DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

Decentralized Navigation of Multiple Quad-rotors using Model Predictive Control

IMRAN KHAN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING

Abstract

In this thesis, we develop a model predictive control (MPC) scheme for the navigation of multiple quadrotors in an environment with obstacles. The overall control scheme is decentralized, since each quadrotor calculates its own signal based on local information. The MPC constraints take care of collision with the static obstacles, inter-agent collisions as well as input saturations. Firstly, we formulate and solve the problem using a nonlinear MPC framework, where the agents and the obstacles are modelled as 3D spheres. Secondly, to deal with complexity issues, we linearize the model and constraints by employing polyhedral sets, and we solve the problem with linear MPC. Thirdly, we use a mixed logical dynamical (MLD) framework to solve our problem, which is then incorporated into a hybrid MPC problem. The performance of the proposed solutions is demonstrated through computer simulations and real-time experiments.

Abstrakt

I denna uppsats utvecklar vi en modellprediktiv reglerstrategi (MPC) för navigering av multipla quadrotor-drönare i en miljö med hinder. Den övergripande reglerstrategin är decentraliserad då varje quadrotor beräknar sin styrsignal baserad på lokal information. Bivillkoren i MPC-formuleringen tar hänsyn till kollisioner med statiska objekt, kollisioner mellan agenter samt villkor för insignal. Reglerproblemet formuleras och löses genom ett olinjärt MPC-ramverk där agenterna samt hindren är modellerade som 3D-sfärer. Vidare, för att hantera formuleringens komplexitet, linjäriseras modellen och bivillkoren uttrycks via polyhedra mängder vilket möjliggör för en linjär MPC. Slutligen används ett ramverk för system av typen mixed logical dynamical (MLD) systems for att formulera regleringen som ett hybrid-MPC-problem. De föreslagna lösningarna är utvärderade genom datorsimuleringar och realtidsexperiment.

Acknowledgements

There are many people at the automatic control department, who guided me whenever I needed their help. First of all, I would like to thank my examiner Professor Dimos V. Dimarogonas who gave me an opportunity to work on real quad-rotors at smart mobility lab KTH under the supervision of Christos Verginis.

A special thanks to my supervisor Christos Verginis, who encouraged me a lot during my thesis and always motivated and helped me at the difficult times. This work wouldn't be possible to complete without you.

I also want to thank Pedro F. Lima, a PhD student at automatic control department for his help. I am also grateful to Pedro Roque, a research engineer at smart mobility lab for his help while carrying out real experiments on quad-rotors.

Last but not least, I would like to express gratitude to my wife who always stood with me throughout the whole journey of my Master program at KTH. Your love and care encouraged me to overcome the hurdles of this journey.

Imran Khan
July 2017
KTH-Stockholm

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the recent past, quad-rotors have become the topic of interest in research institutions. Quad-rotors are finding applications in every field ranging from military and police to fire control departments and from monitoring the agricultural sector to the coverage of an incident or event by electronic media. These useful applications have led the manufacturers to produce excessive number of quad-copters. With this increase in the production of quad-rotors, there is a need to develop algorithms for decentralized navigation of multiple quad-copters in a constrained environment such that they can avoid the hurdles in the form of static and dynamic obstacles.

This thesis is primarily focused on obstacle avoidance in scenarios of multiple quad-copters navigation as shown in Fig. 1.1. As the quad-copter approaches the static obstacle or other quad-copter, a control algorithm maneuvering the quad-copter finds the optimal path such that the obstacles in its path are avoided. There are several model-dependent and model-independent approaches that have been used for collision avoidance. Commonly used among them are the potential field method [25] and predictive control method [4] based on the dynamic model of the quad-rotor.

In this thesis, the main goal is to formulate the model predictive control (MPC) problem Eq. (1.1) based on the position and size information of static and dynamic obstacles, dynamic model of quad-rotor and state and input constraint as follows:

$$\begin{aligned} &\text{minimize:} && \text{cost function} \\ &\text{subject to:} && \text{dynamics of quad-rotor} \\ &&& \text{collision avoidance} \\ &&& \text{input and state constraints} \end{aligned} \tag{1.1}$$

The solution of the optimization problem defined in Eq.(1.1) gives a sequence of optimal inputs at each time instant that make the quad-rotor follow an optimal path free of obstacles.

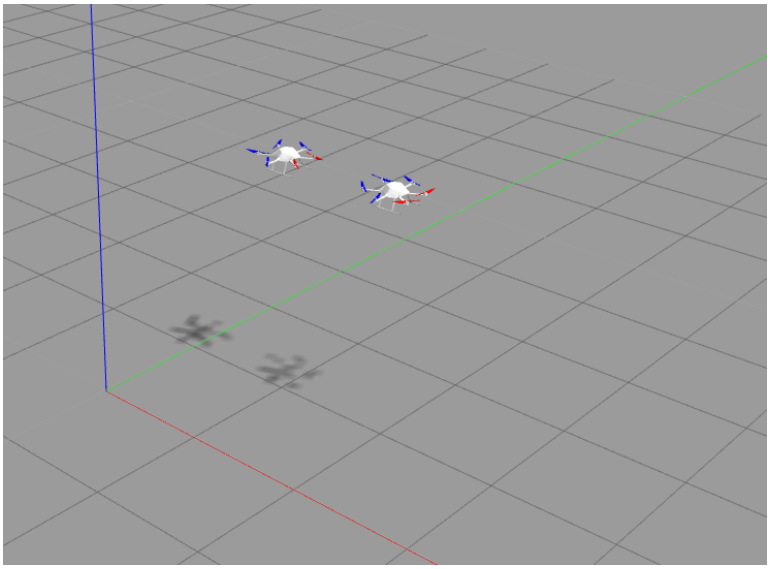

Figure 1.1: Visualization of 2x quad-copter navigation in Gazebo simulator

It has been noticed that model predictive control is becoming popular for the control of autonomous vehicles such as cars [8], [19] and quad-rotors [9] and a lot of research has been done in the recent past on the quad-rotor navigation and obstacle avoidance.

1.1 Literature Overview

Since 2000, various control methods have been suggested for decentralized or centralized motion and trajectory tracking of quad-rotors in a constrained environment. The aim of the research work in this field is to find a control scheme that allows the states of quad-copter to converge to the reference trajectories while keeping them in a safe region.

In 2009, a hierarchical approach is developed for autonomous and decentralized navigation of formation of quad-rotors in [5] subject to collision avoidance constraints. In this method, the hybrid MPC controller is deployed at upper level for each vehicle based on the dynamic model of the quad-rotor which is operated at slower sampling rate. This controller in turn gives the way points to the lower linear MPC stabilizing controller with an integral action running at the higher sampling rate which regulates the vehicle to the desired way points. In 2011, the same author extends his previous work and introduces a hierarchical method which uses linear time varying (LTV) MPC instead of hybrid MPC at the upper level in [6]. These approaches are only limited to the decentralized navigation of multiple quad-rotors in a leader follower approach.

The hierarchical model predictive control (MPC) approach for flight control and stabilization of formation of UAVs while avoiding the inter agent collision by sharing the information with each other is proposed in [10].

Robust Model Predictive Control (RMPC) approach is adopted for the deployment of a quad-rotor in [1], which considers the avoidance of static obstacles in the

local map generated by perception algorithm subject to the constraints on inputs and outputs. Furthermore, this work also uses a relaxation method to formulate the RMPC in terms of linear matrix inequalities (LMI), which makes the the number of LMI constraints to grow linearly with the number of uncertainties. Thus, this approach is computationally efficient compared to previous work in terms of dealing the uncertainties. But on the other hand, it is limited to single agent navigation and static obstacle avoidance.

In 2013, navigation of a small helicopter between two points while avoiding static obstacle is formulated as a hybrid MPC problem in [18]. For the obstacle avoidance, a hybrid model is considered as a piecewise affine (PWA) form, which is then converted to a mixed logical dynamical (MLD) model. The extent of this paper is only for a helicopter navigation and static obstacle avoidance.

In 2016, a control strategy based on decentralized navigation functions integrated with temporal logic is proposed for motion planning of a formation of quad-rotors while guaranteeing the inter-agent collision avoidance in [25].

Control algorithm is derived based on the combination of sliding model control and linear quadratic control in [15] to navigate multiple air vehicles in a leader follower approach. [26] uses reachable set for air vehicles to define inter-agent collision avoidance algorithms.

In [14] and [23], a nonlinear MPC problem is formulated for navigation and control of quad-rotors lacking the obstacle and inter-agent collision avoidance algorithms.

Algorithms for inter-agent obstacle avoidance by adopting velocity obstacle method for quad-copters while following the air rules are proposed in [16] and [17] without, however, dealing with static obstacles.

In this document, MPC based schemes are developed for the navigation of multiple quad-rotors in a decentralized fashion subject to static and dynamic obstacle avoidance and constraints on states and inputs. First the problem is formulated as a non-linear predictive control scheme based on the non-linear dynamic model of quad-rotors. Since non-linear MPC problem is non-convex, it is then converted to a linear MPC problem subject to linear model and constraints on states and inputs. In the end, the method suggested in [18] is extended to multiple quad-rotors navigation and a hybrid MPC scheme is derived for the desired problem by converting whole system into mixed logical dynamical system (MLD).

1.2 System Description

For this thesis, we consider N quad-rotors occupying a body sphere of radius r_i $\forall i \in \{1, 2, ...\mathrm{N}\}$, navigating in a bounded region X defined as:

$$\mathrm{X} = \{x \in \mathbb{R}^3 \mid x_{min} \leq x \leq x_{max}\} \tag{1.2}$$

and it is convex set.

Moreover, there exist K > N spherical regions of interests π_k in this bounded region X with radius $r_{\pi_k} > r_i$ $\forall k \in \{1, 2, ...\mathrm{K}\}$ as shown in the Fig. 1.2.

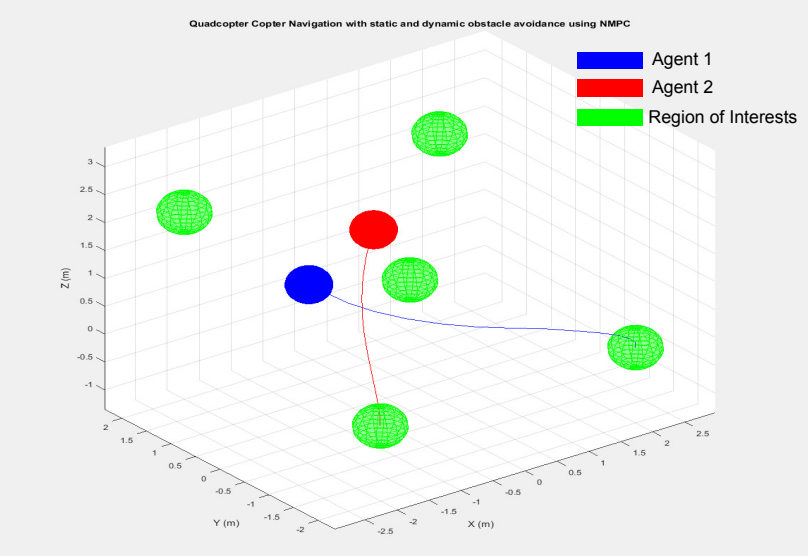

Figure 1.2: Bounded region showing 2x quad-rotors and 5x region of interests

It is assumed that each agent has a sensing radius R $> \max_{i,j=\{1,2,..N\}}(r_i + r_j)$. Therefore, each agent knows the position of all other agents within its sensing radius R. Furthermore, it is considered that each quad-rotor has a priority number p_i $\forall i \in \{1, 2, ...N\}$. Hence, during navigation, priority is given to the quad-copters ranked with higher priority numbers, whenever two or more quad-rotors approach each other in the workspace and are within the sensing radius of each other.

In addition to this, each agent knows the positions of all the regions of interests. Initially, each quad-rotor is in a region of interest and plans to navigate to other region of interest. Thus, during navigation, all the regions of interests π_k except the initial and goal regions of interests are considered to be the static obstacles.

1.3 Problem Statement

The problem tackled in this thesis is the decentralized navigation of multiple quad-rotors in a bounded region with the special purpose of inter-agents collision avoidance and collision avoidance with static obstacles. We aim to design a controller that steers the quad-rotors to the goal region of interests subject to input and state constraints.

In brief, the problem that is addressed in this document is to find the sequence of control inputs that solves the following optimization problem:

$$\begin{aligned} &\text{minimize:} && \text{distance to the goal region of interest} \\ &\text{subject to:} && \text{dynamics of quad-rotor} \\ &&& \text{static obstacle avoidance} \\ &&& \text{inter-agent collision avoidance} \\ &&& \text{input and state constraints} \end{aligned} \tag{1.3}$$

1.4 Methodology

In this thesis, the problem defined in Eq. (1.3) is first designed as a non-linear MPC optimization problem subject to the quad-rotor dynamics, where the agents and static obstacles are modelled as spheres. The non linear MPC is computationally in-efficient and may be non-convex. Therefore, a linear MPC scheme is derived for solving the problem (1.3), which is based on a linear model of the quad-rotors and the obstacles and agents are modelled as polyhedrons which define the linear constraints on the states. Furthermore, the problem defined in Eq. (1.3) is solved as a hybrid MPC problem subject to a mixed logical dynamical system. For this, the working region is converted into two regions; the flying zone and the obstacle zone. The hybrid model is obtained by considering two different dynamics for the obstacle and flying zones. This hybrid model is then converted into a mixed logical dynamical system MLD. For the static obstacle avoidance, the method proposed in [18] for a small scale helicopter model is extended to decentralized navigation of multiple quad-rotors.

For the dynamic obstacle avoidance it is proposed, that whenever two or more quad-rotors encounter each other with in their respective sensing radius, quad-rotor(s) with low priority number will stop and hover until the quad-rotors having higher priority leave their respective sensing radius. Meanwhile, the lower priority quad-rotor(s) will be treated as obstacles by the higher priority quad-rotor within their sensing radius.

1.5 Thesis outline

In this part, the structure of thesis is outlined:

Chapter 2: Modelling of Quad-rotor
This chapter presents the dynamic model of the quad-rotor. This model is then linearized in order to use it in the design of linear and hybrid model predictive controllers.

Chapter 3: Model Predictive Control
In this chapter, the basic working principle of model predictive control (MPC) scheme is given. A brief introduction is given about the predictive controllers that are used in this document. These include non-linear MPC, linear MPC and hybrid MPC.

Chapter 4: MPC for decentralized navigation and obstacle avoidance
This chapter intends to introduce the predictive controllers for multi-agent decentralized navigation subject to constraints. First, the non-linear MPC controller is derived subject to the non-linear model where the agents and obstacles are modelled as spheres and the considered collisions and input saturations act as non-linear constraints. A linear MPC controller is then designed subject to a linearized version of

the quad-rotor model and linear constraints on the states, where the obstacles are modelled as polyhedrons. For the hybrid MPC, first the hybrid model is designed and is then converted into a MLD by introducing logical variables for each obstacle. Finally, the hybrid MPC is presented for navigation subject to MLD.

Chapter 5: Simulations
This chapter is dedicated to the simulation results of all the derived controllers.

Chapter 6: Experimental evaluation
This chapter concerns the experimental evaluation of the proposed controllers.

Chapter 7: Conclusion
Finally, in this chapter, conclusions and possible future developments of the project are made.

Chapter 2

Modelling of Quad-rotor

This chapter depicts the operating principle of a quad-rotor and presents the differential equations-based dynamic model of a quad-rotor. At the end the non-linear model is linearized to get the linear model of the quad-rotor.

2.1 Mathematical model of quad-rotor

A quad-rotor is controlled by the rotation speeds of four motors during flight as shown in Fig. 2.1. Each motor M_i produces a thrust f_i and torque τ_{M_i} and rotates with angular speed ω_i. There are four types of movements; throttle movements, roll movements, pitch movements and yaw movements. Throttle movements lead to forces in the upward direction which move the quad-rotor up and down. Such movements are attained by increasing (or decreasing) the speed of all motors by the same amount. The pitch and roll movements in helicopters are controlled by a mechanical device known as swashplate, whereas same movements are achieved in quad-rotors by varying the angular speed of four motors of quad-rotor. In quad-rotors, the pitch movement is controlled by varying the speeds of the front (M1) and back (M3) motors, while the left (M4) and right (M2) motors are used to control the roll movement. In order to move the quad-rotor in the yaw direction the speed of the front and rear motors is increased while the speed of the left and right motors is decreased at the same time or vice versa. Furthermore, to reduce the gyroscopic and aerodynamic effects, the front and back motors rotate counter-clockwise while the left and right motors rotate clockwise as indicated in Fig. 2.1.

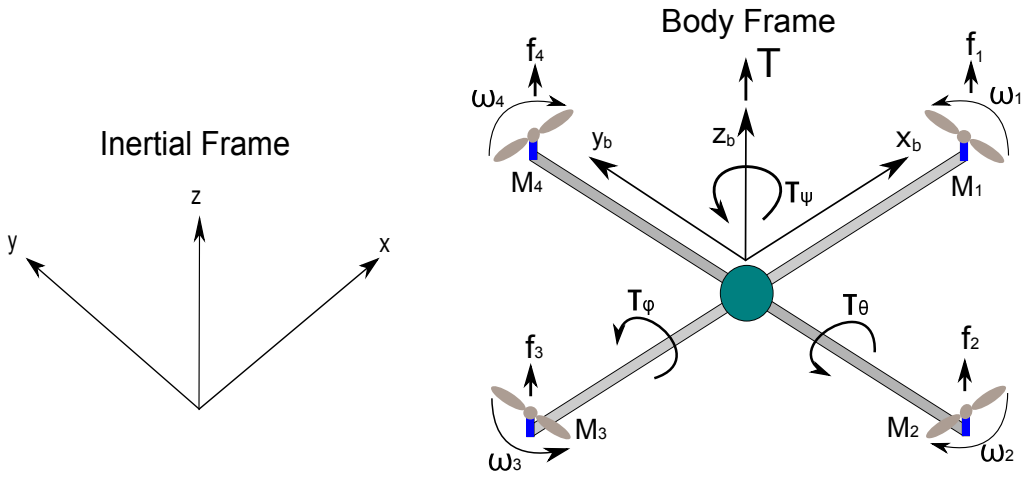

Figure 2.1: Inertial frame (x, y, z) and quad-rotor body frame (x_b, y_b, z_b), where f_i and $\omega_i \ \forall \ i \in [1, 4]$ represent the forces and angular velocities in the body frame respectively produced by each motor M_i while τ_ϕ, τ_θ and τ_ψ are the three rotational torques in the same frame and T is the thrust in the body z-axis.

Let the absolute position of the quad-rotor in the inertial frame is expressed by

$$\mathbf{q} = [\xi^T, \eta^T]^T$$

where $\xi = [x, y, x]^T$ is the linear position of the quad-rotor in the inertial frame and $\eta = [\phi, \theta, \psi]^T$ represents the Euler angles, i.e., the orientation of the quad-rotor in the inertial frame as shown in Fig. 2.1.

The thrust T and rotational torques can be defined as [13] [22]

$$\begin{aligned} \mathrm{T} &= \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} \omega_i^2 \\ \tau_\phi &= lk(\omega_4^2 - \omega_2^2) \\ \tau_\theta &= lk(\omega_3^2 - \omega_1^2) \\ \tau_\psi &= \sum_{i=1}^{4} \tau_{M_i} \end{aligned}$$

where, l is the distance of the rotor from the center of the quad-rotor, ω_i is the angular speed of the i_{th} motor and $k > 0$ is the lift constant.

The rotor torque τ_{M_i} around its rotation axis is opposed by the aerodynamics drag, such that

$$\tau_{M_i} = I_M \dot{\omega}_i + b\omega_i^2$$

in which I_M is the moment of inertia of rotor and b is the drag constant. For quasi-stationary movements, $\dot{\omega}$ is very small and neglected [13], and hence

$$\tau_{M_i} = b\omega_i^2$$

The rotation axis of each rotor also moves in the body-fixed frame result in gyroscopic torques τ_G

$$\tau_G = \sum_{i=1}^{4} I_M(\Omega \times E_z)\omega_i$$

where, E_z is the unit vector in inertial z-axis.

The rotation matrix from the body frame to the inertial frame is mathematically defined as [22]

$$\boldsymbol{R} = \begin{bmatrix} \mathrm{C}\psi\mathrm{C}\theta & \mathrm{C}\psi\mathrm{S}\theta\mathrm{S}\phi - \mathrm{S}\psi\mathrm{C}\phi & \mathrm{C}\psi\mathrm{S}\theta\mathrm{C}\phi + \mathrm{S}\psi\mathrm{S}\phi \\ \mathrm{S}\psi\mathrm{C}\theta & \mathrm{S}\psi\mathrm{S}\theta\mathrm{S}\phi + \mathrm{C}\psi\mathrm{C}\phi & \mathrm{S}\psi\mathrm{S}\theta\mathrm{C}\phi - \mathrm{C}\psi\mathrm{S}\phi \\ -\mathrm{S}\theta & \mathrm{C}\theta\mathrm{S}\phi & \mathrm{C}\theta\mathrm{C}\phi \end{bmatrix} \tag{2.1}$$

where, S and C stands for trigonometric function sin and cos respectively.

The angular velocities $\dot{\eta}$ in the inertial frame are related to the angular velocity vector Ω in the body frame by:

$$\Omega = \boldsymbol{W}\dot{\eta} \tag{2.2}$$

where $\Omega = [p, q, r]^T$, $\dot{\eta} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ and transformation matrix $\boldsymbol{W}$ which relates velocities in both frames is [22]

$$\boldsymbol{W} = \begin{bmatrix} 1 & 0 & -\mathrm{S}\theta \\ 0 & \mathrm{C}\phi & \mathrm{C}\theta\mathrm{S}\phi \\ 0 & -\mathrm{S}\phi & \mathrm{C}\theta\mathrm{C}\phi \end{bmatrix} \tag{2.3}$$

There are different methods adopted by researchers for modelling the quad-rotors. The most popular approaches among them are the Newton-Euler and the Euler-Lagrange methods[13]. In this project, we use the Newton-Euler approach to model the dynamics of quad-rotor.

In order to obtain the mathematical model of the quad-rotor, consider a solid body moving in a 3-D space Fig. 2.1 and is subject to a thrust T in the direction of the body $z-$axis and three torques τ_ϕ, τ_θ and τ_ψ.

In the inertial frame, the quad-rotor is accelerated due to the thrust vector F_B, the gravity and friction force f_d. As the thrust acts on the quad-rotor in the body frame, the rotation matrix $\boldsymbol{R}$ Eq. (2.1) is used to convert the thrust from the body to inertial frame.

Thus, according to Newton's equation of 3-D motion, the following holds in the inertial frame

$$m\ddot{\xi} = -mgE_z + \boldsymbol{R}F_B$$

$$\ddot{\xi} = -gE_z + \frac{\boldsymbol{R}F_B}{m}$$

where $E_z = [0, 0, 1]^T$, $\ddot{\xi} = [\ddot{x}, \ddot{y}, \ddot{z}]^T$, $F_B = [0, 0, \mathrm{T}]^T$ and $\boldsymbol{R}$ is the rotation matrix.

The drag force f_d induced by the air resistance is included in order to give a more realistic behaviour of the quad-rotor. So the above equation can be written as

$$\ddot{\xi} = -gE_z + \frac{\boldsymbol{R}F_B}{m} - \frac{f_d}{m} \tag{2.4}$$

where, f_d is diagonal matrix of the following form:

$$f_d = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{2.5}$$

where, k_x, k_y and k_z are coefficients that relate the linear velocities to the drag force. Eq. (2.4) becomes

$$\begin{aligned} \ddot{x} &= \frac{\mathrm{T}}{m}(\mathrm{C}\psi\mathrm{S}\theta\mathrm{C}\phi + \mathrm{S}\psi\mathrm{S}\phi) - \frac{k_x}{m}\dot{x} \\ \ddot{y} &= \frac{\mathrm{T}}{m}(\mathrm{S}\psi\mathrm{S}\theta\mathrm{C}\phi - \mathrm{C}\psi\mathrm{S}\phi) - \frac{k_y}{m}\dot{y} \\ \ddot{z} &= \frac{\mathrm{T}}{m}\mathrm{C}\theta\mathrm{C}\phi - g - \frac{k_z}{m}\dot{z} \end{aligned} \tag{2.6}$$

According to Euler's equation, the total torque applied to the quad-rotor in the body frame is

$$\begin{aligned} &\mathbf{I}\dot{\Omega} + \Omega \times (\mathbf{I}\Omega) = \tau - \tau_G \\ &\dot{\Omega} = \mathbf{I}^{-1}(\tau - \tau_G - \Omega \times (\mathbf{I}\Omega)) \end{aligned} \tag{2.7}$$

where, $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]$ is the generalized torque acting on the body in the body frame, τ_G is the gyroscopic torque and $\mathbf{I}$ is the moment of inertia. It is assumed that the quad-rotor has symmetric structure and therefore, the inertia matrix is diagonal

$$\mathbf{I} = \begin{bmatrix} \mathbf{I}_{xx} & 0 & 0 \\ 0 & \mathbf{I}_{yy} & 0 \\ 0 & 0 & \mathbf{I}_{zz} \end{bmatrix} \tag{2.8}$$

Moreover, it is also assumed that the moment of inertia of each motor I_M is very small and hence the gyroscopic torque τ_G can be neglected in designing the model [13]. Hence, Eq. (2.7) becomes

$$\begin{aligned} \dot{p} &= \frac{(\mathbf{I}_{yy} - \mathbf{I}_{zz})qr}{\mathbf{I}_{xx}} + \frac{\tau_\phi}{\mathbf{I}_{xx}} \\ \dot{q} &= \frac{(\mathbf{I}_{zz} - \mathbf{I}_{xx})pr}{\mathbf{I}_{yy}} + \frac{\tau_\theta}{\mathbf{I}_{yy}} \\ \dot{r} &= \frac{(\mathbf{I}_{xx} - \mathbf{I}_{yy})pq}{\mathbf{I}_{zz}} + \frac{\tau_\psi}{\mathbf{I}_{zz}} \end{aligned} \tag{2.9}$$

where, $(\dot{p}, \dot{q}, \dot{r})$ is the angular acceleration vector in the body frame with respect to inertial frame. Following Eq. (2.2), we obtain

$$\begin{aligned} \dot{\eta} &= \boldsymbol{W}^{-1}\Omega \\ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 1 & \mathrm{S}\phi\mathrm{T}\theta & \mathrm{C}\phi\mathrm{T}\theta \\ 0 & \mathrm{C}\phi & -\mathrm{S}\phi \\ 0 & \frac{\mathrm{S}\phi}{\mathrm{C}\theta} & \frac{\mathrm{C}\phi}{\mathrm{C}\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{aligned}$$

Expanding above equation results in

$$\begin{aligned}\dot{\phi} &= p + \mathrm{S}\phi\mathrm{T}\theta q + \mathrm{C}\phi\mathrm{T}\theta r \\ \dot{\theta} &= \mathrm{C}\phi q - \mathrm{S}\phi r \\ \dot{\psi} &= \frac{\mathrm{S}\phi}{\mathrm{C}\theta} q + \frac{\mathrm{C}\phi}{\mathrm{C}\theta} r\end{aligned} \tag{2.10}$$

where, $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ are the angular velocities around x, y, and z axis, respectively in the inertial frame.

2.2 State Space Model

By introducing the parameters $u = \dot{x}$, $v = \dot{y}$ and $w = \dot{z}$, the set of equations Eq. (2.6), Eq. (2.9) and Eq. (2.10) can be written as:

$$\begin{aligned}\dot{x} &= u \\ \dot{y} &= v \\ \dot{z} &= w \\ \dot{u} &= \frac{\mathrm{T}}{m}(\mathrm{C}\psi\mathrm{S}\theta\mathrm{C}\phi + \mathrm{S}\psi\mathrm{S}\phi) - \frac{k_x}{m}u \\ \dot{v} &= \frac{\mathrm{T}}{m}(\mathrm{S}\psi\mathrm{S}\theta\mathrm{C}\phi - \mathrm{C}\psi\mathrm{S}\phi) - \frac{k_y}{m}v \\ \dot{w} &= \frac{\mathrm{T}}{m}\mathrm{C}\theta\mathrm{C}\phi - g - \frac{k_z}{m}w \\ \dot{p} &= \frac{(\mathbf{I}_{yy} - \mathbf{I}_{zz})qr}{\mathbf{I}_{xx}} + \frac{\tau_\phi}{\mathbf{I}_{xx}} \\ \dot{q} &= \frac{(\mathbf{I}_{zz} - \mathbf{I}_{xx})pr}{\mathbf{I}_{yy}} + \frac{\tau_\theta}{\mathbf{I}_{yy}} \\ \dot{r} &= \frac{(\mathbf{I}_{xx} - \mathbf{I}_{yy})pq}{\mathbf{I}_{zz}} + \frac{\tau_\psi}{\mathbf{I}_{zz}} \\ \dot{\phi} &= p + \mathrm{S}\phi\mathrm{T}\theta q + \mathrm{C}\phi\mathrm{T}\theta r \\ \dot{\theta} &= \mathrm{C}\phi q - \mathrm{S}\phi r \\ \dot{\psi} &= \frac{\mathrm{S}\phi}{\mathrm{C}\theta} q + \frac{\mathrm{C}\phi}{\mathrm{C}\theta} r\end{aligned} \tag{2.11}$$

or in compact state-space form:

$$\dot{X} = f(X, U)$$

where $X = (x, y, z, u, v, w, p, q, r, \phi, \theta, \psi)^T$ is the state vector and $U = (T, \tau_\phi, \tau_\theta, \tau_\psi)$ is the thrust vector acting as control input.

2.2.1 Linear Model

The solution of the state space model Eq. (2.11) may give non-unique results as trigonometric functions appear in a non-elementary way. Hence, the model is simplified by assuming that the Euler angles are negligibly small. Therefore by using

the Tailor series expansion, sin function is approximated by its argument and the cos function by unity. The state space model Eq. (2.11) is then simplified to

$$\begin{aligned}
\dot{x} &= u \\
\dot{y} &= v \\
\dot{z} &= w \\
\dot{u} &= \frac{\mathrm{T}}{m}(\theta + \psi\phi) - \frac{k_x}{m}u \\
\dot{v} &= \frac{\mathrm{T}}{m}(\psi\theta - \phi) - \frac{k_y}{m}v \\
\dot{w} &= \frac{\mathrm{T}}{m} - g - \frac{k_z}{m}w \\
\dot{p} &= \frac{(\mathbf{I}_{yy} - \mathbf{I}_{zz})qr}{\mathbf{I}_{xx}} + \frac{\tau_\phi}{\mathbf{I}_{xx}} \\
\dot{q} &= \frac{(\mathbf{I}_{zz} - \mathbf{I}_{xx})pr}{\mathbf{I}_{yy}} + \frac{\tau_\theta}{\mathbf{I}_{yy}} \\
\dot{r} &= \frac{(\mathbf{I}_{xx} - \mathbf{I}_{yy})pq}{\mathbf{I}_{zz}} + \frac{\tau_\psi}{\mathbf{I}_{zz}} \\
\dot{\phi} &= p + \phi\theta q + \theta r \\
\dot{\theta} &= q - \phi r \\
\dot{\psi} &= \phi q + r
\end{aligned} \tag{2.12}$$

or in a compact state-space form:

$$\dot{X} = \hat{f}(X, U)$$

Linearization is performed on the simplified state space model, Eq. (2.12). The nonlinear system $\hat{f}(X, U)$ is linearized around the equilibrium points.

$$\dot{X} = 0 \Longrightarrow \hat{f}(\bar{X}, \bar{U}) = 0$$

Solving this equation gives the equilibrium point around which the linearization is peformed.

$$\begin{aligned}
\bar{X} &= [\bar{x}, \bar{y}, \bar{z}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \\
\bar{U} &= [\mathrm{mg}, 0, 0, 0]
\end{aligned}$$

The state matrix related to the linear model is

$$\mathbf{A} = \frac{\partial \hat{f}(X,U)}{\partial X}\Big| X = \bar{X} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_x}{m} & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{g} & 0 \\ 0 & 0 & 0 & 0 & -\frac{k_y}{m} & 0 & 0 & 0 & 0 & -\mathrm{g} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{k_z}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.13}$$

and the input matrix is

$$\mathbf{B} = \frac{\partial \hat{f}(X,U)}{\partial U}\Big|_{X=\bar{X}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{\mathrm{m}} & 0 & 0 & 0 \\ 0 & \frac{1}{\mathbf{I}_{xx}} & 0 & 0 \\ 0 & 0 & \frac{1}{\mathbf{I}_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{\mathbf{I}_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.14}$$

Hence the linear state space model is

$$
\begin{aligned}
\dot{x} &= u \\
\dot{y} &= v \\
\dot{z} &= w \\
\dot{u} &= -\frac{k_x}{m}u + \mathrm{g}\theta \\
\dot{v} &= -\frac{k_y}{m}v - \mathrm{g}\phi \\
\dot{w} &= -\frac{k_z}{m}w + \frac{\mathrm{T}}{m} \\
\dot{p} &= \frac{\tau_\phi}{\mathbf{I}_{xx}} \\
\dot{q} &= \frac{\tau_\theta}{\mathbf{I}_{yy}} \\
\dot{r} &= \frac{\tau_\psi}{\mathbf{I}_{zz}} \\
\dot{\phi} &= p \\
\dot{\theta} &= q \\
\dot{\psi} &= r
\end{aligned}
\tag{2.15}
$$

Chapter 3

Model Predictive Control

In Model predictive control (MPC), an open loop optimization problem is solved iteratively over a finite time horizon. The input to the problem is the current state of the system while the output is a sequence of optimal control inputs calculated over the finite time horizon. The solution of the optimal control problem depends on the predicted states of the system. These states of the system are predicted over a discrete time horizon using a dynamic model of the plant. At each time instant the problem is solved and only the first sample of the optimal control is fed to the plant and the remaining samples are discarded. This process of predicting the states and solving the problem for the optimal control is iteratively repeated at a constant interval of time. The general abstract of the model predictive control scheme is shown in Fig. 3.1

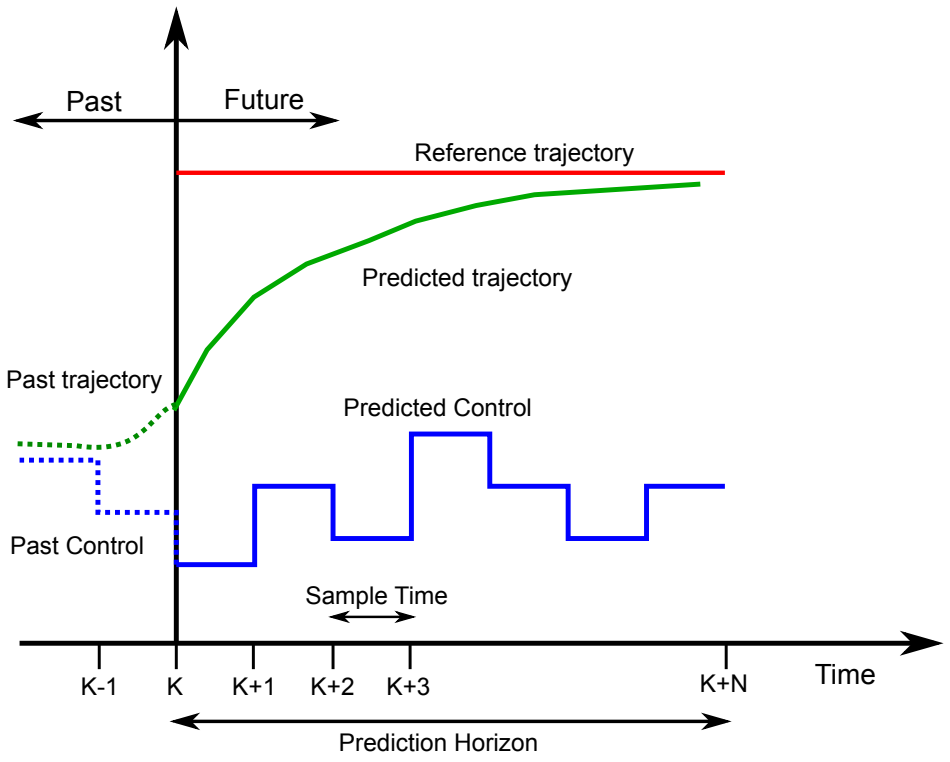

Figure 3.1: Abstract of Model Predictive Control

The advantage of MPC over the other control schemes is that it can deal with the constraints explicitly. But at the same time such type of optimization methods are computationally inefficient and depend on the horizon length and number of states of the model. Typically, such techniques have been used in the industries where the plants or models have slow dynamics and the sampling time is more than the computational time of the problem. More recently, MPC has been applied to systems with fast dynamics such as autonomous vehicles [8] [19] while guaranteeing the closed loop stability. In this chapter, we give a brief overview of nonlinear MPC, linear MPC and hybrid MPC and analyse them for achieving stability.

3.1 Nonlinear Model Predictive Control (NMPC)

The model predictive control scheme which is based on the non-linear dynamics of the plant is known as nonlinear model predictive control (NMPC). The cost function may or may not be quadratic and depends on the type of the problem. NMPC is used to predict the system behaviour subject to non-linear constraints on the inputs and states. Linear MPC scheme is preferred mostly because it takes less computation time. But sometimes it is better to use the nonlinear model instead of the linear one in order to improve the quality specifications of the product because non-linear models are rich in describing the dynamics of plant. On the other hand, NMPC is inherently a state feedback control approach [11] and is based on the assumption that full state information is available or must be estimated from available measurements for solving the optimization problem. In case of output feedback NMPC, one cannot guarantee the closed loop stability of the system even if the controller and observer both are stable as there doesn't exist any separation principle for nonlinear systems [11].

As output feedback NMPC has stability issues for non-linear models [11], in this document state feedback NMPC approach is used. Hence, it is assumed that full system state information is available. This section describes the state feedback NMPC and is limited to the sampled-data NMPC where an open loop optimal problem is solved at discrete sampling instants.

3.1.1 Problem formulation - NMPC

Consider a discrete time non-linear system of the form

$$x(k+1) = f(x(k), u(k)), \quad x(0) = x_0 \tag{3.1}$$

where, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is locally Lipschitz continuous and describes the discretized system dynamics at discrete time instants $k \geq 0$, $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$ are state and input vectors at time instant k. Furthermore, the system in Eq. (3.1) is subject to state and input constraints:

$$\begin{aligned} x(k) &\in \mathrm{X}, \quad \forall k \geq 0 \\ u(k) &\in \mathrm{U}, \quad \forall k \geq 0 \end{aligned} \tag{3.2}$$

where, X and U are convex sets of the form

$$\begin{aligned} \mathrm{X} &= \{x \in \mathbb{R}^n \mid x_{min} \leq x \leq x_{max}\} \\ \mathrm{U} &= \{x \in \mathbb{R}^m \mid u_{min} \leq u \leq u_{max}\} \end{aligned} \tag{3.3}$$

The problem of driving the states of the system defined by Eq. (3.1) to origin while optimizing the control can be formulated as

$$\begin{aligned} &\underset{u}{\text{minimize}}\ J(x,u) \\ &\text{subject to:} \\ &\quad x(k+1) = f(x(k),u(k)), \forall\ k \in [0,\mathrm{N}-1] \\ &\quad u(k) \in \mathrm{U}, \quad \forall\ k \in [0,\mathrm{N}-1] \\ &\quad x(k) \in \mathrm{X}, \quad \forall\ k \in [0,\mathrm{N}] \\ &\quad x(0) = x_0 \end{aligned} \tag{3.4}$$

where, N is the length of the prediction horizon over which the cost function J is minimized. The cost function $J(.)$ is given by

$$J(x,u) = \sum_{k=0}^{\mathrm{N}-1} F(x(k),u(k)) + E(x(\mathrm{N})) \tag{3.5}$$

The stage cost $F(.)$ and terminal penalty term $E(.)$ are often defined as

$$\begin{aligned} F(x,u) &= x^T \mathrm{Q} x + u^T \mathrm{R} u \\ E(x) &= x^T \mathrm{Q}_f x \end{aligned}$$

where the performance weights are positive semi definite and positive definite i.e., $\mathrm{Q} \succeq 0$, $\mathrm{Q}_f \succeq 0$ and $\mathrm{R} \succ 0$.

Theoretically, it would better to set prediction horizon N in problem Eq. (3.4) to infinity. This would lead to minimizing of the infinity cost which is computationally inefficient. Therefore, a finite horizon length is used in solving the problem Eq. (3.4). But this leads to the difference between the actual closed-loop states and inputs and predicted open-loop ones, even if there is no mismatch between plant and its model and no disturbances. This difference between the predicted and actual closed loop trajectories have two effects. Firstly, the actual goal to minimize an infinite horizon problem is not achieved and secondly, one cannot guarantee the closed-loop stability of NMPC [11].

Ideally, one would prefer to use the NMPC scheme that guarantees closed-loop stability and, if possible, approximates infinite horizon NMPC approach. Different approaches that are commonly used to achieve the closed-loop stability of NMPC are discussed in [11]. These methods include **Infinite Horizon NMPC**; where the horizon length is set to infinity and leads to computational problems, **Finite Horizon NMPC with Guaranteed Stability**; where, the terminal constraint set X_f and terminal state penalty term $E(.)$ are added in problem formulation Eq. (3.4) to guarantee stability and **Quasi-infinite horizon NMPC**; where, the

terminal constraint set X_f and terminal state penalty term $E(.)$ are obtained offline by stabilizing the linear control law locally.

In this project, the *Finite Horizon NMPC with Guaranteed Stability* approach is used because of its simplicity to amend in the control problem.

3.1.2 Finite Horizon NMPC with Guaranteed Stability

In this approach, the standard formulation of NMPC Eq. (3.4) is modified in order to achieve the closed loop stability. This is done by adding suitable equality/ inequality constraints, sometimes called stability constraints, and additional penalty terms. One of the possibilities is to add a zero terminal constraint to the standard formulation of NMPC problem. (3.4) i.e.,

$$x(\mathrm{N}) = 0$$

The main disadvantage of adding the zero terminal constraint is that the optimization problem (3.4) becomes infeasible, as the predicted state of the system is forced to go to zero in finite time which is not possible. To over come this issue, most of the strategies use the terminal region constraint set i.e.,

$$x(\mathrm{N}) \in \mathrm{X}_f$$

and/or a terminal cost $E(.)$

3.1.3 Stability of sampled-data NMPC

The following theorem states the conditions for the stability of the closed loop system [11].

Theorem 3.1: Suppose that

(a) the terminal region $\mathrm{X}_f \subseteq \mathrm{X}$ is closed and contains 0, and that the terminal cost $E(x)$ is locally Lipschitz continuous and positive semi-definite

(b) the terminal constraint set X_f and terminal cost $E(x)$ are chosen in a way that $\forall x \in \mathrm{X}_f$, there exists an admissible input u such that $x(\mathrm{N}) \in \mathrm{X}_f$ and that the minimal value of cost function defined in Eq. (3.5) is decreasing.

(c) the NMPC open-loop optimal control problem is feasible for $t = 0$.

Then in the closed-loop system Eq. (3.4), $x(k)$ converges to the origin for $k \to \infty$.

Hence, the terminal region constraint enforces feasibility of the optimal control problem at the next sampling instant and thus makes the minimal value of the cost function to strictly decrease and hence stability can be achieved.

3.2 Linear Model Predictive Control (LMPC)

The optimal control problem defined in Eq. 3.4 is computationally expensive and cannot be implemented in real-time applications. Sometimes, it is also difficult to ensure the closed loop stability of the system for the complex nonlinear models. Moreover, the computational time is so high that NMPC cannot be used for plants that have fast dynamics. Therefore, linear MPC is used instead of NMPC for real time applications with faster dynamics, where the linear time invariant model is approximated from the nonlinear dynamics of the plant model. Moreover, the closed loop stability of the system can be also achieved by using output feedback LMPC as there exists a separation principle for linear models. The linear MPC is a type of MPC in which the model of the plant as well as the constraints on states and inputs are linear.

3.2.1 Problem formulation - LMPC

Consider a discrete time linear system of the form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3.6}$$

where, $x(k) \in \mathrm{X}\ \forall\ k > 0$ and $u(k) \in \mathrm{U}\ \forall\ k > 0$. The sets X and U are bounded convex sets.

The problem of driving the states of the linear system defined by Eq. (3.6) to origin while optimizing the cost can be written as

$$\underset{u}{\text{minimize}} \sum_{k=0}^{\mathrm{N}-1} x^T \mathrm{Q} x + u^T \mathrm{R} u + x^T \mathrm{Q}_f x$$

subject to:

$$\begin{aligned} & x(k+1) = Ax(k) + Bu(k), \forall\ k \in [0, \mathrm{N}-1] \\ & u(k) \in \mathrm{U}, \quad \forall\ k \in [0, \mathrm{N}-1] \\ & x(k) \in \mathrm{X}, \quad \forall\ k \in [0, \mathrm{N}] \\ & x(0) = x_0 \\ & x(\mathrm{N}) \in \mathrm{X}_f \end{aligned} \tag{3.7}$$

where, N is the length of the prediction horizon over which the cost function is minimized. The penalty matrices (or performance weights) are the positive semi definite and positive definite i.e., $\mathrm{Q} \succeq 0$, $\mathrm{Q}_f \succeq 0$ and $\mathrm{R} \succ 0$, so that the control problem is convex [21]. The terminal cost and the terminal constraints are added in the problem Eq. (3.7) to achieve the closed loop system stability.

3.2.2 Stability of linear MPC

The following theorem states the conditions for the stability of the closed loop system using linear MPC [21].

Theorem 3.2: Consider the linear system Eq.(3.6) with (A,B) reachable. Let the penalty matrix for control input is positive definite i.e., R$\succ$0 and (A,$Q^{1/2}$) be observable. Then, if $Q_f = P$ where

$$P = A^T PA + Q - (B^T PA)^T (R + B^T PB)^{-1} B^T PA \tag{3.8}$$

the control problem Eq. (3.7) results in an asymptotically stable closed-loop system for all values of N>1.

Hence, the terminal cost ensures the stability of closed loop system for linear MPC problem if the terminal penalty matrix is chosen as the solution to the algebraic riccati equation (ARE) Eq. (3.8).

3.3 Hybrid Model Predictive Control (HMPC)

Often, systems are described by the linear dynamical equations and linear inequalities which involve real and logical variables. These systems are known as mixed logical dynamical systems MLDs [3]. The optimal control problem of such systems can be solved using linear MPC and heuristic rules inferred by practical operations like if-else statements. This approach is sometimes not feasible and inefficient when there are a lot of if-else statements.

This section describes a predictive control scheme [3] for MLDs known as hybrid model predictive control. The optimization problem defined for MLDs can be solved as mixed integer quadratic problem (MIQP). This section starts with a brief introduction about boolean algebra and mixed logical dynamical MLDs systems and then ends with deriving the problem formulation problem for HMPC subject to MLDs.

3.3.1 Boolean Algebra

Consider a discrete time linear system of the form

$$x(k+1) = f(x(k)) \tag{3.9}$$

where $x(k) \in \mathrm{X}\ \forall\ k > 0$ and X is a bounded set. Let

$$M = \max(f(x)) \tag{3.10}$$

$$m = \min(f(x)) \tag{3.11}$$

To establish a link between the logical variables and dynamical system, we associate a logical variable $\delta \in \{0, 1\}$ for $f(x)$ which has a value of 1 if $f(x) \leq 0$ and 0 otherwise. It can be easily verified that

$$[f(x) \leq 0] \ \leftrightarrow\ [\delta = 1] \Leftrightarrow \begin{cases} f(x) \leq M(1-\delta) \\ f(x) \geq \epsilon + (m-\epsilon)\delta \end{cases} \tag{3.12}$$

where, $\epsilon > 0$ represents the tolerance in the process/ plant.

Sometimes, the statements involve the multiplication of a binary and real variable such as the term $\delta f(x)$. In order to convert it into set of linear inequalities, we introduce an auxiliary real variable z and replace the statement $\delta f(x)$ by z which satisfies

$$\begin{aligned} [\delta = 0] &\rightarrow [z = 0] \\ [\delta = 1] &\rightarrow [z = f(x)] \end{aligned} \tag{3.13}$$

Eq. (3.13) can be written as set of linear inequalities of the following form

$$\begin{aligned} z &\leq M\delta \\ z &\geq m\delta \\ z &\leq f(x) - m(1-\delta) \\ z &\geq f(x) - M(1-\delta) \end{aligned} \tag{3.14}$$

3.3.2 Mixed Logical Dynamical System

Mixed logical dynamical (MLD) systems describe the evaluation of the linear discrete time systems which involve the logical variables. The general form of a mixed logical dynamical system [3] is

$$\begin{aligned} x(k+1) &= Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \\ y(k) &= Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \\ E_2\delta(k) + E_3z(k) &\leq E_1u(k) + E_4x(k) + E_5 \end{aligned} \tag{3.15}$$

where in Eq.(3.15), the first line is a state-update equation, the second is an output equation, and the third is a set of linear inequalities which describe the switching conditions between the different modes of the hybrid model. In this model, x is the state vector, u is the input vector, δ represents the binary variables, z is set the auxiliary variables and $A,\ B_1,\ B_2,\ B_3,\ C,\ D_1,\ D_2,\ D_3,\ E_2,\ E_3,\ E_1,\ E_4$ and E_5 are matrices of suitable dimensions. Hysdel tool box can be used to generate these matrices.

3.3.3 HYSDEL Tool Box

HYSDEL (HYbrid SYstem DEscription Language) is a tool box that models the mixed logical dynamical (MLD) system [24]. It uses a high-level modeling language to describe the whole system and then translates it into mixed logical dynamical (MLD) system suitable for solving the desired optimization problem.

3.3.4 Optimal Control of MLD systems

The optimal control problem for MLDs can be defined as

$$\begin{aligned} &\underset{u}{\text{minimize}}\ J(x, u, \delta, y, z) \\ &\text{subject to:} \\ &\quad \text{MLD dynamics described by Eq.(3.15) and} \\ &\quad x(\mathrm{N}) \in \mathrm{X}_f \end{aligned} \tag{3.16}$$

where, N is the prediction horizon and X_f is terminal state. The cost function $J(x, u, \delta, y, z)$ is usually defined as

$$\begin{aligned} J(x, u, \delta, y, z) = \sum_{k=0}^{N-1} & x^T(k)Q_1x(k) + u^T(k)Q_2u(k) + \delta^T(k)Q_3\delta(k) \\ & + z^T(k)Q_4z(k) + y^T(k)Q_5y(k) \end{aligned} \tag{3.17}$$

where, Q_1, Q_2, Q_3, Q_4 and Q_5 are positive definite matrices. The optimization problem for hybrid systems defined in Eq. (3.16) can be solved as a mixed integer quadratic program (MIQP).

Chapter 4

MPC for decentralized navigation and obstacle avoidance

This section describes the methods that are adopted to design the MPC based controllers for decentralized navigation of multiple quad-rotors in a constraint environment. First, a non-linear MPC problem is derived subject to the nonlinear model of the quad-rotor and non-linear state constraints. Afterwards, a linear predictive control scheme is derived to steer the aerial vehicles to the desired goal points subject to linear constraints on states and inputs. In the end, the whole system is converted into a mixed logical dynamical system and a hybrid MPC controller is derived for the same problem subject to a mixed logical dynamical model of the system. In the linear and hybrid MPC, the obstacles and agents are modelled as polyhedral sets while in non-linear MPC, these are modelled as spheres.

4.1 Non-linear MPC and obstacle avoidance

The first predictive controller that we derive to solve our desired problem is based on the non-linear dynamical model of the quad-rotor Eq.(2.11). The nonlinear MPC optimization problem defined in Eq. (3.4) which is based on the non-linear dynamics needs to be modified for navigating a quad-rotor to the goal region of interest such that static and dynamic obstacles can be avoided. Hence, in order to formulate the nonlinear MPC control problem for navigating a quad-rotor to the goal region of interest and obstacle avoidance, we need to model the obstacles as constraints on the states.

4.1.1 Obstacle modelling

For a nonlinear MPC problem, the obstacles are modelled as spherical objects. These obstacles include the static as well as dynamic obstacles. Let there are K number of static obstacles positioned at $(x_{si}, y_{si}, z_{si})\ \forall\ i \in \mathrm{K}$, and M number of dynamic obstacles positioned at $(x_{dj}, y_{dj}, z_{dj})\ \forall\ j \in \mathrm{M}$ respectively. It is assumed that all static and dynamic obstacles have body radius r_s and r_d respectively then

the spherical objects can be defined as:

$$\begin{aligned}(x - x_{si})^2 + (y - y_{si})^2 + (z - z_{si})^2 = r_s^2 \quad \forall\ i \in \mathrm{K} \\ (x - x_{dj})^2 + (y - y_{di})^2 + (z - z_{di})^2 = r_d^2 \quad \forall\ j \in \mathrm{M}\end{aligned} \tag{4.1}$$

The static and dynamic obstacles are the non-linear constraints on states x, y and z. As we want our vehicle to never enter the regions defined by Eq. (4.1), the constraints on the states x, y and z for the static and dynamic obstacles avoidance are

$$(x - x_{si})^2 + (y - y_{si})^2 + (z - z_{si})^2 > R_s^2 \quad \forall\ i \in \mathrm{N} \tag{4.2}$$

and

$$(x - x_{dj})^2 + (y - y_{dj})^2 + (z - z_{dj})^2 > R_d^2 \quad \forall\ j \in \mathrm{N} \tag{4.3}$$

where, R_s is equal to or more than the sum of the radial sizes of the static and dynamic obstacles for static obstacle avoidance. Similarly, in order to avoid the dynamic obstacles, R_d must be equal to or more than the sum of the radial sizes of encountering agents.

4.1.2 Problem Formulation - NMPC

Consider the nonlinear model of the quad-rotor Eq. (2.11) and define it a function $f(x, u)$. Let x_0 be the initial position of the quad-rotor and $\mathbf{r}$ the reference point where we want our quad-rotor to navigate. In other words, x_0 and $\mathbf{r}$ represent the positions of two different regions of interest π_i and π_j, $i \neq j$, such that the agent navigates from x_0 to $\mathbf{r}$ subject to non-linear dynamics $f(x, u)$ while avoiding all types of obstacles.

The constraints Eq. (4.2) and Eq. (4.3) depend on the positions of static and dynamic obstacles. In this thesis, it is assumed that each agent has a certain sensing radius and it knows the positions of other agents only when they are with in its sensing radius. Hence we define two optimization problems. The first one is only dedicated to static obstacle avoidance while the second one is solved only when there is a dynamic obstacle in the vicinity of the agent.

The non-linear MPC problem named as "NPMC 1" afterwards for navigating a quad-rotor to goal region of interest $\mathbf{r}$ from initial region of interest x_0 subject to

non-linear model and static obstacles avoidance is

$$\begin{aligned}
&\underset{u}{\text{minimize}} \sum_{k=0}^{\mathrm{N}-1}(x(k)-\mathbf{r})^T\mathrm{Q}(x(k)-\mathbf{r})+u(k)^T\mathrm{R}u(k)+\\
&\qquad (x(\mathrm{N})-\mathbf{r})^T\mathrm{Q}_f(x(\mathrm{N})-\mathbf{r})\\
&\text{subject to:}\\
&\qquad x(k+1)=f(x(k),u(k)),\forall\ k\in[0,\mathrm{N}-1]\\
&\qquad u(k)\in\mathrm{U},\quad \forall\ k\in[0,\mathrm{N}-1]\\
&\qquad x(k)\in\mathrm{X},\quad \forall\ k\in[0,\mathrm{N}]\\
&\qquad (x-x_{si})^2+(y-y_{si})^2+(z-z_{si})^2>R_s^2\quad \forall\ i\in\mathrm{K}\\
&\qquad x(0)=x_0\\
&\qquad x(\mathrm{N})\in\mathrm{X}_f
\end{aligned}\tag{4.4}$$

and the non-linear MPC controller "NPMC 2" for steering the agent to the desired goal point while avoiding static and dynamic obstacles avoidance subject to non-linear dynamics is

$$\begin{aligned}
&\underset{u}{\text{minimize}} \sum_{k=0}^{\mathrm{N}-1}(x(k)-\mathbf{r})^T\mathrm{Q}(x(k)-\mathbf{r})+u(k)^T\mathrm{R}u(k)+\\
&\qquad (x(\mathrm{N})-\mathbf{r})^T\mathrm{Q}_f(x(\mathrm{N})-\mathbf{r})\\
&\text{subject to:}\\
&\qquad x(k+1)=f(x(k),u(k)),\forall\ k\in[0,\mathrm{N}-1]\\
&\qquad u(k)\in\mathrm{U},\quad \forall\ k\in[0,\mathrm{N}-1]\\
&\qquad x(k)\in\mathrm{X},\quad \forall\ k\in[0,\mathrm{N}]\\
&\qquad (x-x_{si})^2+(y-y_{si})^2+(z-z_{si})^2>R_s^2\quad \forall\ i\in\mathrm{K}\\
&\qquad (x-x_{dj})^2+(y-y_{dj})^2+(z-z_{dj})^2>R_d^2\quad \forall\ j\in\mathrm{M}\\
&\qquad x(0)=x_0\\
&\qquad x(\mathrm{N})\in\mathrm{X}_f
\end{aligned}\tag{4.5}$$

where, N is the prediction horizon length and X_f is the terminal constraint set which is the spherical region around the goal region of interest. It is chosen in a way that the optimization problem is feasible and at the same time it is computationally efficient. The matrices Q, R and Q_f are positive definite matrices.

4.1.3 Flow chart of NMPC Algorithm

The flow chart of the NMPC algorithm for the decentralized motion of agents is shown in Fig. 4.1.

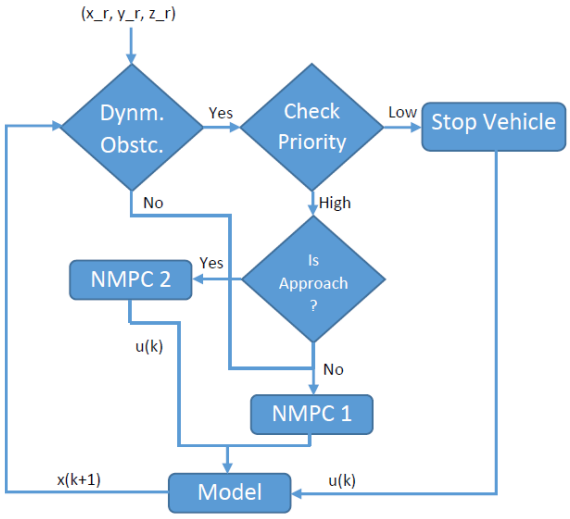

Figure 4.1: Flow chart of NMPC Algorithm for static and dynamic obstacle avoidance algorithm, NMPC 1 for Static Obstacle avoidance and NMPC 2 for static and dynamic obstacle avoidance

At each time instant, it is checked whether there is a dynamic obstacle with in the sensing radius of an agent or not. If there is no dynamic obstacle within the sensing radius, then the problem is solved using NMPC 1 which only takes care of the static obstacle avoidance. If there exist a dynamic obstacle in the vicinity of an agent, the priority is compared to the priority of other agent. If its priority is low compared to other agent, then vehicle is slow down and stopped. If it has a higher priority compared to the other agent, it is investigated if other agent is approaching or not. If it is approaching, then optimization problem is solved with NMPC 2, which handles the static as well as dynamic obstacle avoidance otherwise NMPC 1 is used to solve the problem. Only the first sample of the control input obtained from the NMPC 1 or NMPC 2 is fed to the plant/ model afterwards.

4.2 Linear MPC and obstacle avoidance

This section describes the derivation of the optimal control scheme for moving the quad-rotor to the goal region of interest subject to the linear model of quad-rotor Eq. (2.15) and linear constraints such that the static and dynamic obstacles are avoided during the decentralized motion of multiple agents. The linear MPC problem Eq. (3.7) which is based on the linear dynamics and linear constraints on states and inputs is updated for static and dynamic obstacle avoidance.

4.2.1 Obstacle modelling

In order to avoid the obstacles using linear MPC, the avoidance of obstacles is modelled as linear constraints on states x, y and z. In this project, we model the

obstacles as polyhedrons for linear MPC problem. Consider an obstacle centered at position (x_c, y_c, z_c) having facet length l as shown in Fig. 4.2.

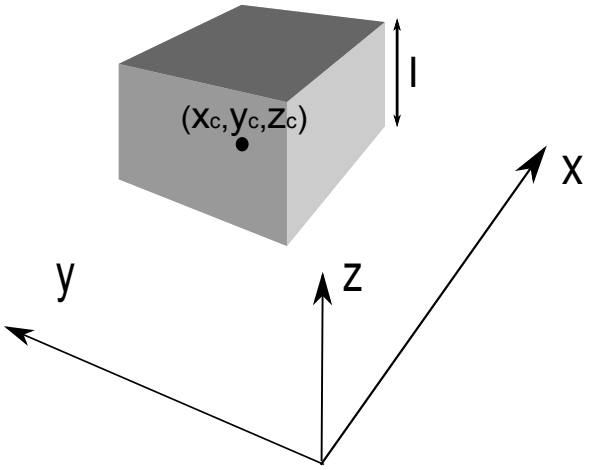

Figure 4.2: Obstacle modelled as polyhedron

It is assumed, that the size l of all facets of obstacle are equal. The region spanned by this obstacle can be written in the form of the following linear inequalities.

$$\begin{aligned} x &\leq x_c + l/2 \\ y &\leq y_c + l/2 \\ z &\leq z_c + l/2 \\ x &\geq x_c - l/2 \\ y &\geq y_c - l/2 \\ z &\geq z_c - l/2 \end{aligned} \tag{4.6}$$

which can be written as

$$\mathrm{PZ} \leq \mathrm{d} \tag{4.7}$$

where, $\mathrm{Z} = [x, y, z]^T$, $\mathrm{P} = [I_3, -I_3]^T$ is a 6x3 matrix and d is a 6x1 vector equal to

$$d = \begin{bmatrix} x_c + l/2 \\ y_c + l/2 \\ z_c + l/2 \\ -x_c + l/2 \\ -y_c + l/2 \\ -z_c + l/2 \end{bmatrix} \tag{4.8}$$

In order to avoid collisions with static obstacles and other agents, it is assumed that the length of facet l is equal to the sum of the radial sizes of the static obstacle and dynamic obstacles for static obstacle avoidance while it is equal to the sum of the radial sizes of two agents when dynamic obstacle is avoided.

4.2.2 Obstacle avoidance

In order to avoid the static and dynamic obstacles, the linear optimization problem is defined and solved at each sample point. At each time instant, the previous predicted trajectories are checked by employing heuristic rules such as if-else statements. If these trajectories enter the obstacle region the constraints are put on the states related to the linear position of quad-rotor. In order to understand the concept of obstacle avoidance using LMPC, consider the obstacle in the path of trajectory of agent in 2-D as shown in Fig.4.3.

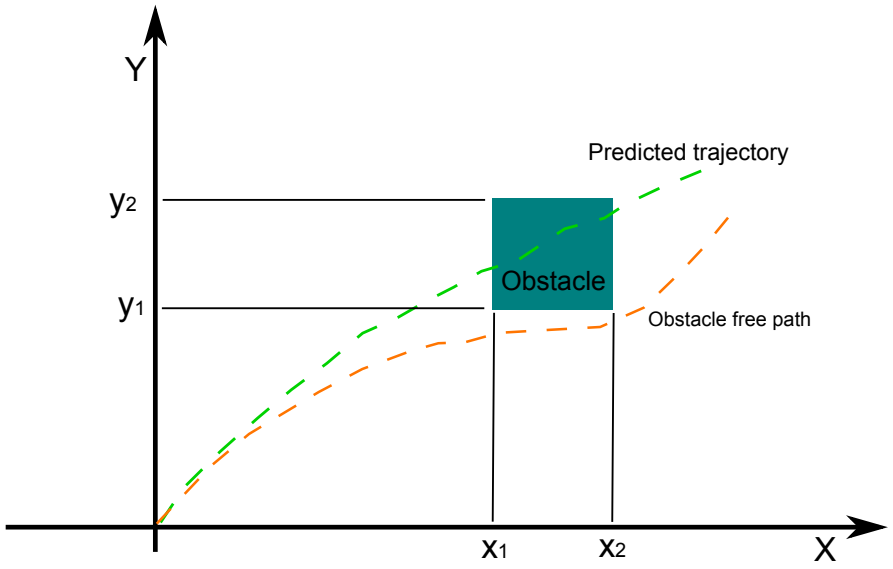

Figure 4.3: Obstacle in the predicted trajectory of quad-rotor

where, (x_1,x_2) and (y_1,y_2) specify the boundaries of the obstacle and $X_{pred} = [x_{pred},y_{pred}]$ is the predicted trajectory for prediction horizon length at previous time instant. The proposed algorithm for obstacle avoidance using linear MPC is

Algorithm 1 Obstacle avoidance algorithm

while $k \leq N$ **do**
  if $x_{pred}(k) > x_1$ & $x_{pred}(k) < x_2$ **then**
    if $y_{pred}(k) > y_1$ & $y_{pred}(k) < y_2$ **then**
      if $y_{pred}(k) > z_1$ & $z_{pred}(k) < z_2$ **then**
        if $abs(y(k) - y_1) < abs(y_2 - y(k))$ **then**
          $y(k) < y_1$
        else
          $y(k) > y_2$
        end if
      end if
    end if
  end if
end while

where, $\mathrm{y}(k) < \mathrm{y}_1$ or $\mathrm{y}(k) > \mathrm{y}_2$ are the constraints on the states of the agent and N is the prediction horizon. If the initial state and the final goal of the targets are such that they are aligned on x-axis then constraints are put on $\mathrm{y}(k)$, and if they are aligned on y-axis, constraints are put on $\mathrm{x}(k)$. In general form, the constraints on the states for obstacle avoidance using linear MPC can be written as

$$\mathrm{AZ} < b \tag{4.9}$$

where, $\mathrm{Z} = (x, y, z)$.

This algorithm is for only one obstacle and can be easily extended for the multiple obstacles in the same way.

4.2.3 Problem formulation - LMPC

Consider a LTI system of the form

$$\bar{x}(k+1) = \bar{A}\bar{x}(k) + \bar{B}\bar{u}(k) \tag{4.10}$$

For error-free tracking in the z-axis, an integral state in the dynamic model is added. Hence the augmented model is

$$\begin{bmatrix} \bar{x}(k+1) \\ \bar{x}_I(k+1) \end{bmatrix} = \begin{bmatrix} \bar{A} & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} \bar{x}(k) \\ \bar{x}_I(k) \end{bmatrix} + \begin{bmatrix} \bar{B} \\ 0 \end{bmatrix} \bar{u}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) \tag{4.11}$$

which leads to

$$x(k+1) = Ax(k) + Bu(k) + Dr(k) \tag{4.12}$$

The optimal control problem of making the quad-rotor go to reference region of interest $\mathbf{r}$ subject to the linear model Eq. (4.12) and linear constraints on the states of the form Eq. (4.9) can be written as

$$\underset{u}{\text{minimize}} \sum_{k=0}^{\mathrm{N}-1} (x - \mathbf{r})^T \mathrm{Q}(x - \mathbf{r}) + u^T \mathrm{R}u + (x - \mathbf{r})^T \mathrm{Q}_f (x - \mathbf{r})$$

subject to:

$$\begin{aligned} & x(k+1) = Ax(k) + Bu(k) + Dr(k), \forall\ k \in [0, \mathrm{N}-1] \\ & u(k) \in \mathrm{U}, \quad \forall\ k \in [0, \mathrm{N}-1] \\ & x(k) \in \mathrm{X}, \quad \forall\ k \in [0, \mathrm{N}] \\ & \mathrm{AZ} < b, \\ & x(0) = x_0, \\ & x(\mathrm{N}) \in \mathrm{X}_f \end{aligned} \tag{4.13}$$

where, $\mathrm{AZ} < b$ defines the constraints on the states $\mathrm{Z} = [x, y, z]^T$ such that the static and dynamic obstacles are avoided safely.

4.2.4 Flow chart of LMPC Algorithm

The flow chart of the LMPC algorithm for the decentralized motion of the agents is depicted in Fig. 4.4. At each sample time, the existence of other agents with in the sensing radius is checked. In the absence of any agent, the linear MPC problem is defined such that it only avoids static obstacles. Otherwise, based on the priority it either decides to stop or avoids the dynamic obstacle. It can be observed that the MPC problem is defined at each time sample which is based on the predicted states of the system at previous sample times in order to ensure that obstacles are avoided.

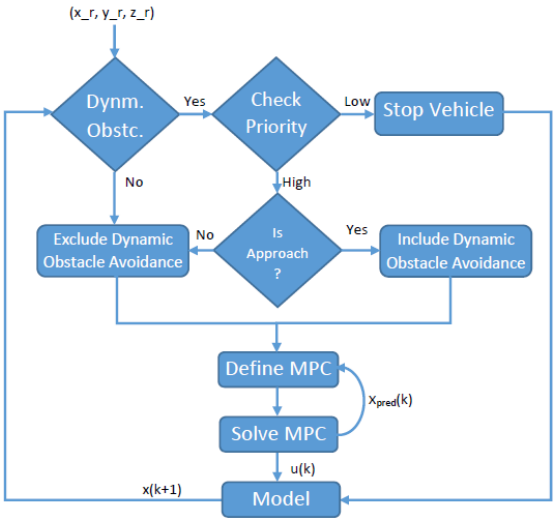

Figure 4.4: Flow chart of LMPC Algorithm for static and dynamic obstacle avoidance algorithm

4.3 Hybrid MPC and obstacle avoidance

In this section, the hybrid MPC problem defined in Eq.(3.16) is modified such that the quad-rotor navigates between two regions subject to avoidance of all types of obstacles. Here, the hybrid MPC is designed for a single static/ dynamic obstacle which can be extended for multiple obstacles. The workspace for quad-rotor navigation is shown in the Fig. 4.5.

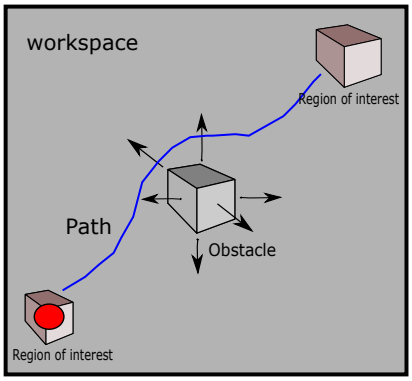

Figure 4.5: Workspace for quad-rotor navigation

4.3.1 Obstacle modelling

The obstacles are modelled as polyhedrons in the same way as we did for the linear MPC Eq. (4.6) and Eq. (4.7).

4.3.2 Hybrid Model

Consider the quad-rotor located in one of the region of interests that aims to navigate to another region of interest as shown in Fig. 4.5. In this project, the work-space shown in Fig. 4.5 is divided in to two regions i.e., the obstacle region and the flying region, with the obstacle region being the sum of individual regions of all obstacles. Let the dynamics of the quad-rotor in the flying region be described by the discrete LTI system which also includes the integral state in the z-axis

$$x(k+1) = Ax(k) + Bu(k) + Dr(k) \tag{4.14}$$

Similarly for the obstacles region, the dynamic model is

$$x(k+1) = Ix(k) \tag{4.15}$$

where I is the identity matrix that has the same dimension as the matrix A.

Therefore, the hybrid model for obstacle avoidance scenario is defined by

$$x(k+1) = \begin{cases} Ax(k) + Bu(k) + Dr(k), & \text{Flying region} \\ Ix(k), & \text{Obstacle region} \end{cases} \tag{4.16}$$

4.3.3 Mixed logical dynamical system modelling

In this section, the hybrid model obtained Eq. (4.16) is converted into a mixed logical dynamical system. A binary variable for each obstacle region can be associated as specified in Fig. 4.6.

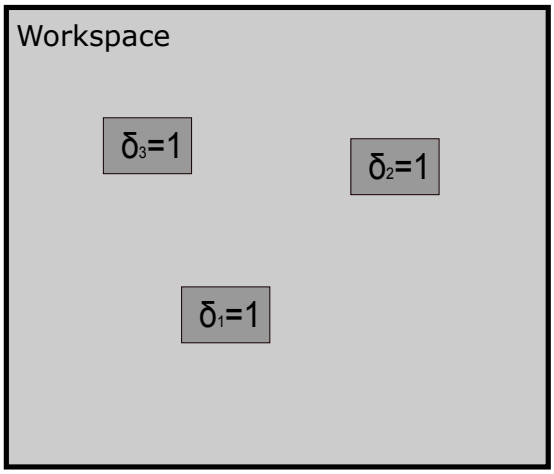

Figure 4.6: Obstacles associated with binary variables in a work-space

Hence, the Hybrid model is

$$x(k+1) = \begin{cases} Ix(k) & \text{if } (\vee\delta_i == 1) \\ Ax(k) + Bu(k) + Dr(k) & \text{else} \end{cases} \tag{4.17}$$

The hybrid dynamic model Eq. (4.17) obtained is modified as piecewise linear affine (PWA) dynamics which is then converted into an equivalent mixed logical dynamical (MLD) system [3] [18].

Following [7] [18], the model of PWA with s number of states can be written in the form as:

$$x(k+1) = \begin{cases} A_1x(k) + B_1u(k) & \text{if} \quad \delta_1(k) = 1 \\ A_2x(k) + B_2u(k) & \text{if} \quad \delta_2(k) = 1 \\ . & \\ . & \\ A_sx(k) + B_nu(k) & \text{if} \quad \delta_s(k) = 1 \end{cases} \tag{4.18}$$

subject to $x(k) \in \Omega_i$, where, $\delta_i(k) \in \{0, 1\}$, $\forall i \in s$ represent the binary variables associated to the polytope Ω_i $\forall i \in s$. $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$ represent the state and control vector respectively at time instant k. The polytopes Ω_i are the convex polyhedral sets of the form $\Omega_i = \{x : P_ix < b_i\}$ in the state vector.

The hybrid model Eq. (4.17) is a piecewise affine (PWA) system with four events each triggered by particular polytope Ω_i for specific values of the states of the plant. According to [7] [2], the piecwise affine form can be converted to the mixed logical dynamical framework using the HYSDEL tool box in MATLAB. The model MLD obtained is the desired form suitable for optimization control problems. Hence, in this project, the MLD model is obtained by using the hysdel tool box [24].

4.3.4 Problem formulation - HMPC

Consider a MLD system obtained in the previous section and **r** be the goal region of interest where we want our quad-rotor to maneuver. The optimal control problem

is defined as

$$\begin{aligned}\underset{u}{\text{minimize}} \sum_{k=0}^{\mathrm{N}-1} &(x(k)-\mathbf{r})^T \mathrm{Q}_1 (x(k)-\mathbf{r}) + (z(k)-\mathbf{r})^T \mathrm{Q}_2 (z(k)-\mathbf{r}) + \\ &(d(k)-d_f)^T \mathrm{Q}_3 (d(k)-d_f) + u(k)^T \mathrm{R} u(k) + \\ &(x(\mathrm{N})-\mathbf{r})^T \mathrm{Q}_f (x(\mathrm{N})-\mathbf{r})\end{aligned}$$

subject to:

$$\begin{aligned}&x(k+1) = Ax(k) + B1u(k) + B2\delta(k) + B3z(k) \\ &E2\delta(k) + E3z(k) \leq E1u(k) + E4x(k) + E5 \\ &x(0) = x_0 \\ &x(\mathrm{N}) \in \mathrm{X}_f\end{aligned} \tag{4.19}$$

where, N is the prediction horizon length and X_f is the terminal constraint set which defines the region around the goal region of interest. It is chosen in a way that the optimization problem is feasible and at the same time it is computationally efficient. The matrices Q_1, Q_2, Q_3, R and Q_f are positive definite matrices and d_f is the final states of the logical variables. In order to avoid the obstacles, the logical variables linked to the obstacles region are set as zero in d_f. Hence, in the control problem the binary variables related to the obstacles' region are penalized to zeros.

4.3.5 Flow chart of HMPC Algorithm

The flow chart of the HMPC algorithm for the decentralized motion of agents is almost identical to that of the NMPC, Fig. 4.7.

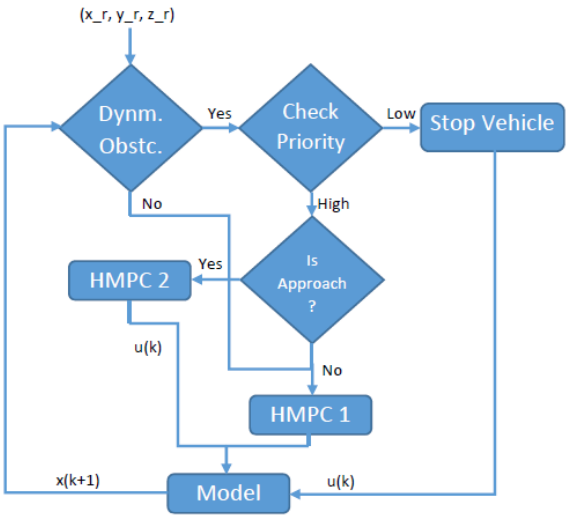

Figure 4.7: Flow chart of HMPC Algorithm for static and dynamic obstacle avoidance algorithm, HMPC 1 for Static Obstacle avoidance and HMPC 2 for static and dynamic obstacle avoidance

Two optimal control problems are also defined using HMPC which are used for avoiding the obstacles in the path while navigating between two regions of interests.

4.4 Linear MPC vs Hybrid MPC

Two hybrid MPC based controllers are defined for the desired task of navigating the quad-rotors to the goal point while avoiding all types of obstacles where the first one only avoids the static obstacles and the second one tackles the problem of both static and dynamic obstacles avoidance. In the linear MPC, the optimization problem is defined at each sample instant which makes it good for adding the obstacles online. In the hybrid MPC, the problem is defined based on the information about number of the dynamic obstacles. But on the other hand in the linear MPC, we can define/ update the problem based on the number of agents in the sensing radius. When the hybrid model is converted into the desired MLD form, the size of the matrices becomes too large and grows linearly with the number of obstacles. This makes the hybrid MPC difficult to debug and thus the structure of hybrid MPC becomes complex in comparison to linear MPC. But on the other hand, the drawback of the approach using LMPC is that we loose optimality of the optimization problem which is maintained in hybrid MPC case.

4.5 Dynamic Obstacle avoidance

In order to explain the moving obstacle avoidance, consider two agents having different priority numbers and same radius navigating in the work-space as shown in Fig. 4.8. Each quad has a certain sensing radius. Therefore, each quad will know the position of other quad-rotor within that radius. When these quads are with in the sensing radius of each other and approaching each other, they will share the priority numbers. The low priority agent will slow down and stop while the high priority agent will treat the other agent as obstacle and will avoid it. The predicted path of both agents while avoiding each other is shown in Fig. 4.8

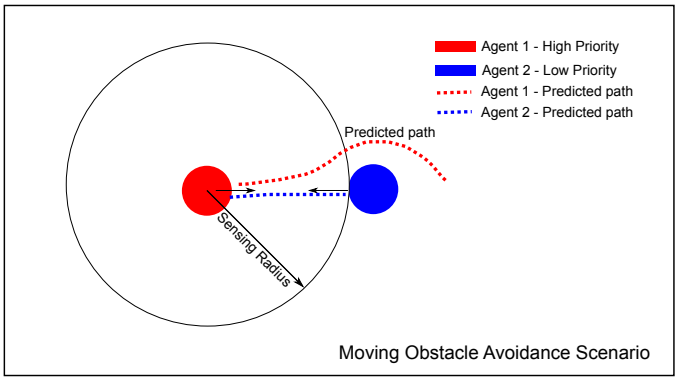

Figure 4.8: Obstacle avoidance method for other moving agents; this method is common for NMPC, LMPC and HMPC

Chapter 5

Simulation Results

In this chapter, simulation results for navigation of quad-rotors using non-linear, linear and hybrid MPC are presented. The necessary parameter values of the quad-rotor model are taken from [22]. Performance of each controller is presented and analyzed for feasibility and collision avoidance. The obstacle avoided path, trajectories of necessary states and inputs for each quad-rotor are plotted in each case. Matlab with Yalmip [20] is used for the modelling and simulation of controllers. Non-linear MPC problem is solved with non-linear solver ***Fmincon***, linear MPC with ***Quadprog*** while hybrid MPC problem is solved using ***Gurobi*** which deals with integer programming. The general parameters of the quad-rotor used during the simulations are given in the Table. 5.1.

S/No.	**Description**	**Parameter**	**Value**
1	mass of quad-rotor	m	1.56779 Kg
2	drag coefficient	k_x	0.25 Kg/sec
3	drag coefficient	k_y	0.25 Kg/sec
4	drag coefficient	k_z	0.25 Kg/sec
5	Inertia in $x-x$ axis	I_{xx}	4.856x10^{-3} Kgm^2
6	Inertia in $y-y$ axis	I_{yy}	4.856x10^{-3} Kgm^2
7	Inertia in $z-z$ axis	I_{zz}	4.856x10^{-3} Kgm^2

Table 5.1: Quad-rotor parameters

The simulation environment consists of five region of interests and two quad-rotors Fig. 5.1. Hence, for each quad-rotor, there are three static obstacles and one dynamic obstacle. Table. 5.2 lists the positions of the region of interests in the working boundary. Initially agent-1 that has high priority is placed at π_1 while the low priority agent-2 is placed at π_2 as shown in Fig. 5.1.

Region	x	y	z
π_1	-2	-2	0
π_2	2	-2	0
π_3	2	2	2
π_4	-2	2	2
π_5	1	1	2

Table 5.2: Positions of Region of interests

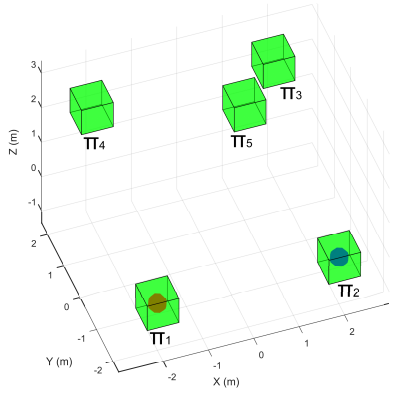

Figure 5.1: A simulation workspace showing five regions of interests and two quad-rotors positioned at π_1 and π_2

In this thesis, we study two cases for each controller. In the first case, agent-1 which has high priority navigates form region π_1 to π_3 while agent-2 is moved from π_2 to π_4. Due to the symmetrical positioning of these regions of interests, agents will collide each other in the middle of the workspace if we don't use the dynamical obstacle avoidance algorithms. Moreover, region π_5 is also in the path of agent-1 which acts as a static obstacle for agent-1. Hence in this case, the side on collision avoidance between two agents as well as the static obstacle avoidance is tested.

In the second case, the high priority agent-1 placed at π_1 navigates to π_2 while the other agent moves from π_2 to π_1. Hence a head on collision avoidance between two agents is tested in this case.

5.1 Tuning of Parameters

The parameters that have to be tuned include the prediction horizon of the problem, sampling time, sampling distance and spherical sizes of the obstacles.

5.1.1 Prediction Horizon

The prediction horizon is tuned according to the current simulation setup so that the problem doesn't become computationally expensive. Moreover, it should be

feasible so that the terminal state of the model lies with in the terminal constraint set. The prediction horizon is related to the terminal constraint set. If we choose the terminal constraint set large than we can keep the prediction horizon low and vice verse. The prediction horizon varies for all controllers.

5.1.2 Sampling time

For all the simulations, the sampling time is set as **0.1** msec. Increasing the sampling time makes the problem feasible for low prediction horizon but at the same time it also increases the sampling distance which makes it difficult to avoid the obstacles. Hence the penalty matrices of the cost functions are tuned to decrease the sampling distance if we raise the sampling time.

5.1.3 Sampling distance

Sampling distance is related to the velocity and sampling time and must be small enough than the size of the obstacle so that min of 1-2 samples can be read with in the obstacle area. The sampling distance is not constant as the velocity is not constant. There are different ways to make the velocity of the aerial vehicle small. The velocity can be be made small by penalizing the velocity deviation more in the cost function and hence the sampling distance can be made small. If the sampling time is increased, it also increases the sampling distance and hence the lateral velocities of the agents need to be penalized more in order to decrease the sampling distances. Other ways to adjust the velocity and sampling distance is by giving less control that is to penalize the control more. Hence, the weight matrices for the state deviations and inputs are tuned accordingly if the sampling time is changed. Moreover, it has been observed from the dynamical model of quad-rotor Eq. (2.15) that the velocities u and v of the quad-rotor are dependent on the states θ and ϕ respectively. Hence the velocity of the quad-rotor can also be controlled by penalizing these states more or by putting hard constraints according to demand.

5.1.4 Spherical sizes of the obstacle

One of the most important parameters that affect the simulation time and the feasibility of the optimal control problem is the sizes of the quad-rotors and static obstacles. They affect the computational time and the feasibility of the simulations. Increasing the sizes increases the computational time and hence makes the problem infeasible. The upper bounds on the spherical radius of the static and dynamic obstacles for each controller which have been able to guarantee the feasibility of the optimal control control problems are found in Table. 5.3.

S/No.	Obstacle type	Radius (m)	Controller type
1	Static	0.35	LMPC, NLMPC, HMPC
2	Dynamic	0.15	NLMPC
3	Dynamic	0.2	LMPC, HMPC

Table 5.3: Upper bounds on the sizes of the static and dynamic obstacles for each controller

5.2 Non-linear MPC results

In this section, the results of the non-linear MPC controller are presented

5.2.1 CASE-I

In this case high priority agent is moved from region π_1 to π_3 and other agent which has a low priority navigates from π_2 to π_4 as shown in Fig. 5.2. It can be seen that they avoid each other during motion. Moreover, agent 1 also avoids the static obstacle π_5. The linear positions and corresponding Euler angles of each agent are shown in Fig. 5.3. The control input required for decentralized navigation of these agents are depicted in Fig. 5.4.

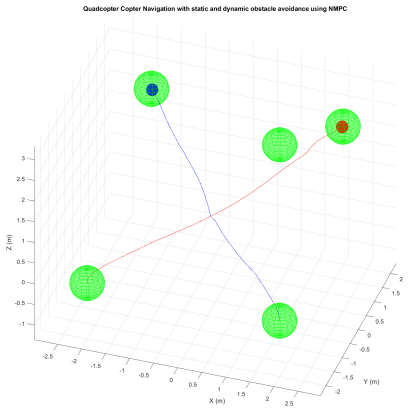

Figure 5.2: 3D trajectory showing the decentralized motion of two agents using NMPC for sideon collision avoidance between two agents and static obstacle avoidance with region π_5

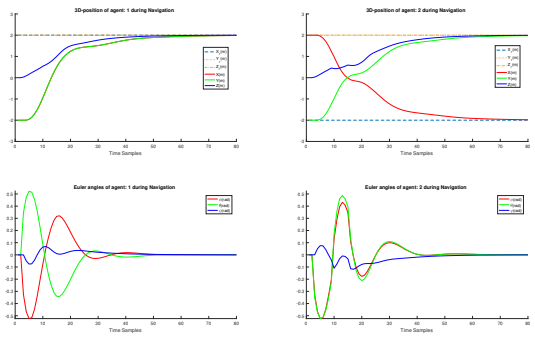

Figure 5.3: Linear positions and Euler angles of agents

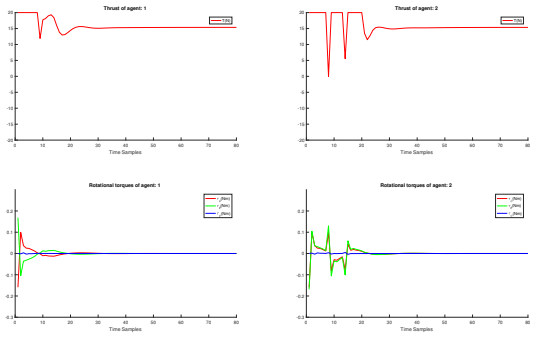

Figure 5.4: Inputs applied to agents

5.2.2 CASE-II

The head-on collision between agents are studied in this case. It can be noticed that Fig. 5.5, they avoid each other during their motion towards the goal points. Linear and angular positions, and control generated by the optimal control problem are shown in Fig. 5.6 and Fig. 5.7.

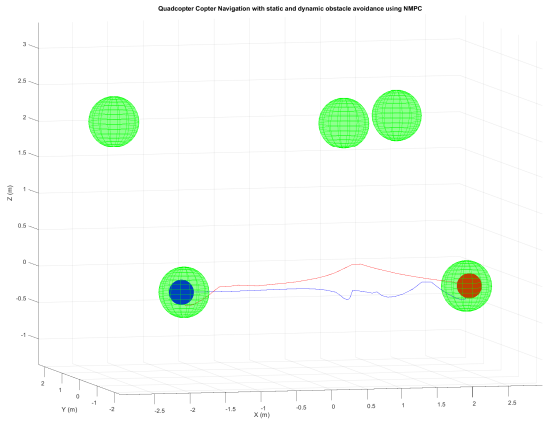

Figure 5.5: 3D trajectory showing the decentralized motion of two agents using NMPC for headon collision avoidance between two agents

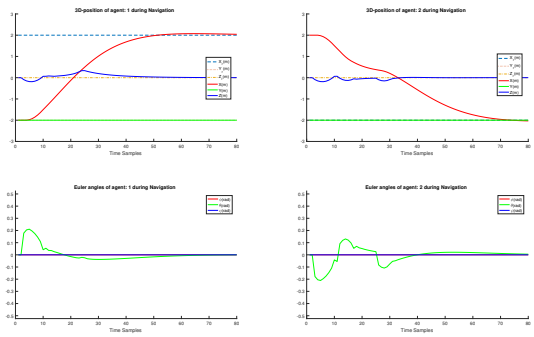

Figure 5.6: Linear positions and Euler angles of agents

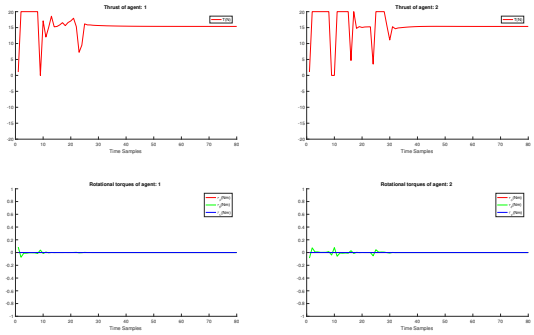

Figure 5.7: Inputs applied to agents

The simulation results for these cases solved using non-linear MPC are available at `https://youtu.be/qb7fEwLXWOA` and `https://youtu.be/inSTa70TnMA`.

5.3 Linear MPC results

In this section, we present the simulation results for both cases using linear MPC.

5.3.1 CASE-I

In this case the linear MPC based obstacle avoidance algorithm is tested for side on collision avoidance between two agents and as well as the collision with the static obstacle. During the navigation agent-1 and agent-2 have to avoid each other. Hence agent-2 having low priority slows down and stops in order to avoid the collision with high priority agent. Moreover agent-1 also avoids the static obstacle in its path before reaching the goal position as seen in Fig. 5.8.

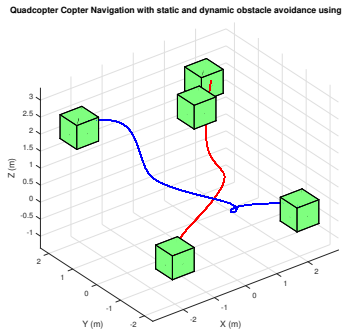

Figure 5.8: 3D trajectory showing the decentralized motion of two agents using LMPC for sideon collision avoidance between two agents and static obstacle avoidance with region π_5

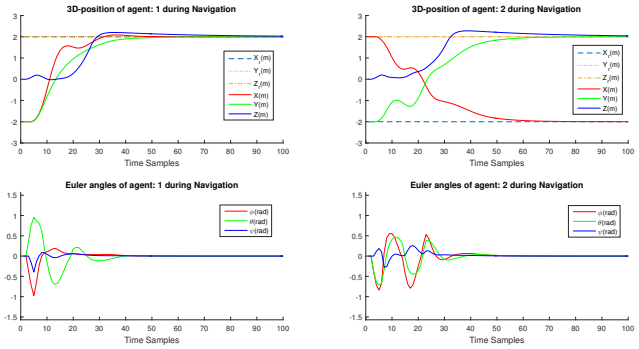

Figure 5.9: Linear positions and Euler angles of agents

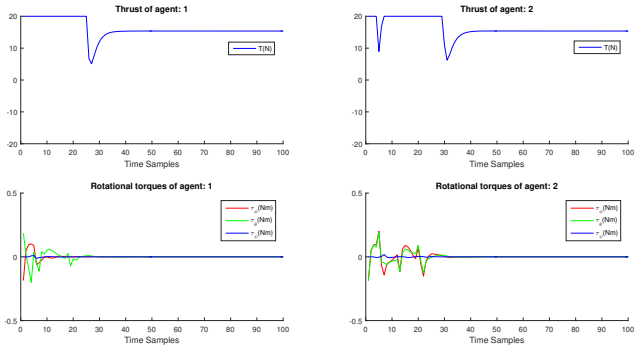

Figure 5.10: Inputs applied to agents in order to reach the goal while avoiding obstacles

The absolute positions $\mathbf{q} = (x, y, z, \phi, \theta, \psi)$ of both agents are shown in Fig. 5.9. It can be seen that both agents reach their destination points whereas response time of agent 2 is high as it slows down and stops during the navigation. Fig. 5.10 depicts the inputs generated by the linear MPC controller which are required to reach the desired goals subject to the linear constraints on states and inputs.

5.3.2 CASE-II

In this case head-on collision avoidance between two agents is studied. Agent-1 and agent-2 swap their positions. During the motion agent-2 slows down and stops as soon as agent-1 is in its scanning radius and agent-1 being the high priority agent avoids the agent-2 Fig. 5.11. The absolute positions of both agents and the inputs required for the current problem are show in Fig.5.12 and Fig. 5.13.

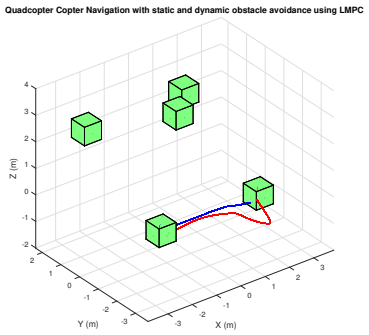

Figure 5.11: 3D trajectory showing the decentralized motion of two agents using NMPC for headon collision avoidance between two agents

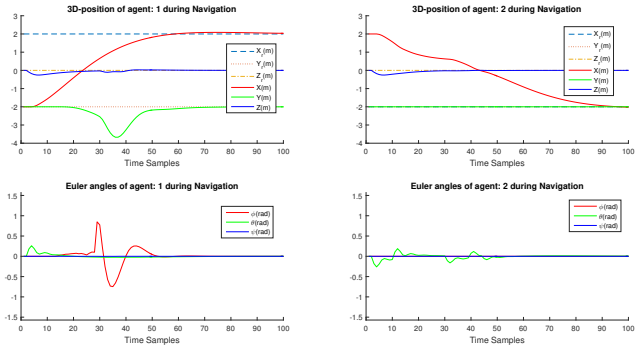

Figure 5.12: Linear positions and Euler angles of agents

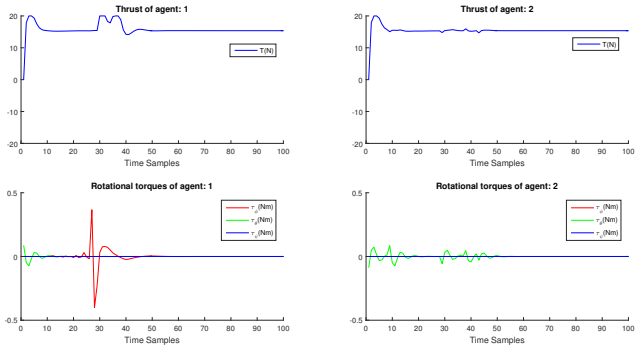

Figure 5.13: Inputs applied to agents in order to reach the goal while avoiding obstacles

Simulation results for decentralized navigation of two quad-rotors for these two cases using linear MPC are available at `https://youtu.be/a6Sexymzll0` and `https://youtu.be/fFFE83XGb1o`.

5.4 Hybrid MPC results

The hybrid MPC based obstacles avoidance algorithms are also tested for under study cases.

5.4.1 CASE-I

This section presents the results of the obstacles avoidance algorithms based on hybrid MPC for side on collision between the two agents and also for static obstacle avoidance Fig. 5.14. Agent 1 being the higher priority agent treats the agent 2 as obstacle and avoids it while the agent 2 slows down until the other agent has left. Fig. 5.15 and Fig. 5.16 present the absolute positions of both agents and control input obtained by solving by HMPC problem at each time sample.

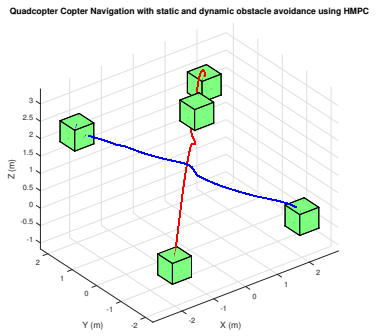

Figure 5.14: 3D trajectory showing the decentralized motion of two agents using HMPC for sideon collision avoidance between two agents and static obstacle avoidance with region π_5

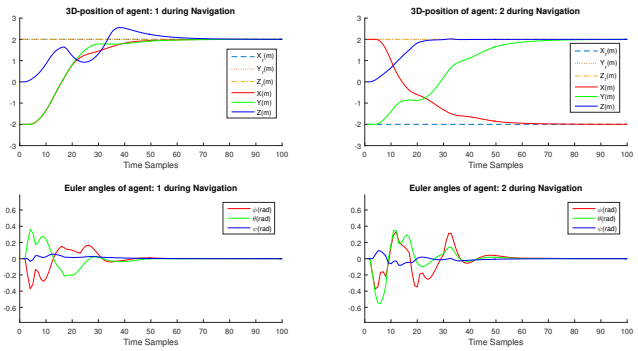

Figure 5.15: Linear positions and Euler angles of agents

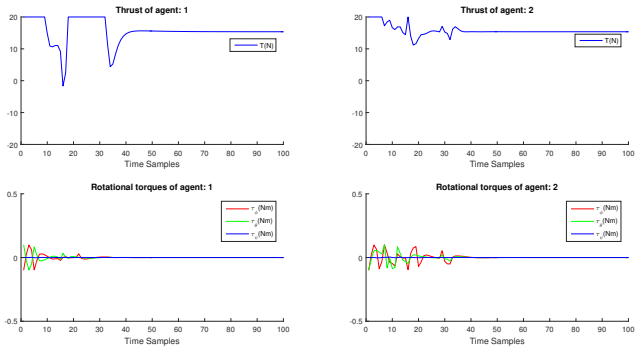

Figure 5.16: Inputs applied to agents in order to reach the goal while avoiding obstacles

5.4.2 CASE-II

The simulation results for head on collision avoidance between the agents using HMPC are shown in Fig. 5.17 to Fig. 5.19. It can be seen that the agents avoid each other in order to swap their positions.

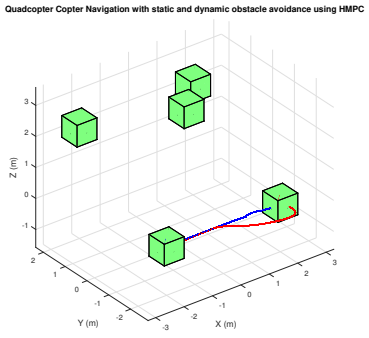

Figure 5.17: 3D trajectory showing the decentralized motion of two agents using NMPC for headon collision avoidance between two agents

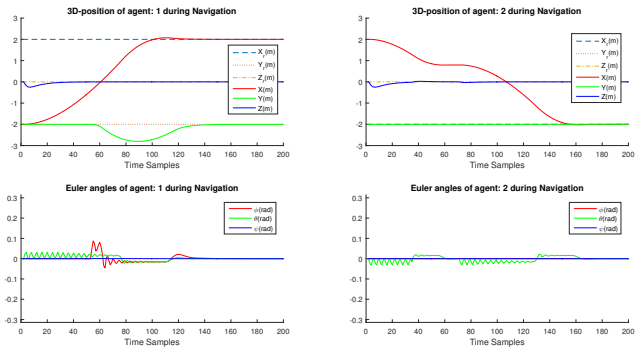

Figure 5.18: Linear positions and Euler angles of agents

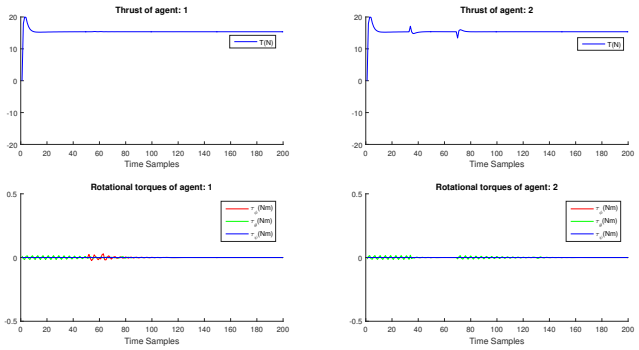

Figure 5.19: Inputs applied to agents in order to reach the goal while avoiding obstacles

Simulation results for decentralized motion of two quad-rotors for under study cases using hybrid MPC are available at `https://youtu.be/E1ye_YK7QQo` and `https://youtu.be/1wi1l1GJzWI`.

Chapter 6

Experimental Evaluation

The experimental evaluation is done only for the linear MPC controller. The Gurobi solver in CVXPY created problems with python interface and couldn't solve the hybrid MPC problems efficiently in time. The experiments are divided into two parts. In the first part, the linear MPC is implemented on firefly drone [12] in the Gazebo simulator. In the second part, the real time experiments are performed using Crazyflie mini-drones. Moreover, during experimentation only the head on collision case is studied and evaluated for the controller.

The controllers that we designed calculate the control inputs applied to the plant based on the predicted states of the system such that the obstacles are avoided. For the implementation part, the problem is made simpler and we use the linear MPC to generate the way points to lower level controller subject to obstacle avoidance. The lower level PID controller steers the quad to way points published at each sample instant.

The flow chart for performing experiments for obstacle avoidance is shown in Fig. 6.1

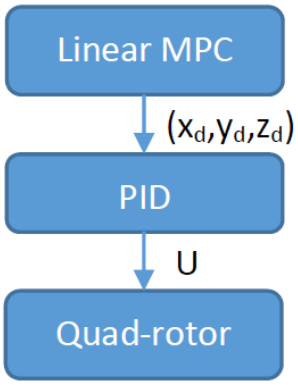

Figure 6.1: Hierarchical control structure for performing experiments using LMPC as a path planner

6.1 Implementation

Robot operating system (ROS) which provides a better interface to communicate in between different nodes/ components is used to implement the MPC controller while CVXPY provides a good interface to python in order to define and solve the optimization problem such as linear MPC. To estimate the states of the agent in the environment, a motion capture (MOCAP) is used. It has been observed by experience that MOCAP doesn't give good estimates when the quad-rotors avoid each other in z-axis. Therefore, the obstacle avoidance is done in x-y plane.

6.2 Velocity Model

The position of the agent in x-y plane can be controlled by the velocity model Eq. (6.1) [25]. This model is used to design the way point publisher. For experiments this model is used as we need only prediction of the states x and y.

$$\dot{p} = v \tag{6.1}$$

where, $p = (x, y)$ and $v = (v_x, v_y)$

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \end{aligned} \tag{6.2}$$

6.3 Experiments

As discussed earlier, two types of experiments are done. In each case only the head-on collision case is studied so there will be only dynamic obstacle that will be avoided.

6.3.1 Gazebo Simulator

Gazebo is a 3D dynamic simulator which is used to simulate the robots under the necessary environmental conditions. **RotorS** [12] which provides a modular Gazebo (Micro Aerial Vehicle) MAV framework is used to simulate the obstacle avoidance algorithms.

The initial positions of both air-vehicles are (-1.5,-1.5) and (1.5,1.5) respectively. In-order to test the obstacle avoidance algorithm, the agents are tested for position swapping. So each quad will go to the initial position of other quad. During navigation they will face each other. At that time the quad-rotor with low priority will stop and the one with high priority will avoid the other dynamic agent. There results are shown in Fig. 6.2 and the video is available at `https://youtu.be/LAVANoWWReU`.

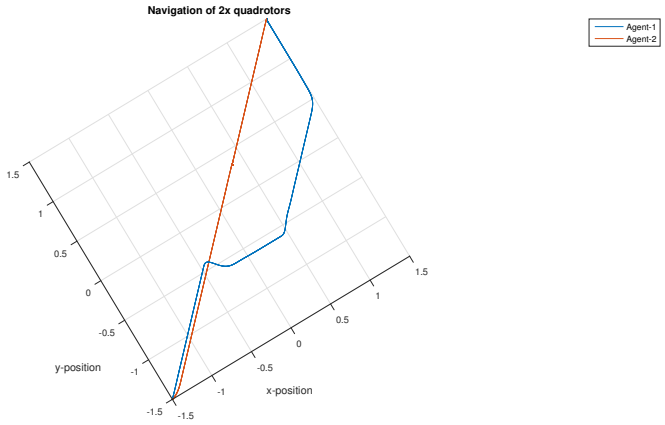

Figure 6.2: 2D Path showing the obstacle avoidance between two agents in Gazebo

The communication between different nodes in the ROS are done by topic. The one that publish the topic is called publisher and the one that listens to it is called listener. The RQT-graph Fig. 6.3 shows how the different nodes communicate each other in ROS in order to perform specific task.

Figure 6.3: RQT graph showing different topics published by different nodes during decentralized navigation of 2x firefly

The path planner and the lower level PID controller are highlighted in red and green color Fig. 6.3. The path planner has two nodes **TrajectoryPlanner** and **TrajectoryPlanner2** for each agent. Each of these nodes listens to the topic $/firefly/ground\_truth/odometry$ and $/firefly\_two/ground\_truth/odometry$ published by node **Gazebo**. These nodes publish the topics $/uav\_posereference$ and $/uav\_posereference2$ to lower level PID controllers represented by nodes **controller** and **controller2** which maneuver the quad-rotors to the desired points published by the trajectory planners.

6.3.2 Real time

The real time experiments are performed on Crazyflie 2.0 mini-drones. Two crazyflies are placed initially at (0,-1.5) and (0,1.5) respectively. The real time experiment is also carried out for the second case where the dynamic obstacle avoidance algorithm is tested on real drones. The communication between different nodes for the real time experiments is shown in RQT graph Fig. 6.4.

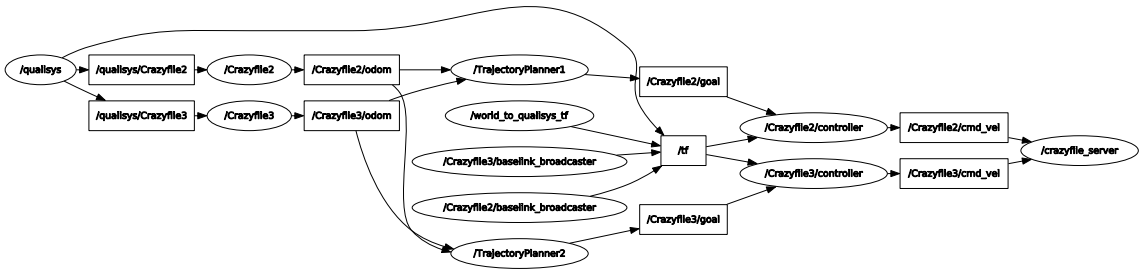

Figure 6.4: RQT graph showing different topics published by different nodes during decentralized navigation of 2x Crazyflies

There are two path planners **TrajectoryPlanner** and **TrajectoryPlanner2** for each Crazyflie. These nodes listen to the topics $/Crazyflie2/odom$ and $/Crazyflie3/odom$ and publish the topics $/Crazyflie2/goal$ and $/Crazyflie3/goal$ respectivley. These published points represent the way points of obstacle free path. These topics are listened by two nodes named as **/Crazyflie2/controller** and **/Crazyflie3/controller**. These are the lower level PID controllers which steer the agents to the desired way point published by the trajectory planner. Fig. 6.5 shows the linear positions of the two agents while navigating and avoiding each other. Video for the real time experiment is available at `https://youtu.be/qjon-zx3GqQ`.

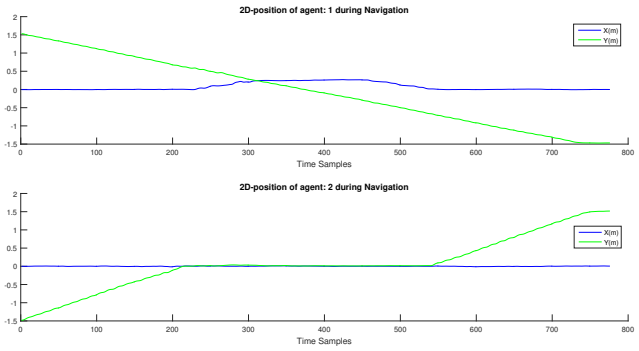

Figure 6.5: Linear positions of two agents during decentralized navigation subject to inter-agent collision avoidance

Chapter 7

Conclusion

In this project, we proposed algorithms for MPC based decentralized navigation of multiple quad-rotors subject to the dynamical model of quad-rotor and avoidance of static obstacles and inter-agent collision. First, we derived the non-linear model of a quad-rotor using Newton-Euler approach and modelled the obstacle collisions as non-linear constraints on the states x,y and z. Then the desired problem was formulated as a non-linear MPC problem subject to nonlinear model and non-linear constraints. Secondly, we linearized the model around hovering points and modelled the obstacles as linear constraints on states x,y and z respectively. Then we derived the linear MPC problem for the stated problem by using heuristic rules such as if-else statements for obstacle avoidance. Thirdly, we converted the workspace into two regions; the flying region and the obstacle region and then associated the binary variables to each obstacle region and obtained the hybrid model which is then converted into mixed logical dynamical system suitable form for optimal control problems. Based on MLD, we formulated the stated problem as hybrid MPC problem. Moreover, linear MPC and hybrid MPC based algorithm are also compared for desired problem.

It has been observed form the simulation results that each controller is able to steer the each agent to the desired goal point in such a way that they follow a path free of obstacles.

7.1 Future Work

In this thesis, we didn't use a separate path planner and a controller for simulations rather we designed a predictive controller which calculates the desired control based on the predicted trajectories such that all types of obstacles are avoided. While performing experiments, it has been learned that sometimes the solver takes more time than the desired sampling time and hence if we apply that input to the model, it may lead to instability of the closed loop system and the quad-rotor may-not converge to the desired goal point. Therefore, it is better to have a lower level controller PID or linear MPC which calculates the control based on the way points published by the upper level path planner. Hence, we suggest to use the hierarchical approach

having path planner at upper level and controller at lower level for decentralized navigation of multiple quad-rotors subject to desired tasks.

It is also suggested to use velocity model for path planner at upper level. It may simplify the problem at upper level because of having less number of states to handle. Hence, the simulations can be extended to more than two agents.

In this thesis, we define the priorities prior to the start of the simulation. The other suggestion is to dynamically assign the priorities of the agents.

Bibliography

[1] Armin Ataei and Ioannis Ch Paschalidis. "Quadrotor deployment for emergency response in smart cities: A robust MPC approach". In: *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE. 2015, pp. 5130–5135.

[2] Alberto Bemporad. "Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form". In: *IEEE Transactions on Automatic Control* 49.5 (2004), pp. 832–838.

[3] Alberto Bemporad and Manfred Morari. "Control of systems integrating logic, dynamics, and constraints". In: vol. 35. 3. Elsevier, 1999, pp. 407–427.

[4] Alberto Bemporad, Carlo A Pascucci, and Claudio Rocchi. "Hierarchical and hybrid model predictive control of quadcopter air vehicles". In: vol. 42. 17. Elsevier, 2009, pp. 14–19.

[5] Alberto Bemporad and Claudio Rocchi. "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles". In: vol. 44. 1. Elsevier, 2011, pp. 11900–11906.

[6] Alberto Bemporad and Claudio Rocchi. "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles". In: *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE. 2011, pp. 7488–7493.

[7] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[8] Francesco Borrelli et al. "MPC-based approach to active steering for autonomous vehicle systems". In: *International Journal of Vehicle Autonomous Systems* 3.2-4 (2005), pp. 265–291.

[9] Patrick Bouffard. *On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments*. Tech. rep. CALIFORNIA UNIV BERKELEY DEPT OF COMPUTER SCIENCES, 2012.

[10] Abolfazl Eskandarpour and Vahid Johari Majd. "Cooperative formation control of quadrotors with obstacle avoidance and self collisions based on a hierarchical MPC approach". In: *Robotics and Mechatronics (ICRoM), 2014 Second RSI/ISM International Conference on*. IEEE. 2014, pp. 351–356.

[11] Rolf Findeisen et al. "State and output feedback nonlinear model predictive control: An overview". In: vol. 9. 2-3. Elsevier, 2003, pp. 190–206.

[12] Fadri Furrer et al. “Rotors—A modular gazebo mav simulator framework”. In: *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[13] Luis Rodolfo G Arc AC Arrillo, Rogelio Loz Ano, Cl Aude P g Ard, et al. *Quad Rotorcraft Control*. 2014.

[14] Gonzalo Garcia and Shahriar Keshmiri. “Nonlinear model predictive controller for navigation, guidance and control of a fixed-wing UAV”. In: *AIAA Guidance, Navigation, and Control Conference, American Institute of Aeronautics and Astronautics, Portland, OR, Aug*. 2011, pp. 8–11.

[15] Khaled A Ghamry and Youmin Zhang. “Formation control of multiple quadrotors based on leader-follower method”. In: *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE. 2015, pp. 1037–1042.

[16] Yazdi I Jenie et al. “Selective velocity obstacle method for cooperative autonomous collision avoidance system for UAVs”. In: *AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, MA*. 2013.

[17] Yazdi I Jenie et al. “Velocity obstacle method for non-cooperative autonomous collision avoidance system for UAVs”. In: *AIAAGuidance, Navigation, and Control Conf*. 2014.

[18] Endra Joelianto et al. “Model predictive control for obstacle avoidance as hybrid systems of small scale helicopter”. In: *Instrumentation Control and Automation (ICA), 2013 3rd International Conference on*. IEEE. 2013, pp. 127–132.

[19] Tamás Keviczky et al. “Predictive control approach to autonomous vehicle steering”. In: *American Control Conference, 2006*. IEEE. 2006, 6–pp.

[20] Johan Lofberg. “YALMIP: A toolbox for modeling and optimization in MATLAB”. In: *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*. IEEE. 2004, pp. 284–289.

[21] Johan Löfberg. “Linear model predictive control: Stability and robustness”. PhD thesis. Linköping University Electronic Press, 2001.

[22] Teppo Luukkonen. “Modelling and control of quadcopter”. In: *Independent research project in applied mathematics, Espoo* (2011).

[23] H Merabti, I Bouchachi, and K Belarbi. “Nonlinear model predictive control of quadcopter”. In: *Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2015 16th International Conference on*. IEEE. 2015, pp. 208–211.

[24] Fabio Danilo Torrisi and Alberto Bemporad. “HYSDEL-a tool for generating computational hybrid models for analysis and synthesis problems”. In: vol. 12. 2. IEEE, 2004, pp. 235–249.

[25] Christos K Verginis, Ziwei Xu, and Dimos V Dimarogonas. “Decentralized Motion Planning with Collision Avoidance for a Team of UAVs under High Level Goals”. In: 2016.

[26] Yuchen Zhou and John S Baras. "Reachable set approach to collision avoidance for UAVs". In: *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE. 2015, pp. 5947–5952.

TRITA TRITA-EE 2017:095
ISSN 1653-5146