementation of both exact linear and nonlinear velocity constraints, which provide an accurate

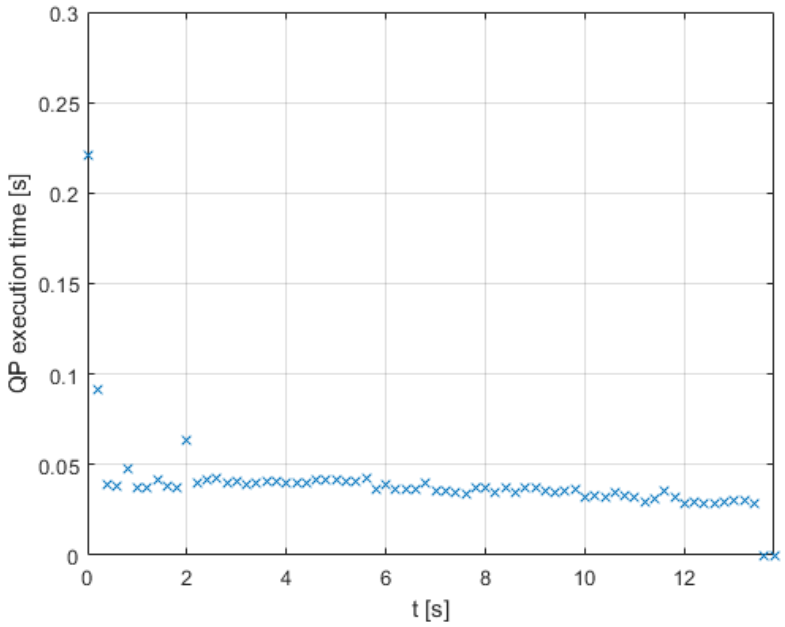



*Figure 6.6: Computational time of the simulation with quadratic velocity constraints related to Figure 6.3*

solution without affecting significantly the solver performance. In fact, as it can be seen from Figure 6.6 (reporting the computational time of the slowest simulation related to Figure 6.3, i.e., the one with quadratic velocity constraints), the MPC computational time is definitely acceptable, excluding the first iteration where the optimization environment is initialized. Notably, a C++ implementation would be even faster that the current Matlab implementation.

In order to guarantee the fulfillment of the control requirements, the vehicle stop needs to be achieved within the prediction horizon $T_{ph}$. Therefore, the time window $T_{ph}$ must assume a higher value than the worst case braking time, i.e., starting from the maximum reachable speed. As a consequence

$$T_{ph} = 4[s] > \frac{v_{max}}{a_{max}} \tag{6.22}$$

Moreover, the choice of the maximum longitudinal velocity value $v_{max}$ and of the maximum velocity variation $\Delta v_{max}$ must take into account the comfort requirements and the intrinsic delays on the system. These maximum reported values were chosen with reference to the experiments

on the real system in [8]. As for the maximum velocity variation, it was chosen to set a limit on the maximum average acceleration equal to:

$$a_{max} = 0.2[m/s^2] \tag{6.23}$$

which is reasonable in terms of comfort. From this, it is possible to retrieve the value of the maximum velocity variation by considering the product between the controller sampling time $\tau$ and maximum value of the average acceleration $\bar{a}_{max}$:

$$\Delta v_{max} = \bar{a}_{max}\ \tau \tag{6.24}$$

The choice of the maximum longitudinal velocity value $v_{max}$ was made by taking into account the experimental tests in [8] that placed the real system in a limit situation (providing the control algorithm with a time-varying rotating reference outside the admissible region) in order to observe the effects of uncertainties and delays of the system over the compliance with the imposed constraints. The limit value of:

$$v_{max} = 0.55\ [m/s] \tag{6.25}$$

was chosen, since it is conservative and reasonably safe for autonomous navigation.

## 6.3 Pedestrian Avoidance Simulations

In order to prove the consistency of the constraints on the status (and of the relative slack variables) in the definition of a convex obstacle-free trust region, the wheelchair behaviaour when encountering an approaching pedestrian was simulated in MATLAB. First of all, it is necessary to introduce the pedestrian kinematic model used in the following simulations. In fact, this work is developed under the assumption that, during the controller sampling time $\tau$, the velocity of each moving obstacle, in particular of pedestrians, remains constant. Since the controller sampling time was set as $\tau = 0.2[s]$, this assumption can be easily accepted. Therefore, the kinematic of a pedestrian can be characterized as a particle motion model, which is described by the following discrete time model:

$$\begin{cases} x(k+1) = x(k) + \tau\ v_x(k) \\ y(k+1) = y(k) + \tau\ v_y(k) \end{cases} \tag{6.26}$$

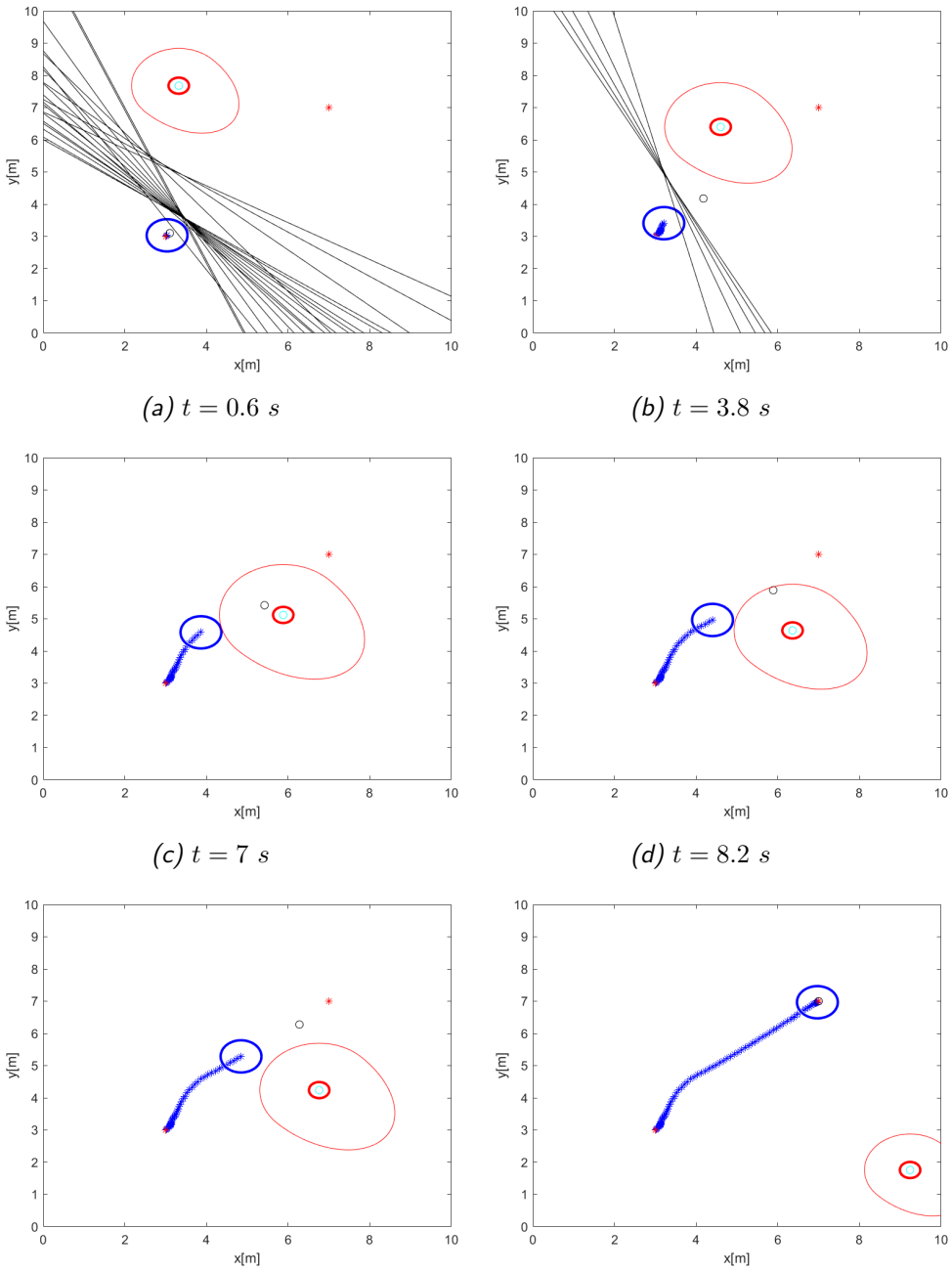
The above characterization allows to prove the consistency of position constraints, as highlighted in the following simulations.
The first simulation, as reported in Figure 6.7 and Figure 6.8, takes into account an approaching pedestrian heading towards the wheelchair from the left, starting from position $P_{ped} = [3, 8]$ with velocity $v_{ped} = [0.4, -0.4]$. The wheelchair starts from position $P_s = [3, 3]$ and needs to reach $P_g = [7, 7]$, following the reference trajectory. Figure 6.7 shows the evolution of the pedestrian avoiding trajectory, while in Figure 6.8 the final trajectory of the wheelchair is reported, as well as the reference trajectory planned by the Global Planner and the pedestrian trajectory. In this simulation, the robot is forced to deviate from the planned trajectory, in order to avoid collision. Not only is the pedestrian avoided, but also both the actuation constraints and the personal space are respected, as shown in Figures 6.9, 6.10 (reporting the velocity and acceleration profile corresponding to the simulated trajectory) and Figure 6.11 (highlighting the distance from the pedestrian).
To avoid collision, the wheelchair centre must not exceed a distance of $0.75[m]$ from the pedestrian estimated centre, as indicated by the red dashed line in Figure 6.11; this limit distance considers the wheelchair dimensions as well as those of an average pedestrian. The smallest distance detected in simulation was $1.8926[m]$. This distance takes into account the pedestrian personal space function value (with the introduction of slack variables) measured between the wheelchair position and the pedestrian one. The pedestrian personal space function is composed, as presented in Chapter 4, by two Gaussian functions: below the X-axis, the function is a symmetric Gaussian with

$$\sigma_x = \sigma_y = 3v \tag{6.27}$$

where $v$ is the relative velocity between the person and the wheelchair; above the X-axis, it is an elliptical function with

$$\sigma_y = 1.5\sigma_x \tag{6.28}$$

Since no other obstacle is detected within the path, the pedestrian Personal Space is fully respected, as shown in Figure 6.11.




(a) $t = 0.6\ s$ (b) $t = 3.8\ s$

(c) $t = 7\ s$ (d) $t = 8.2\ s$

(e) $t = 9.2\ s$ (f) $t = 15.6\ s$

*Figure 6.7: First Navigation simulation: Evolution of the pedestrian avoiding trajectory. The blue circle and stars represent the wheelchair occupied space, past and current position, whereas the cyan and red circle are the pedestrian position and virtual box, and the ellipsoidal red shape highlight the personal space. The black circle represents the reference trajectory, connecting the PRM waypoints (red stars), at the current time instant. The black lines are the constraints on the state space imposed along the prediction horizon.*

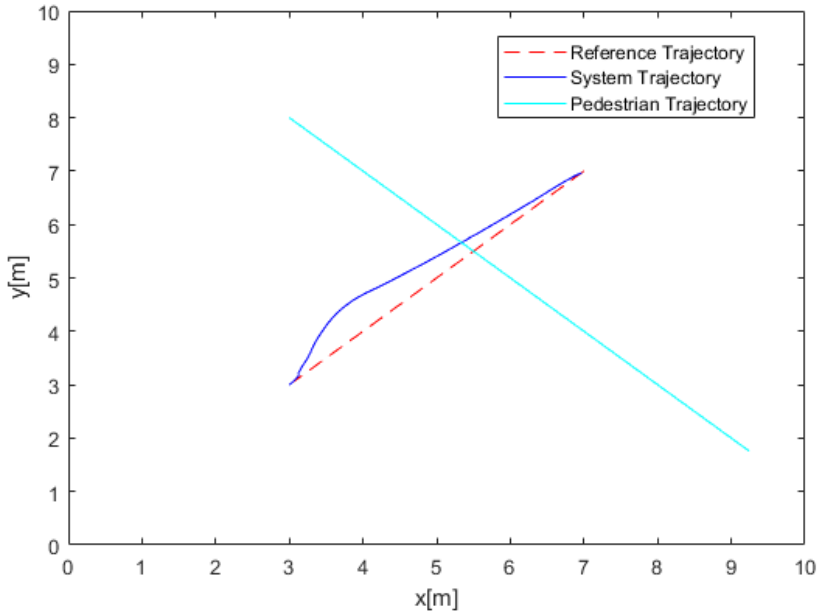



*Figure 6.8: First Navigation simulation: Final trajectory from start to goal*

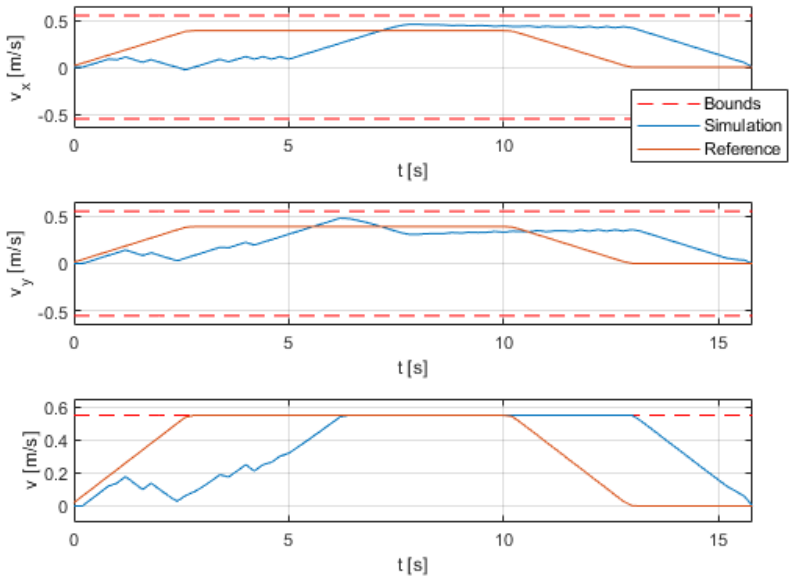



*Figure 6.9: First Navigation simulation: Velocity profiles*

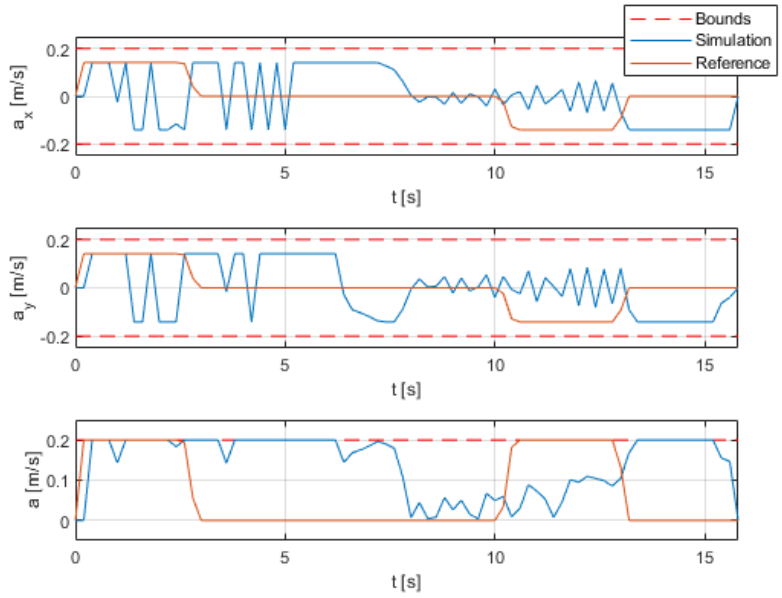



*Figure 6.10: First Navigation simulation: Acceleration profiles*

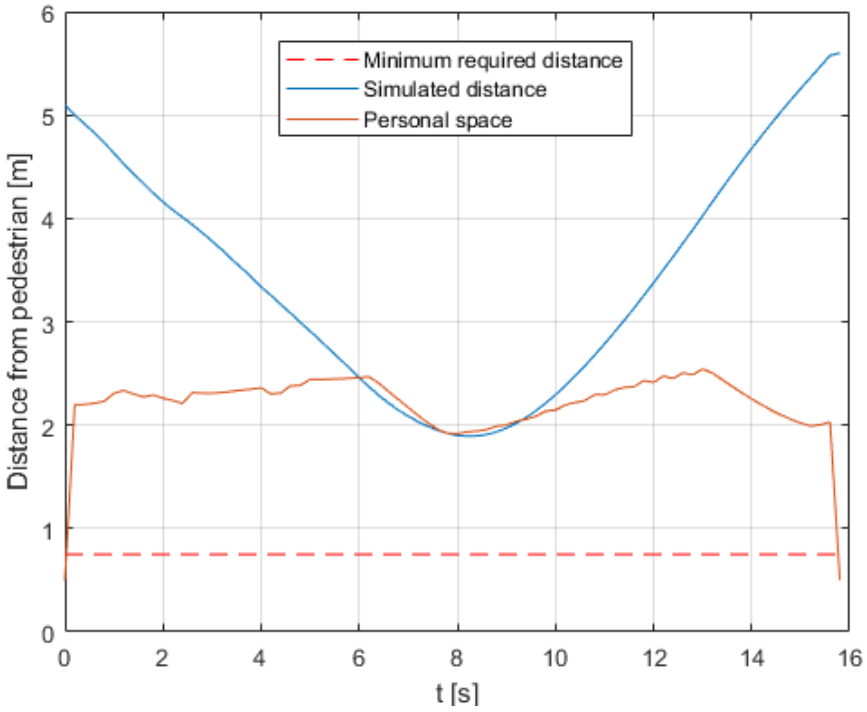
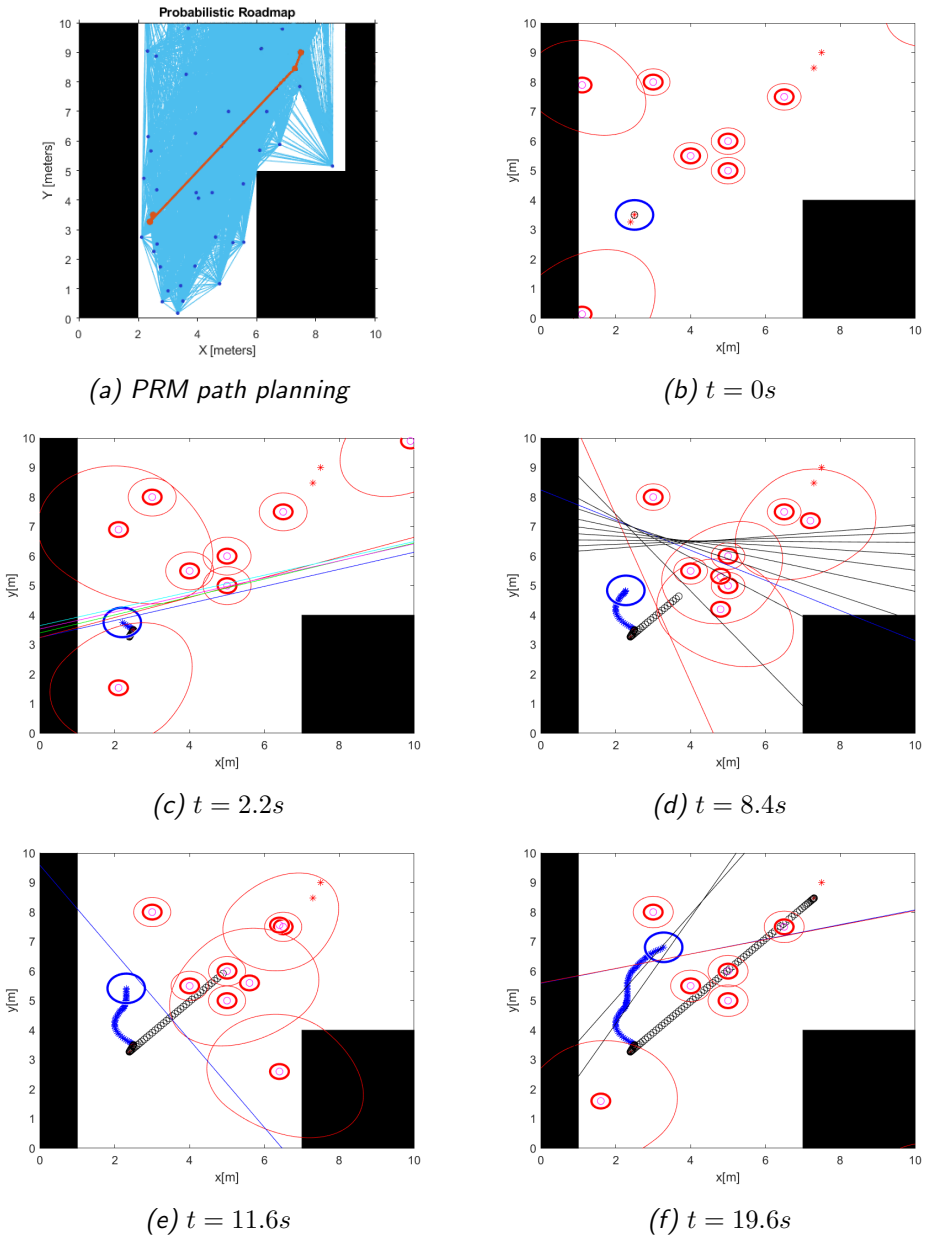



*Figure 6.11: First Navigation simulation: Distance from pedestrian*

## 6.4 Simulation in a crowded environment

To test the behavior of the proposed integrated strategy, a simulation in a crowded environments was performed. The navigation simulation presented in Figures 6.12, 6.13, 6.14, 6.15, 6.16, and 6.17, describes the results of a test carried out in a narrow environment, a room, where the wheelchair is surrounded by eight pedestrians, either standing or in motion. Five pedestrian are static, and are positioned in $P_{ped1} = [5, 5]$, $P_{ped2} = [5, 6]$, $P_{ped3} = [4, 5.5]$, $P_{ped4} = [6.5, 7.5]$ and $P_{ped5} = [3, 8]$. Three pedestrians are walking, starting from $P_{ped6} = [1, 8]$, $P_{ped7} = [11, 11]$ and $P_{ped8} = [1, 0]$, with velocities $v_{ped6} = [0.5, -0.5]$, $v_{ped7} = [-0.5, -0.5]$ and $v_{ped8} = [0.5, 0.7]$. The wheelchair starts from position $P_s = [2.5, 3.5]$ and needs to reach $P_g = [7.5, 9]$, following the reference trajectory planned by the Global Planner. Figures 6.12 and 6.13 show the evolution of the pedestrian avoiding trajectory from start to goal position, highlighting the trajectory tracking, the pedestrian personal space avoidance and the trajectory replanning in unexpectedly crowded circumstances. In Figure 6.14 the final trajectory of the wheelchair is reported, as well as the reference trajectories. In the figures, the blue star and circle represent the wheelchair position and occupied space, respectively, whereas the magenta circles contained in the red circles and ellipses highlight the pedestrian positions, virtual boxes and personal spaces. The small black circles represents the reference trajectory up to the current time instant, while the red stars represent the PRM waypoints. The coloured lines are the constraints on the state space imposed along the prediction horizon. The black areas represent the obstacle region occupied by fixed obstacles (walls). Since multiple pedestrians are encountered, the wheelchair deviates from the planned reference trajectory - which is not optimal and generated without the obstacle avoidance action enforced by the MPC controller - and needs to follow a different trajectory. Thus, as it can be seen in Figure 6.12f, the trajectory is replanned, in order to reach the goal position. In this way, the vehicle manages to reach the destination safely, complying with control and comfort requirements as shown in Figures 6.15 and 6.16 (reporting the velocity and acceleration profile corresponding to the simulated trajectory), avoiding each pedestrian, and respecting their personal space as much as possible, i.e., without compromising the solvability of the optimization problem. To avoid collisions, the wheelchair must not exceed a distance of 0.75[m] from the pedestrian estimated centre, as indicated with the red dashed line in Figure 6.17;

this limit distance considers the wheelchair dimensions as well as those of an average pedestrian. The smallest distance from the closest pedestrian detected in simulation was $1.1810[m]$. This value takes into account the pedestrian Personal Space function. This is a smaller value from the one reported in the previous section, due to different and more complex environmental conditions (more people and walls).




*(a) PRM path planning*

*(b)* $t = 0s$

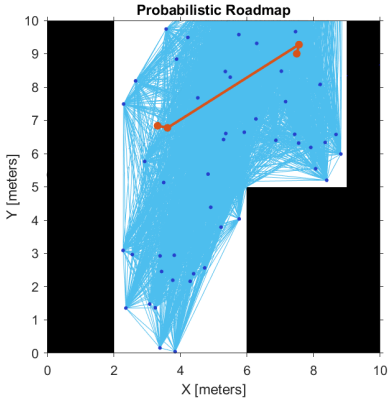
*(c)* $t = 2.2s$

*(d)* $t = 8.4s$

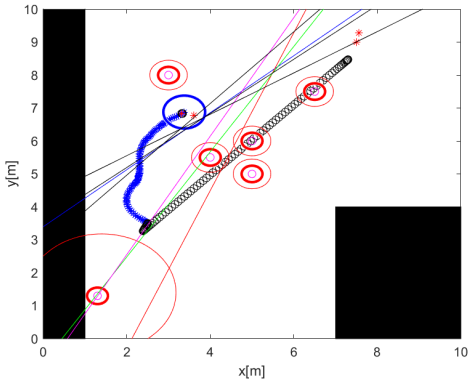
*(e)* $t = 11.6s$

*(f)* $t = 19.6s$

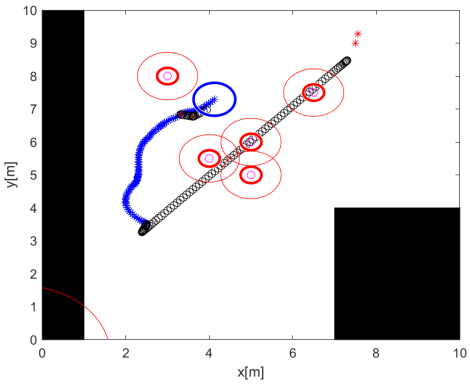
*Figure 6.12: Simulation in a crowded room: Evolution of the pedestrian avoiding trajectory until replanning is needed*



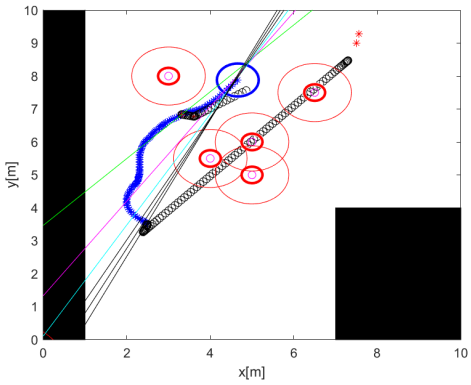

(a) PRM path replanning



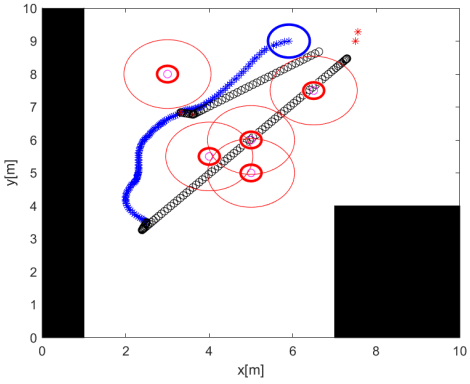

(b) $t = 20.2s$



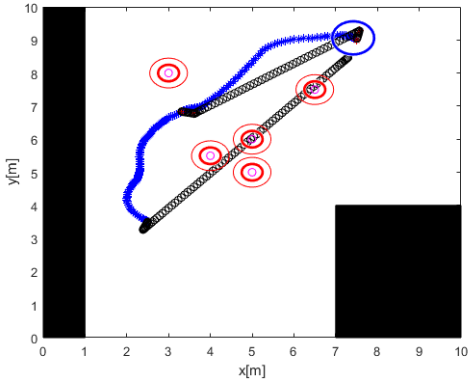

(c) $t = 24s$




(d) $t = 26s$




(e) $t = 29.8s$




(f) $t = 34.8s$

*Figure 6.13: Simulation in a crowded room: Evolution of the pedestrian avoiding trajectory after replanning*

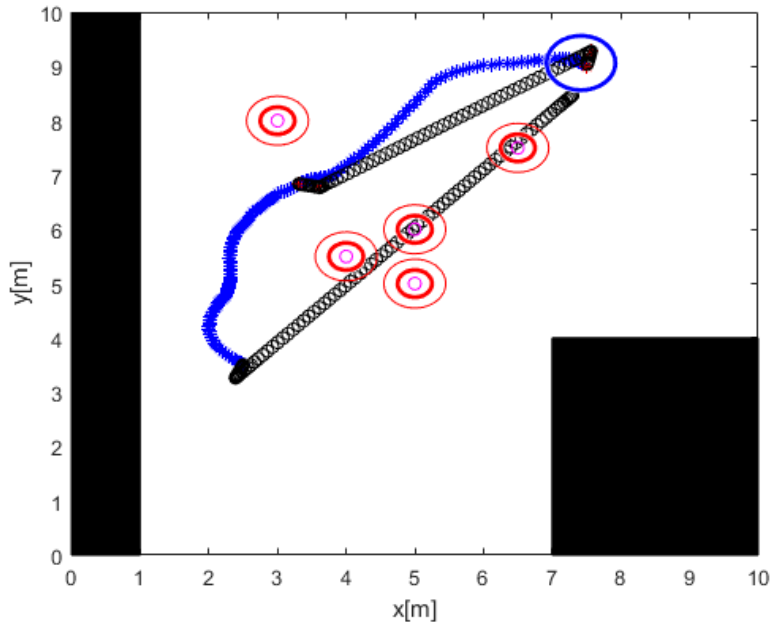



*Figure 6.14: Simulation in a crowded room: Final trajectory from start to goal*

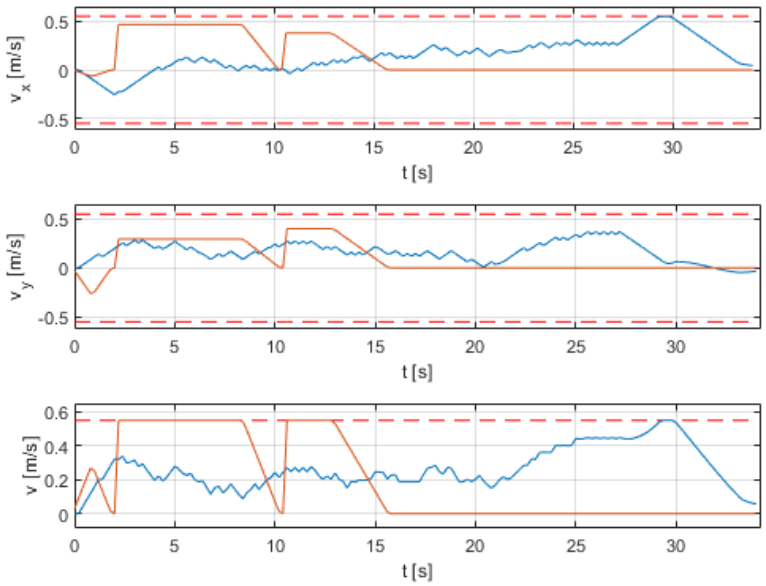



*Figure 6.15: Simulation in a crowded room: Velocity profiles. The orange line represents the planned reference velocity profile, while the blue line reports the simulated one. The red dashed lines highlight the set lower and upper velocity limitations.*

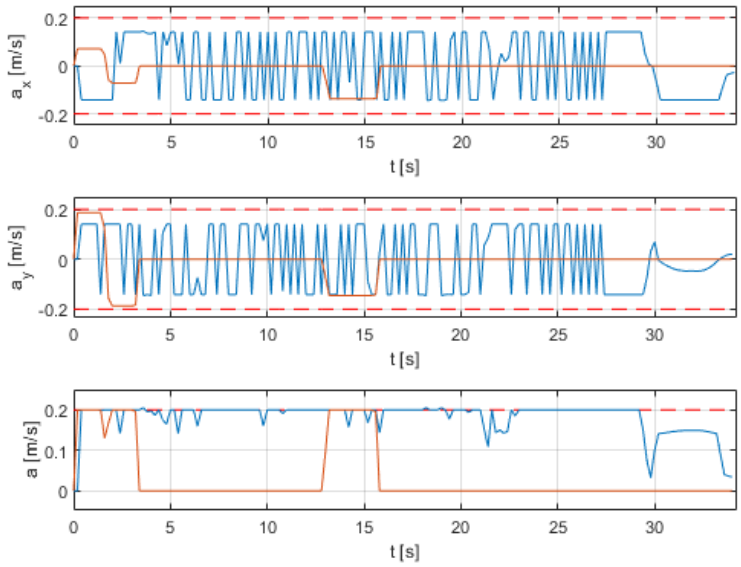



*Figure 6.16: Simulation in a crowded room: Acceleration profiles. The orange line represents the planned reference acceleration profile, while the blue line reports the simulated one. The red dashed lines highlight the lower and upper limitations.*

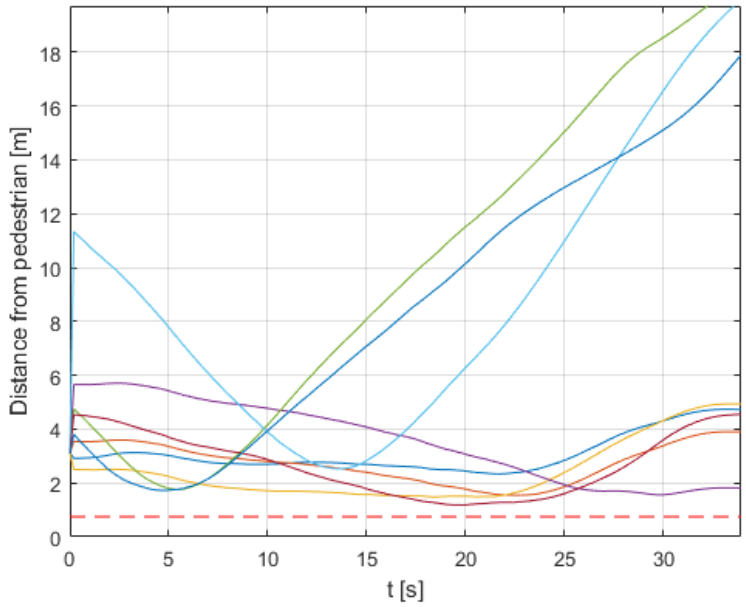



*Figure 6.17: Simulation in a crowded room: Distance from pedestrian. The red dashed lines highlight the minimum required distance, while continuous lines report the simulated distances from the different pedestrians.*

## 6.5 Simulation in a real environment

Finally, a simulation was performed considering a real environment, exploiting the BIWI Walking Pedestrians Dataset (EWAP) [33] from The Computer Vision Laboratory, ETH Zurich. This manually annotated dataset includes data about walking pedestrians in busy scenarios from a bird eye view, reporting a video together with the information about the pedestrians position, velocity and destination estimated from the video, as well as the obstacle map. The sequence considered in the following simulation was acquired at 25 fps from the 4th floor of Hotel Schweizerhof in Bahanhofstr, Zurich, by Stefano Pellegrini and Andreas Ess in 2009. However the annotation was done at 2.5 fps, that is with a timestep of 0.4 seconds. Since the controller sampling time is of 0.2 seconds, the data about position and velocity of each detected pedestrian needed to be interpolated, in order to provide a value for each time instant. Not all pedestrians in the video are reported in the dataset, especially the subjects at the border of the scene. The sequence reports the pedestrian positions and velocities in meters (meters/seconds), obtained with the given homography matrix, as well as annonation regarding people that seemed to walk in groups and the assumed destinations for all the subjects walking in the scene. The third direction, i.e., the perpendicular to the ground is not used. Being already given, the pedestrian trajectories do not consider the wheelchair presence.
The simulation takes into account the scenario of a wheelchair getting off the bus, that is required to reach the entrance of the hotel, as shown in Figures 6.18 and 6.19. Since multiple pedestrians are encountered, the wheelchair deviates from the planned reference trajectory. In this way, the vehicle manages to reach the destination safely, complying with control and comfort requirements as shown in Figures 6.20 and 6.21 (reporting the velocity and acceleration profile corresponding to the simulated trajectory), avoiding each pedestrian, and respecting their personal space as much as possible, i.e., without compromising the solvability of the optimization problem. To avoid collision, the wheelchair must not exceed a distance of $0.75[m]$ from the pedestrian estimated centre, as indicated with the red dashed line in Figure 6.22; this limit distance considers the wheelchair dimensions as well as those of an average pedestrian. The smallest distance from the closest pedestrian pedestrian in the dataset detected in simulation was $1.3459[m]$, which is consistent with the avoidance of pedestrian and their personal space. The fixed obstacles in the

scene are reported in Figure 6.23.
The implemented approach produced satisfactory simulation results even in a real environment, accomplishing the given task while ensuring a socially compliant navigation.

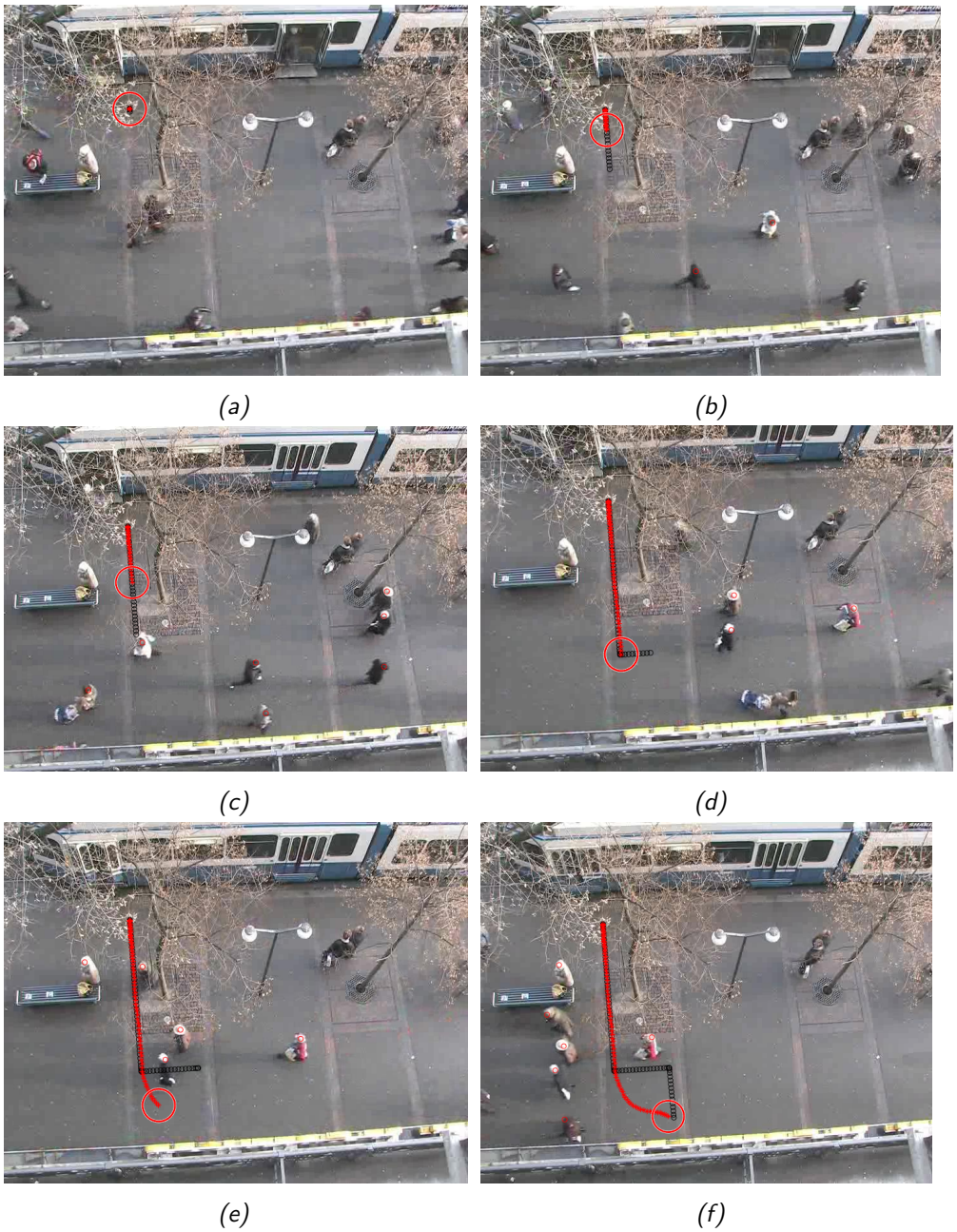


*Figure 6.18: Simulation in a real environment: Evolution of the pedestrian avoiding trajectory. In black the reference trajectory, in red the actual system trajectory, the wheelchair dimensions, and the detected pedestrian estimated center.*

*Figure 6.19: Simulation in a real environment: Final trajectory from start to goal*

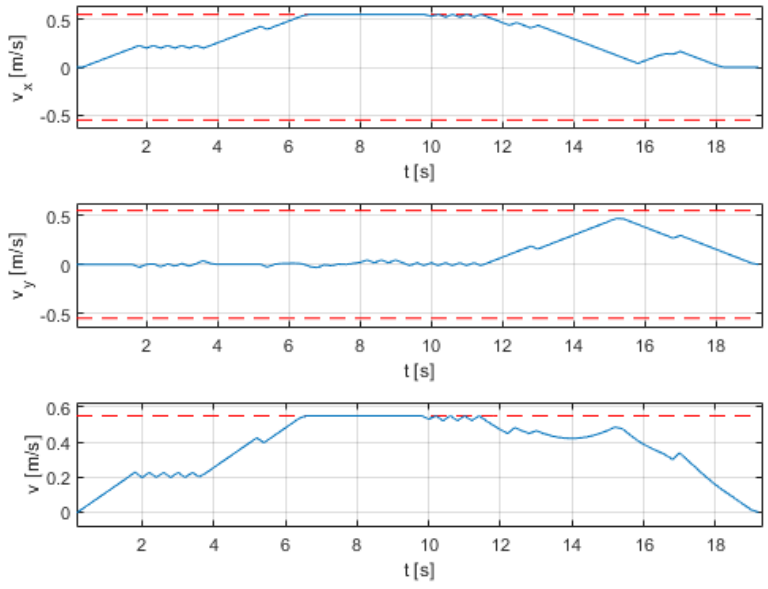



*Figure 6.20: Simulation in a real environment: Velocity profiles. The blue lines report the simulated velocity profile, while the red dashed lines highlight the set lower and upper velocity limitations.*

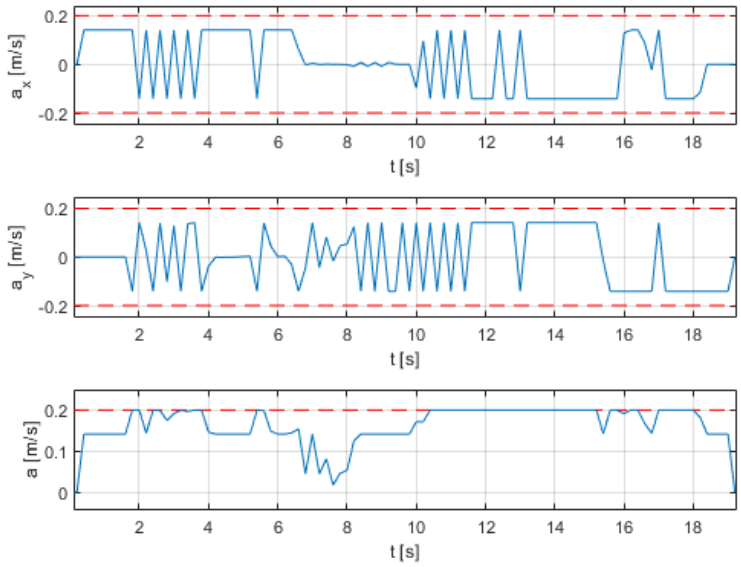



*Figure 6.21: Simulation in a real environment: Acceleration profiles. The blue lines report the simulated acceleration profile, while the red dashed lines highlight the set lower and upper acceleration limitations.*

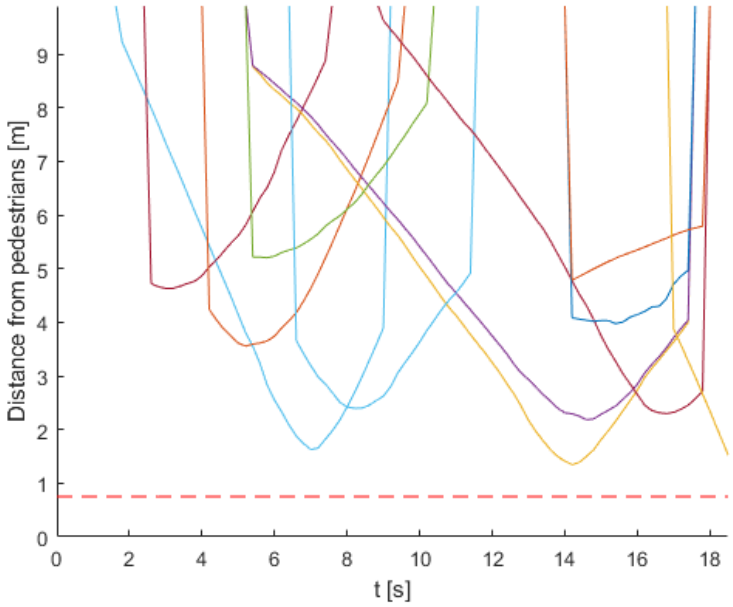



*Figure 6.22: Simulation in a real environment: Distance from pedestrian. The red dashed lines highlight the minimum required distance, while continuous lines report the simulated distances from the different pedestrians.*

*Figure 6.23: Map reporting the fixed obstacles in the scene of Figures 6.18 and 6.19.*

# Chapter 7

# Conclusion and future work

In this thesis a solution for the problem of socially compliant autonomous navigation in human crowded environments is developed. The approach is based on the integration of two layers, with the Local Planner that can correct the trajectory designed by the Global Planner according to the obstacles detected, and the Global Planner that can intervene and replan a new trajectory whenever the Local Planner is not able to solve the trajectory tracking problem. The Global Planner computes a feasible trajectory in the given environment. It generates a feasible geometric path exploiting a Probabilistic Roadmaps (PRM) algorithm, which is then endowed with the time information through a Trapezoidal Velocity Profile based approach. The Local Planner solves a trajectory tracking problem while reactively acting in order to detect obstacles from sensor data and ensure collision avoidance, pedestrian safety and comfort. A novel approach for obstacle avoidance and socially compliant human-robot interactions is introduced, which is based on the introduction of a circular virtual box surrounding the detected obstacle as well as the pedestrian personal spaces. The Local sensor-based Planner is implemented via Model Predictive Control (MPC) casting the motion design into an optimization problem, with operational, collision avoidance, safety and human comfort requirements as constraints. To prove the effectiveness of the solution, some simulations were run where the MPC parameters were calibrated to guarantee satisfactory results in terms of passenger comfort and safety as well as dynamic performance, reaching an acceptable trade-off that takes into account the system limitations. Then the control algorithm was tested in the simple case of trajectory tracking without obstacles, verifying the compliance with the constraints

on velocity (both linear and quadratic) and acceleration. Finally, a set of tests designed to verify the compliance with the human aware pedestrian avoidance requirements were performed in different contexts and situations: a single pedestrian scenario, a simulation in a crowded room, and a real environment based on the ETH Walking Pedestrians (EWAP) Dataset [33]. Pedestrian avoidance, in compliance with velocity and acceleration constraints, is always achieved, and the personal space is fully respected, unless the wheelchair is in critical or overcrowded situations, where a slightly shorter distance is maintained from pedestrians.
Despite the satisfactory results, some issues were identified that may be part of future works: the trajectory generation could be improved by using an asymptotically optimal sampling-based motion planning algorithm for real-time navigation in dynamic environments, such as RRTX [30], instead of PRM that provides only a feasible path. Moreover, another interesting aspect would be to adopt pedestrian stochastic models, in order to allow the wheelchair to predict the formation of groups of people and avoid generating a trajectory too close to areas with a high human-density.

# Bibliography


[1] Inc. © 1994-2019 The MathWorks. *Probabilistic Roadmaps (PRM)*. URL: `https://www.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html?searchHighlight=PRM%5C%20example&s_tid=doc_srchtitle`.

[2] B. Authors: Siciliano et al. *Robotics (Modelling, Planning and Control)*. Chapter 4: Trajectory planning, pages 161-189. Springer, 2009.

[3] H. Bai et al. "Intention-aware online POMDP planning for autonomous driving in a crowd". In: *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), pp. 454–460.

[4] Gianni Ferretti Basak Sakcak Luca Bascetta and Maria Prandini. *Using motion primitives to enforce vehicle motion constraints in sampling-based optimal planners*. 2018.

[5] Luca Bascetta et al. "MPC-based control architecture of an autonomous wheelchair for indoor environments". In: *Control Engineering Practice* (2018).

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] Matthew W. Brault. *Americans With Disabilities: 2010*. URL: `www.census.gov/prod/2012pubs/p70-131.pdf`.

[8] Eugenio Ceravolo and Mauro Gabellone. "Controllo del moto di una sedia a rotelle autonoma mediante tecniche di controllo predittivo". In: *Master thesis, Politecnico di Milano* (Academic Year 2015/2016).

[9] Yu Fan Chen et al. "Socially Aware Motion Planning with Deep Reinforcement Learning". In: (2017).

[10] *CVX: Matlab Software for Disciplined Convex Programming*. URL: http://cvxr.com/cvx/. Version 2.1, December 2018, Build 1127.

[11] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[12] Pierpaolo Durante. "Human-aware planning and control of an indoor autonomous wheelchair". In: *Master thesis, Politecnico di Milano* (Academic Year 2017/2018).

[13] M. Farina, A. Perizzato, and R. Scattolini. "Application of distributed predictive control to motion and coordination problems for unicycle autonomous robots." In: *Robotics and Autonomous Systems* (2015).

[14] Paolo Fiorini and Zvi Shiller. *Motion Planning in Dynamic Environments Using the Relative Velocity Paradigm*. 1993.

[15] Paolo Fiorini and Zvi Shiller. *Motion Planning in Dynamic Environments using Velocity Obstacles*.

[16] D.and Burgard W. Fox and S. Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics Automation Magazine* 4.1 (1997), pp. 23–33.

[17] C.E. Garcia, D.M. Prett, and M. Morari. "Model predictive control: theory and practice - a survey". In: *Automatica* 25.3 (1989), pp. 335–348.

[18] McFadyen BJ Gérin-Lajoie M Richards CL. "The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space." In: *Motor Control* 9 (2005), pp. 242–269.

[19] C. Sprunk H. Kretzschmar M. Spies and W. Burgard. "Socially compliant mobile robot navigation via inverse reinforcement learning". In: *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA) The International Journal of Robotics Research* (2016).

[20] Amy Haddad, Regina Doherty, and Ruth Purtilo. *Health Professional and Patient Interaction*. 2019.

[21] Edward T. Hall. *The Hidden Dimension*. Garden City, N.Y., Doubleday, 1966.

[22] Y. Hwang and N. Ahuja. “A potential field approach to path planning”. In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 23–32.

[23] J.M.Maciejowski and C.Kerrigan. “Soft Constraints and Exact Penalty Functions in Model Predictive Control”. In: *UKACC International Conference (Control 2000)* (2000).

[24] Hadi Kheyruri and Daniel Frey. “Comparison of People Detection Techniques from 2D Laser Range Data”. In: (2010).

[25] Rachel Kirby, Reid Simmons, and Jodi Forlizzi. “COMPANION: A Constraint-Optimizing Method for Person–Acceptable Navigation”. In: *The 18th IEEE International Symposium on Robot and Human Interactive Communication, Toyama, Japan* (2009).

[26] Kavraki L.E. et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[27] S. M. LaValle. “Rapidly-Exploring Random Trees: A New Tool for Path Planning”. In: (1998).

[28] Lalo Magni and Riccardo Scattolini. *Advanced and multivariable control*. Pitagora, 2014.

[29] G. Oriolo, A. De Luca, and M. Vendittelli. *WMR control via dynamic feedback linearization: design, implementation, and experimental validation.* IEEE Transactions on Control Systems Technology. 2002.

[30] Michael W. Otte and Emilio Frazzoli. *RRTX: Real-Time Motion Planning/Replanning for Environments with Unpredictable Obstacles*. WAFR, 2014.

[31] N. J. Nilsson P. E. Hart and B. Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[32] E. C. Pacchierotti. “Evaluation of Passing Distance for Social Robots”. In: *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication* (2006), pp. 315–320.

[33] Stefano Pellegrini and Andreas Ess. *BIWI Walking Pedestrians dataset*. URL: `http://www.vision.ee.ethz.ch/en/datasets/`. (accessed: 01.10.2019).

[34] J. Rawlings and D. Mayne. *Model predictive control: theory and design*. Nob Hill Publishing LLC., 2009.

[35] Bearee Richard. 2005. URL: `https://www.mathworks.com/matlabcentral/fileexchange/15310-gentraj-m?s_tid=prof_contriblnk`.

[36] C. Rösmann, F. Hoffmann, and T. Bertram. "Planning of multiple robot trajectories in distinctive topologies". In: *2015 IEEE European conference on mobile robots* (2015), pp. 1–6.

[37] C. Rösmann et al. "Efficient trajectory optimization using a sparse model". In: *2013 IEEE European conference on mobile robots* (2013), pp. 138–143.

[38] C. Rösmann et al. "Trajectory modification considering dynamic constraints of autonomous robots". In: *2012 German conference on robotics* (2012), pp. 74–79.

[39] Kazuo Sugihara and John Smith. "Genetic algorithms for adaptive motion planning of an autonomous mobile robot". In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'* (1997), pp. 138–143.

[40] Simon X. Yanga and Max Mengb. "An efficient neural network approach to dynamic robot motion planning". In: *Neural Networks* 13 (2000), pp. 143–148.

[41] M. Yoda and Y. Shiota. "The mobile robot which passes a man". In: *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97 SENDAI* (1997), pp. 112–117.

# Appendix A

# Ellipsoidal State Constraints

The idea behind this approach is to determine an ellipsoidal admissible state space, generating a quadratically constrained quadratic optimization problem. First of all, by analyzing the obstacles data gathered from the available equipment, the velocity and the dimensions of the obstacle are determined. Its approximate center can thus be defined, as well as its closest point to the vehicle. The nearest points collected among all the detected obstacles, as well as from the known environment from the plant, will become the vertexes of a polygon (which can be described as a set of linear inequalities) if the resulting area is convex, as shown in Figures A.1 and A.2. If that is not the case, it will require adjustments in order to ensure the admissibility of the optimization problem.
By determining the maximum Area inscribed ellipse, the ellipsoidal admissible state space constraint is defined.

### Maximum Area inscribed ellipsoid

As stated in [6], if we consider the problem of finding the ellipsoid of maximum volume that lies inside a convex set C, the ellipsoid can be parametrized as the image of the unit ball under an affine transformation, i.e., as

$$\xi = \{Bu + d \mid u \leq 1\} \tag{A.1}$$

Considering the case where C is a polyhedron described by a set of linear inequalities:

$$C = \{x \mid a_i^T x \leq b_i, i = 1, ..., m\} \tag{A.2}$$

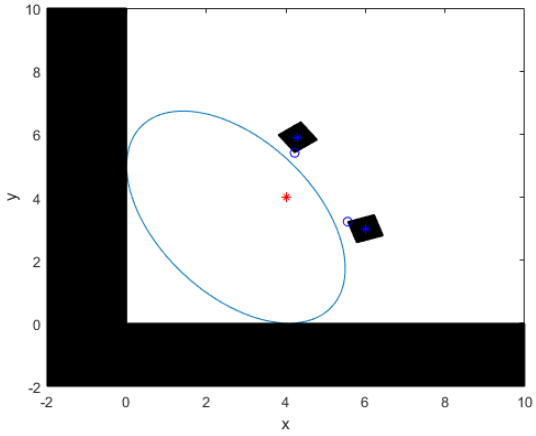



*Figure A.1: Ellipsoidal constraint. The red star represents the vehicle position, while the black areas highlight the walls and detected obstacles positions.*

By rearraging the constraints a convex optimization problem in $B$ and $d$ can be formulated as:

$$\begin{aligned} &\max \log\det B^{-1} \\ &\text{s.t.} \|Ba_i\| + a_i^T d \leq B_i \end{aligned} \tag{A.3}$$

The Matlab implementation for this optimization problem exploited the CVX code from [10]. However, the implementation in the MPC framework resulted to be too computationally intensive, thus not applicable in practice.

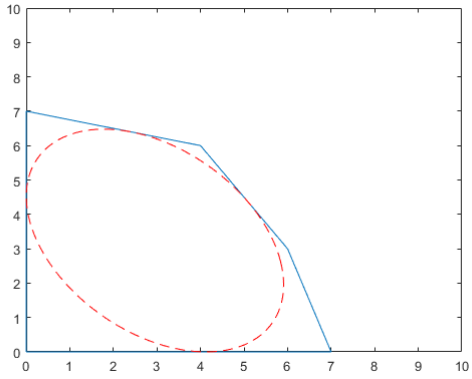



*Figure A.2: Maximum Area inscribed ellipse.*