Model Predictive Control for Safe Navigation in Multi-agent Systems

Project-II (ME47602) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Mechanical Engineering

by

Atharva Navsalkar

(18ME33002)

Under the supervision of

Prof. Ashish Ranjan Hota

Department of Mechanical Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2021-22

April 21, 2021

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 21, 2021 (Atharva Navsalkar)
Place: Kharagpur (18ME33002)

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Model Predictive Control for Safe Navigation in Multi-agent Systems**" submitted by **Atharva Navsalkar** (Roll No. 18ME33002) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Mechanical Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2021-22.

Date: April 21, 2021
Place: Kharagpur

Prof. Ashish Ranjan Hota
Department of Electrical Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Atharva Navsalkar** Roll No: **18ME33002**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Mechanical Engineering**

Thesis title: **Model Predictive Control for Safe Navigation in Multi-agent Systems**

Thesis supervisor: **Prof. Ashish Ranjan Hota**

Month and year of thesis submission: **April 21, 2021**

Drones have recently seen as strong rise in terms of popularity and deployment. This is mainly due to their ability to traverse difficult environments. For advanced use cases, autonomy and safe navigation in dynamic and uncertain environments has become essential. The capabilities of the flying machines increase even more if deployed in swarms or teams, adding additional need for coordination. This work discusses the use of optimization-based model predictive control for such swarm to navigate safely in cluttered environments. This work is of greater importance when the obstacles have irregular shape difficult to approximate using ellipsoids or spheres. This project furthers aims to address the challenge of distributed computation and motion uncertainty in real world.

Acknowledgements

Firstly, I sincerely thank my thesis supervisor Prof. Ashish Hota, Department of Electrical Engineering, IIT Kharagpur for the guidance throughout the project duration and previously as well. I also thank AI & Robotics Technology Park, Indian Institute of Science for providing financial support for equipment purchase for this project through Innovation Grant Program. Lastly, I thank my friends and family for supporting me during the work of this thesis.

Contents

Chapter 1

Introduction

1.1 Overview

Multi-robotic swarms, including drones, ground robots, or autonomous vehicles, have seen tremendous development due to applications ranging from military, search and rescue missions, cave explorations, indoor motion, or entertainment purposes. Motion planning of multiple robots in such a diverse set of environments remains an essential challenge. Each individual agent must safely avoid obstacles and other members of the group. The real world is, however, not completely deterministic in terms of both dynamics and state measurements. In this thesis, over the course of the year, my aim is to develop and implement computationally efficient centralized and distributed model predictive control (MPC) algorithms for risk-sensitive safe navigation of multiple robots under uncertainty.

MPC has been widely used for drones and provides flexibility to modify the problem formulations for different purposes. It is powerful tool that solves a finite horizon numerical optimization problem repeatedly to compute the control commands. The primary advantage of using MPC is its ability to accommodate various performance metrics and state/input constraints. The real world systems have an element of uncertainty and noise, and also surrounded by non-cooperative robots or obstacles. To handle the uncertain motion of obstacles, I plan to develop collision avoidance constraints in terms of conditional value-at-risk (CVaR) of the distance between full dimensional controlled objects and obstacles (instead of conveniently assuming them to be point-sized or make a conservative spherical/ellipsoid approximation). In order to capture the uncertainty inherent in the environment, I plan to leverage recent results in distributionally robust optimization to construct ambiguity sets for the uncertainty distribution from observed data in an online manner. The proposed scheme would provide rigorous (finite sample) guarantees on risk sensitive collision avoidance.

For this part of the project, we focus on identifying the current research gaps in the literature, and develop algorithms for deterministic multi-agent MPC. This report primarily focuses on the centralised and decentralised methods based on separate techniques. Results are presented for each of them based on realistic simulations on Gazebo. We will then compare the benefits and limitations of both to proceed with adding the uncertainty consideration. This chapter discussed the detailed literature survey and specific project goals. The achievements and ongoing work is briefly discussed in the last section of the chapter.

1.2 Literature Survey

Motion planning techniques for robotic teams have been thoroughly studied but some important challenges still exist in collision avoidance for a large number of agents, as reviewed in (Huang et al., 2019). Authors in (Luis et al., 2020) develop a distributed MPC scheme for multiple robots to generate trajectories in real-time. Though the approach has been shown to give excellent experimental results, other agents' current state (and planned states) have been assumed to be known to all agents for avoiding collisions. This information might not be practically possible to communicate amongst each other, making it challenging to implement in a decentralised manner. Moreover, uncertainties in the actual motion of robots or other dynamic obstacles have not been considered.

Reference (Park and Kim, 2020) develops a decentralised planner and constructs safe corridors using linear constraints. The authors estimate the reachable area by limiting the velocity or acceleration of other bodies, resulting in robust but conservative constraints. Other optimization-based methods like (Zhang et al., 2020) assume static, deterministic, but complex environments to find the least intrusive trajectories by solving MPC with signed-distance constraints formulation. One significant advantage of this method is that all dimensions of robots are accounted for, which is essential for large-sized robotic agents, but has been currently implemented for a single robot in a 2D world. Other works like (Zhu et al., 2021; Cong et al., 2021) use neural networks to predict the motion of neighbouring robots. Both these works use MPC to impose constraints using predicted states and work with decentralised computation. Robust MPC approaches as studied in (Kamel et al., 2017) ensure guaranteed safety but lead to a computationally expensive and overly conservative solution.

An alternative to model the uncertainty is using stochastic optimization. Given a probability distribution of possible transitions, probabilistic chance-based constraints can be applied to the MPC problem to limit the collision probability (Zhu and Alonso-Mora, 2019). Building on this, the authors in (Castillo-Lopez et al., 2020) propose tighter constraints for a single-agent with uncertain dynamic obstacles using chance constraints. Another work (Arul and Manocha, 2021) discusses decentralised implementation of MPC with probabilistic OCRA (Van Den Berg et al., 2011), which basically computes a set of collision-free velocities. Authors in (Arul and Manocha, 2021) compare both Gaussian and non-Gaussian distribution for evaluating chance constraints. Since chance constraints give probabilistic guarantees over large sample runs, it fails for worst-case scenarios. Hence, a hybrid approach was proposed in (Brüdigam et al., 2021) that computes two possibilities, a failsafe trajectory and chance-based stochastic MPC and applies the appropriate controller, giving a safety guarantee in worst-case scenarios.

Recently, the metric Conditional Value-at-risk (CVaR) has been used in robotics. The CVaR of a random loss is equal to the conditional expectation of the loss within the $(1 - \alpha)$ worst-case quantile of the loss distribution (Hakobyan et al., 2019). Authors in (Hakobyan et al., 2019) propose constraints on the CVaR values, which can assess the worst-case tail events of a probability distribution. This formulation is, so far, one of the most appropriate quantification of risk associated with the motion plan as it also bounds the magnitude of constraint violation. Though (Hakobyan et al., 2019) considers Gaussian randomness in obstacle motion, the work (Hakobyan and Yang, 2021) shows that this formulation works for collected sample data from observation of movement. The CVaR constraint has been adopted for constructing barrier functions as a safety filter (Ahmadi et al., 2022). However, these constraints are mostly derived for the case where the controlled object is a point mass and are studied in a centralized/single-robot setting.

Table 1.1 summarises the previous works in this domain and highlights the gap in the literature. It can be seen that methods for distributed and decentralised MPC do not consider the dimensions of the obstacles, which can be important in some cases. Also, the works that consider the dimensions and risk-aware motion planning have not been extended to multi-agent systems.

Table 1.1: Previous Works

References	Agents	Obstacle Modelling	MPC type	Uncertainty
Luis et al. (2020)	N	Ellipsoids (linearised)	Decentralised with information exchange	Not considered
Park and Kim (2020)	N	Ellipsoids (linearised)	Decentralised trajectory optimization with no information exchange	Robust obstacle space using error bounds
Zhang et al. (2020)	1	Polyhedral spaces	Single-agent MPC	Not considered
Zhu et al. (2021)	N	Ellipsoids	Decentralised MPC with no information exchange	Neural Networks for prediction
Kamel et al. (2017)	N	Sphere	Decentralised MPC with limited information exchange	Inflated uncertainty over the time horizon
Zhu and Alonso-Mora (2019)	N	Ellipsoids	Centralised	Chance constraints with Gaussian distribution
Castillo-Lopez et al. (2020)	1	Cuboids	Single-agent MPC	Chance constraints
Arul and Manocha (2021)	N	Spheres	Decentralised MPC with limited information exchange	Chance constraints with Gaussian mixture distribution
Hakobyan et al. (2019)	1	Polyhedral	Single-agent MPC	CVaR constraints with Gaussian distribution

1.3 Project Goals and Achievements

In this project aim to develop and implement computationally efficient and decentralised algorithms for multi-robot systems. We rely on the model predictive control (MPC) framework to impose constraints and ensure obstacle and collision avoidance. The approach is validated on realistic Gazebo simulations, using hexacopter drone as the robot. Theoretical formulation of the work is generic to accommodate other types of robots as well. Based on the literature survey, my work aims to address three key challenges.

1. **Obstacle Avoidance**: In addition to the other agents, the robot must effectively navigate in the environment with obstacles. Since obstacles are unknown initially, they are modelled as 3D polyhedrons composed of multiple half-planes. The work (Zhang et al., 2020) highlights the non-convex constraints to include obstacle avoidance.

2. **Decentralised Computation**: For better scalability and non-dependence on a central authority, each agent must share the computational load, with minimal communication among each other. This can be using distributed computation of a combined centralised MPC optimization. A particular method useful for this purpose is Alternate Direction of Method of Multipliers (ADMM) (Ferranti et al., 2018; Rey et al., 2018). In this method, each agent carries a local copy of states of other agents and all the agent communicate to converge to a consensus for the MPC solution. Alternatively, each agent can solve its independent problem with the assumption of cooperative behaviour, and minimal communication. An Optimal Reciprocal Collision Avoidance (ORCA) (Berg et al., 2011; Cheng et al., 2017) -based planner yields computationally inexpensive solutions for cooperative and decentralised agents.

3. **Handling Uncertainty**: An important aspect of the project involves risk-aware behaviour considering the uncertainty in the motion and perception of surrounding objects/agents. Based on available data or assumption of probability distribution, we narrowed down to two major approaches, chance constraints (Zhu and Alonso-Mora, 2019; Castillo-Lopez et al., 2020; Arul and Manocha, 2021) and risk consraints (Hakobyan et al., 2019). I plan to use

Conditional Value-at-risk (CVaR) approach to limit the risk associated with worst-case probabilities, leading to a safer path. The collected onboard data will be used to pick distributionally robust samples (Hakobyan and Yang, 2021) for modelling the constraints.

Currently, objectives 1 & 2 have be achieved to a large extent, and the on-going work focuses mainly on part 3. In the subsequent chapters, I introduce the system dynamics and controls, basic constraints, and vanilla model predictive control (MPC). Then, I discuss the centralised implementation of the multi-agent problem with polyhedral representation of the objects around the robot. Subsequently, simpler and faster technique (ORCA) is discussed for decentralised implementation, which relaxes the shape of agents to a sphere. Numerous simulation tests are conducted to back the proposed approach and an extensive code has been developed in the process. The MPC codebase used for centralised formulation can be accessed on Github at `https://bit.ly/3qzRSJC`. Similarly, the code for decentralised and parallel MPC based on ORCA constraints can be accessed on Github at `https://bit.ly/3D99TDC`.

Chapter 2

Problem Description

2.1 System State and Dynamics

We begin by describing the system i.e., drone. As seen in figure 2.1, two reference frames are considered: (i) inertial frame $\mathcal{A}$ with axes $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ and (ii) body frame $\mathcal{B}$ with axes $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$. The position $\mathbf{r}$ in $\mathcal{A}$ is represented by $\begin{bmatrix} x & y & z \end{bmatrix}^\top$. We use the $[Z - X - Y]$ Euler angle notation to represent the orientation of the drone. The orientation angles are roll ϕ (along X-axis), pitch θ (along Y-axis), and yaw ψ (along Z-axis). The rotation matrix for this convention (used in (Michael et al., 2010)) is

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\theta s\phi c\psi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}, \tag{2.1}$$

where c: *cos* and s: *sin* is used for shorter representation. The angular velocity in the body-fixed frame is given by $\boldsymbol{\omega} = p\mathbf{b}_1 + q\mathbf{b}_2 + r\mathbf{b}_3$. The state of the system is represented by the position and velocity, the Euler Angles (in the $[Z - X - Y]$ sequence), and the angular velocities. The control inputs are the thrust and the moments about the three axes. The state and control input is denoted by

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & p & q & r \end{bmatrix}^T, \tag{2.2}$$

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T.$$

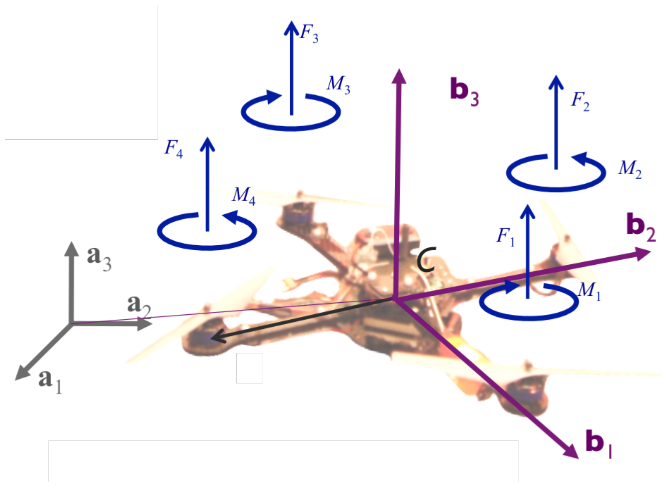

FIGURE 2.1: The inertial and body-fixed reference frames, and forces and moments by each of the rotors. Image obtained from 'Aerial Robotics' on Coursera by Prof. Vijay Kumar.

The dynamic equations for translational motion of the drone are

$$m\ddot{\mathbf{r}} = m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\0\\-mg\end{bmatrix} + R\begin{bmatrix}0\\0\\u_1\end{bmatrix}, \tag{2.3}$$

where the components are denoted in the inertial frame along $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$; m and g represent mass and gravitational acceleration respectively, and R is the rotation matrix from $\mathcal{B}$ to $\mathcal{A}$ as stated in (2.1). The input

$$u_1 = F_1 + F_2 + F_3 + F_4,$$

is the combined thrust obtained, where F_i is the thrust produced by i^{th} propeller. The equations for the rotation are:

$$I\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}u_2\\u_3\\u_4\end{bmatrix} - \begin{bmatrix}p\\q\\r\end{bmatrix} \times I\begin{bmatrix}p\\q\\r\end{bmatrix}, \tag{2.4}$$

where the components are along the body-fixed principal axes $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$; I is the inertia matrix and L is the distance between the rotor and the center of mass of the drone; the rotational control inputs are

$$\begin{aligned} u_2 &= L(F_2 - F_4), \\ u_3 &= L(F_3 - F_1), \\ u_4 &= M_1 - M_2 + M_3 - M_4, \end{aligned}$$

where u_2, u_3, and u_4 are the moments along the body axes and M_i is the moment produced by i^{th} propeller. The Euler-angular rates $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ are related to angular velocities in body-fixed frame (p, q, r) as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c\theta & 0 & s\theta \\ s\theta s\phi/c\phi & 1 & -c\theta s\phi/c\phi \\ -s\theta/c\phi & 0 & c\theta/c\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.5}$$

The dynamics of the multirotor can be represented in the standard $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ as represented in (Sabatino, 2015) form as follows

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \frac{1}{m}(c\psi s\theta + c\theta s\phi s\psi)u_1 \\ \frac{1}{m}(s\psi s\theta - c\theta s\phi c\psi)u_1 \\ \frac{1}{m}(c\phi c\theta)u_1 - g \\ p(c\theta) + r(s\theta) \\ p(s\theta s\phi/c\phi) + q - r(c\theta s\phi/c\phi) \\ -p(s\theta)/c\phi) + q(c\theta/c\phi) \\ \frac{1}{I_{xx}}(u_2 - (I_{zz} - I_{yy})qr) \\ \frac{1}{I_{yy}}(u_3 - (I_{xx} - I_{zz})pr) \\ \frac{1}{I_{zz}}(u_4 - (I_{yy} - I_{xx})pq) \end{bmatrix}, \tag{2.6}$$

where I has only the diagonal components I_{xx}, I_{yy}, I_{zz} as $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ are principal axes. As seen in equation (2.6), the system has nonlinearities making it harder to solve. Hence, the model is linearised around the hover position such that thrust is almost equal to mg and the euler angles are small such that $\sin(\theta) \approx \theta$. Hence, the second order angular terms can be neglected. The linear dynamic model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \theta g \\ -\phi g \\ \frac{1}{m}u_1 - g \\ p \\ q \\ r \\ \frac{u_2}{I_{xx}} \\ \frac{u_3}{I_{yy}} \\ \frac{u_4}{I_{zz}} \end{bmatrix}, \tag{2.7}$$

The actual control commands to the quadrotor are the angular velocities of the four rotors (Ω_i). We use the total thrust and moments about body axes as the control inputs ($\mathbf{u}$) for the MPC problem, as they are versatile and intuitive. It is known that thrust and moment of individual motors is given by

$$F_i = k_F \Omega_i^2 \;\; \forall i = 1, 2, 3, 4,$$

$$M_i = k_M \Omega_i^2 \;\; \forall i = 1, 2, 3, 4,$$

where k_F, k_M are force and moment coefficients respectively, and Ω_i is the angular velocity of the respective rotor (Bouabdallah, 2007). A low-level controller converts the control thrust and moments control commands to the individual rotors speeds accordingly.

In case of computational limitations, the dynamics of the system are assumed to be linear time-invariant (LTI) from equation (2.7). Assuming a total of N symmetrical agents, dynamics of agent i is of the form

$$\mathbf{x}_i(k+1) = A\mathbf{x}_i(k) + b\mathbf{u}_i(k) \qquad \forall i \in 1, 2..., N, \tag{2.8}$$

where $\mathbf{x}_i(k) \in \mathbb{R}^{n_x}$ represents the state vector of agent i at time step k, $\mathbf{u}_i(k) \in \mathbb{R}^{n_u}$ is the control input, $A \in \mathbb{R}^{n_x \times n_x}$ and $b \in \mathbb{R}^{n_x \times n_u}$ are the constant matrices. Specific limits on states and input as per system constraints $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{U}$ are applied for all agent at all times. These include environment boundaries, quadrotor orientation bounds to preserve linear dynamics, actuators limit among other constraints. Contrarily, full state dynamics 2.6 can be utilized for better results and greater angular operation range at the cost of slight computational load.

2.2 Baseline MPC formulation

Model Predictive Control (MPC) is a recursive finite-horizon optimal control problem. Figure 2.2 depicts the solution obtained for finite time horizon with a discrete time sample. The predicted states using the dynamics model aim to track the reference trajectory. Due to the use of numerical optimization techniques, various constraints on states and controls including the dynamics and limits can be applied. I particularly leverage on this capability to formulate types of constraints to achieve particular objectives.

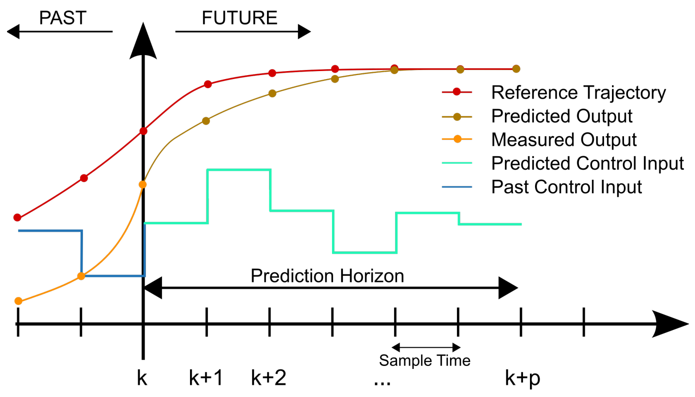

FIGURE 2.2: Visual representation of receding-horizon control or MPC. Image obtained from Wikipedia `https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg`

The goal of each agent is to reach a final position or track a desired trajectory. Irrespective of this, the stage cost function of proximity to desired state and control effort for a single agent i for time $k+l$ at time step k is

$$J_i(k+l|k) = \mathbf{x}_{e,i}^{\intercal} Q\, \mathbf{x}_{e,i} + \mathbf{u}^{\intercal} R\, \mathbf{u}, \tag{2.9}$$

where $\mathbf{x}_{e,i} = \mathbf{x}_{\text{ref},i} - \mathbf{x}_i(k+l)$ is the difference between predicted state at time $k+l$ and reference state depicting the deviation from desired values, and the other term penalises the control effort and matrices Q and R are positive semidefinite. Thus, for each agent, the overall cost $J_i = \sum_{k=1}^{T} J_i(k+l|k)$ is the sum over the time horizon of T. When the problem is solved in a centralised fashion the total cost of the optimization problem will in turn be sum of the costs of individual agents.

Finally, the objectives and constraints are formulated as a centralised optimization problem. The optimal control problem given by

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=1}^{N}\sum_{l=1}^{T} J_i(k+l|k) \\ \text{s.t.} \quad & x_i(k+l|k) = Ax_i(k+l-1|k) + bu_i(k+l-1|k), \\ & x \in \mathcal{X},\ u \in \mathcal{U}, \\ & \text{for } l \in \{0, ..., T-1\}, i \in \{1, ..., N\}, \\ & \text{plus additional constraints for collision avoidance,} \end{aligned} \tag{2.10}$$

is solved for states $\mathbf{x}$, control inputs $\mathbf{u}$ for all agents at each time step, where $\mathcal{X}$ and $\mathcal{U}$ denote the limits on states and control input.

Chapter 3

Collision Avoidance for Polyhedral Objects

Obstacle modelling is important for collision avoidance strategies to avoid computing overly conservative solutions. Lot of methods focus on spherical/ellipsoidal representation on obstacles. Though they provide linearised constraints and faster solutions, many obstacles need to be assumed as inflated ellipsoids resulting in inaccurate modelling and conservative trajectory. On the other hand, many obstacles can be almost exactly modelled as geometric shapes, that can be approximated as polyhedrons. This chapter discusses the non-linear constraints for such obstacle representation. We begin by discussing the theoretical outline of the approach, followed by simulation validation. Then, we move on to the centralised multi-agent case, that uses the constraints for other agents as obstacles. The chapter concludes by presenting simulation results for different scenarios consisting of multiple agents.

3.1 Obstacle Representation

Each obstacle occupies some space which is forbidden for the controlled object. This is embedded as constraint in the optimization problem. The obstacles are modelled as convex polyhedral sets composed as union of multiple half-spaces. The space occupied by m^{th} obstacle $\mathcal{O}^{(m)}$ is represented as

$$\mathcal{O}^{(m)} = \{\mathbf{p} \in \mathbb{R}^3 : G^{(m)}\mathbf{p} \leq h^{(m)}\}, \;\; m \in \{1, ..., M\}, \tag{3.1}$$

where M denotes the total number of obstacles, $G \in \mathbb{R}^{n_m \times 3}$ and $h \in \mathbb{R}^{n_m}$ are constant matrices, and n_m is the number of half spaces required to model the obstacle. We assume that any non-convex obstacle can be conservatively approximated to an enclosed polyhedral. If $z_i(k)$ is the position of the agent i at time step k, then desired condition for avoiding the obstacles is

$$\text{dist}(z_i(k), \mathcal{O}^{(m)}) > 0 \;\; \forall i \in \{1, 2, ..., N\}, \quad \forall m \in \{1, 2, ..., M\}, \tag{3.2}$$

where the dist() function is the distance between the current position of the agent and obstacle space. It is mathematically defined as

$$\begin{aligned} &\text{dist}(z_i(k), \mathcal{O}^{(m)}) := \min_{r \in \mathcal{O}^{(m)}} ||z_i(k) - r|| \text{ or,} \\ &\text{dist}(z_i(k), \mathcal{O}^{(m)}) = \min_{d}(||\mathbf{d}|| : G^{(m)}(z_i(k) + \mathbf{d}) \leq h^{(m)}). \end{aligned} \tag{3.3}$$

Currently, we consider the controlled robot to be point-sized. In future, this will be extended to consider the dimensions. It is evident that the constraint (3.2) is non-trivial to impose on the optimization problem. This will be reformulated to a different form, as discussed further in Section 3.2.

Apart from obstacles, we need to ensure collision-free trajectories between multiple agents. Since we had considered the robots as point sized, a simple approach is to put constraints between inter-agent distance to be greater than a user-defined value. But for our case, this formulation has few important disadvantages. Quadrotors cause a significant downwash i.e., strong airflow beneath the plane of quadrotor. Hence, it is important to avoid this vertical column-like region. Using a larger value for minimum distance between agent will cause overly conservative planning, as quadrotors can fly closely when besides each other.

Each agent considers itself as a point sized object and other agents as polygonal prism modelling the turbulent region below and above the other agent. As the agents are considered dynamic obstacles, the polyhedral representation of each obstacles is also the function of time. Let $\mathcal{N}_j$ be the forbidden space around the agent j, the desired constraint is

$$\text{dist}(z_i(k+l|k), \mathcal{N}_j) > 0, \;\; \forall l \in \{1, 2, ..., T\}, \forall i, j \in \{1, 2, ..., N\}, i \neq j. \tag{3.4}$$

The space $\mathcal{N}_j$ is a dynamic quantity, and hence is defined by time dependent matrices $G^{(j)}(k+l|k)$ and $h^{(j)}(k+l|k)$ in the form as defined in equation (3.1). These matrices can be computed from the predicted states of the other agents.

3.2 Constraints Reformulation

The constraint like (3.2) and (3.4) cannot be directly imposed, as the "dist()" function (in (3.3)) itself involves a minimisation problem. The work (Zhang et al., 2020) proposes an equivalent form for these constraints that can be framed for optimization solver. We have

$$\text{dist}(z, \mathcal{O}) \geq 0 \Longleftrightarrow \exists \lambda \geq 0 : (G\; z - h)^\intercal \lambda \geq 0, \;\; ||G^\intercal \lambda||_2 \leq 1, \tag{3.5}$$

obtained by finding the dual of the minimization problem.

Proof. From the definition (3.3) $\text{dist}(z_i(k), \mathcal{O}) = \min_d(||\mathbf{d}|| : G(z_i(k) + \mathbf{d}) \leq h)$ is a minimization problem. The dual of this problem is $\max_\lambda (Gz_i(k) - h)^\intercal \lambda : ||G^\intercal \lambda||_2 \leq 1, \lambda \geq 0$. Since $\mathcal{O}$ is assumed to have nonempty relative interior, strong duality holds (Zhang et al., 2020) as follows, $\text{dist}(z_i(k), \mathcal{O}) = \max_\lambda (Gz_i(k) - h)^\intercal \lambda : ||G^\intercal \lambda||_2 \leq 1, \lambda \geq 0$. Hence, we can reformulate the constraint on distance to the form specified in (3.5). □

Thus, if any λ satisfying the properties can be found out, it is proven that the collision constraint is satisfied. Moreover, these conditions can encoded as constraints with λ as control variable.

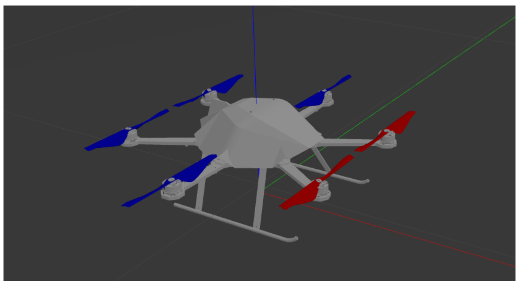
FIGURE 3.1: Closeup view of the drone

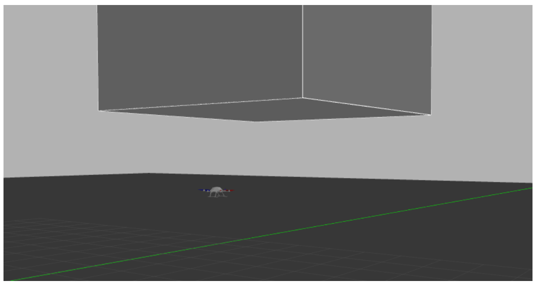
FIGURE 3.2: Gazebo snapshot during mission

3.3 Basic Demo

The above formulation was initially tested on a single agent and obtacle. Figures 3.1 and 3.2 show the visual representation of the robot and obstacle during the simulation. Starting from the origin the drone has the target setpoint as (10, 10, 10) in 3D space. We simulate the mission without any obstacles and with a cube of size 4m as an obstacle, centered at point (5, 5, 5). As we can see in the Fig. (3.3), the proposed obstacle avoidance scheme successfully avoids the obstacle that comes in the original path. It can be observed in figure 3.3 how the drone maintains constant altitude below the obstacle due to flat face of the cube. In further sections, we use the similar constraints for multiple agents posing as obstacles.

3.4 Centralised Approach for Multi-agent Case

Multi-agent model predictive control strategies can be implemented in a centralized or decentralized manner. This depends on where the computation occurs i.e., on a single node or multiple nodes. The optimal control problem with the reformulated

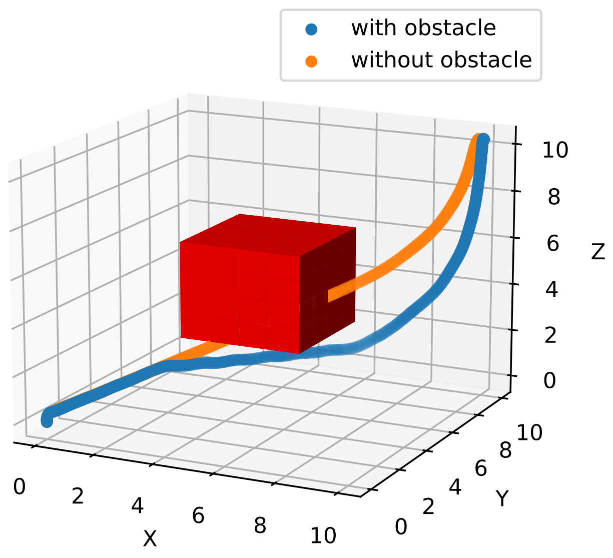

FIGURE 3.3: 3D plot for the mission

constraints for N agents at time step k is given by

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u},\lambda} \quad & \sum_{i=1}^{N}\sum_{l=1}^{T} J_i(k+l|k) \\
\text{s.t.} \quad & x_i(k+l|k) = Ax_i(k+l-1|k) + bu_i(k+l-1|k), \\
& x \in \mathcal{X},\ u \in \mathcal{U}, \\
& (G^{(a)}\, z_i - h^{(a)})\lambda^{(a)} \geq 0, \\
& ||G^{(a)}\lambda^{(a)}||_2 \leq 1, \lambda^{(a)} \geq 0 \\
& \text{for } l \in \{0, ..., T-1\}, i \in \{1, ..., N\},
\end{aligned}
\tag{3.6}
$$

is solved for states $\mathbf{x}$, control inputs $\mathbf{u}$ and dual variable $\boldsymbol{\lambda}$ for all agents. a is the member of the set of predicted polygon space for other agents or obstacles. The optimization problem (3.6) is repeatedly solved for a finite time horizon to obtain a optimal set of control inputs $\mathbf{u}^*(k)$. Only the first component $\mathbf{u}^*(k|k)$ is used as the control input. After one time step, states $\mathbf{x}(k+1)$ are obtained and used for solving the optimization problem again. This is a nonlinear optimization problem, that can be solved using the popular Interior Point OPTimizer (IPOPT) algorithm (Wächter and Biegler, 2006).

The problem described in (3.6) includes the sum of individual cost of each agent. The control commands of all the agents are computed simultaneously and communicated to each of the agent. This does not require the communication between different agents as all of them report the estimated state to this central node.

Algorithm 1 Centralised MPC algorithm for a multiple agents

```
Input: Initial Positions $\mathbf{x}(k|k)$ and control sequence $\mathbf{u}^*(k-1)$ for all agents
Output: Control command $\mathbf{u}_i^*(k)$ for all agents
 1: $\mathbf{x}_{\text{ref},i} \leftarrow$ getReference($k$)    ▷ reference over time horizon $T$ after $k$
 2: for $j \in \{1, ..., N\}$ and $l \in \{1, ..., T\}$ do
 3:     $\hat{\mathbf{x}}_j(k+l|k) \leftarrow$ getStatePredictions($\mathbf{u}_j^*(k+l-1|k-1)$)    ▷ Over time horizon $T$
 4: end for
 5: $\hat{\mathbf{X}}(k) \leftarrow \{\hat{\mathbf{x}}_1(k), ..., \hat{\mathbf{x}}_N(k)\}$    ▷ for all $N$ agents over the time horizon $T$
 6: $(G_i^{(a)}, h_i^{(a)}) \leftarrow$ getObstacleSpace($\mathcal{O}$, $\hat{\mathbf{X}}(k)$)
 7: OCP $\leftarrow$ buildMPC($\mathbf{x}_{\text{ref},i}$, $\mathbf{x}_i(k|k)$, $G_i^{(a)}$, $h_i^{(a)}$, $\mathbf{A}_i$)
 8: $\mathbf{u}_i^*(k) \leftarrow$ solveMPC(OCP)
 9: $\mathbf{u}(k) \leftarrow \mathbf{u}_i^*(k|k)$    ▷ use only the first component for all agents
10: broadcast($\mathbf{u}_i^*(k)$)
```

Algorithm 1 describes the steps for one iteration of MPC. It needs the current observed state and previous optimal control values for all agents. Then the predicted state of all the agents is computed based on dynamics model and previous optimal control commands. The matrices ($G^{(a)}$ and $h^{(a)}$) representing the obstacle space of other agents and obstacles can then be calculated. These matrices for the other agents are time-varying over the time horizon, unlike the stationary obstacles. The cost function is the sum of individual objective cost function obtained using state reference $\mathbf{x}_{\text{ref},i}$ for agent i. We can then formulate the optimization problem (3.6) by imposing appropriate constraints. Only the first component of the obtained $\mathbf{u}^*$ is utilised and the algorithm is re-iterated for next time step.

3.5 Simulation Results

We use the AsTec Firefly hexacopter as the controlled object for simulation. High-fidelity simulations are performed using RotorS simulator (Furrer et al., 2016) in

Gazebo. The controller communicates with the Gazebo using Robot Operating System (ROS). The MPC problem is solved using a nonlinear programming solver IPOPT (Wächter and Biegler, 2006). We use the "do-mpc" interface (Lucia et al., 2017) for the solver, that also uses CasADi package (Andersson et al., 2019). All the simulations are perfromed using HP Pavilion Gaming Laptop with AMD Ryzen 5800H CPU, 16GB RAM, Nvidia RTX3050 GPU.

Being a hexacopter, each agent is modelled as hexagonal prism using eight half-planes enclosing the robot. We now demonstrate the multi-agent missions involving position exchange in different formations with six drones. Figures 3.5 and 3.4 show the trajectories of each of the drone, and Figure 3.6 shows a snapshot of the gazebo simulation for the case with hexagonal formation. It can be seen how each agent successfully avoids the others to reach the final positions. Video links for the same have been provided in the description.

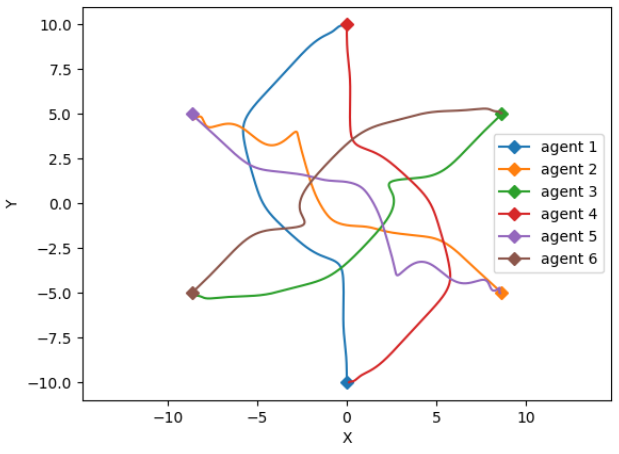

FIGURE 3.4: Top view with hexagonal formation. The video can be accessed at `bit.ly/3q3Ymzg`

All the simulations and optimizations stated above are computationally expensive, more due to the nonlinear and nonconvex constraints. The multi-agent simulation on Gazebo can run at a real time factor of 0.2. Over the 200 time-steps of the MPC problem, the computation time was found to be 0.29s on an average with a standard deviation of 0.04s. Figure 3.7 shows the CPU utilization of the machine. It can be observed that percentage utilization of each CPU core oscillates. A single

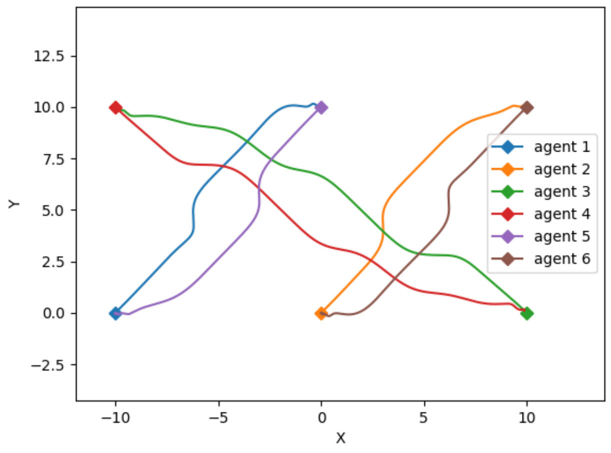

FIGURE 3.5: Top view mission with rectangular formation. The video can be accessed at `bit.ly/3dVptHd`

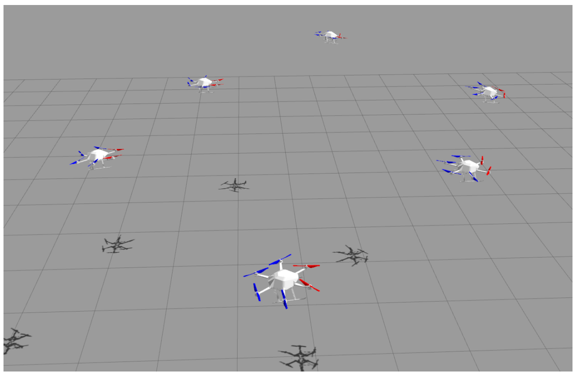

FIGURE 3.6: Gazebo snapshot during the mission

optimization problem is repeatedly solved on different computer threads, but lacks efficient parallelization.

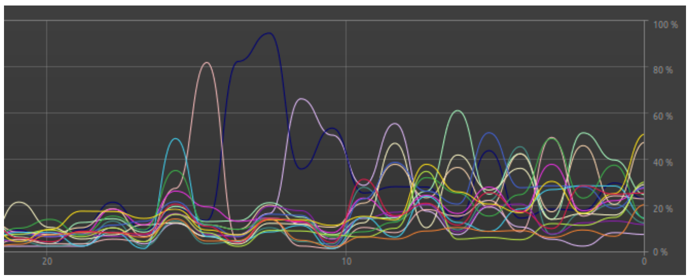

FIGURE 3.7: CPU usage during the centralised MPC optimizations.

Chapter 4

Decentralised MPC

We now focus on decentralised approach for solving the MPC on independent parallel nodes. Realistically, each agent agent has some decent computational power, hence it must be utilised as much as possible. This efficient parallel computation results in faster solving times and good scalibility. One approach, namely Alternate Direction of Method of Multipliers (ADMM) (Ferranti et al., 2018; Rey et al., 2018) works by efficiently fragmenting the single MPC problem (like the one discussed in previous chapter), that can be solved on multiple nodes. At each time step, multiple rounds of optimization and communication are required to obtain a consensus on true states and predictions of all the agents. Though it can handle all types of constraints, the computational load is usually very high for a time-critical system like ours. Hence, I shifted to Optimal Reciprocal Collision Avoidance (ORCA) method for decentralized implementation. The only limitation of this method is the assumption of the circular shape of the agents. In this chapter we discuss the mathematical framwork for ORCA, parallel computation architecture on the PC, and simulation results for the same.

4.1 ORCA Algorithm

This is a distributed collision avoidance algorithm that enables each agent to compute optimal velocities with guaranteed collision avoidance for finite time horizon. This algorithm applies constraint on relative velocities for each pair of agents to avert collisions with each other. We define a velocity obstacle, for agent B as obstacle, with respect to agent A as

$$VO^{\tau}_{A|B} = \{\mathbf{v} | \forall t \in [0, \tau] :: ||\mathbf{v}t - (\mathbf{p}_B - \mathbf{p}_A)|| < (r_A + r_B)\},$$

where $\mathbf{p}_B$, $\mathbf{p}_A$ are positions of agent A and B, and r_A, r_B being their respective radius (assuming circular shape of agents). The velocity obstacle is basically a set of forbidden relative velocities of agent A with respect to B to avoid collision with B in time horizon τ. With $\mathbf{v}_A$, $\mathbf{v}_B$ being the current velocities, let $\mathbf{u}$ be the perpendicular vector from relative velocity $(\mathbf{v}_A - \mathbf{v}_B)$ to the velocity obstacle, as shown in Fig. 4.1. The illustration shows the geometric interpretation of the velocity obstacle. Mathematically, we define

$$\mathbf{u} = (\arg\min_{\mathbf{v} \in VO^{\tau}_{A|B}} ||\mathbf{v} - (\mathbf{v}_A - \mathbf{v}_B)||) - (\mathbf{v}_A - \mathbf{v}_B).$$

This represents the minimum distance from the relative velocity vector to the boundary of the velocity obstacle. The vector $\mathbf{u}$ is analytically computed during the runtime for any set of position and velocity values using the known geometric properties of the $VO^{\tau}_{A|B}$, circles and tangents.

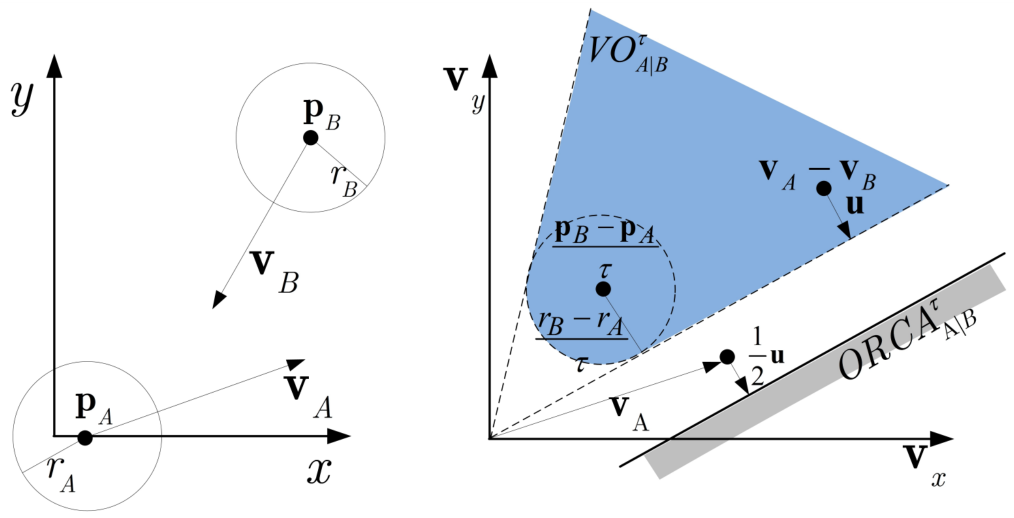

FIGURE 4.1: Left: Positions and velocities of agents in global perspective. Right: Geometric representation of velocity obstacle $VO^{\tau}_{A|B}$ and $ORCA^{\tau}_{A|B}$ constraint from reference frame of agent A, when $\mathbf{v}_A - \mathbf{v}_B$ is inside $VO^{\tau}_{A|B}$. Image from (Cheng et al., 2017).

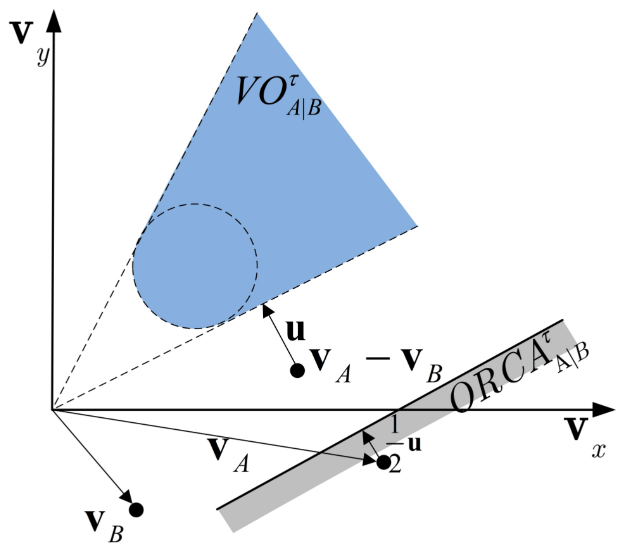

FIGURE 4.2: $ORCA^{\tau}_{A|B}$ constraint when $\mathbf{v}_A - \mathbf{v}_B$ is outside $VO^{\tau}_{A|B}$. Image from (Cheng et al., 2017).

If $(\mathbf{v}_A - \mathbf{v}_B)$ belongs to the velocity obstacle $VO^{\tau}_{A|B}$, the ORCA region for velocity of agent A (Berg et al., 2011) is (Fig. 4.1)

$$ORCA^{\tau}_{A|B}(\mathbf{v}_A, \mathbf{u}) = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \frac{1}{2}\mathbf{u}))\mathbf{n} \geq 0\},$$

where $\mathbf{n}$ is the normal vector of $\mathbf{u}$. The term $\frac{1}{2}\mathbf{u}$ denotes that agent A takes half the correction is the velocity and assumes that the other half be corrected by agent B similarly. This is applied as a constraint on the velocity of the agent A. When $(\mathbf{v}_A - \mathbf{v}_B)$ is already outside $VO^{\tau}_{A|B}$, the current trajectory is guaranteed to be collision-free for atleast time horixon τ. In this case, the constraint on the velocity becomes (Cheng et al., 2017) (Fig. 4.1)

$$ORCA^{\tau}_{A|B}(\mathbf{v}_A, \mathbf{u}) = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \frac{1}{2}\mathbf{u}))\mathbf{n} \leq 0\}.$$

Note that these constraints as linear constraints, hence computationally very fast. These constraints are applied for all time steps, and hence requires the relative velocities and positions at all time steps of MPC horizon. The expected states of agent A (on which the problem is solved) can be approximated using the previous solution. Whereas for the other agent B, we make a constant velocity assumption to calculate a tentative relative positions and velocities. Also, if a communication channel may be established to use the previously calculated solution on agent B for better accuracy.

4.2 Decentralized MPC Problem

Based on the previous MPC formulation, we discuss the decentralized version of MPC using ORCA constraints for inter-agent cooperation. For an agent A at time step k, the optimal control problem is

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{l=1}^{T} J_A^{k+l|k}(\mathbf{x},\mathbf{u},\mathbf{x}_{\text{ref}}) \\ \text{s.t.} \quad & \mathbf{x}_A(k+l|k) = A\mathbf{x}_A(k+l-1|k) + b\mathbf{u}_A(k+l-1|k), \\ & x \in \mathcal{X},\ u \in \mathcal{U}, \\ & \text{vel}(\mathbf{x}_A) \in ORCA_{A|B}^{\tau-l}(\mathbf{x}_A^{k+l|k}, \mathbf{x}_B^{k+l|k}) \quad \forall l \in [1,T], \\ & \text{and} \quad B \in [1,N] \neq A, \end{aligned} \tag{4.1}$$

where $J_A^{k+l|k}(\mathbf{x},\mathbf{u},\mathbf{x}_{\text{ref}})$ represents the quadratic cost function comprising of distance from target setpoint and control inputs, A and b are matrices representing the linearised dynamics, $\mathcal{X}$ and $\mathcal{U}$ being the limits, and vel($\mathbf{x}_A$) function denoting the velocity component of state $\mathbf{x}_A$. This problem is solved simultaneously and independently on each of the agents. This is further described in algorithm 2. This algorithm must parallely run on all the agents. The initial state $\mathbf{x}_A(k|k)$, the previous solution $\mathbf{u}_A^*(k-1)$, and the position and velocity of the other agent is required as an input for any time step. Firstly, from the previous solution and constant velocity assumption, we calculate the prediction for the relative velocities as shown in steps 3 and 4. We then use the geometric properties to calculate the vector $\mathbf{u}$ defined earlier (Step 5). Based on this, a linear ORCA constraint on the velocity is calculated for each pair of agents at all time steps in step 6. Subsequent steps follow the general procedure for building and solving the MPC to apply the solution at the first time step.

Algorithm 2 Decentralised MPC algorithm (for agent A) with ORCA constraints

Input: Initial state $\mathbf{x}_A(k|k)$ and control sequence $\mathbf{u}_A^*(k-1)$ and position $\mathbf{p}_B$ and velocity $\mathbf{v}_B$ for all other agents

Output: Control command $\mathbf{u}_A^*(k)$ for agent A

```
1: $\mathbf{x}_{\mathrm{ref},A} \leftarrow$ getReference($k$)    ▷ reference over time horizon $T$ after $k$
2: for $B \in \{1,...,N\} \neq A$ and $l \in \{1,...,T\}$ do
3:     $\hat{\mathbf{p}}_B(k+l|k) = \mathbf{p}_B + l\mathbf{v}_B\Delta t$    ▷ constant velocity for B over time horizon $T$
4:     $(\hat{\mathbf{p}}_A^{k+l|k}, \hat{\mathbf{v}}_A^{k+l|k}) \leftarrow$ getPredictions($\mathbf{x}_A^*(k-1), l$)
5:     $\mathbf{u}_{A|B}^{\tau-l} \leftarrow$ getU($\hat{\mathbf{p}}_A^{k+l|k}, \hat{\mathbf{p}}_B^{k+l|k}, \hat{\mathbf{v}}_A^{k+l|k}, \mathbf{v}_B$)    ▷ calculate $\mathbf{u}$ vector geometrically
6:     $ORCA_{A|B}^{\tau-l} \leftarrow$ getORCA($\hat{\mathbf{v}}_A^{k+l|k}$, $\mathbf{u}_{A|B}^{\tau-l}$)
7: end for
8: OCP $\leftarrow$ buildMPC($\mathbf{x}_{\mathrm{ref},A}$, $\mathbf{x}_A(k|k), ORCA_{A|B}$)    ▷ here $ORCA_{A|B}$ denotes set of all linear constraints
9: $\mathbf{u}_i^*(k) \leftarrow$ solveMPC(OCP)
10: $\mathbf{u}(k) \leftarrow \mathbf{u}_i^*(k|k)$    ▷ use only the first component for all agents
```

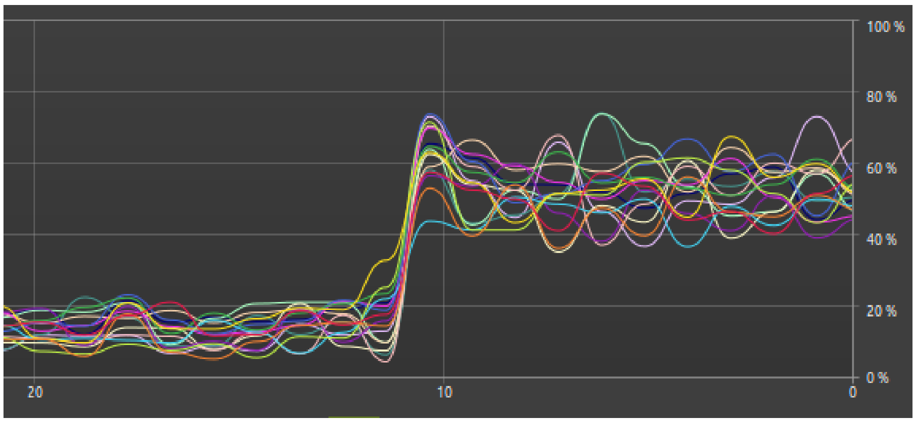

FIGURE 4.3: CPU usage during the parallel computation of multiple MPC optimizations.

4.3 Software Architecture

Simulation of such scenarios require independent computation of MPC interacting with a common Gazebo simulation. For this purpose, I developed a codebase for parallel processing for multiple agents. This involves a single Python script initialising multiple processes and ROS nodes (acting as independent agents), that invoke the algorithm 2. All these processes communicate their states and control input to other agents and Gazebo over ROS Master. We use the `Multiprocessing` module in Python that initiates new processes for each agent. Fig. 4.3 depicts the change in CPU usage as the MPC starts. We can observe that all the cores are being used considerably. The initial part shows the CPU usage for the Gazebo Simulation and ROS. This can be compared with Fig. 3.7 from the previous chapter to understand the utilization of the parallel processing. The code is written in a generic nature to facilitate quick implementation of any other decentralised MPC algorithm.

4.4 Simulation Results

We conduct multiple cases to test our approach involving point-to-point transition missions. In the first case, we consider the case of robots approaching each other in head on direction. Fig. 4.4 shows the trajectories for all the agents. It can be observed in the video (link in the caption) how the agents in the middle area pause for some time to let the other agents pass due to insufficient space. Next, we consider

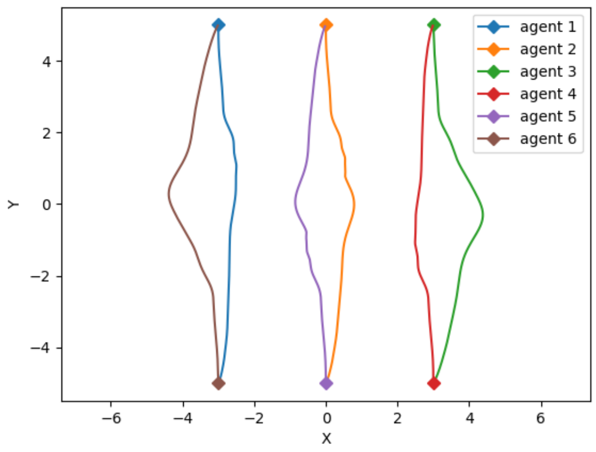

FIGURE 4.4: Trajectories of six agents linearly approaching each other. The video can be accessed at `https://bit.ly/3NkCw5s`

a symmetric hexagonal formation similar to the one discussed in previous chapter. The video (and trajectory plots in Fig. 4.5) shows how the agents keep a safe distance from each other and moving in a round-about like motion. Similarly, Fig. 4.6 shows another case with diagonal position switching in a rectangular formation. In the next case (Fig. 4.7), we consider two teams of three agents exchange positions in a triangular orientation. A snapshot of Gazebo simulation (Fig. 4.8) shows the positions when the agents closely pass each other.

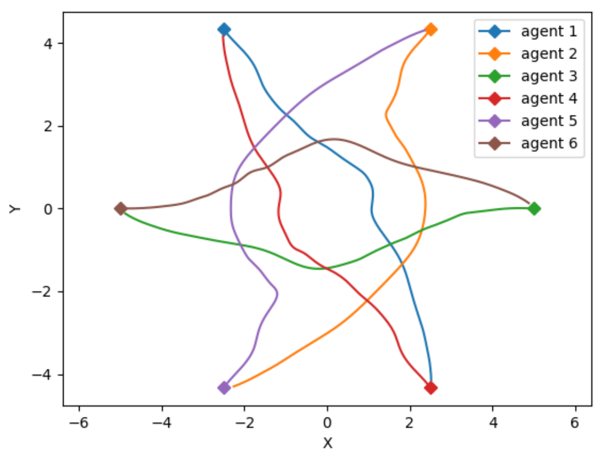

FIGURE 4.5: Trajectories of six agents in symmetrical hexagonal formation. The video can be accessed at `https://bit.ly/3D7YKmI`

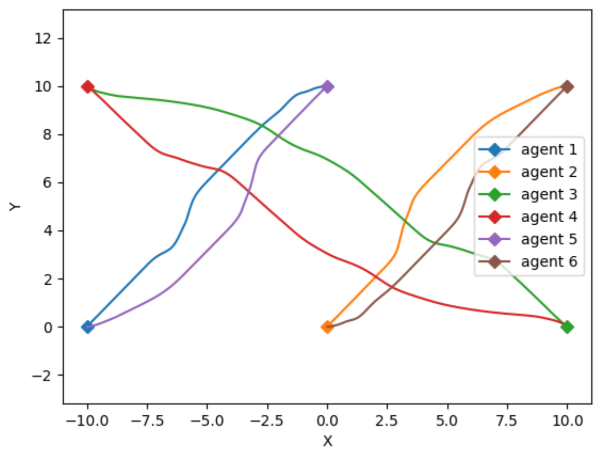

FIGURE 4.6: Trajectories for the mission with rectangular formation. The video can be accessed at `https://bit.ly/3iwOjjb`

Video links for all the cases are provided in the captions of the figures. This method is significantly faster in compared with the previously discussed method with mean computation time of 0.06s with a standard deviation of 0.018s for a data set of over 200 iterations. The entire simulation run with a real time factor of 0.5 in the Gazebo.

It was observed in some cases that collisions (or slight brushing of the circles) occurred due to difference dynamic model of the realistic simulation and low-level controller tracking error. A higher value of time horizon for the ORCA constraint τ mitigates

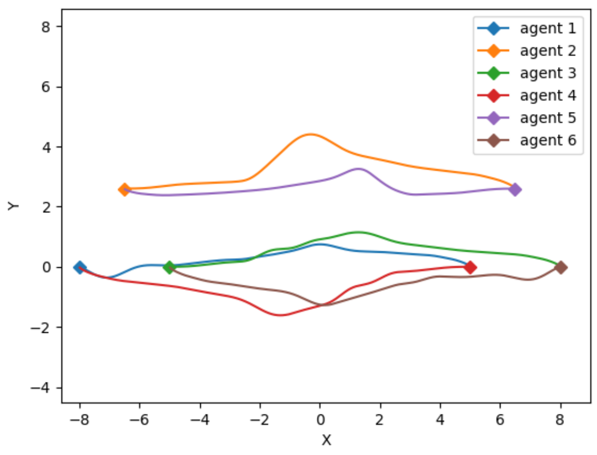

FIGURE 4.7: Trajectories for the mission with two teams in triangular configuration. The video can be accessed at `https://bit.ly/3Nl1STd`

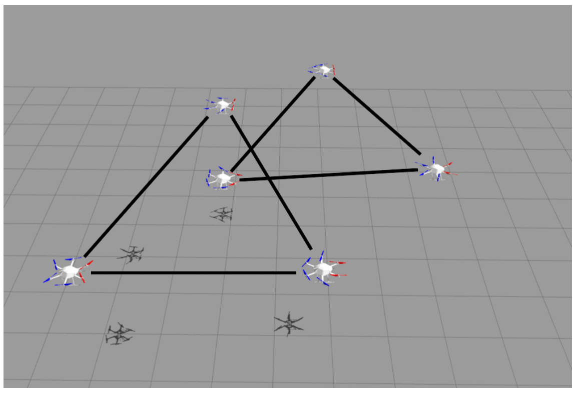

FIGURE 4.8: Snapshot of the Gazebo simulation.

this problem. Meanwhile, we can consider the minute overlap of the circles like a slack variable, and high quadratic cost for the overlap distance. These slight modifications significantly increase the collision avoidance performance of the proposed controller.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This report comprehensively discusses the Model Predictive Control for navigation in multi-agent system in environment with obstacles. We begin by surveying the existing work to identify the problem statement of the project and bridge the research gap. We objectively discuss three pillars of focus in the project namely, obstacle avoidance, decentralised computation, uncertainly handling. Meanwhile, the first two goals have been achieved, further work would mostly focus on the aspect of adding constraints considering uncertainty in state perception and motion of surrounding obstacles. We describe the states, control representation and derived dynamics of a multi-rotor drone. Further we consider two approaches, first of them being the polyhedral-shaped assumption of the obstacles. The constraints are then reformulated using the dual problem in a form that can be directly used in vanilla MPC code. Contrary to the centralised MPC, in the next chapter, we introduced ORCA-based method for faster and distributed implementation. The ORCA algorithm is described for a pair of agents and is extended to multiple agents solving parallely. For both the methods, we illustrate multiple test cases and videos of the simulations. We also compare the computational aspects associated with both the methods. In conclusion, the ORCA method works significantly better for simple multi-agent cases, with a trade-off with the assumption of circular shape of the agent. While this might not be a problem for robots like drones, it may be inaccurate or overly conservative for modelling generic obstacles.

In future I plan to focus of three aspects –

1. **Planning with Uncertainty**: Current formulation assumes perfect values of the states of the agents, which is not observed in real world. Moreover, other dynamic obstacles might have uncertainty in the motion. I will be using Distributionally Robust approach (Hakobyan and Yang, 2021) for filtering out the onboard observed data for other agents and obstacles. This can then be used for adding CVaR and Chance constraints to the current parallel simulation architecture.
2. **Multi-element Realistic Simulations**: Current tests mostly involve position exchange mission, we plan to test more extensively on realistic mission. Though we have validated collision avoidance with polyhedral elements, next tests would involve both the ORCA for other agents and dynamic obstacles.
3. **Embedded Implementation**: I am actively working on using the Raspberry Pi to compute a single agent decentralised MPC problem. Current simulation framework is modular to accommodate other independent computation nodes (RPi) with the already running multi-agent MPC.

Bibliography

Ahmadi, M., Xiong, X., and Ames, A. D. (2022). Risk-averse control via cvar barrier functions: Application to bipedal robot locomotion. *IEEE Control Systems Letters*, 6:878–883.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36.

Arul, S. H. and Manocha, D. (2021). Swarmcco: Probabilistic reactive collision avoidance for quadrotor swarms under uncertainty. *IEEE Robotics and Automation Letters*, 6(2):2437–2444.

Berg, J. v. d., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer.

Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying. Technical report, Epfl.

Brüdigam, T., Olbrich, M., Wollherr, D., and Leibold, M. (2021). Stochastic model predictive control with a safety guarantee for automated driving. *IEEE Transactions on Intelligent Vehicles*.

Castillo-Lopez, M., Ludivig, P., Sajadi-Alamdari, S. A., Sanchez-Lopez, J. L., Olivares-Mendez, M. A., and Voos, H. (2020). A real-time approach for chance-constrained motion planning with dynamic obstacles. *IEEE Robotics and Automation Letters*, 5(2):3620–3625.

Cheng, H., Zhu, Q., Liu, Z., Xu, T., and Lin, L. (2017). Decentralized navigation of multiple agents based on orca and model predictive control. In *2017 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 3446–3451. IEEE.

Cong, S., Wang, W., Liang, J., Chen, L., and Cai, Y. (2021). An automatic vehicle avoidance control model for dangerous lane-changing behavior. *IEEE Transactions on Intelligent Transportation Systems.*

Ferranti, L., Negenborn, R. R., Keviczky, T., and Alonso-Mora, J. (2018). Coordination of multiple vessels via distributed nonlinear model predictive control. In *2018 European Control Conference (ECC)*, pages 2523–2528. IEEE.

Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. (2016). Rotors—a modular gazebo mav simulator framework. In *Robot operating system (ROS)*, pages 595–625. Springer.

Hakobyan, A., Kim, G. C., and Yang, I. (2019). Risk-aware motion planning and control using cvar-constrained optimization. *IEEE Robotics and Automation Letters*, 4(4):3924–3931.

Hakobyan, A. and Yang, I. (2021). Wasserstein distributionally robust motion control for collision avoidance using conditional value at risk. *IEEE Transactions on Robotics.*

Huang, S., Teo, R. S. H., and Tan, K. K. (2019). Collision avoidance of multi unmanned aerial vehicles: A review. *Annual Reviews in Control*, 48:147–164.

Kamel, M., Alonso-Mora, J., Siegwart, R., and Nieto, J. (2017). Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243. IEEE.

Lucia, S., Tătulea-Codrean, A., Schoppmeyer, C., and Engell, S. (2017). Rapid development of modular and sustainable nonlinear model predictive control solutions. *Control Engineering Practice*, 60:51–62.

Luis, C. E., Vukosavljev, M., and Schoellig, A. P. (2020). Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611.

Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V. (2010). The grasp multiple micro-uav testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65.

Park, J. and Kim, H. J. (2020). Online trajectory planning for multiple quadrotors in dynamic environments using relative safe flight corridor. *IEEE Robotics and Automation Letters*, 6(2):659–666.

Rey, F., Pan, Z., Hauswirth, A., and Lygeros, J. (2018). Fully decentralized admm for coordination and collision avoidance. In *2018 European Control Conference (ECC)*, pages 825–830. IEEE.

Sabatino, F. (2015). Quadrotor control: modeling, nonlinearcontrol design, and simulation.

Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57.

Zhang, X., Liniger, A., and Borrelli, F. (2020). Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983.

Zhu, H. and Alonso-Mora, J. (2019). Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783.

Zhu, H., Claramunt, F. M., Brito, B., and Alonso-Mora, J. (2021). Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 6(2):2256–2263.