SAPIENZA
UNIVERSITÀ DI ROMA

Real-time Nonlinear Predictive Control for Human-Quadrupedal Shared Environments

Faculty of Information Engineering, Informatics, and Statistics
Master Degree in Control Engineering

Weihao Wang
ID number 1988339

Advisor
Prof. Gainluca Pepe

Co-Advisor
Prof. Antonio Carcaterra

Academic Year 2022/2023

Thesis defended on 30 January 2024
in front of a Board of Examiners composed by:

Prof. Alessandro De Luca (chairman)
Prof. Claudia Califano
Prof. Andrea Cristofaro
Prof. Francesco Delli Priscoli
Prof. Alessandro Giuseppi
Prof. Leonardo Lanari
Prof. Gianluca Pepe
Prof. Marilena Vendittelli

Real-time Nonlinear Predictive Control for Human-Quadrupedal Shared Environments
Master thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: wang.1988339@studenti.uniroma1.it

To my grandmother Yingchun Song, whose legacy of love and grace forever surrounds us.

Abstract

Navigation and obstacle avoidance are two important maneuvers for autonomous robots, especially in robot-human shared environments, as robots increasingly appear in various fields. This article proposes a real-time nonlinear predictive control framework for a quadruped robot model. This framework enables the automatic generation of collision-free paths while navigating in dynamic environments by solving trajectory optimization problems in a receding-horizon NMPC structure. A novel collision prediction method based on velocity obstacles (VO) is utilized to identify the interaction state, and a Control Policy Decision-maker is defined to choose the correct Optimal Control Problem (OCP). Three different OCPs are formulated to correspond to different maneuvers. The proposed framework is validated through multiple laboratory experiments, demonstrating its capabilities. Additionally, the method is compared with two different avoidance constraints (Artificial Potential Field (APF) based constraint and Velocity Obstacle Potential Field (VOPF) based constraint to investigate efficiency. The results indicate that the framework can operate in dynamic obstacle avoidance scenarios, with a maximum robot velocity of 1.2 m/s and the average computation time less then 9.9 ms. The avoidance motion generated by OCP with VOPF constraint is smoother than OCP with APF.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Gianluca Pepe, for his invaluable guidance, insightful advice, steadfast encouragement and constructive feedback throughout the entire duration of my master's thesis.

I am deeply thankful to Dr. Maicol Laurenza for his patient guidance and his significant contribution on the obstacle avoidance framework code, upon which I developed my work.

A heartfelt thank you to my parents and girlfriend for their unwavering support, understanding, and encouragement all the time. Their unconditional love and belief in me have been a constant source of motivation.

Finally, thanks to myself on this journey.

Contents

Chapter 1

Introduction

Quadrupedal robots have developed significantly over the years, such as Unitree Go1, depicted in Figure 1.1. They are increasingly applied across various industries, including industrial, service, rescue sectors, and so on due to their robust and versatile load-carrying capabilities in uneven terrains and commercial feasibility. However, in many scenarios, quadruped robots need to share environments with humans, especially in settings where humans are moving.

To successfully achieve the goal of safe shared spaces between humans and robots, robots must autonomously and safely plan trajectories in dynamic environments, which also pose the need for obstacle avoidance ability. Robot path planning is a broad subject with numerous methods are currently available[8], such as roadmap techniques[18][3], randomized probabilistic methods[16], artificial potential methods[14], and path planning methods based on the solution of Optimal Control Problem (OCP)[2]. The last method includes Model Predictive Control (MPC) has been increasingly utilized recently not only due to the improvement of computational power with the development of hardware, but also its predictive nature and direct connection to robot dynamics, particularly suitable for scenarios where robots need to navigate among moving obstacles. This property enables robots not only to react responsively to potential collisions with moving obstacles but also to proactively plan their responses.

MPC solves an OCP at each control cycle to ensure the real-time generation

Figure 1.1. Unitree Go1 in a human-robot interaction environment

of motions that satisfy both the kinodynamic feasibility and collision avoidance conditions, using robot dynamics as a predictive model and considers the current state and input constraints. This holds true even in the case of Nonlinear Model Predictive Control(NMPC). NMPC can be applied to robots with more complex dynamics and constrains, presenting a significant advantage over traditional dynamic window approaches[7]. With the recent availability of OCP solvers, such as *OptimTraj*[13], real-time NMPC computation has become possible in practical applications. For these reasons, many studies have adopted MPC to achieve collision-free robot navigation amidst moving human scenarios[25][1][21]. Similar to them, in this article, we also employ the widely used *artificial potential field* (APF) as one of the obstacle avoidance constraints for MPC, with modifications made to this constraint. Moreover, to address the dynamic environment, it is beneficial to formulate multiple control problems with different constraints to ensure navigation safety in robot-human shared space, especially when the situation is critical.

It is worth to mention that all the works mentioned above used distance-based perception, where the obstacle detection and avoidance abilities strongly depend on the prediction horizon and the drive capability of the robot. In practice, a long prediction horizon requires more computational resources, and it is impossible to predict future dynamics arbitrarily far into the future. Therefore, distance-based obstacle avoidance perception may not guarantee safety in situations where the robot detects danger but lacks sufficient actuation power to prevent it. The *Velocity Obstacle* (*VO*) technique extends the analysis to the velocity space of the robot, allowing for the prediction of control for vehicle and robot navigation [15][17]. In this article, we draw inspiration from these works and utilize the *VO* to define a new dynamic perception, not only utilized as a avoidance constraint but also as a means for the robot to perceive obstacles more effectively and identify future interactive states with the robot itself.

Building on the state-of-the-art, the authors propose a real-time NMPC framework to address the challenge of quadruped robot navigation while avoidance among moving humans in the same operate space. Based on this, we define a dynamic perception mechanism using *VO* to identify the predicted interaction state. A selection algorithm is then employed to choose the suitable control problem based on the information of dynamic perception. The nature of *VO* also enables the robot to more effectively respond to rapidly moving human obstacles. Three distinct control problems are defined: navigation, obstacle avoidance, and emergency obstacle avoidance, enhancing the versatility and applicability of our framework across various scenarios and minimizing the risk of collision. For each control problem, we formulate an OCP with different constraints and formulation. Furthermore, we present the principle and practice of the transcription method, which reduces the non-convex optimization problem into Non-linear Programming (NLP) with finite dimensions for efficient computation of numerical solutions. In the end, we conduct multiple simulation experiments using a quadruped robot model in various scenarios to validate the proposed framework's promising capabilities and superiority in real-time conditions for safe autonomous navigation and obstacle avoidance in human-robots shared environments.

The rest of the article is structured as follows: In Chapter 2, we begin by formulating the high-level quadrupedal robot dynamics in Section 2.1.1 and define

the objective function for the navigation task in Section 2.1.2. Following this, we introduce the background of velocity obstacle and the definition of dynamic perception in Section 2.2.1. The Control Policy decision-maker work flow is illustrated in Section 2.2.2 based on criteria information generated from dynamic perception. We outline the formulations of two obstacle avoidance constraints based on APF and *VO* in Section 2.3. Section 2.4.1 provides an introduction to Trajectory Optimization, and in Section 2.4.2, we present the formulations of three NMPC problems. The real-time NMPC workflow is presented in Section 2.5. Simulation results are presented in Chapter 3, while Chapter 4 summarizes the findings and limitations of this work, and offers directions for future research.

Chapter 2

Architecture

2.1 Problem Formulation

The completed architecture proposed by us is illustrated in Figure 2.1. The main objective is to enable obstacle avoidance while navigating the robot within a dynamic environment. In our work, we assume the accurate acquisition of the robot's current state and the velocity and position of all obstacles in the world coordinate system. This assumption is grounded in our focus on the design and feasibility verification of the NMPC architecture in this paper. It allows us to concentrate on the crucial aspects of the NMPC framework, without being constrained by sensor accuracy or robot state estimation limitations, enabling a more in-depth exploration of the potential of NMPC in achieving safe navigation and obstacle avoidance.

2.1.1 Robot Modelling

The Unitree Go1 quadruped robot features a built-in walking controller, allowing us to manage the high-level model instead of dealing with the complexities of the low-level dynamics model. The coordinate systems of the high-level model are illustrated in Figure 2.2, where $(x^{\mathbb{G}}, y^{\mathbb{G}}, z^{\mathbb{G}})$ represent the global coordinate system, and $(x^{\mathbb{B}}, y^{\mathbb{B}}, z^{\mathbb{B}})$ represent the body-fixed coordinate system. The high-level model states are denoted as $x = [p_x^{\mathbb{G}}, p_y^{\mathbb{G}}, v_x^{\mathbb{B}}, v_y^{\mathbb{B}}, \psi]^T$, and the control inputs are represented by $u = [u_{vx}^{\mathbb{B}}, u_{vy}^{\mathbb{B}}, u_\omega]^T$. In this context, we focus solely on the planned walking motion of the robot. The velocities of the robot in the body frame $\mathbb{B}$ simply correspond to velocities with a rotation around the z-axis, with a heading angle ψ. The high-level robot model is defined by the following set of equations (see Equation 2.1):

$$\begin{aligned}
\dot{p}_x^{\mathbb{G}}(t) &= \cos\psi(t) v_x^{\mathbb{B}}(t) - \sin\psi(t) v_y^{\mathbb{B}}(t) \\
\dot{p}_y^{\mathbb{G}}(t) &= \sin\psi(t) v_x^{\mathbb{B}}(t) + \cos\psi(t) v_y^{\mathbb{B}}(t) \\
\dot{v}_x^{\mathbb{B}}(t) &= 1/\tau_{vx}(\kappa_{vx}(u_{vx}^{\mathbb{B}}(t) - v_x^{\mathbb{B}}(t)) \\
\dot{v}_y^{\mathbb{B}}(t) &= 1/\tau_{vy}(\kappa_{vy}(u_{vy}^{\mathbb{B}}(t) - v_y^{\mathbb{B}}(t)) \\
\dot{\psi}(t) &= \kappa_\omega u_\omega(t)
\end{aligned} \tag{2.1}$$

The evolution of velocity terms is modeled as a first-order system linking the body velocities and the control inputs $u_{vx}^{\mathbb{B}} \in \mathbb{R}$ and $u_{vy}^{\mathbb{B}} \in \mathbb{R}$, with gains $\kappa_{vx}, \kappa_{vy} \in \mathbb{R}$

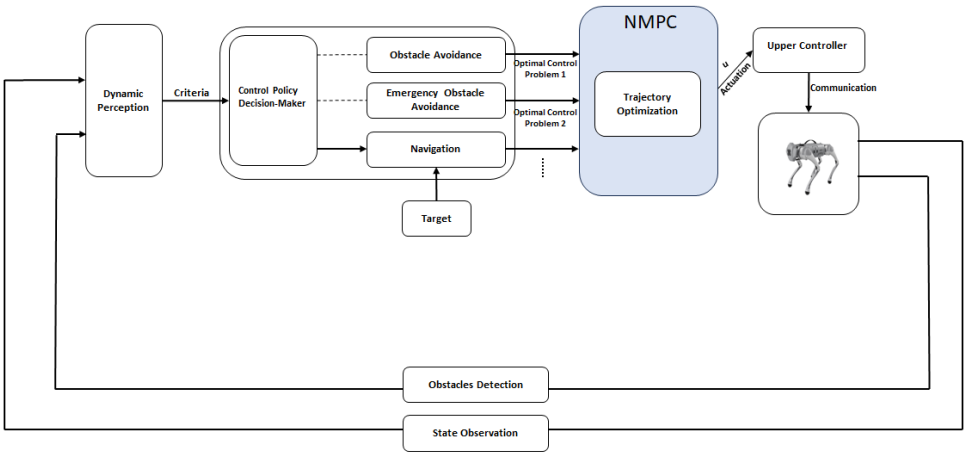


Figure 2.1. Proposed architecture. The dynamic perception provides collision prediction analysis based on the robot state and obstacle information to Decision-maker. The Decision-maker chooses the appropriate maneuver and the corresponding OCP will be solve. The use of NMPC structure allows for the generation of the control commend in each control cycle by solving NLP. Unitree Go1 robot can automatically transfer high-level control commend into low-level commend using the built-in walking controller.

and time constants $\tau_{vx}, \tau_{vy} \in \mathbb{R}$. These terms model the closed-loop behavior of a low-level Unitree Go1 controller, while $\kappa_\omega \in \mathbb{R}$ represents the gain associated with rotational movements.

2.1.2 Objective Function

The objective function defines the control objective; in our case, we aim to make the robot reach the reference state $x_{\text{ref}} = [p^{\mathbb{G}}_{\text{ref}}, v^{\mathbb{B}}_{\text{ref}}, \psi_{\text{ref}}]^T$. This implies that the robot reaches the target location $p^{\mathbb{G}}_{\text{ref}}$ with a specific velocity $v^{\mathbb{B}}_{\text{ref}}$ and heading angle ψ_{ref}. Simultaneously, we seek to minimize actuation efforts. The initial robot state and initial control action are defined as $x(t_0) = x_0$ and $u(t_0) = u_0$, while $t_0 = 0$. Therefore, we formulate the running cost function in the time interval $t \in [0, t_f]$ as follows:

$$J(x(t), u(t)) = \int_0^{t_f} L(x(t), u(t), t)dt \tag{2.2}$$

where

$$\begin{aligned} L(x(t), u(t), t) =& ||p^{\mathbb{G}}_{ref} - p(t)^{\mathbb{G}}||^2_{Q_p} + ||v^{\mathbb{B}}_{ref} - v(t)^{\mathbb{B}}||^2_{Q_v} \\ &+ ||\psi_{ref} - \psi(t)||^2_{Q_\psi} + u(t)^2_{Q_u} \end{aligned} \tag{2.3}$$

here, $Q_p, Q_v \in \mathbb{R}^{2\times2}$, $Q_u \in \mathbb{R}^{3\times3}$, and $Q_\psi \in \mathbb{R}$ are positive definite diagonal weight matrices for position, velocity, heading angle, and input, as well as the terminal time, respectively, while the robot states are function of time. In equation 2.3, the first, second, and third terms denote the state cost, penalizing deviations from a defined state reference x_{ref}. The term $u(t)^2_{Q_u}$ represents the energy consumption

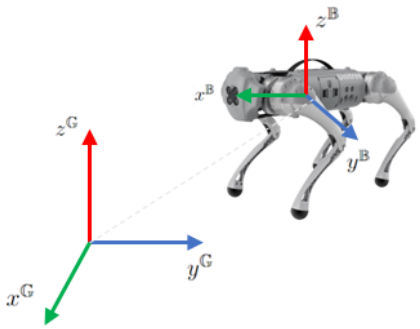


Figure 2.2. Utilized coordinate frames, where $\mathbb{G}$ and $\mathbb{B}$ denote the world and body coordinate frames respectively.

of control actions that we aim to minimize. Additionally, considering the real Go1 robot's dynamic limitations, we impose hard bounds on the control inputs:

$$u_{min} \leq u \leq u_{max} \tag{2.4}$$

2.2 Dynamic Perception

There are many techniques used for detecting and identifying moving obstacles within the MPC framework, such as those discussed in [24, 5, 10]. In this work, we define the dynamic perception based on Velocity Obstacle (*VO*) technique, which allows the analysis to extend into the velocity space of the robot. To define dynamic perception, we introduce the preliminary knowledge of *VO* as detailed in [6].

2.2.1 Definition

We model the Go1 robot and human obstacle as circles, as illustrated in Figure 2.4. This is not a strict limitation, as general polygons can be represented by a combination of circles [20]. Circle A represents the Go1 robot, with its center at the geometric center of the robot and a diameter calculated as $1.3 \times l_{\text{robot}}$. This means the circle increments by 30% from the boundary circle using l_{robot} as the diameter, where l_{robot} is the length of the robot. The circles labeled B represent the human obstacle, with a radius r_o of 0.4 meters. To define *VO*, we first need to map B to the configuration space of A. The configuration space C is a transformation from the physical space, where the agent is of finite size, into another space where the robot is treated as a point, as shown in the Figure 2.3.

Thus, we obtain $\hat{A}$ and $\hat{B}$ by reducing A and enlarging B by r_A to $\hat{B}$, where the r_A is the radius of A. We define the Collision Cone, $C_{A,B}$, as follows:

$$C_{A,B} = \{v_{A,B} | \lambda_{A,B} \cap \hat{B} \neq \emptyset\} \tag{2.5}$$

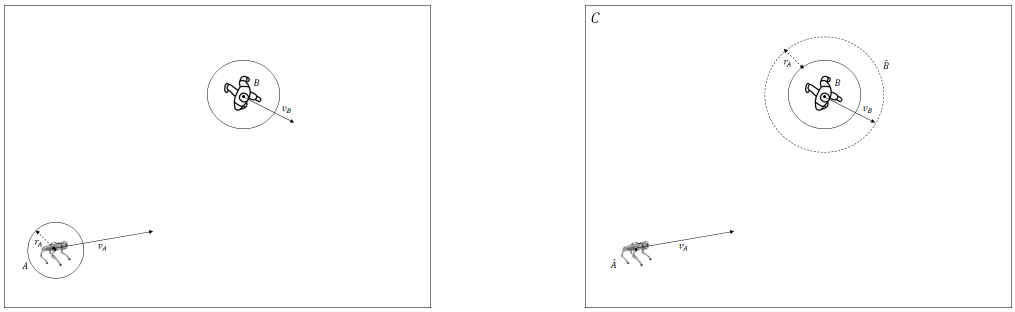


Figure 2.3. The left figure is the physical environment space while the right figure is the corresponding configuration space

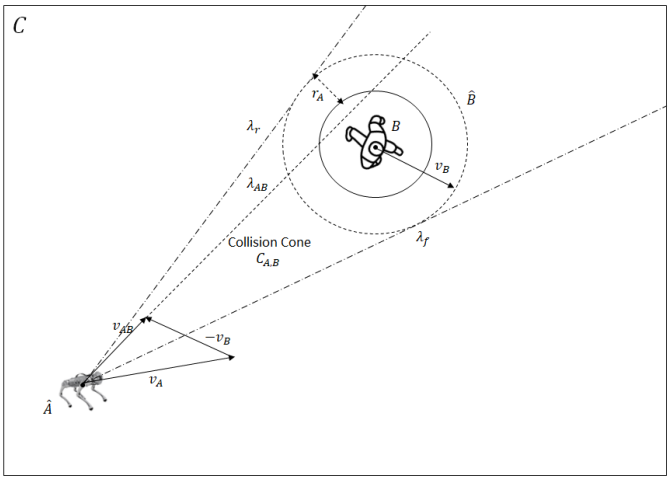


Figure 2.4. The relative velocity $v_{A,B}$ and the *Collision Cone* $C_{A,B}$

$v_{A,B}$ is the relative velocity of $\hat{A}$ with respect to $\hat{B}$. The $C_{A,B}$ is bounded by two tangents, λ_f and λ_r, and $\lambda_{A,B}$ is the extended line of the relative velocity $v_{A,B}$. Any relative velocity $v_{A,B}$ lies inside the Collision Cone will result in a future collision between A and B, as illustrated in Figure 2.4.

It is evident that the Collision Cone is specific to a designated pair of the robot and obstacle. In the case of multiple obstacles, we have to create a corresponding numbers of Collision Cone and check each relative velocity. To reduce complexity, we can add the velocity vector of B, $\overrightarrow{v_B}$, into each velocity vector in $C_{A,B}$; in other word, using the absolute velocity v_A as an equivalent. The VO is shown in the Figure 2.5 and defined as:

$$VO = C_{A,B} \oplus \overrightarrow{v_B} \tag{2.6}$$

here, $\oplus$ represents the *Minkowski* vector sum operator. In the filed of geometry, the *Minkowski* sum of two sets of position vectors, denoted as P_1 and P_2 in Euclidean space, is generated by summing each vector from P_1 with every vector from P_2:

$$P_1 + P_2 = \{p_1 + p_2 | p_1 \in P_1, p_2 \in P_2\} \tag{2.7}$$

if the absolute velocity v_A lies outside the VO, collision-free is ensured in the future;

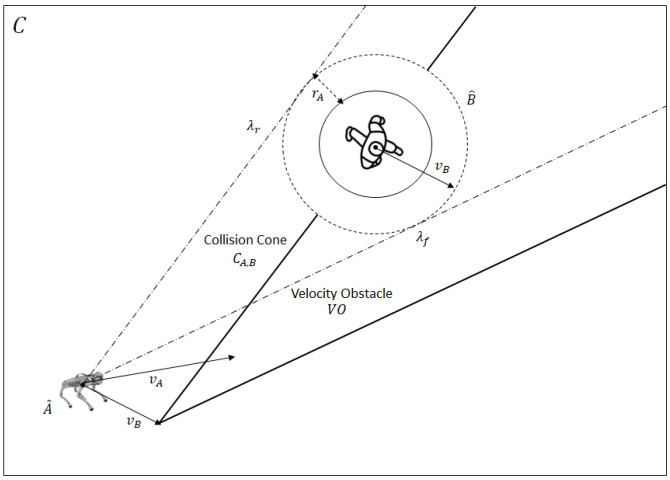


Figure 2.5. Velocity obstacle

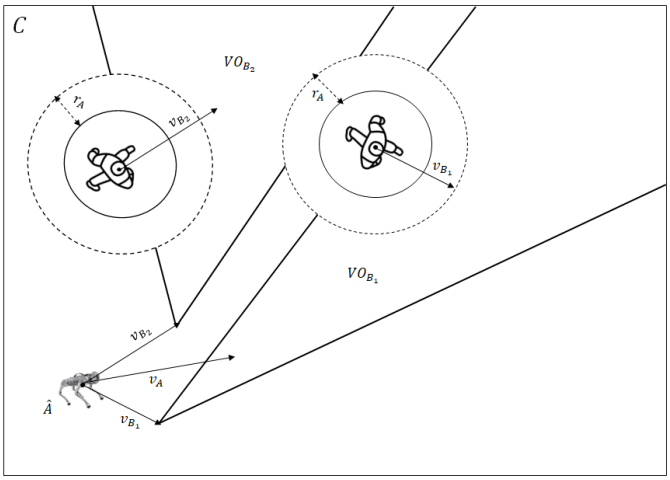


Figure 2.6. *Velocity obstacles for B_1 and B_2*

conversely, a collision is anticipated in the future if v_A lies inside the VO, as shown in the Figure 2.5. To evade multiple obstacles, we consider the union of the individual VO, as expressed in the following:

$$VO_u = \cup_{i=1}^{m} VO_{B_i} \tag{2.8}$$

where m is the number of obstacles, VO_{B_i} is the i_{th} obstacle's VO, illustrated in Figure 2.6.

Since the VO is a linear approximation based on obstacle trajectories, it is employed to predict the interaction state with the robot over a time horizon. However, its accuracy diminishes when obstacles do not move in a straight line. Therefore, we must choose an appropriate time horizon, τ_h, selected based on the dynamic system, which denotes the occurrence of an imminent collision at time t between the obstacle and the robot if $t < \tau_h$. Considering the imminent collision, we adapt the set of VO

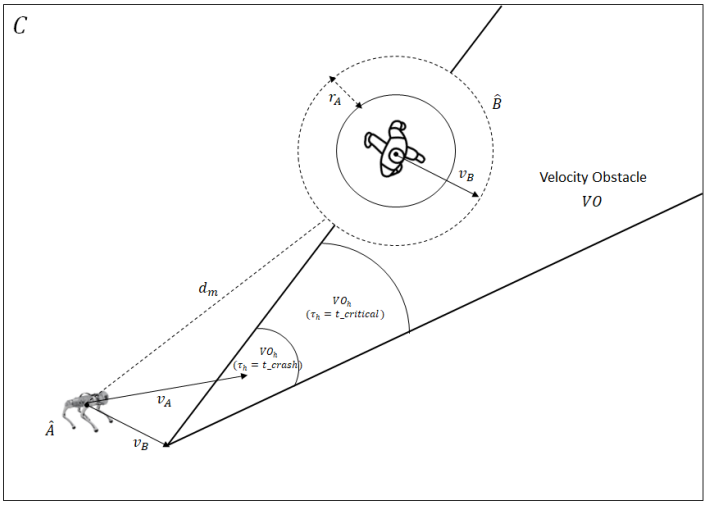


Figure 2.7. The velocity obstacle for different time horizons

by subtracting the elements in the set VO_h, defined as:

$$VO_h = \{v_A | v_A \in VO, ||v_{A,B}|| \leq \frac{d_m}{\tau_h}\} \tag{2.9}$$

showing in the figure 2.7, where d_m is the current relative distance between the robot and obstacle and the set VO_h denotes the robot velocity that would cause a collision over time τ_h, the different collision time horizon generates different size VO_h, while $t_{crash} > t_{critical}$.

Building upon the preliminary knowledge, dynamic perception can be defined as follows: Dynamic perception is a collision prediction method based on VO and distance information. It involves assessing, at each sampling time, whether the absolute velocity of the robot lies inside the union of individual VO generated by different obstacles after subtracting the sets of imminent collision. Simultaneously, it evaluates whether all relative distances between the robot and obstacles are within a safe distance range, constituting distance-based perception. As shown in Figure 2.8, where d_{m_i} denotes the relative distance between the robot and the i_{th} obstacle, and VO_{h_i} denotes the set of imminent collisions generated by different time thresholds t_{crash} and $t_{critical}$ for the i_{th} obstacle. All value of those parameters are defined based on the system dynamics. For consistency, we use VO_u from Equation 2.8 to represent the union of VO_{B_i} while excluding the set of imminent collisions VO_{h_i}.

To apply dynamic perception in practical programs, criteria can be defined based on the interaction information collected from dynamic perception at each sampling time between the robot and human obstacles. This information assists the Control Policy Decision-maker in identifying an encounter situation, which is subsequently addressed through the corresponding OCP. The decision-making process will be detailed after the definition of criteria. The criteria defined based on dynamic perception are as follows:

C_1^{DP}: There will occur a potential collision after τ_h, where $t_{critical} < \tau_h = t_{crash}$, i.e.,

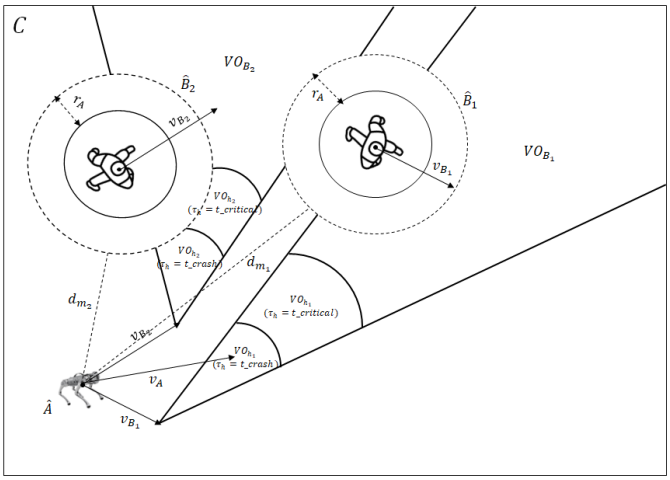


Figure 2.8. Illustration of dynamic perception

$v_A \in VO_u(\tau_h = t_{crash})$.

C_2^{DP}: Any relative distance between the robot and obstacles is less than or equal to the safe moving distance, i.e., $d_{ms} \leq d_{safe}$, where d_{ms} is the smallest distance between the robot and the i_{th} obstacle and d_{safe} is the safe boundary distance defined base on dynamics.

C_3^{DP}: Any relative distance between the robot and obstacles is less than or equal to the critical moving distance, i.e., $d_{ms} \leq d_{ct} < d_{safe}$, where d_{ct} is the dangerous distance defined base on dynamics. Alternatively, there will occur a potential collision after τ_h, where $\tau_h = t_{critical}$, i.e., $v_A \in VO_u(\tau_h = t_{critical})$.

The first criterion, C_1^{DP}, indicates that a collision is predicted to occur in t_{crash} later, given the current state. The second criterion, C_2^{DP}, helps avoid potential collision risks that cannot be solely determined by the VO when the absolute velocity of the robot and humans is parallel. The last criterion, C_3^{DP}, introduces the most critical situation where the distance or predicted collision time is less than critical thresholds. Activating this criterion for any relative distance smaller than this threshold and a coming collision after $t_{critical}$ guarantees the safety requirement.The variable values in these criteria depend on the system dynamics.

2.2.2 Control Policy Decision-maker

In our proposed framework, we have a total of three different control maneuvers: navigation, obstacle avoidance, and emergency obstacle avoidance, each control maneuver corresponds to a control problem. Switching between these control maneuvers relies on the decision-making process of the Control Policy Decision-maker, which depends on the criteria collected through dynamic perception.

The algorithm of the Control Policy Decision-maker, depicted in Figure 2.9, operates as follows: In each sampling time, the Decision-maker assesses the situation

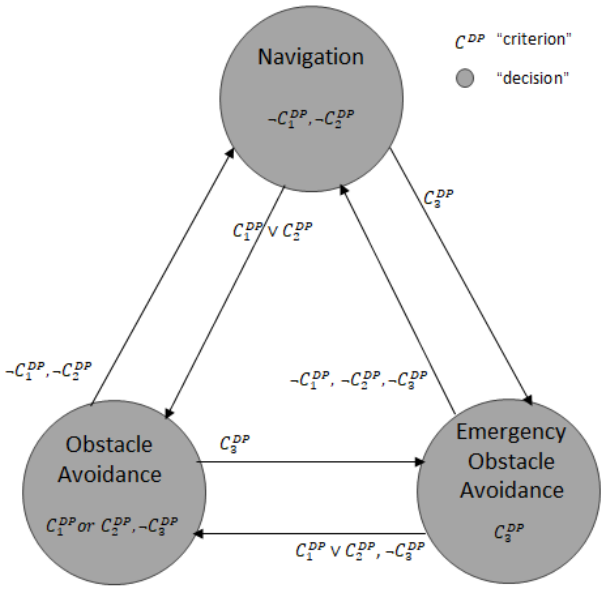


Figure 2.9. Control Policy Decision-maker

by evaluating certain criteria from dynamic perception. If, at the current sampling time, the predicted collision time is larger than t_{crash} ($\neg C_1^{DP}$) and any relative distance between the robot and all obstacles is greater than d_{safe} ($\neg C_2^{DP}$), then our robot will address the navigation control problem. If the the predicted collision time is less than or equal to t_{crash} but larger than $t_{critical}$ (C_1^{DP} and $\neg C_3^{DP}$), or the distance between the robot and the i_{th} obstacle is less than or equal to d_{safe} but greater than d_{ct} (C_2^{DP} and $\neg C_3^{DP}$), then our robot will address the control problem of obstacle avoidance. In any case where the distance between the robot and the i_{th} obstacle is less than or equal to d_{ct} (C_3^{DP}), or there is a potential collision that will happen less than or equal to $t_{critical}$ (C_3^{DP}), the robot will address the control problem of emergency obstacle avoidance.

The synergy between dynamic perception and the Control Policy Decision-maker results in accurate and deterministic actions by the robot, whether for navigation or obstacle avoidance, in dynamic scenarios.

2.3 Obstacle Avoidance Constraints

In this section, obstacle avoidance constraints are introduced to prevent the robot from colliding with human obstacles in its environment. The criteria are employed to determine upcoming control problems through the Control Policy Decision-maker. Subsequently, the OCP is solved with specified constraints corresponding to different control problems.

After defining the objective function, obstacle avoidance constraints are also

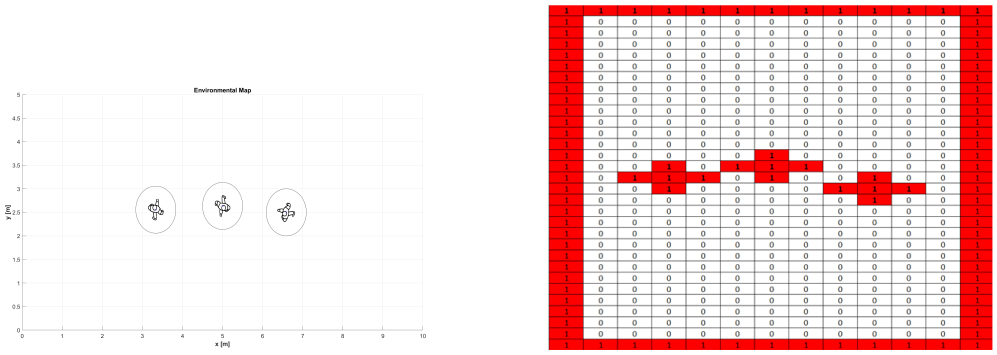


Figure 2.10. The current environment map and its schematic diagram of corresponding Boolean matrix

implemented within the framework of NMPC. In this work, two constraints are applied: one based on Artificial Potential Fields (APF), and the other based on *VO*.

For the constraint based on APF, the approach follows the first method proposed by the authors in the article [19] with some modifications. This method involves generating potential functions using the Gaussian filter. In our implementation, we discretize the two-dimensional environment map into many grids where the robot operates, denoted as an $M \times M$ Boolean matrix I_{obs}. The value of M depends on how many grid points we create. If a grid point is inside a circular area with center $p_{obs}^{\mathbb{G}}$ and radius $r_o + r_a$, this element of the matrix is marked as 1, where $p_{obs}^{\mathbb{G}}$ is the coordinate of the human obstacle geometric center point in the world frame and r_a is the radius of robot circular area; otherwise, it is marked as 0. Moreover the room boundaries are also considered as obstacles, where is marked as 1 as well. Therefore, an occupation environment matrix can be obtained that contains the information of obstacle and room boundary positions, as illustrated in Figure 2.10, where the left figure is the environment map while the right figure is the corresponding Boolean matrix I_{obs}.

The Gaussian filter is then used to create a smooth potential field. The 2D Gaussian kernel function is given by:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.10}$$

where (x, y) are the distance from the origin in the coordinate, and σ is the standard deviation, controlling the spread of the filter.

To generate potential field based on the obstacle positions, we can create $2D$ Gaussian distribution at each element of I_{obs}, and this is achieved by discrete convolution. The general expression of a $2D$ discrete convolution is

$$g(x, y) = \omega * f(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \omega(i, j) f(x - i, y - j) \tag{2.11}$$

where $g(x, y)$ is the output function, $f(x, y)$ is the original function, ω is another original function (in some applications it called filter/kernel). This expression means that each element of the $g(x, y)$ is a function of the nearby elements of the $f(x, y)$. In

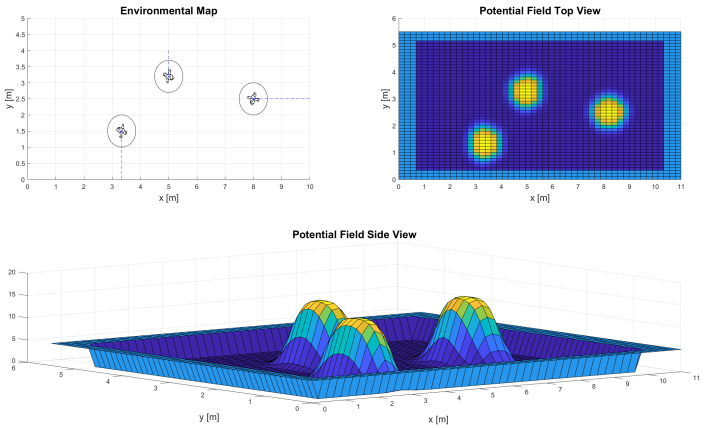


Figure 2.11. The sub-figure in the top-left represents the trajectories and positions of three human obstacles in the environment map. The sub-figure in the top-right displays the top view of the corresponding potential field of that environment map, while the sub-figure below represents the side view of the corresponding potential field of that environment map. The size and dispersion of the potential field can be adjusted by tuning the parameters of the Gaussian distribution.

our formulation the function ω is the Gaussian filter function G we defined. So the $2D$ discrete convolution of the potential filed environment map with the Gaussian kernel is expressed as:

$$P_{APF}(x,y) = G * I_{obs}(x,y) = \sum_{i=-M}^{M} \sum_{j=-M}^{M} G(i,j) I_{obs}(x-i, y-j) \tag{2.12}$$

P_{APF} represents the potential field map function, where $I_{obs}(x,y)$ is the occupation environment matrix in the environment map, and $(x-i, y-j)$ indicates the relative coordinates between the Gaussian distribution function and the environment map. This function denotes the potential value at a given point in the environment. The closer the point is to an obstacle or the boundary, the larger the value, following a Gaussian distribution. Conversely, the farther the point is from an obstacle, the smaller the value, approaching zero.

To ensure consistency in size between the input occupancy grid and the output grid matrix, we need to pad the input array such that the output size is the same as the input size. For better illustration, the visualization of this function is shown in Figure 2.11. This function provides smooth, globally influential potential fields, tuning adaptability with mathematical simplicity and efficiency.

Similar to the constraint based on APF, the VO-based potential field (VOPF) constraint follows a comparable idea with the first constraint. However, in this case, the velocity map is discretized instead of the environment map. A Boolean matrix I_{vo} of dimensions $M \times M$ is created to represent the grids of the velocity map. If the grids are inside the area of VO_u, the corresponding element of the matrix is set to 1. We then construct the Gaussian distribution on the I_{vo} as same as before;

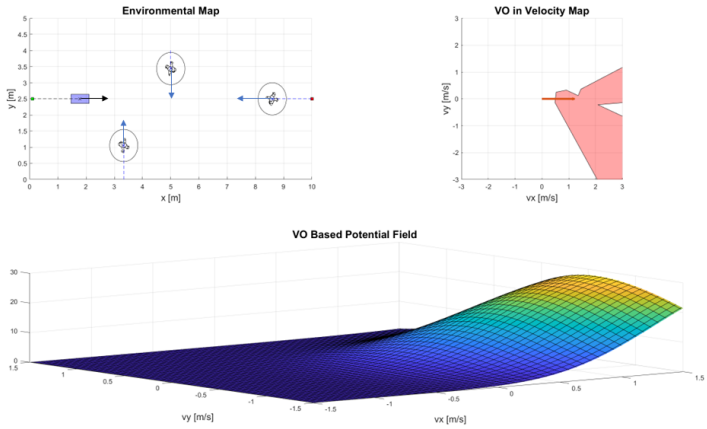


Figure 2.12. The environment map and the corresponding VOPF based on VO

otherwise, the potential is set to 0. Building on this, the formula of VOPF function can be expressed as:

$$P_{VO}(v_x, v_y) = G * I_{vo}(v_x, v_y) = \sum_{i=-M}^{M} \sum_{j=-M}^{M} G(i,j) I_{vo}(v_x - i, v_y - j) \tag{2.13}$$

where the coordinate is the velocity space coordinate. In practical applications, when a collision happens between a robot and an obstacle, the generated VO is set as a circle with the current velocity of the obstacle as its center.

Figure 2.12 represents the VOPF generated according on the VO_u. The sub-figure in the top-left corner illustrates the environment map, where the blue rectangle represents the robot, and the arrows attached on the robot and obstacles denote the velocity vectors. At the moment shown in this Figure, the corresponding VO is shown in the top-right sub-figure, where the red polygon represents the VO_u, and the orange color arrow indicates the robot's current velocity in the velocity map. The sub-figure below represents the VOPF generated at the current moment.

In our work, both mentioned obstacle avoidance constraints are applied separately to test and compare their effects. To incorporate these constraints fit our framework, we reformulate them as inequality constraints for the NMPC framework. The final expressions of constraint can be stated as follows:

$$P_{APF}(p^{\mathbb{G}}_{x,k}, p^{\mathbb{G}}_{y,k}) \leq 0 \tag{2.14}$$

$$P_{VO}(v^{\mathbb{B}}_{x,k}, v^{\mathbb{B}}_{y,k}) \leq 0 \tag{2.15}$$

2.4 Optimization

This section explores the methodologies employed for optimization, specifically focusing on trajectory optimization and how to formulate our three NMPC control problems with trapezoid collocation method.

Figure 2.13. Contrasting an open-loop solution (optimal trajectory) with a closed-loop solution (optimal policy, the open-loop solution (depicted on the left) for an optimal control problem comprises a sequence of controls, denoted as $u(t)$, guiding the system from a singular starting point A to the designated destination point B. Conversely, the closed-loop solution provides controls, denoted as $u(x)$, capable of guiding the system from any point within the state space to the same destination point B.

2.4.1 Trajectory Optimization

Given the objective and the constraints of the system from Section 2.1.2, the OCP to be solved at each time instant is formulated with

$$\min_{x(.),u(.)} \quad J(x(t),u(t)) \tag{2.16a}$$

$$\text{subject to} \quad \dot{x}(t) = f(t,x(t),u(t)) \tag{2.16b}$$

$$u_{min} \leq u(t) \leq u_{max} \tag{2.16c}$$

$$x_{min} \leq x(t) \leq x_{max} \tag{2.16d}$$

where 2.16c and 2.16d represent continuous bounds, and $x(.), u(.)$ are decision variables that are functions to be computed. The OCP of scheme 2.16 leads to an high-dimensional problem.

In addressing the challenges posed by high-dimensional systems, the trajectory optimization method is employed to simplify the optimization problem. Instead of attempting to solve for the optimal feedback controller across the entire state space, the focus shifts to discovering open-loop solutions to an optimal control problem, commonly referred to as the optimal trajectory[12]. An illustration comparing closed-loop and open-loop solutions is provided in Figure 2.13.

The approaches for solving trajectory optimization problems can generally be categorized as direct or indirect. In this work, direct methods are employed, which is the extension to the simultaneous methods and it is widely used in practical applications. The main idea of direct methods is to discretize the trajectory optimization problem itself. This typically involves transforming the original trajectory optimization problem into a NLP, this process commonly known as *transcription*.

In general, direct transcription methods has the capability to discretize a continuous trajectory optimization problem by representing all continuous functions in the problem statement as polynomial splines. In our work, we employ the trapezoidal collocation method, the trapezoidal method is a numerical integration method used to approximate the definite integral of a function. It works by approximating the area under the curve by dividing the interval into small trapezoids, and is utilized to enforce the dynamics constraints at the collocation points. This method involves

discretizing the trajectory optimization problem using polynomial splines, making it particularly effective for our purposes.

To solve trajectory optimization problems using the trapezoidal collocation method, the trajectory needs to be discretized as follows:

$$\begin{aligned} t &\rightarrow [t_0, t_1, ..., t_N] \\ x(t) &\rightarrow [x_0, x_1, ..., x_N] \\ u(t) &\rightarrow [x_0, x_1, ..., x_N] \end{aligned} \tag{2.17}$$

we discretize the time horizon $[t_0, t_h]$ into N intervals defined in the equaltion 2.18, where t_h is the control time horizon, and define states and actions at those instants. The system dynamics of the legged robot is discretized using direct collocation methods, which is introduced in 2.21. This discretized model serves as the *prediction model* for the NMPC. Prediction occurs with a receding horizon, considering a designated number of future steps. This designated number is referred to as the prediction horizon, denoted as H, while N and δ represent the number of intervals in the prediction horizon and the sampling interval respectively, where

$$N = \frac{H}{\delta} \tag{2.18}$$

to ensure the capture of collisions between two objects at the sampling interval, δ is determined as:

$$\delta = r_o/(v^{\mathbb{B}}_{max} + vo_{max}) \tag{2.19}$$

where r_o denotes the obstacle radius (in our case the radius are same, chose the smallest radius if they are different), $v^{\mathbb{B}}_{\max}$ represents the maximum velocity of the robot, and $vo_{\max}$ is the maximum velocity of obstacles in the defined scenarios.

In the next step, we convert our robot continues dynamics in Equation 2.1 into the collocation constrants, where is trapezoid quadrature used, the general predictive form as shown below, where $h_k = (t_{k+1} - t_k)$:

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), u(t)) \\ &\approx \\ x_{k+1} &= x_k + \frac{h_k}{2}(f_{k+1} + f_k) \end{aligned} \tag{2.20}$$

Since our objective function is an integral expression, the idea of the trapezoidal collocation method is to convert the objective function by approximating the continuous integral form as a summation form between each collocation point. This yields the original objective function 2.2 in the following equation:

$$\begin{aligned} &\int_0^{t_f} L(x(t), u(t), t)dt \\ &\approx \\ &\sum_{k=0}^{N-1} \frac{1}{2} h_k (L_k + L_{k+1}) \end{aligned} \tag{2.21}$$

where $L_k = L(x(t_k), x(t_k), u(t_k))$. Thus, the optimization problem posed in 2.16 is expressed in NMPC problem with:

$$\min_{\substack{x_0 \dots x_N \\ u_0 \dots u_N}} \quad \sum_{k=0}^{N-1} \frac{1}{2} h_k (L_k + L_{k+1}) \tag{2.22a}$$

$$\text{subject to} \quad \frac{h_k}{2}(f_k + f_{k+1}) = x_{k+1} - x_k \tag{2.22b}$$

$$u_{min} \leq u_k \leq u_{max} \tag{2.22c}$$

$$x_{min} \leq x_k \leq x_{max} \tag{2.22d}$$

$$k \in 0 \dots (N-1)$$

The solution to the NMPC problem consists of the set of control inputs and the system dynamics at each collocation point. To approximate the control trajectory and system dynamics as functions, we can use a spline. In the trapezoidal collocation method, the knot points of the spline coincide with the collocation points. First, to construct the function of the control trajectory, we create a piecewise linear function within each time interval $t \in [t_k, t_{k+1}]$, and define $\tau = t - t_k$, thus, $u(t)$ is a linear spline:

$$u(t) \approx u_k + \frac{\tau}{h_k}(u_{k+1} - u_k) \tag{2.23}$$

The difference between state trajectory and control trajectory is that state trajectory is a piecewise quadratic function. The reason for this lies in the trapezoidal collocation method, where the system dynamics are approximated by a linear function between any two collocation points over the segment interval $t \in [t_k, t_{k+1}]$, as shown below:

$$f(t) = \dot{x}(t) \approx f_k + \frac{\tau}{h_k}(f_{k+1} - f_k) \tag{2.24}$$

Since the system state function is what we need instead of the system dynamics, the system state function can be obtain by integrating both side of equation 2.24:

$$x(t) = \int \dot{x}(t) d\tau \approx c + f_k \tau + \frac{\tau^2}{2h_k}(f_{k+1} - f_k) \tag{2.25}$$

The constant c can be solved by using the value of the state at the boundary $\tau = 0$. Therefore, the final expression of the state function is:

$$x(t) \approx x_k + f_k \tau + \frac{\tau^2}{2h_k}(f_{k+1} - f_k) \tag{2.26}$$

2.4.2 NMPC Scheme

In this section we focus on the formulations of the three different NMPC problems: navigation, obstacle avoidance and emergency obstacle avoidance.

Since the trajectory optimization problems are formulated over a finite horizon, each optimization can be viewed as an inference about the next N time steps. thanks to the characteristics of MPC, it is a good idea that we continue solving for an N step and let the dynamics evolve for one step and repeat. Figure 2.14 illustrates the working principle of trajectory optimization as a feedback policy in receded horizon MPC.

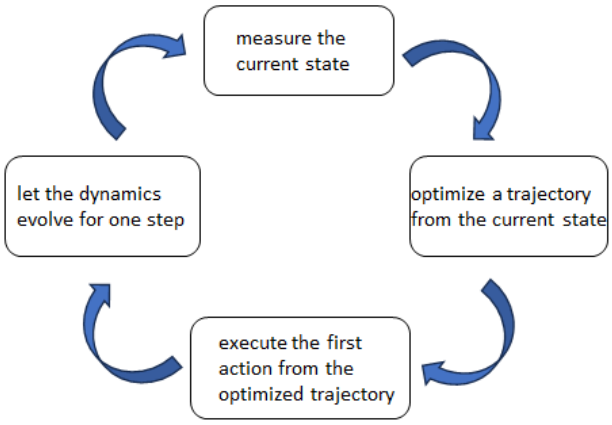


Figure 2.14. Trajectory optimization as a feedback policy within MPC framwork.

Building on this property, we define the three NMPC problems as follows. In the first control problem, when the robot is in a navigation state identified by the decision-maker, its primary objective is to reach the target point as quickly as possible. This leads to the formulation of the NMPC problem where the optimization focuses on solving the objective function subject to robot dynamics and bounds. We have presented this NMPC problem in the last subsection already (2.22a-2.22d).

The seconed control problem is, when the Decision-maker determines that the current interaction state between the robot and the obstacles is "obstacle avoidance," it implies that the robot needs to generate a safe trajectory by solving a constrained OCP. This leads to a NMPC problem that incorporates both the objective function and obstacle avoidance constraints. The formulation can be expressed as:

$$\min_{\substack{x_0 \dots x_N \\ u_0 \dots u_N}} \quad \sum_{k=0}^{N-1} \frac{1}{2} h_k (L_k + L_{k+1}) \tag{2.27a}$$

$$\text{subject to} \quad \frac{h_k}{2}(f_k + f_{k+1}) = x_{k+1} - x_k \tag{2.27b}$$

$$u_{min} \leq u_k \leq u_{max} \tag{2.27c}$$

$$x_{min} \leq x_k \leq x_{max} \tag{2.27d}$$

$$\textit{obstacle avoidance constraints} \tag{2.27e}$$

$$k \in 0 \dots (N-1)$$

here, the term *obstacle avoidance constraints*, as presented in 2.3, contributes to the avoidance constraint for the OCP, thereby generating safe motion by solving this constrained control problem.

In the emergency obstacle avoidance control problem, the priority shifts to ensuring the safety of human obstacles as fast as possible instead of approaching to target. The resulting NMPC problem solely includes obstacle avoidance constraints without minimizing the objective function, subject to dynamic constraints and

bounds, aims to find feasible and safety control sequence under the critical situation. The formulation is presented in the following equation:

$$
\begin{aligned}
& find && u_k, x_k && \text{(2.28a)}\\
& \text{subject to} && \frac{h_k}{2}(f_k + f_{k+1}) = x_{k+1} - x_k && \text{(2.28b)}\\
& && u_{min} \leq u_k \leq u_{max} && \text{(2.28c)}\\
& && x_{min} \leq x_k \leq x_{max} && \text{(2.28d)}\\
& && \textit{obstacle avoidance constraints} && \text{(2.28e)}\\
& && k \in 0...(N-1)
\end{aligned}
$$

This formulation ensures that the robot generates a trajectory for emergency obstacle avoidance faster by simplifying the optimization problem. Since the goal of emergency obstacle avoidance is to find a collision-free solution as fast as possible, computation time becomes an important metric when solving this NMPC problem. Generally, the grid number N is set along the trajectory based on the dynamic system. A small number of collocation points can result in fast computation of cost but may increase numerical errors. On the other hand, a large number of collocation points can provide an accurate solution but require higher computational effort. Therefore, when solving the emergency obstacle avoidance NMPC problem, an appropriate number of N can be considered to reduce the complexity of the optimization problem while reducing transcription accuracy to an acceptable range based on dynamics.

2.5 Implement

To implement our proposed NMPC framework and simulate the computational consummations, the designed real-time workflow is illustrated in the diagram 2.15 Firstly, we perform initialization. Based on the robot's initial state x_0 and objective function, we calculate the initial control outputs and system state sequence within the NMPC horizon off-line. The terms $u_{out|x_0}$ and $x_{out|x_0}$ denote the initial outputs based on the initial robot state x_0, serving as the initial control input for the system. Since we are uncertain about the duration required for the controller to compute a solution, we estimate a time Δt_c^* as a predicted computation time while Δt_c is the real computation time that can be measured after the computation cycle, and $t_c = 0$ where t_c is the time when the solution becomes available. The estimation idea is, at each NMPC control cycle, the prediction control output should be calculate based on the estimated future system state, as shown as in the Figure 2.16, where x^* is the estimated system state while x is the real system state that can be observed.

Subsequently, we check whether t_c is less than or equal to t, where t is the platform simulation time (the real running time). if t_c is less than or equal to t, it indicates the need to compute the predicted control output for using in the future (because the prediction horizon is finite). Given our lack of knowledge about the robot's precise future state, we employ a linear approximation to estimate the robot's state as

$$x_1^* = x_0 + Ax_0\Delta {t_c}^* + Bu_0 \tag{2.29}$$

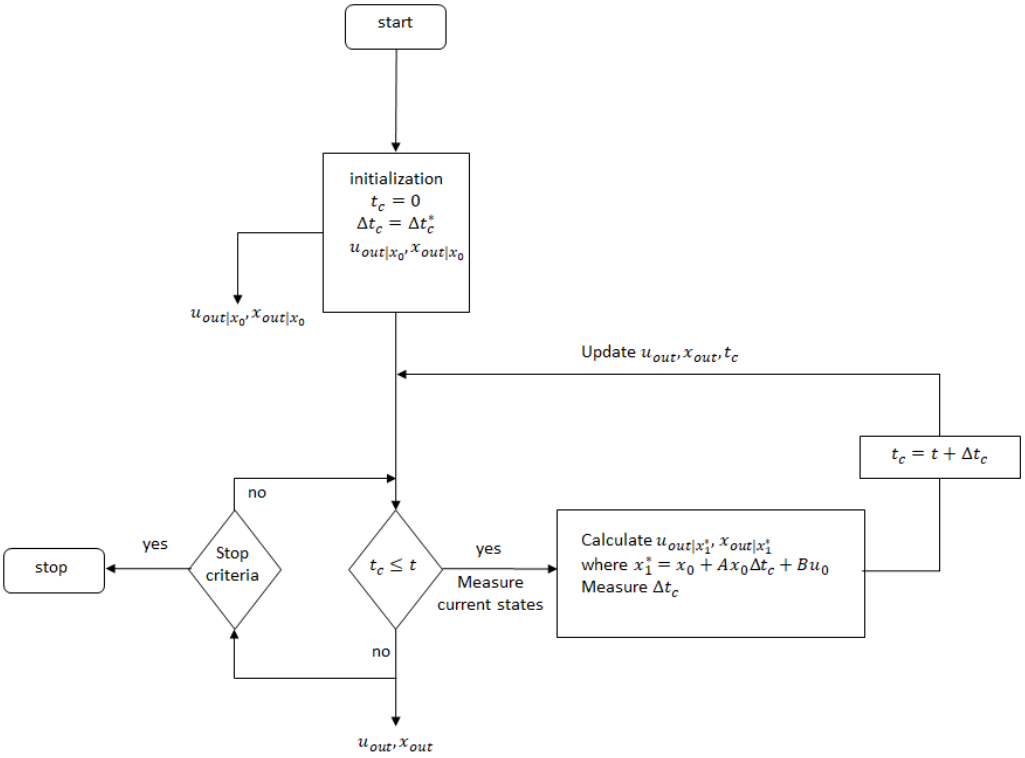


Figure 2.15. Real-time workflow

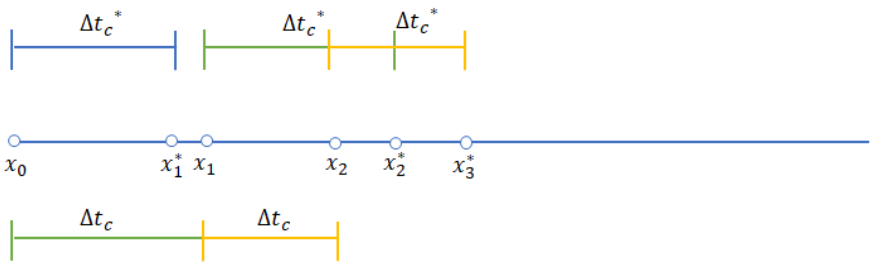


Figure 2.16. The estimated states base on the computational consumption

This approach means that we estimate the future state x_1^* obtained by evolving linear model over the assumed time duration $\Delta {t_c}^*$. The new computed output $u_{out|x_1^*}, x_{out|x_1^*}$ are computed based on the anticipated future state of robot x_1^*, where the x_0 is measured when $t_c \leq t$ is true. After computation, we record the computation time Δt_c and update t_c by $t_c = t + \Delta t_c$, storing this output for the subsequent control cycle. If t_c is greater than t, the system dynamics still be evaluated with control output from the previous control horizon, if t_c is less than or equal to t, means the robot dose not have a available control outputs after current control horizon, the framework has to compute a new predicted control based on the estimated state obtaining from 2.29, the $\Delta {t_c}^*$ is the measured computation time of last control circle.

The control loop repeats until the termination criterion is met. The design of this real-time workflow aims to align the performance exhibited by the robots in the simulation environment more closely with the hardware and software capabilities of the actual robot since the prediction horizon can not be infinite. This is done to enhance the reliability and authenticity of the conclusions drawn from simulations.

The solution of those NMPC problems require NLP solver to deduce the optimal solution for the NMPC problem. the generic NLP framework for the direct method is expressed with

$$\begin{aligned} \min_{z} \quad & f(z) \\ \text{subject to} \quad & g(z) \leq 0 \\ & h(z) = 0 \\ & z_{low} \leq z \leq z_{up} \end{aligned} \tag{2.30}$$

where z is the optimization which depends upon the direct method formulation utilized. This work focuses on the implementation NLP instead of solve NLP itself, a detailed discussion can be found in [4].

Most NLP solvers require an initial guess for the optimization process. A good initial guess often leads to accurate results, while poor guesses can potentially cause the solver to become 'stuck'. For efficient and convenient practical applications, it is reasonable to use the outputs from the previous horizon as the initial guess for the next NLP. This is particularly relevant for navigation problems. In the cases of obstacle avoidance and emergency obstacle avoidance problems, we formulate the initial guess using the following steps: 1) at each control loop, we measure the current robot states. 2) linear interpolation between current states and the target states. 3) initialize with zero control effort. 4) repeat until the control problem turn back to navigation problem.

Additionally, providing gradients of the objectives and constraints to the solver is efficient that for smooth problems the solvers are faster and more robust when exact gradients are provided. In our work, we provide the gradients for the direct transcription methods that provide the gradients for each constraint individually.

Chapter 3

Simulations

3.1 Experimental Set-up

The proposed real-time NMPC architecture was implemented in Matlab and Simulink. We utilized Matlab and Simulink for co-simulating real-time human-quadrupedal shared environments. The Matlab library *OptimTraj*[13]had been employed, which offers four different methods for trajectory optimization: trapezoidal direct collocation, Hermite–Simpson direct collocation, fourth-order Runge–Kutta direct multiple shooting, and Chebyshev orthogonal collocation. In all simulation experiments, the trapezoidal direct collocation method was chosen. For the NLP problem, we employed another Matlab library, *FMINCON*, where the *interior-point* and *sqp* (Sequential Quadratic Programming) algorithms are available for solving the NLP. The solution is returned to the user at each grid point along the trajectory, and we use spline function to approximate the control and state trajectory (which can be done by the Matlab function *interp*). Additionally, the maximum solver iteration count is set to 15.

The parameters of the quadruped robot used in simulation are consistent with those of the Unitree Robot Go1, with a robot length l_{robot} of 0.645 meters and a robot width w_{robot} of 0.28 meters. The circle representing the robot's boundary has a center of circle located at the geometric center of the robot, and the circle has a radius $r_a = \frac{1.3 * l_{robot}}{2}$ as mentioned in Section 2.2. Similarly, circular representations of human obstacles have their centers at the top view center of an adult human and a radius of 0.4 m. We disregard the height of both the robot and the human as this simulation is conducted in a $2D$ plane. According to the robot physical characteristics, the moving velocity and the control inputs are bounded as, $-0.12 \leq u_{vx}^{\mathbb{B}} \leq 1.2\ m/s$ and $-0.012 \leq u_{vy}^{\mathbb{B}} \leq 0.012\ m/s$, $-0.12 \leq u_{vx}^{\mathbb{B}} \leq 1.2\ m/s$ and $-0.012 \leq u_{vy}^{\mathbb{B}} \leq 0.012\ m/s$, respectively.

The sampling time δ of both robot and obstacles is 0.15 s under the setting that the maximum robot velocity is 1.2 m/s, while the time horizon H is configured as 2.5 seconds. All simulations were conducted on an AMD Ryzen 7 5800H 16-core processor platform. Moreover, we apply a linear approximation of the predicted obstacle positions at each sampling time, where the time horizon of predicted position is represented as $H * 1.3$. This prediction of position helps the robot in proactively avoiding potential collisions when solving the obstacle avoidance problem with the

APF-based constraint. For the following experiments, the corresponding model parameters presented in equation 2.1 are chosen as $k_{vx}, k_{vy}, k_{\omega} = 1$ to align with the response of a low-level controller operating on our robot model, while configuring the time constant terms as $\tau_{vx}, \tau_{vy} = 0.4$. The weights in the objective function 2.3 are chosen as $Q_p = diag(50, 50), Q_v = diag(50, 50), Q_u = diag(3, 3, 3)$ and $Q_{\psi} = 50$.

3.2 Experimental Evaluations

The experiments take place in an environment represented by a rectangular $2D$ space with dimensions 20 m in length and 12 m in width. The robot's initial state is $x_0 = [1, 1.5, 0.7, 0, 0]^T$, while the target state of the robot is $x_G = [16.35, 9.6, 0, 0, 45]^T$, indicating that the robot needs to reach the target point $(16.35, 9.6)$ with a specific heading angle $\psi = 45°$. Non-linear motion trajectories for all human obstacles are generated using the Matlab tool *Driving Scenario Designer* by creating a sequence of viapoints. These non-linear motions are designed to test the capabilities of the proposed framework.

To validate the capabilities of the architecture proposed in our work, it is assessed in a series of scenarios involving different dynamic environment settings. Scenario 1 depicts a situation where a robot is going to meet with a human. In this scenario, the robot is tasked with reaching a target state while the human moves towards the robot at varying speeds, comes to a sudden slow down, and then resumes movement. In Scenario 2, the robot has to reach a target state while simultaneously avoiding two human obstacles. One human moves towards the robot in a constant velocity, while the other moves in an 'S' shape with varying speed over time. Scenario 3 represents the most challenging scenario, incorporating three moving human obstacles. Two of them follow the same setup as in Scenario 2, while the third moves in the normal direction of the line connecting the robot to the target position, then making a counterclockwise movement after crossing that line. The details of the scenarios are presented in Table 3.1.

Scenario	1	2	3
Number of obstacles	1	2	3
Maximum obstacle v	0.9	0.86/ 0.8	0.86/ 0.8/ 0.64
Average obstacle v	0.68	0.84/ 0.4	0.86/ 0.4/ 0.53

Table 3.1. A table depicting the max and average obstacle velocities (in m/s) in the different scenarios and the number of obstacles.

The two obstacle avoidance constraints, APF and VOPF, have been applied individually in these three scenarios, and each repeated 5 times, to evaluate the capabilities of our NMPC framework. Furthermore, the effects of these two constraints are compared. The simulation results are presented in Table 3.2, showcasing the time taken to reach the target point, the average traveling speed of the quadruped robot, the average computation time of NMPC, and the success rate (considering collision occurrences as failures). These values are averaged over 5 runs in each simulation scenario. For a visual representation of the simulation results, a video clip is accessible at `https://github.com/Wealhour/`

`Real-time-Nonlinear-Predictive-Control-for-Human-Quadrupedal-Shared-Environments`.

Scenario	Selected constraint	Arriving time (t)	Average velocity (m/s)	Average computation time (ms)	Successful times
1	APF	18.33	0.848	9.8	5/5
	VOPF	18.81	0.874	9.9	5/5
2	APF	19.14	0.832	9.8	5/5
	VOPF	18.62	0.850	9.9	5/5
3	APF	19.26	0.873	9.9	4/5
	VOPF	18.53	0.822	9.9	5/5

Table 3.2. Performance data across all experiments

In the case of scenario 1, the experimental results applying two different constraints exhibit similar performances, as shown in Table 3.2. During the experiments, the correct switching of the control policy based on dynamic perception is also observed, as depicted in the Figures 3.1 and 3.2, the combination of blue circular rings in front of the human obstacle represents predictions of the obstacle positions. The red dashed line in front of the robot represents the generated robot position predictions obtained by solving OCP, and the blue dashed line and the gray dashed line represent the actual trajectories of the robot and the obstacles, respectively. The green bar in the top left corner indicates the current velocity of the robot.

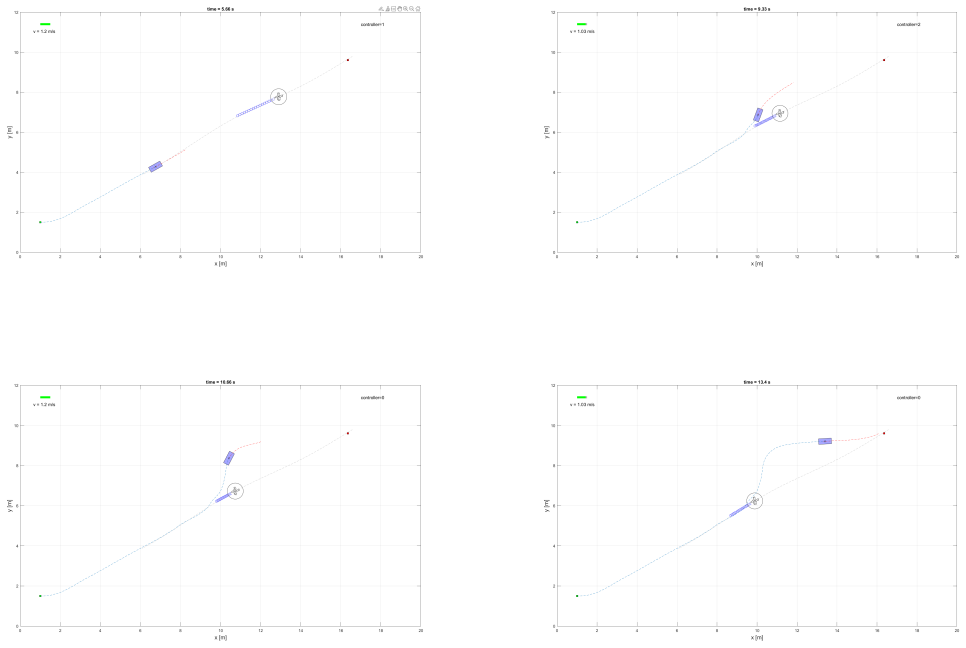


Figure 3.1. Simulation snapshots of scenario 1 using the APF obstacle avoidance constraint.

we can observe that despite maintaining a certain distance from human obstacles, the robot switches to the obstacle avoidance maneuver (indicated as "controller

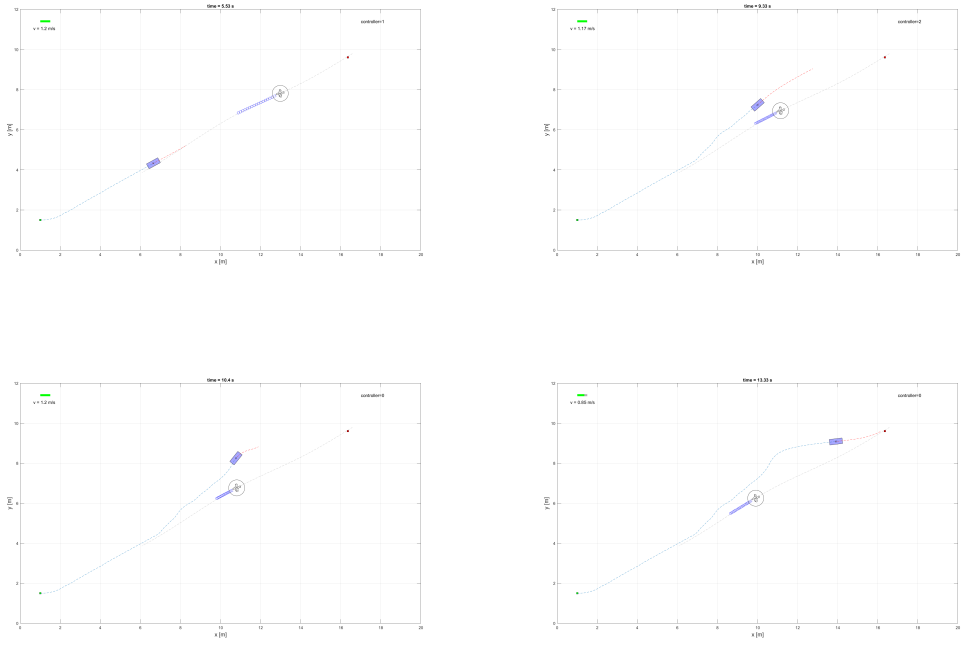
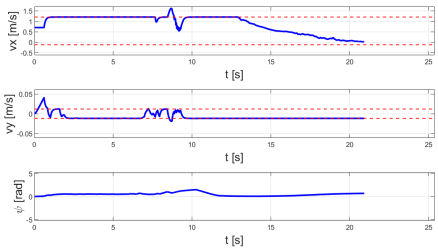


Figure 3.2. Simulation snapshots of scenario 1 using the VOPF obstacle avoidance constraint.

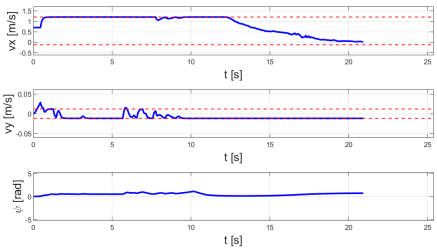
= 1" in the simulation, where the navigation control maneuver is represented as "controller = 0," and the emergency obstacle avoidance maneuver as "controller = 2") because its velocity lies within the VO_u generated by the interaction with the human obstacle at that moment. the trajectory applying the VOPF constraint appears smoother compared to the trajectory applying the APF constraint, as illustrated in Figure 3.3. Apparent spikes can be seen in 3.3a, while the trajectory from 3.3b is smoother during avoidance action. Additionally, in the sub-figure of the robot's trajectory with the APF constraint (Figure 3.3a), it can be observed that the v_x (the robot velocity along x axis of the robot body frame $v_x^{\mathbb{B}}$) component exceeds the set state boundary during obstacle avoidance (because the boundary constraints in this NMPC framework are soft constraints). While this phenomenon occurs infrequently, it remains a challenge to address in our subsequent work, specifically focusing on resolving infeasibility problems during optimization.

The computation time of NMPC recorded from scenario 1, shown in Figure 3.4, displays experimental results applying the two constraints. It is evident that, given our real-time NMPC structure shown in Figure 2.15, the shorter the computation time, the faster the predicted control solution updates. This property can explain the delay of avoidance movement in both experiments due to the need for computation time. Although the average computation time is around 9.9 ms, the maximum computation time is a crucial factor that could lead to a potential collision. Reducing computation time is deemed necessary in our upcoming work, and we will discuss this aspect in the last chapter as one of the areas for further research.

In the case of scenario 2, the experimental performance applying the two obstacle



(a) Trajectory generated by the robot applying the APF obstacle avoidance constraint.



(b) Trajectory generated by the robot applying the VOPF obstacle avoidance constraint.

Figure 3.3. Trajectory generated by the robot in scenario 1, where the red dashed line represents the boundary of robot states.

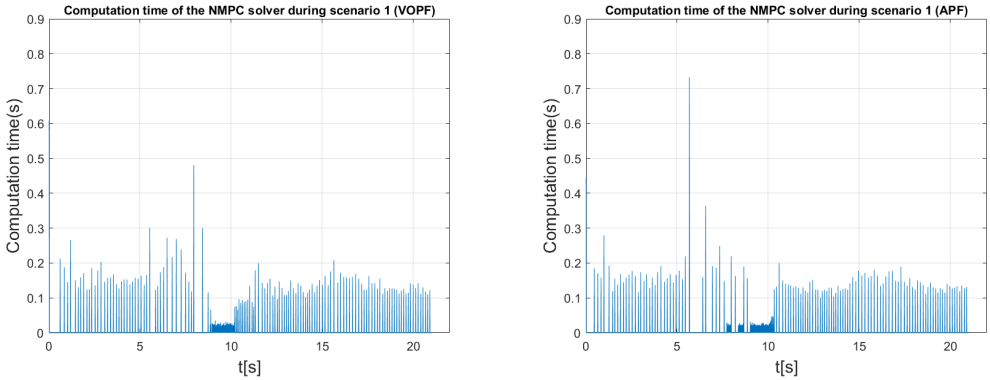


Figure 3.4. Computation time of the NMPC during scenario 1

avoidance constraints is also quite similar. However, the different constraints result in different path trajectories. In Figures 3.5 and 3.6, from both the first sub-figures, we can observe that the robot, under the APF constraint, takes avoidance actions to avoid the first moving human induced by the generated potential field around the obstacle. In contrast, the robot under the VOPF constraint does not trigger avoidance actions because, at that moment, the robot's absolute velocity is not within the VO_u, avoiding unnecessary evasive maneuvers.

Additionally, dynamic perception proves to be an effective predictor. In the third subfigure of Figure 3.5, it can be observed that, even though the relative distance between the robot and the human obstacle remains within a safe range ($d_{safe} < d_{ms}$), the robot's absolute velocity already lies within the VO_u represented by the criterion C_3^{DP}. As a result, the robot proactively initiates avoidance actions in advance, even though it is still at a safe distance from the human.

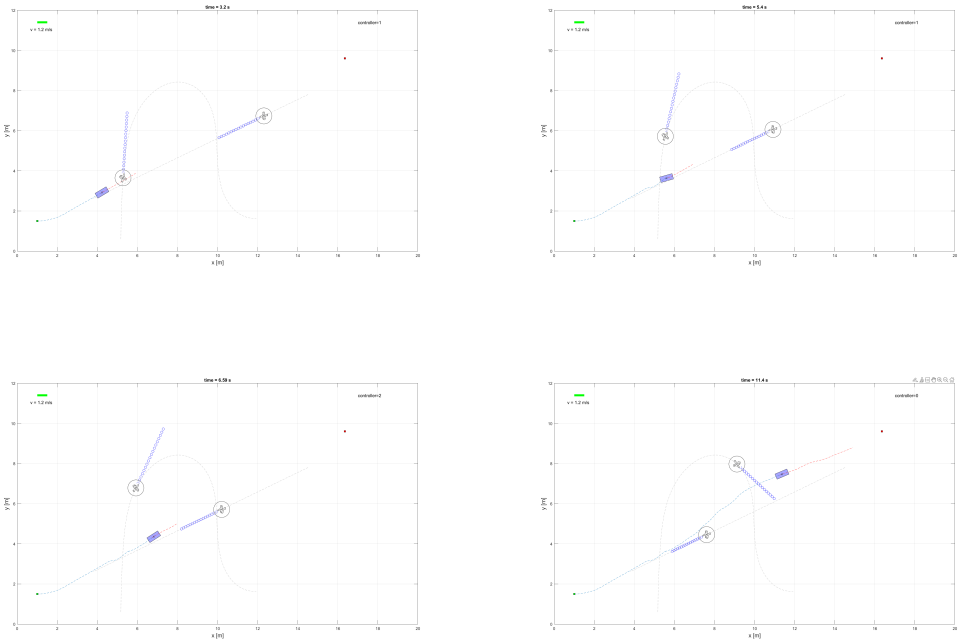


Figure 3.5. Simulation snapshots of scenario 2 using the APF obstacle avoidance constraint.

Comparing the trajectories of the robots applying two different obstacle avoidance methods in scenario 2, as shown in Figure 3.7, overall, the experimental trajectory applying the VOPF constraint appears smoother compared to the trajectory applying the APF constraint, but both of them can lead a collision-free path successfully through all experiments. In aspect of computation time, scenarios 1, 2, and 3 exhibit similar average computation times, indicating that our proposed NMPC framework can handle these three scenarios as we have set even sometimes the computation time spiked to 0.6 seconds.The computation time of NMPC recorded from scenario 2, shown in Figure 3.8.

In the case of scenario 3, as the results indicated in Table 3.2, the success rate of experiments using the APF constraint is 4 out of 5, with one collision occurring

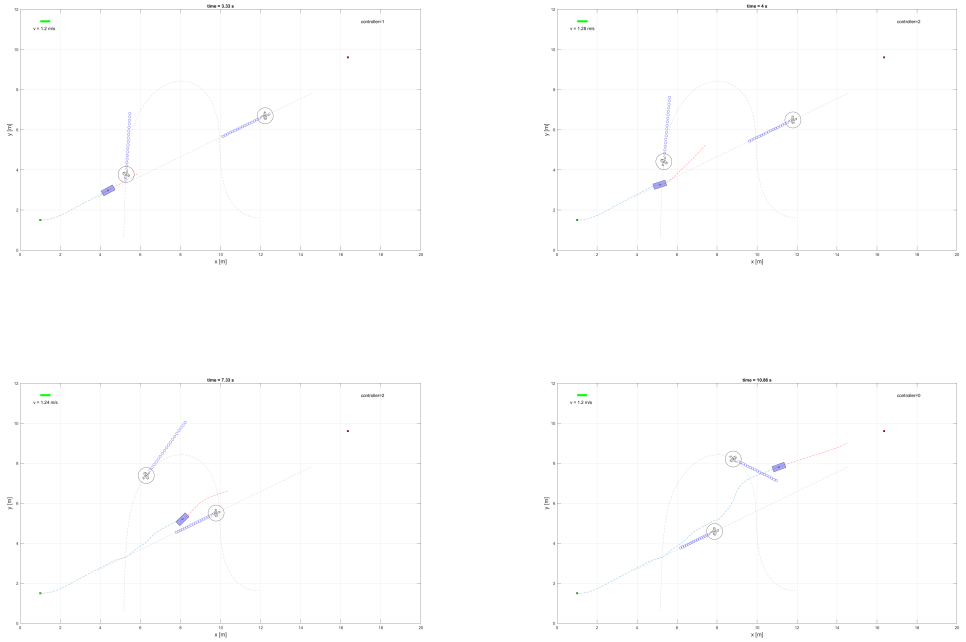
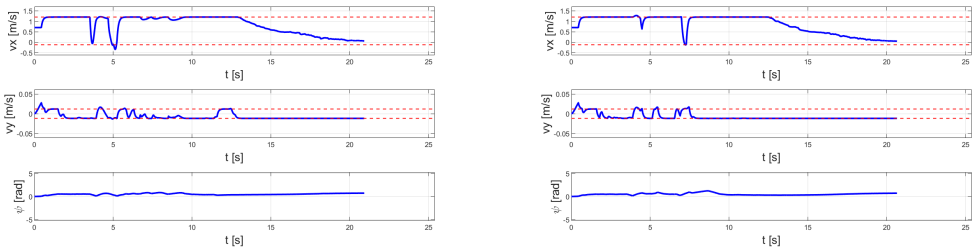


Figure 3.6. Simulation snapshots of scenario 2 using the VOPF obstacle avoidance constraint.

during tests. The increase in the number of moving obstacles to 3 causes elevated complexity in the OCP, leading to a narrower feasible solution space. This complexity is particularly notable for the APF-based constraint, as the potential field generated by obstacles increases with their quantity, resulting in longer computation times for finding feasible solutions (sometimes leading to infeasible situations and collisions). In contrast, the VOPF-based constraint, due to the predictive nature of *VO*, provides more feasible directions during the NMPC. This allows the robot enough time to apply the avoidance maneuver, ensuring that the actuators' actions are within the admissible actuators action space.

The simulation snapshots in Figures 3.9 and 3.10 show where the robot successfully reaches the target state. The trajectory of the robot based on the VOPF constraint is smoother, as shown in the comparison in Figure 3.11, due to the predictive nature of VOPF and the broader feasible solution space. However, if the speed of solving the NLP is fast enough, trajectories based on the APF constraint generally tend to avoid extreme obstacle avoidance situations. This is because the feasible solution space in the OCP with the VOPF constraint includes these extreme cases, while the generated motion of the robot applying the APF constraint should be outside the area of the potential field of obstacles. This ability to stay outside the potential field helps avoid extreme situations, such as passing by in a narrow distance.

For different maneuvers, the control input trajectories generated by the corresponding control problems align with our designed objectives. Following figures presents a continuous sequence of predicted control input trajectories corresponding



(a) Trajectory generated by the robot applying the APF obstacle avoidance constraint.

(b) Trajectory generated by the robot applying the VOPF obstacle avoidance constraint.

Figure 3.7. Trajectory generated by the robot in scenario 2, where the red dashed line represents the boundary of robot states.

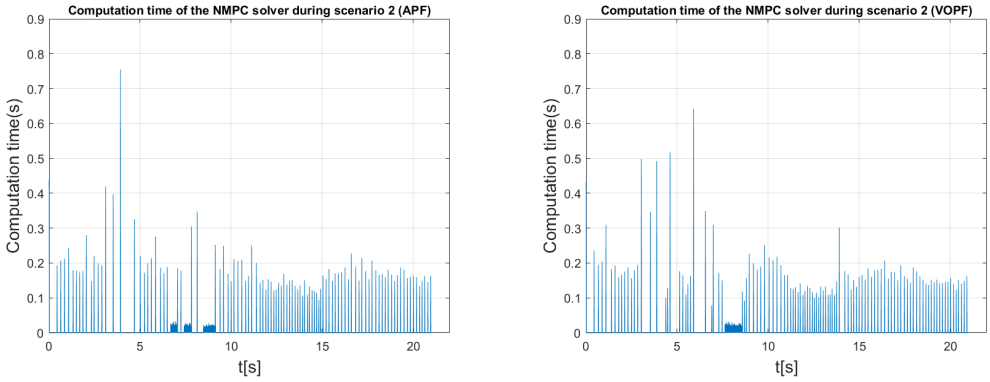


Figure 3.8. Computation time of the NMPC during scenario 2

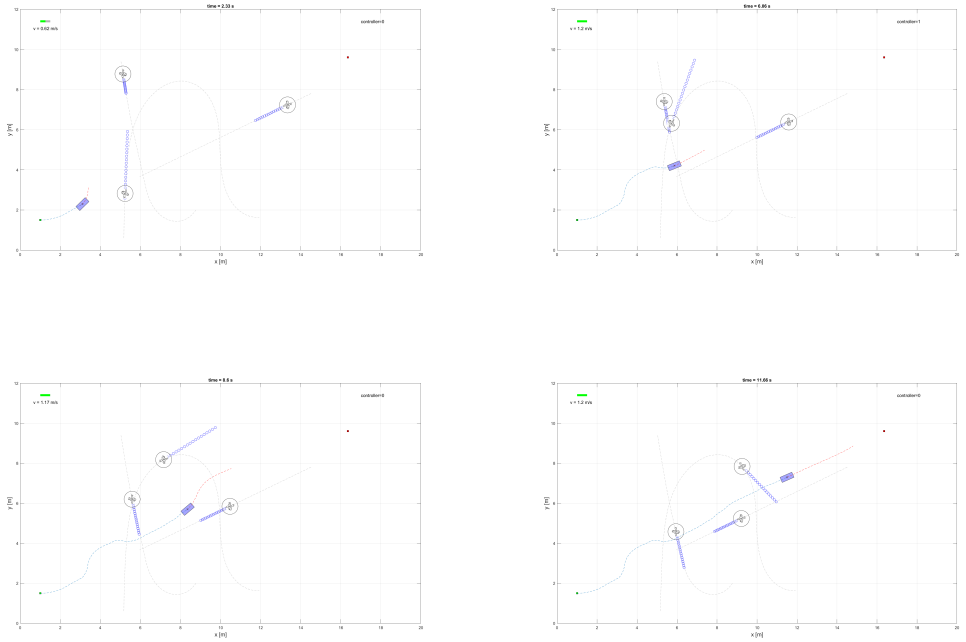


Figure 3.9. Simulation snapshots of scenario 3 using the APF obstacle avoidance constraint.

to three maneuvers during scenario 2 using the APF avoidance constraint. In the navigation maneuver, as shown in Figure 3.12, the predicted control is designed to move from the current state to the target state, minimizing the objective function. The computed velocity input maintains its maximum value of 1.2 m/s, while the heading angle remains nearly constant, directing the robot toward the endpoint state. In Figure 3.13, the prediction control input trajectory during obstacle avoidance maneuver reveals significant changes in angular velocity corresponding to the predicted avoidance action, steering away from the obstacle. Figure 3.14 showcases the input trajectory during the emergency obstacle avoidance maneuver. The OCP for this maneuver is designed to find a collision-free path as fast as possible, disregarding navigation. We can observe that the computed output may not align with the target point direction, while the steering action is evident, and the predicted horizon is longer due to the faster computation speed.

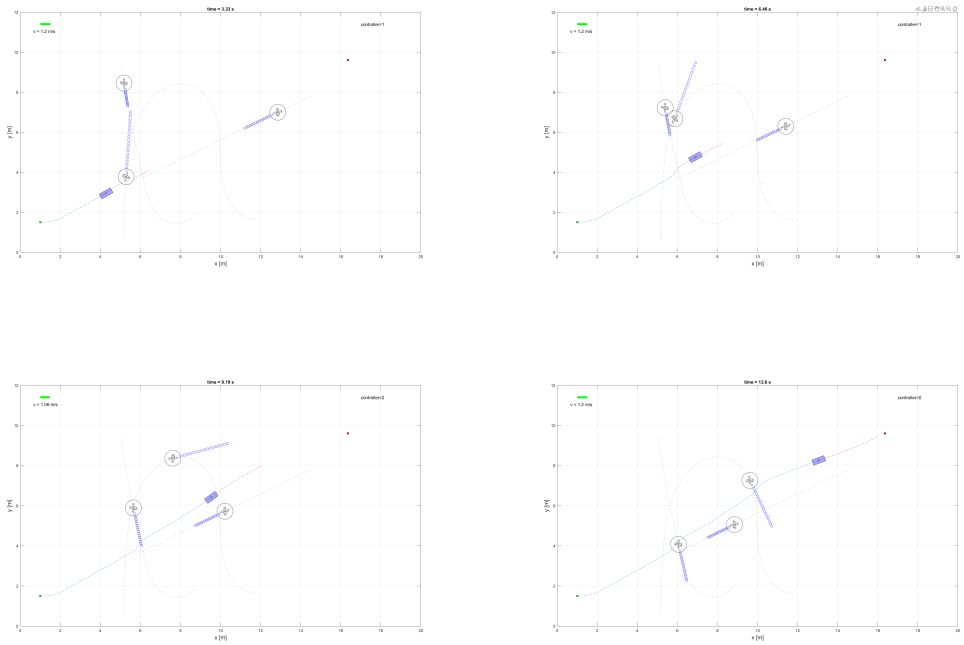
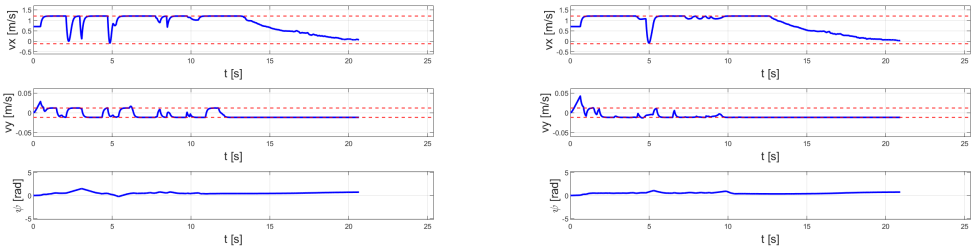


Figure 3.10. Simulation snapshots of scenario 3 using the VOPF obstacle avoidance constraint.



(a) Trajectory generated by the robot applying the APF obstacle avoidance constraint.

(b) Trajectory generated by the robot applying the VOPF obstacle avoidance constraint.

Figure 3.11. Trajectory generated by the robot in scenario 3, where the red dashed line represents the boundary of robot states.

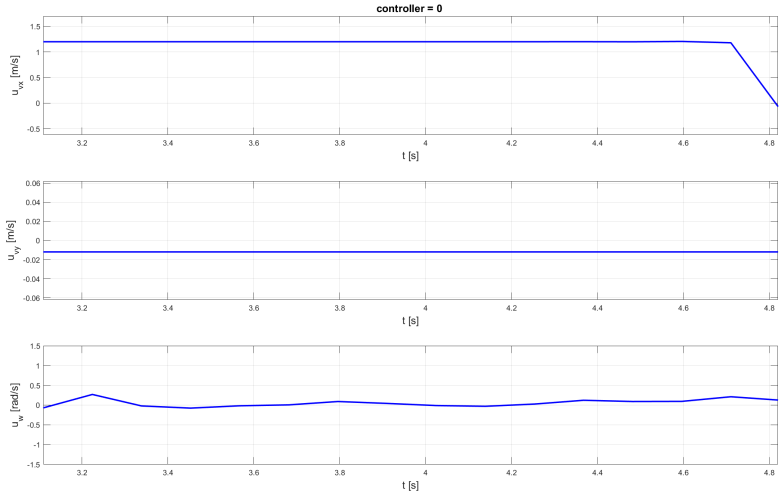


Figure 3.12. Predicted control actions along the horizon as the navigation maneuver in scenario 2 generated from corresponding optimal control problem with APF-based avoidance constraint.

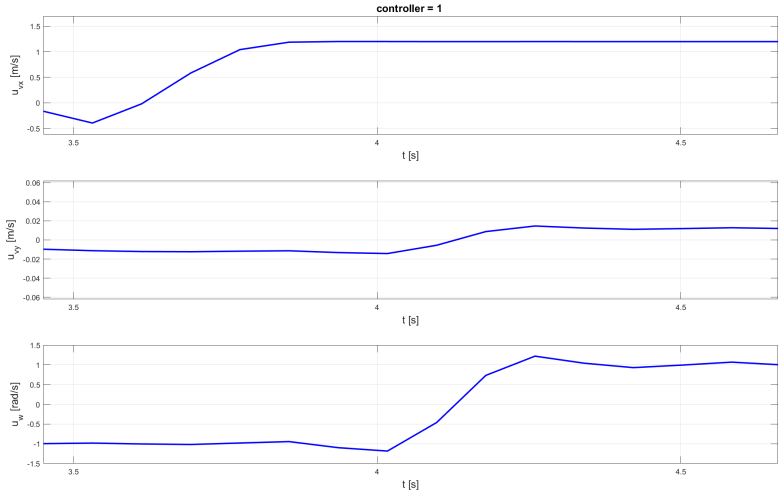


Figure 3.13. Predicted control actions along the horizon as the obstacle avoidance maneuver in scenario 2 generated from corresponding optimal control problem with APF-based avoidance constraint.

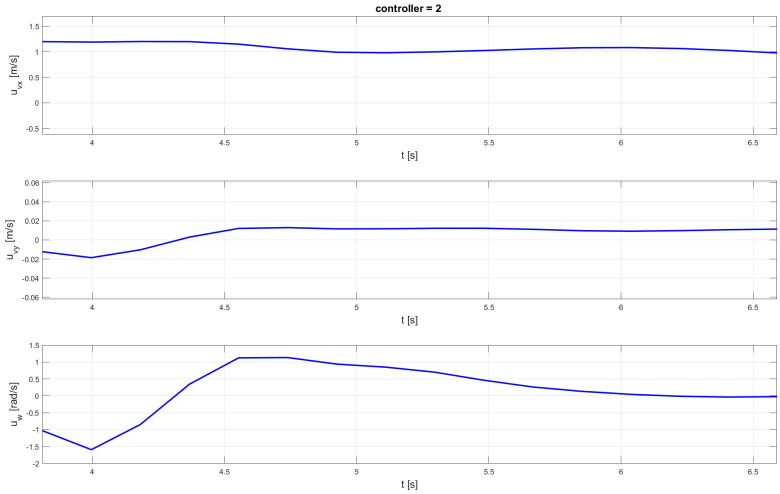


Figure 3.14. Predicted control actions along the horizon as the emergency obstacle avoidance maneuver in scenario 2 generated from corresponding optimal control problem with APF-based avoidance constraint.

Chapter 4

Conclusion

This work has presented a navigation and avoidance framework based on real-time NMPC for a high-level quadruped robot model with the dynamic perception inspired by velocity obstacles technique that has the ability to handle robot-human shared environments where humans are treated as dynamic obstacles. The main focus of this article is to provide a full autonomy framework that integrates advanced perception methods and with three different control maneuvers. Moreover, we present a numerical method, the trapezoidal method, to reduce the trajectory optimization problem to NLP. This framework is simulated in Matlab and Simulink to investigate the ability of keeping a quadrupedal robot collision-free while navigating in various dynamic environments. Moreover, a comparison of the performance between the APF constraint and VOPF constraint has been conducted during comparative simulations.

The proposed architecture has been simulated through three designed scenarios, with the maximal velocities of robot and obstacles reaching 1.2 m/s and 0.9 m/s respectively, providing collision-free paths in each scenario. In the experiments, the combination of dynamic perception and the Control Policy Decision-maker allows the framework to accurately and effectively predict the future human-robot interaction states in all trials and switch to the appropriate control maneuver, then generate safe predicted motion according to the corresponding control problem, thereby enhancing the framework's compatibility and collision-free capabilities in complex and dynamic scenarios. Regarding to the navigation capabilities, this framework consistently demonstrates quick performance in all experiments, reaching the target state in an average time of up to 19.26 s. In all three scenarios, the robot applying this framework successfully avoid obstacles in the majority of simulations. Generally, the avoidance trajectories generated by OCP with the VOPF constraint are smoother than those with the APF constraint. However, both constraints demonstrate the ability to successfully avoid obstacles in the designed scenarios. For real-time simulations, the average NMPC computation time is 9.9 ms. In navigation control problems, the average computation time is the shortest in comparison, and when switching to obstacle avoidance control problems, the average computation time increases significantly, despite at a rare frequency.

While we can demonstrate successful results in the specified scenarios, our experiments are conducted under certain constraints, and our method still has some limitations. Meanwhile, there are multiple directions for further work based on

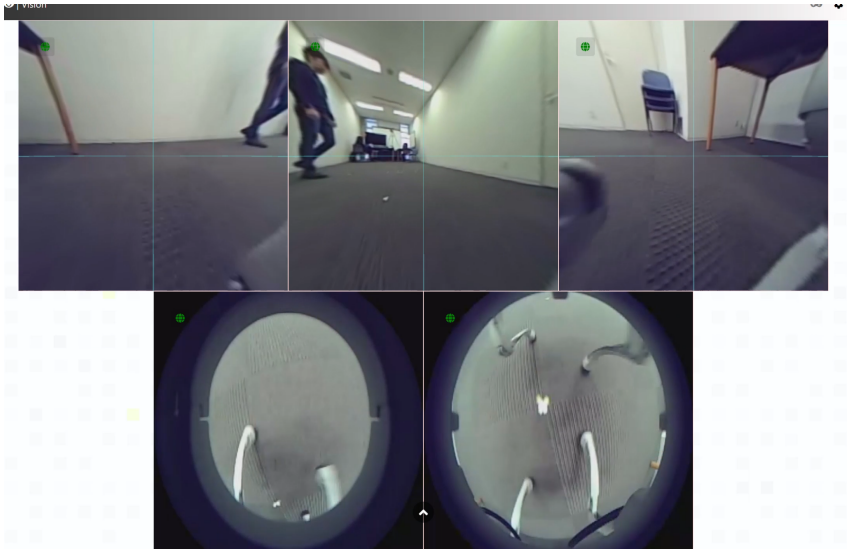


Figure 4.1. Quadrupedal robot obstacle avoidance experiment

those limitations. One limitation of our method is the lack of observation part. We conduct simulations assuming the accurate acquisition of the robot's current state, velocity and information of all obstacles. In practical, we rely on sensors to obtain data, which requires filtering and estimation. Thus, our further work involves developing a sensor-based scheme to apply the method proposed in this article and testing its capabilities as well, and it is currently in progress as shown in the figure 4.1.

Another limitation concerns the feasibility properties of the NMPC. In certain situations, infeasible solutions may be encountered, potentially leading to collisions if the robot follows such solutions. Therefore, a promising area for further research involves developing strategies to recover from potential infeasibilities. An alternative approach, as proposed in [15], is to utilize VO as a pure-feedback mechanism during emergency obstacle avoidance maneuvers. This avoids the need to solve the OCP, reducing computational effort, and enabling instantaneous motion without predicting obstacle movements.

As discussed in Chapter 3, the solving time of the NLP may not be sufficient for more complex environments. While the NMPC framework demonstrates effectiveness in handling the designed scenarios, its real-time performance might be compromised in more intricate dynamic environments due to extended consumption times. Therefore, an additional avenue for further research involves exploring more efficient optimization tools for our NMPC framework. OpEn [22] has gained popularity as a recent tool where the PANOC [23] (Proximal Averaged Newton-type method for Optimal Control) algorithm is coupled with a Penalty Method [9] to solve for trajectories. In a similar study utilizing this optimization tool, it was observed that the general solver time ranged between $5 - 10$ ms, occasionally experiencing spikes with a maximum duration of 56 ms [11].

An intriguing avenue for future exploration is the extension of the proposed NMPC framework to flying robots, where the VO can be employed for $3D$ dynamic

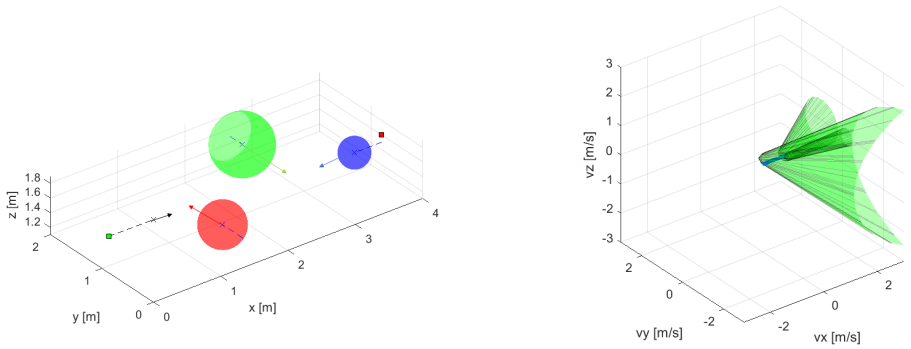


Figure 4.2. 3D environment map and the corresponding *VO*

perception and avoidance constraint in airspace, illustrated as Figure 4.2.

Bibliography

[1] Ivo Batkovic et al. “A robust scenario MPC approach for uncertain multi-modal obstacles”. In: *IEEE Control Systems Letters* 5.3 (2020), pp. 947–952.

[2] John T Betts. “Survey of numerical methods for trajectory optimization”. In: *Journal of guidance, control, and dynamics* 21.2 (1998), pp. 193–207.

[3] John Canny et al. *On the complexity of kinodynamic planning*. Tech. rep. Cornell University, 1988.

[4] JE Dennis, Matthias Heinkenschloss, and Luis N Vicente. “Trust-region interior-point SQP algorithms for a class of nonlinear programming problems”. In: *SIAM Journal on Control and Optimization* 36.5 (1998), pp. 1750–1794.

[5] Huckleberry Febbo et al. “Moving obstacle avoidance for large, high-speed autonomous ground vehicles”. In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 5568–5573.

[6] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using velocity obstacles”. In: *The international journal of robotics research* 17.7 (1998), pp. 760–772.

[7] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[8] Alessandro Gasparetto et al. “Path planning and trajectory planning algorithms: A general overview”. In: *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches* (2015), pp. 3–27.

[9] Ben Hermans, Goele Pipeleers, and Panagiotis Panos Patrinos. “A penalty method for nonlinear programs with set exclusion constraints”. In: *Automatica* 127 (2021), p. 109500.

[10] Xuemin Hu et al. “Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles”. In: *Mechanical systems and signal processing* 100 (2018), pp. 482–500.

[11] Samuel Karlsson, Björn Lindqvist, and George Nikolakopulos. “Ensuring Robot-Human Safety for the BD Spot Using Active Visual Tracking and NMPC With Velocity Obstacles”. In: *IEEE Access* 10 (2022), pp. 100224–100233.

[12] Matthew Kelly. “An introduction to trajectory optimization: How to do your own direct collocation”. In: *SIAM Review* 59.4 (2017), pp. 849–904.

[13] Matthew Peter Kelly. *OptimTraj: Trajectory Optimization for Matlab*. Version 1.7. Dec. 2022. DOI: 10.5281/zenodo.7430524. URL: https://github.com/MatthewPeterKelly/OptimTraj.

[14] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *The international journal of robotics research* 5.1 (1986), pp. 90–98.

[15] Maicol Laurenza et al. "Car collision avoidance with velocity obstacle approach: Evaluation of the reliability and performace of the collision avoidance maneuver". In: *2019 IEEE 5th International forum on Research and Technology for Society and Industry (RTSI)*. IEEE. 2019, pp. 465–470.

[16] Steven M LaValle and James J Kuffner Jr. "Randomized kinodynamic planning". In: *The international journal of robotics research* 20.5 (2001), pp. 378–400.

[17] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2682–2688.

[18] Tomas Lozano-Perez. "Robot programming". In: *Proceedings of the IEEE* 71.7 (1983), pp. 821–841.

[19] Jihyun Mok et al. "Gaussian-mixture based potential field approach for UAV collision avoidance". In: *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE. 2017, pp. 1316–1319.

[20] Joseph O'Rourke and Norman Badler. "Decomposition of three-dimensional objects into spheres". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 3 (1979), pp. 295–305.

[21] Ajay Sathya et al. "Embedded nonlinear model predictive control for obstacle avoidance using PANOC". In: *2018 European control conference (ECC)*. IEEE. 2018, pp. 1523–1528.

[22] P. Sopasakis, E. Fresk, and P. Patrinos. "OpEn: Code Generation for Embedded Nonconvex Optimization". In: *IFAC World Congress*. Berlin, 2020.

[23] Lorenzo Stella et al. "A simple and efficient algorithm for nonlinear model predictive control". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 1939–1944.

[24] Sankaranarayanan Subramanian et al. "Robust nmpc schemes for the control of mobile robots in the presence of dynamic obstacles". In: *2018 23rd international conference on methods & models in automation & robotics (MMAR)*. IEEE. 2018, pp. 768–773.

[25] Xiaoxue Zhang et al. "Trajectory generation by chance-constrained nonlinear MPC with probabilistic prediction". In: *IEEE Transactions on Cybernetics* 51.7 (2020), pp. 3616–3629.