School of Engineering, Discipline of Mechanical Engineering and Mechatronics, pp. 1–21

Autonomous Object Evasion, and Interception Using the Quanser QDrone

Final Year Project Report - ENGG4801A

May 2024

Evan Byrne [1]

[1] *Student of Mechatronics Engineering,*
The University of Newcastle, Callaghan, NSW 2308, AUSTRALIA
Student Number: 3349681
E-mail: Evan.Byrne@uon.edu.au

Abstract

This report focuses on a project in which the Quanser QDrone (1) is required to detect an object, specifically a ball in this scenario, and autonomously recognize and manoeuvre to either avoid or intercept it. This project explores various control systems and advanced object detection capabilities of current unmanned aerial vehicles (UAVs), aiming to enhance trajectory estimation for practical applications such as automated package retrieval and delivery. The current methodology for addressing this challenge involves the use of a Vicon System (2). However, the project's ultimate goal is to achieve complete autonomy by relying solely on onboard sensors, cameras, and computational resources. Currently, MATLAB simulations show that nonlinear model predictive control (NMPC) (3) is able to effectively capture the states of the QDrone. When coupling this with the ball trajectory, simulations show the QDrone successfully intercepting the object.

Contents

1 Introduction

The utilization of UAVs has significantly expanded across various sectors, driven by enhancements in their design and control systems. This project specifically addresses the autonomous capabilities of the Quanser QDrone for object detection and interaction tasks, which are critical in fields such as automated delivery and emergency response. The challenge lies in enabling the QDrone to identify and respond to dynamic objects, such as a moving ball, using only its onboard computational tools, rather than relying solely on external systems like the Vicon System.

Initially, this project leverages said Vicon System to provide accurate spatial tracking and support the development and testing of control algorithms. The project explores the adaptation and integration of advanced control systems and object detection technologies that have been traditionally used in less dynamic settings. Employing nonlinear model predictive control (NMPC) and Kalman Filter algorithms, the project aims to enhance the drone's ability to make real-time decisions based on the trajectory data of moving objects. The ultimate goal is to develop a fully autonomous UAV system capable of performing complex manoeuvres without human intervention, thereby pushing the boundaries of what autonomous drones can achieve in practical applications.

This endeavour not only aims to advance the technological capabilities of UAVs but also to provide scalable solutions that could transform industries by enabling more sophisticated autonomous operations.

2 Literature Review

2.1 System Dynamics

2.1.1 Quadrotor Dynamics

All mathematical models for quadrotor UAVs depend on the blade orientation, which can be either a plus (+) configuration or a cross (X) configuration as seen in Figure.1. The plus configuration aligns each rotor with the primary axes, resulting in symmetrical thrust distribution along these axes. This alignment allows for easier prediction and simplified control algorithms, making it an advantageous choice for straightforward control implementations and basic flight stability. This simplicity also introduces drawbacks, such as reduced manoeuvrability as well as increased complexity in sensor and camera placement. Alternatively, the X configuration allows for easier sensor placement and better stability, manoeuvrability, and aerodynamic efficiency; however, these benefits come at the cost of more complex control algorithms (4).

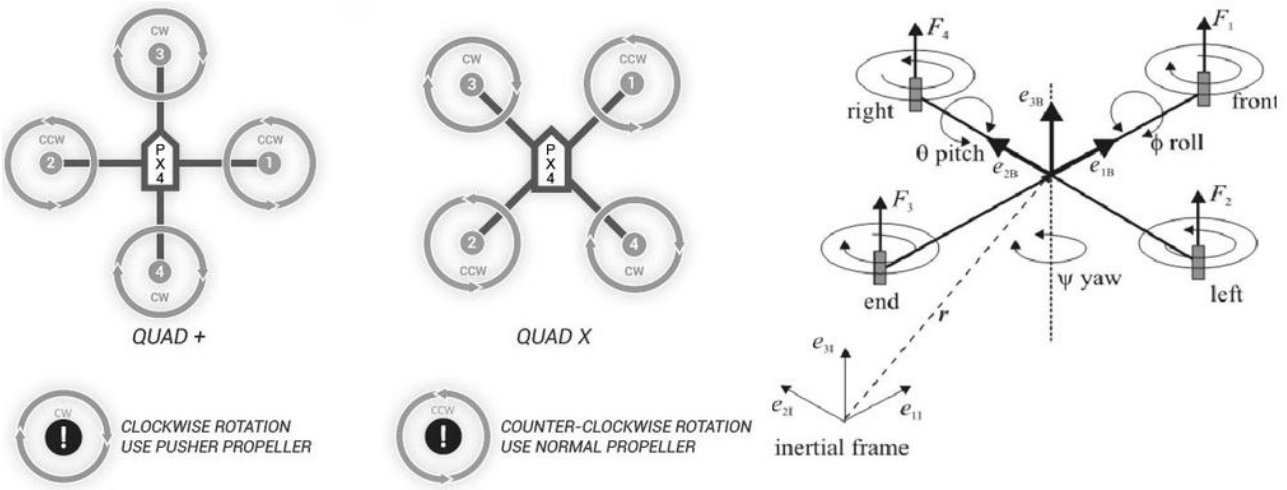

Figure 1: Quadrotor Configuration and Reference Frames

Once a quadrotor orientation is established, the system dynamics can be derived. Quadrotors are six-degree-of-freedom (DOF) systems, encompassing three rotational and three translational movements. These movements are controlled using four individual thrust inputs, each dependent on the motor and blade on each arm. For the majority of literature, such as (5), the torques and forces created from each thrust about the primary axes form the basis of the model. Although this is a common approach in modelling quadrotors, due to the complexity of the system, models can vary from quite low-fidelity models to extremely high-fidelity models that encompass additional forces such as gyroscopic and drag forces. These varying levels of model fidelity allow researchers and engineers to balance the need for accuracy with computational efficiency, depending on the specific application and result in the lower fidelity models being more prevalent.

Regardless of the model's fidelity, an accurate representation of quadrotor dynamics necessitates the proper application of reference frames and the implementation of Euler-Newton and Euler-Lagrange equations (6). Reference frames such as the ones shown in Figure 1 are crucial for understanding the motion relative to different points of view, whether fixed or moving. The Euler-Newton method applies forces and torques to derive the equations of motion, while the Euler-Lagrange approach offers a generalised formulation ideal for capturing complex dynamics. These theoretical foundations are essential for developing robust models that can accurately predict and control the UAV's behaviour under various conditions

As previously mentioned, quadrotors are nonlinear systems which inherently result in nonlinear models. Due to this, many control algorithms require the system to be linearised. This linearisation involves approximating the nonlinear dynamics around a specific operating point by taking the Jacobian (7), thereby making it possible to apply linear control techniques.

2.1.2 Object Dynamics

The kinematics and dynamics of objects travelling through the air can range from simplistic to complex, depending on the geometry of the object and the fidelity of the chosen model. Idealised projectile motion problems allow for easy estimation of trajectory, time of flight, and position by neglecting all external forces except gravity (8). This approach simplifies analysis but overlooks critical factors that

affect real-world objects.

As the complexity of the object increases, understanding the forces acting on it becomes significantly harder. Factors such as mass, surface area, and surface finish play major roles in determining drag forces, rotational biases, and asymmetrical rotation. Aerodynamic effects, such as the Magnus effect, and surface roughness can impact drag and lift characteristics. Computational fluid dynamics (CFD) simulations and wind tunnel experiments are often used to study these effects in detail. While idealised models offer simplicity, real-world dynamics require consideration of various physical phenomena to achieve accurate predictions.

2.2 Controllers

Refining and developing control strategies is vital for enabling quadrotors to perform complex tasks such as catching a ball. These strategies are designed to enhance the agility, stability, and precision of quadrotor systems, which are inherently unstable and require constant adjustments to maintain flight. From the straightforward control strategies such as Proportional-Derivative (PD) control to the more sophisticated Proportional-Integral-Derivative (PID) control, and extending further to advanced techniques like Linear Quadratic Regulator (LQR) control, Model Predictive Control (MPC), and Nonlinear Model Predictive Control (NMPC), researchers have developed and refined various methods to address the challenges of quadrotor dynamics. Each of these control strategies offers distinct advantages and complexities, making them suitable for different aspects of quadrotor flight and task execution (9).

2.2.1 PD Control

Proportional-Derivative (PD) control is one of the simplest and most widely used control strategies in quadrotor systems due to its ease of implementation and effectiveness in improving transient response (10). The PD controller adjusts the control input based on the current error and its rate of change, providing a balance between the speed of response and damping.

In discrete time, the PD control law can be expressed as:

$$u[k] = K_p e[k] + K_d \frac{e[k] - e[k-1]}{T_s} \tag{2.1}$$

where $u[k]$ is the control input at the k-th sample, $e[k]$ is the error at the k-th sample, and T_s is the sampling period (11).

In the context of quadrotor control, the error $e(t)$ or $e[k]$ can be the difference between the desired and actual positions or angles. By tuning K_p and K_d, the PD controller can achieve a fast and stable response, minimizing overshoot and settling time.

2.2.2 PID Control

Proportional-Integral-Derivative (PID) control builds upon the principles of PD control by adding an integral term to address the accumulation of past errors, which can help eliminate steady-state errors . The inclusion of the integral action allows the controller to adjust the control input not only based on the current error and its rate of change but also on the cumulative sum of past errors (11).

In discrete time, the PID control law can be expressed as:

$$u[k] = K_p e[k] + K_d \frac{e[k] - e[k-1]}{T_s} + K_i T_s \sum_{j=0}^{k} e[j] \tag{2.2}$$

where $u[k]$ is the control input at the k-th sample, $e[k]$ is the error at the k-th sample, and T_s is the sampling period (11).

Although the integral term excels at removing steady state error that the proportional and derivative terms alone cannot, it must be carefully tuned as excessive integral gain can lead to integral windup, causing large overshoot and potentially destabilizing the system (12).

By combining the proportional, integral, and derivative actions, the PID controller can provide a balanced approach to managing both transient and steady-state performance. This makes it a versatile and widely used control strategy in various applications, including quadrotor control.

2.2.3 Cost Function in Optimal Control

When moving to more advanced control methods, cost functions are the mathematical tool used to assess the performance of a control system. A cost function provides a measure of how well the system is achieving its desired objectives, typically by penalizing deviations from a target state and the effort required to control the system. The goal of the control strategy is to minimize this cost function, thereby optimizing the system's performance (13).

A common form of cost function used in optimal control strategies is a quadratic function, which can be expressed as:

$$J = \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt, \tag{2.3}$$

where:

- $x(t)$ represents the state deviations.
- $u(t)$ represents the control efforts.
- Q and R are weighting matrices that determine the relative importance of state deviations and control efforts.

The cost function helps to ensure that the system states remain close to the desired values while using control inputs efficiently. By carefully choosing the weights Q and R, the control strategy can balance precision and efficiency, making it effective for systems requiring optimal performance, such as quadrotor control (14).

2.2.4 LQR Control

Linear Quadratic Regulator (LQR) control is a fundamental optimal control strategy designed for linear systems. It aims to find the optimal set of control inputs that minimize a predefined cost function, balancing the trade-off between state deviations and control efforts. This makes LQR particularly effective for systems where precision and efficiency are crucial.

The LQR control strategy leverages the quadratic cost function as shown in (2.3) to determine the optimal feedback control law. By minimizing this cost function, LQR ensures that the system's state remains close to the desired trajectory while using control inputs efficiently.

LQR is advantageous for quadrotor control due to its optimal control solution, robustness to disturbances, and guaranteed stability if the system is controllable. Its simplicity and computational efficiency make it suitable for real-time applications. However, LQR assumes linear system dynamics, which can be inaccurate for the highly nonlinear behaviour of quadrotors, particularly during aggressive manoeuvres. This reliance on accurate models and fixed gains can limit its adaptability to changing conditions. As a result, more advanced nonlinear control methods are often preferred for handling complex scenarios. (15)

2.2.5 MPC Control

Model Predictive Control (MPC) is an advanced control strategy that optimizes control inputs by predicting future system behaviour over a finite time horizon, making it suitable for complex systems like quadrotors. MPC minimizes a cost function, typically in the form:

$$J = \sum_{k=0}^{N-1} \left(x[k]^T Q x[k] + u[k]^T R u[k] \right) + x[N]^T P x[N], \tag{2.4}$$

where N is the prediction horizon, $x[k]$ is the state vector, $u[k]$ is the control input, and Q, R, and P are weighting matrices. The term $x[N]$ represents the state of the system at the end of the prediction horizon, guiding the quadrotor towards achieving its target objectives (16).

MPC effectively manages state and input constraints, handles multivariable control problems, and anticipates future disturbances. However, it requires significant computational resources, relies on an accurate system model, and involves complex parameter tuning. Despite these challenges, MPC's predictive and optimization capabilities make it ideal for precision and adaptability in quadrotor control.

Without a prediction horizon ($N = \infty$), MPC reduces to Linear Quadratic Regulator (LQR) control, where the optimization is performed over an infinite horizon. This makes LQR a special case of MPC, where the control law is derived for a linear system without explicitly considering constraints over a finite horizon.

2.2.6 NMPC Control

Nonlinear Model Predictive Control (NMPC) extends the principles of MPC to handle nonlinear system dynamics, making it well-suited for the highly nonlinear behaviour of quadrotors. Unlike MPC, which assumes linear dynamics, NMPC directly incorporates the nonlinearities of the system into the optimization problem. This is achieved by using nonlinear system models in the prediction of future states and in the formulation of the cost function and constraints.

The NMPC cost function can be expressed as:

$$J = \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N) \tag{2.5}$$

where $\ell(x_k, u_k)$ is the stage cost, representing the cost at each step k, and $\ell_f(x_N)$ is the terminal cost, representing the cost at the final step N. The stage cost typically penalizes deviations from desired

states and control inputs, while the terminal cost ensures the system reaches the desired final state (3).

In NMPC, the stage cost $\ell(x_k, u_k)$ and the system dynamics used for prediction are both nonlinear functions. Specifically, the system dynamics are represented as:

$$x_{k+1} = f(x_k, u_k) \tag{2.6}$$

where $f(\cdot)$ is a nonlinear function that describes the system's evolution. By incorporating this nonlinear function directly into the optimization problem, NMPC can accurately predict future states and optimize control inputs accordingly, even for systems with highly nonlinear behaviors.

By incorporating the actual nonlinear dynamics of the system directly into the optimization problem, NMPC provides more accurate control for systems with significant nonlinearities. This leads to better performance in complex and dynamic environments where linear approximations would be insufficient.

NMPC offers several advantages, including improved accuracy in handling nonlinearities and enhanced flexibility in managing a wide range of constraints and objectives. However, it also presents challenges, such as increased computational complexity due to the need to solve nonlinear optimization problems at each time step, and a higher sensitivity to the accuracy of the system model.

2.2.7 Control Summary

Control strategies vary depending on system requirements. PID is suitable for systems lacking accurate models but may struggle with complexity. LQR provides efficiency for linear models. MPC suits systems with moderate computational resources and accurate models, handling multivariable control and future disturbances. NMPC excels in highly nonlinear systems, requiring significant computational power and expertise. In summary, PID is for simple systems, LQR for linear efficiency, MPC for accurate control with moderate resources, and NMPC for nonlinear precision with computational capacity.

3 Problem Approach

The high-level block diagram illustrated in Figure 2 provides an overview of the system architecture for the quadrotor's task of catching a ball. This system integrates multiple components, including the NMPC controller, quadrotor dynamics, and sensor inputs from various sources. Currently, a Vicon system is utilized to provide precise state measurements of the quadrotor, which are used by the NMPC controller to generate the control inputs necessary for manoeuvring. These control inputs drive the quadrotor dynamics, ensuring accurate adjustments to catch the ball. The ultimate goal of the project is to replace the Vicon system with onboard sensors and sensor fusion techniques for autonomous operation. This structured approach ensures the quadrotor can dynamically respond to the ball's movements, achieving the desired task of catching it.

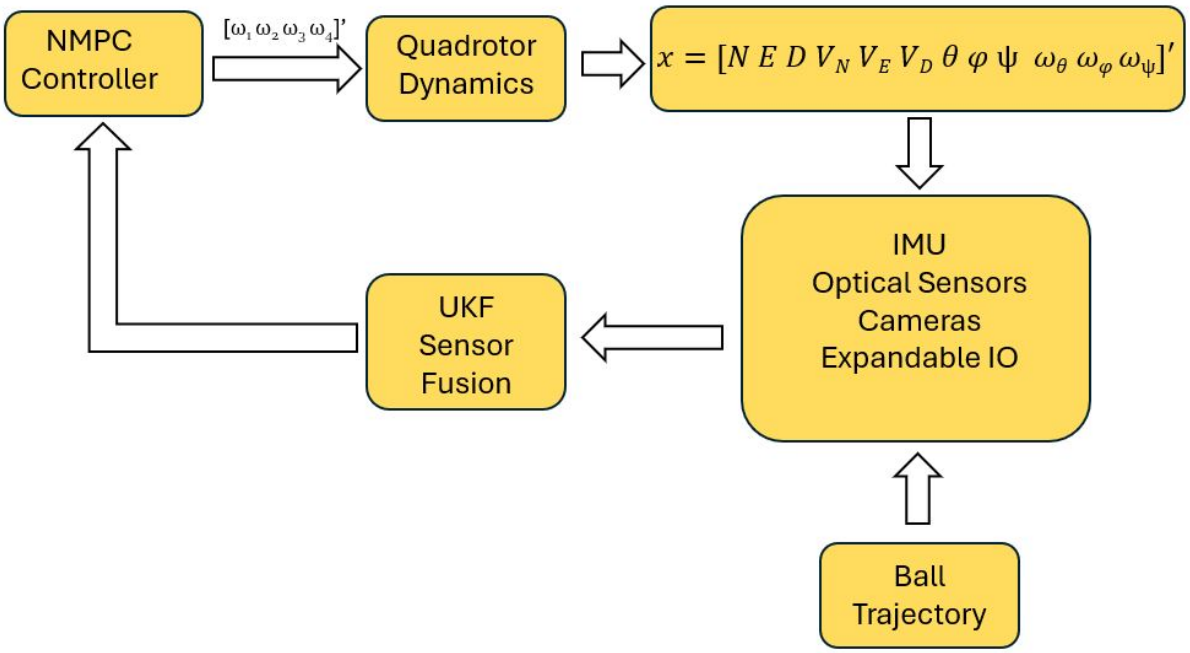

Figure 2: High-level block diagram of the quadrotor system for catching a ball.

4 Quadrotor Dynamics

4.1 Mathematical Model

4.1.1 Assumptions

In developing the quadrotor dynamics for the NMPC control system, several key assumptions were made to simplify the model and focus on the primary control objectives:

- The quadrotor is modelled as a rigid structure with no flexible parts.
- It is assumed to be symmetrical around both the pitch and roll axes.

- Thrust and forces are considered to be directly proportional to the propeller speed.
- Drag is neglected.
- The effects of blade flapping are neglected.
- Gravity is assumed to be constant, with variations due to height being negligible due to the indoor scenario.
- The presence of wind is not considered in the model.
- Ground effect is considered negligible and thus not included in the model.

These assumptions allow for a more tractable analysis and control design, focusing on the primary dynamics and control challenges of the quadrotor.

4.1.2 Coordinates and Rotation Matrices

For the Quanser quadrotor model, a North-East-Down (NED) coordinate system was used, where the north and east axes were defined between the rotors, considering sensor placement. In this system, the down direction is positive in the direction of gravity.-

To define the relationship between the body-fixed frame and the world frame, rotation matrices about the roll (ϕ), pitch (θ), and yaw (ψ) axes are used (17). These matrices are crucial for transforming vectors between frames.

The rotation matrices are:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{4.1}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.2}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The composite rotation matrix R is formed by multiplying these matrices:

$$R = R_z(\psi)R_y(\theta)R_x(\phi) \tag{4.4}$$

$$R = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{4.5}$$

This matrix R is essential for transforming coordinates between the world and body frames, playing a significant role in accurately simulating the quadrotor's behaviour and calculating forces and torques.

By using the NED coordinate system and these rotation matrices, the relationship between the world frame and the body frame is clearly established, enabling precise modelling and control of the Quanser quadrotor.

4.1.3 States and Inputs

To fully describe the condition of the system at any given time, a set of states must be assigned to encompass all the dynamic behaviour. For this system, the states are defined as:

- N: Position in the north direction (world frame).
- E: Position in the east direction (world frame).
- D: Position in the down direction (world frame, positive in the direction of gravity).
- $\dot{N}$: Velocity in the north direction (world frame).
- $\dot{E}$: Velocity in the east direction (world frame).
- $\dot{D}$: Velocity in the down direction (world frame).
- ϕ: Roll angle, describing rotation around the north axis (world frame).
- θ: Pitch angle, describing rotation around the east axis (world frame).
- ψ: Yaw angle, describing rotation around the down axis (world frame).
- $\dot{\phi}$: Angular velocity around the north axis (world frame).
- $\dot{\theta}$: Angular velocity around the east axis (world frame).
- $\dot{\psi}$: Angular velocity around the down axis (world frame).

These states collectively capture the position, orientation, and motion dynamics of the system and can be grouped into subsets depending on their effects. Specifically, $\mathbf{X_n} = [N, E, D]$ represents the position, $\mathbf{W_n} = [\phi, \theta, \psi]$ represents the orientation, $\dot{\mathbf{X}}_\mathbf{n} = [\dot{N}, \dot{E}, \dot{D}]$ represents the linear velocities, and $\dot{\mathbf{W}}_\mathbf{n} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$ represents the angular velocities. Using all of these variables the states of the system are $\mathbf{x} = [\mathbf{X_n}, \dot{\mathbf{X}}_\mathbf{n}, \mathbf{W_n}, \dot{\mathbf{W}}_\mathbf{n}]$

Additionally, two crucial subgroups for system modelling are $\dot{\mathbf{X}}_\mathbf{B} = [u, v, w]$ and $\dot{\mathbf{W}}_\mathbf{B} = [p, q, r]$, the linear and angular velocities in the body coordinates. These are important because they allow for the transformation of velocities between the body and world frames as shown in Equations 4.6 and 4.7 below (18):

$$\dot{\mathbf{X}}_\mathbf{n} = R_{bn}\dot{\mathbf{X}}_\mathbf{B} \tag{4.6}$$

$$\dot{\mathbf{W}}_\mathbf{n} = W_n^{-1}\dot{\mathbf{W}}_\mathbf{B} \tag{4.7}$$

where R is the rotation matrix (4.5) and W_n is the transformation matrix found in Appendix A.

The inputs to the quadrotor system are the individual angular velocities of each motor $\mathbf{u} = [w_1, w_2, w_3, w_4]$. These inputs directly affect the thrust and torques generated by the rotors, influencing the quadrotor's movement and orientation, and are the controllable elements of the system.

4.2 Motor Equations

The thrust generated by a quadrotor motor is the primary driving force of the system. The thrust for each motor can be calculated using Equation 4.8 (18):

$$T_i = k_t \cdot \omega_i^2 \tag{4.8}$$

where T_i is the individual thrust produced by motor i (i is the ith motor for example motor four is i = 4) in the body frame, k_t is the thrust coefficient, and ω_i is the angular velocity of motor i. The thrust coefficient k_t depends on the shape of the propeller and was determined through system identification performed by Quanser for their specific motors. The thrust generated by each motor directly influences the quadrotor's ability to lift, manoeuvre, and stabilize.

4.3 Translational Dynamics

To derive the translational equations of motion, Newton's second law of motion, $F = ma$, is utilized in global coordinates. This can be expressed as:

$$\sum \mathbf{F} = m\ddot{\mathbf{X}}_\mathbf{n} \tag{4.9}$$

In this equation, $\sum \mathbf{F}$ represents the sum of all external forces acting on the system, m denotes the mass of the system, and $\ddot{\mathbf{X}}_\mathbf{n}$ is the linear acceleration of the system. By rearranging this equation we can solve for linear acceleration of the system.

The sum of forces on the system can be defined as:

$$\sum \mathbf{F} = \mathbf{F}_g - \mathbf{F}_T - \mathbf{F}_d \tag{4.10}$$

where $\mathbf{F}_g$ is the force of gravity, $\mathbf{F}_T$ is the total thrust, and $\mathbf{F}_d$ is the drag force. As stated in the assumptions we neglect the force of drag in this analysis.

The gravitational force, $\mathbf{F}_g$, is given by mg, where m is the mass of the quadrotor and g is the acceleration due to gravity. In the coordinate system, this force can be represented as $[0\ 0\ mg]^T$ because it acts only in the D-direction.

Similarly, the total thrust, $\mathbf{F}_T$, acts only in the D-direction. Therefore, it can be represented as $[0\ 0\ -F_T]^T$, where F_T is the magnitude of the total thrust generated by the quadrotor's motors. This can be seen in Equation 4.11.

$$\mathbf{F}_T = R \sum \mathbf{T}_i \tag{4.11}$$

Where R is the rotation matrix required to transform the force from body to earth coordinates and $\mathbf{T}_i$ is the result of Equation 4.8. After deriving these expressions the linear acceleration of the quadrotor can be found (5).

4.4 Rotational Dynamics

The rotational dynamics represent some of the most complex forces acting upon the quadrotor. Due to the symmetrical and rigid body assumptions previously mentioned, the equation for the torque acting on the quadrotor can be expressed as:

$$\mathbf{I}_b \ddot{\boldsymbol{W}}_n = \boldsymbol{\tau}_{\text{body}} - \boldsymbol{\tau}_{\text{gravity}} - (\boldsymbol{\omega}_b \times \mathbf{I}_b \boldsymbol{\omega}_b) \tag{4.12}$$

where $\mathbf{I}_b$ is the inertia matrix of the quadrotor body, $\ddot{\boldsymbol{W}}_n$ is the angular acceleration in the body frame, $\boldsymbol{\tau}_{\text{body}}$ is the torque produced by the rotors, and $\boldsymbol{\tau}_{\text{gravity}}$ is the torque due to gravitational forces. For this project, the torques produced by gravity were neglected. The expression $(\boldsymbol{\omega}_b \times \mathbf{I}_b \boldsymbol{\omega}_b)$ is the gyroscopic torque as derived by (19).

Due to the degrees of freedom of a quadrotor, there are three dimensions of inertia in the body coordinates. The inertia matrix $\mathbf{I}_b$ can be represented as:

$$\mathbf{I}_b = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{4.13}$$

Similar to the inertias, the torques generated by the motors act in three dimensions: roll, pitch, and yaw. The torques can be represented as $\boldsymbol{\tau}_b = \begin{bmatrix} \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T$. The torques generated about the roll and pitch axes are a result of the moments acting on the associated rotor arms. These moments can be expressed as

$$\tau_{\phi\theta} = l_i T_i \tag{4.14}$$

where l_i is the length of the arm and T_i is the thrust shown in Equation 4.8. Assuming l_i is the same for all arms due to the symmetry of the drone and observing Figure 1, the torque equations can be derived as follows:

$$\tau_\phi = k_t l(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \tag{4.15}$$

$$\tau_\theta = k_t l(\omega_1^2 + \omega_4^2 - \omega_2^2 - \omega_3^2) \tag{4.16}$$

The torque acting in the yaw (ψ) direction is a result of the opposite moments from the rotors:

$$\tau_\psi = k_t l(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{4.17}$$

4.5 State Space Equations

After deriving the above equations the dynamics of a quadrotor can be described using the following linear and rotational accelerations (19) (5) (18):

$$\ddot{\mathbf{X}}_n = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \frac{\mathbf{R} \begin{bmatrix} 0 \\ 0 \\ \tau_1 \end{bmatrix}}{m} \tag{4.18}$$

$$\ddot{\mathbf{W}}_n = \begin{bmatrix} \frac{(I_{yy} - I_{zz})pq + \tau_\phi}{I_{xx}} \\ \frac{(I_{zz} - I_{xx})qr + \tau_\theta}{I_{yy}} \\ \frac{(I_{xx} - I_{yy})rp + \tau_\psi}{I_{zz}} \end{bmatrix} \tag{4.19}$$

These equations are second-order differential equations and can be solved using numerical solvers such as ODE45 in matlab (20) to simulate the operation of the drone.

5 NMPC

Choosing Nonlinear Model Predictive Control (NMPC) for the drone project, where the objective is to catch a thrown ball, provides several critical advantages over other control methods. NMPC's ability to incorporate constraints directly into the control design is particularly valuable. By defining specific constraints on the drone's motor velocities and their rates of change, safe and efficient operation within the drone's physical limits can be ensured. The use of a prediction horizon in NMPC allows the controller to use the ball's estimated trajectory, provided by external systems like a Kalman filter, as a reference. This enables the drone to plan its movements proactively, enhancing the likelihood of a successful catch.

The control horizon enables fine-tuning of how far ahead the controller looks when making decisions, ensuring a balance between responsiveness and stability. This is crucial for quickly adapting to the ball's trajectory while maintaining controlled flight. The ability to precisely adjust the weighting of each state variable in NMPC is also essential, as it allows prioritization of certain aspects of the drone's behaviour, such as position accuracy and control effort.

5.1 Implementation

In implementing the NMPC for the drone within a simulation environment, specific values were carefully chosen for the controller's weights and constraints to optimize its performance for the task of catching a thrown ball. The controller is configured with a prediction horizon of 30 steps and a control horizon of 29 steps, ensuring a fast response to the dynamic task. The time step for the controller is set to 0.01 seconds, providing high-frequency updates for precise control.

The controller uses the following state weights to ensure precise control over the drone's position and orientation: [15, 15, 15, 1, 1, 1, 5, 5, 5, 1, 1, 1]. These weights emphasize the importance of accurate position tracking while also considering orientation and other dynamic aspects of the system. The manipulated variable weights are set to [0.5, 0.5, 0.5, 0.5], which help balance control effort and minimize excessive actuator use. Additionally, the rate of change weights for the manipulated variables are [0.2, 0.2, 0.2, 0.2], which further ensure smooth control actions and prevent abrupt changes that could destabilize the drone.

Constraints are also defined to keep the drone's operation within safe and efficient limits. The motor velocities are constrained to a minimum of 0 and a maximum of 20, with rate limits set between -5 and 5. These constraints ensure that the motors operate within their physical capabilities and provide a safe response to control commands.

Currently, the NMPC uses the default cost function, which balances state tracking and control effort. In future iterations, I plan to customize this cost function to better suit the specific requirements of intercepting a moving ball, potentially enhancing the controller's performance even further.

By carefully setting these parameters, the NMPC is tailored to provide robust and efficient control, enabling the drone to effectively track and intercept the ball's trajectory while maintaining stability and precision in the simulation environment.

6 Simulation

6.1 Setup and Validation

After establishing the dynamics, a simulation was created to validate the controller and system performance. This simulation initializes the drone's parameters and specifies the starting states. The main simulation loop uses the ODE45 solver (21) to get state updates and the NMPC toolbox (20) to drive the control variables. The first test to check the dynamics involved manually setting all inputs to 4.9, which balances out gravity and allows the drone to hover, confirming the dynamics were correctly modeled. Following this, a simple trajectory was set for the drone to follow. The drone successfully tracked the trajectory, verifying the controller's effectiveness.

The assessment of performance was done not only through graphical information but also visually through animation. By graphing the states and inputs and animating the drone's movement, the system's correct behavior and control responses were confirmed. This successful validation allowed the progression to simulating the drone catching a ball.

6.2 Ball Trajectory and Assumptions

For the simulation of the drone intercepting a thrown ball, the ball's trajectory is generated using a series of assumptions and random initial conditions. In a real-world scenario, a Vicon system would be used to obtain the ball's velocity, angle, and other necessary parameters. However, for the current simulation, these parameters are randomly generated to simulate different possible trajectories of the ball. The initial conditions for the ball's trajectory can be visualized in Figure 3, including a random initial velocity V_o and random initial angles in both vertical (θ) and horizontal (ψ) directions. The ball is assumed to be thrown from a height of 1.5 meters. When the drone intercepts the ball is determined based on the height at which interception is desired, allowing control over the drone's height suitable for an indoor setting.

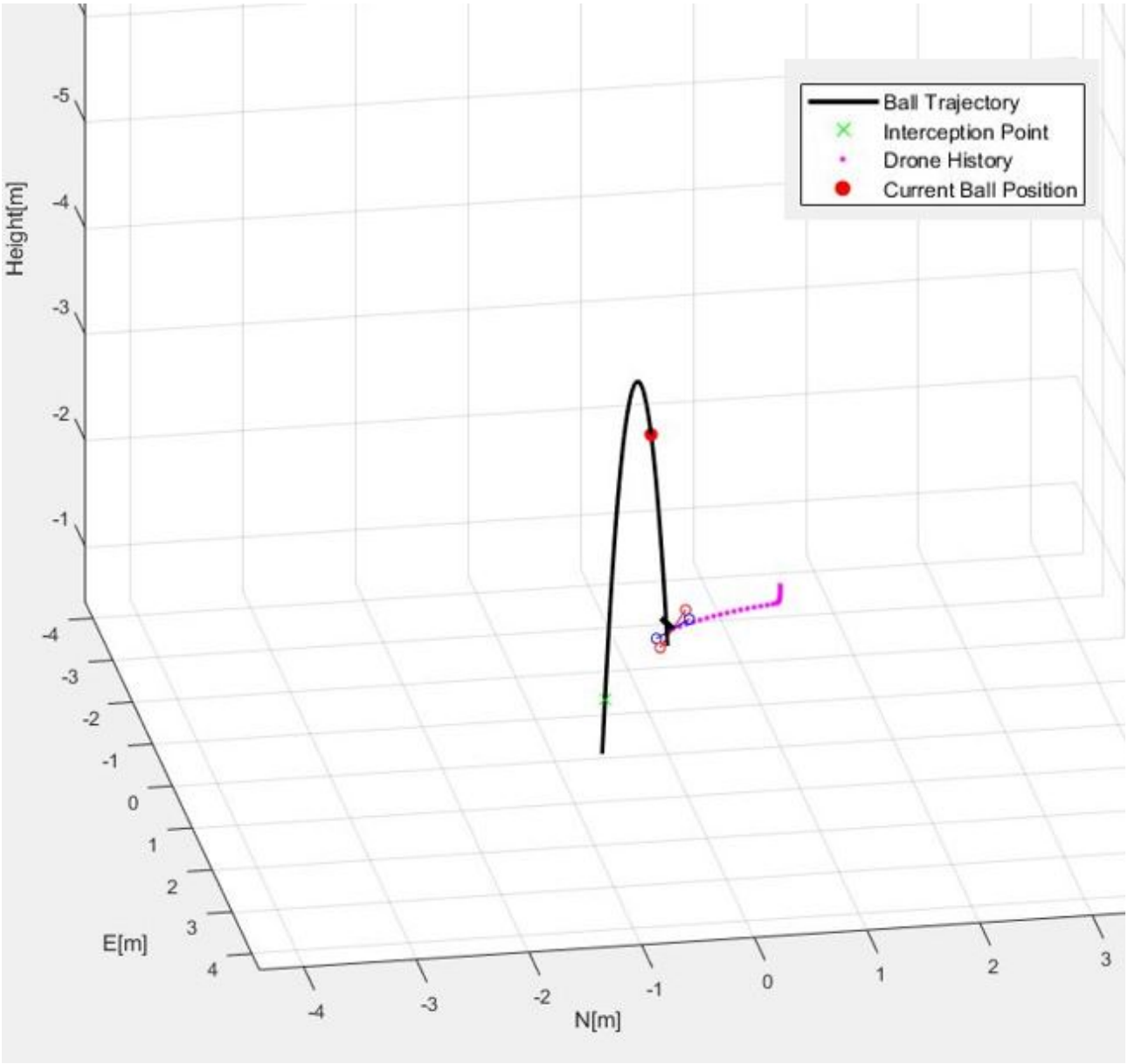

Figure 3: Screenshot of the quadrotor animation.

Assuming perfect projectile motion for simplicity, the trajectory of the ball is calculated, and the time of flight is determined based on the initial conditions and the ball's physical properties. If the calculated time of flight is infeasible for the drone to intercept the ball (less than 1 second), the simulation stops, indicating that the scenario is unrealistic.

The initial velocity, angles, and resulting trajectory provide a reference for the NMPC controller, which is tasked with manoeuvring the drone to intercept the ball. This setup allows for the testing and validation of the drone's ability to follow and intercept the ball's path accurately.

The ball's trajectory is visualized along with the drone's movement in the simulation, allowing for both graphical analysis, as shown in Figure 4, and animation of the interception process. This dual approach ensures that the system's behaviour can be thoroughly assessed, providing insights into the

performance of the controller and the dynamics of the drone in various interception scenarios.

6.3 Results

The simulation effectively allows for tuning and testing of the NMPC controller designed for the drone to intercept a thrown ball. Initial results indicate that while the controller is operational, there is a need for further fine-tuning to improve performance.

Examining the simulation results, it is evident that the controller is currently experiencing issues with overshooting and fine position control on the interception point. This behaviour suggests that the drone might be heading directly to the predicted interception point rather than adjusting its trajectory to catch the ball more effectively.

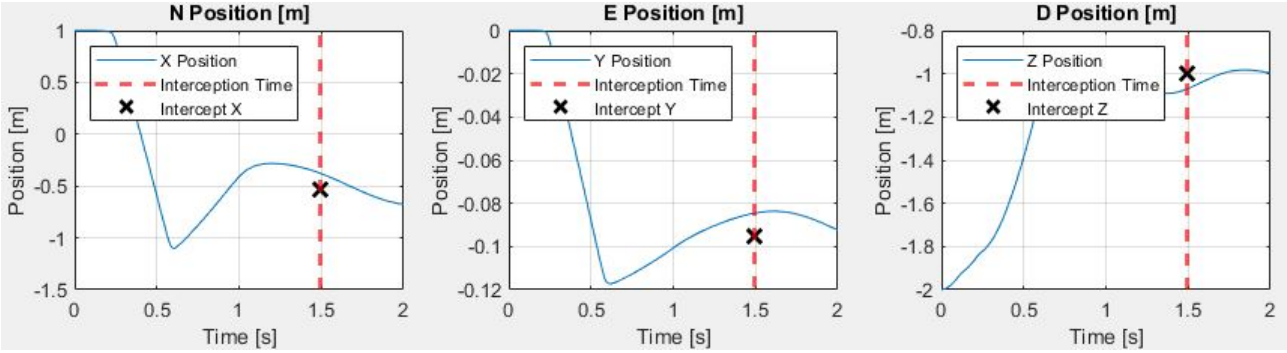

Figure 4: Quadrotor N E and D positions compared to the intercept location and time.

Overall however, the simulation is a success and will be a valuable tool as the project

7 Considerations Moving Forward

Moving forward, the focus will be on enhancing the drone's capability to capture the ball with a more sophisticated strategy. Instead of simply intercepting the ball at a point in space, the goal is for the drone to anticipate the ball's trajectory and position itself at a point on the trajectory before the ball arrives. This approach will enable the drone to move down and perform a more controlled catch rather than a mere interception. Achieving this requires the drone to have a better understanding of the ball's location and its future path.

To attain the fine control necessary for this advanced manoeuvre, it may be essential to implement a custom cost function within the NMPC framework. This custom cost function will allow for more precise tuning of the controller, emphasizing the importance of positioning and timing in the catching process. Additionally, extensive tuning of the controller's parameters will be required to ensure optimal performance.

Once the drone is capable of gently and accurately catching the ball, the next step will be to integrate an obstacle avoidance feature. This feature will involve generating a trajectory that takes the drone away from the ball instead of towards it, thereby avoiding the ball. This requires the drone to not only track the ball's position but also dynamically adjust its path to avoid a potential collision.

Successfully integrating this capability will make the system more versatile and adaptable to different scenarios.

After achieving these milestones, the plan is to utilize the Vicon system for tracking both the ball and the drone in real-time. The Vicon system will provide precise and accurate position data, which will be crucial for improving the performance of the NMPC controller. The final implementation will involve integrating the simulation controller with ROS (Robot Operating System) to facilitate real-world testing and deployment.

These advancements will significantly enhance the drone's capabilities, making it more efficient and reliable in performing complex tasks such as catching a ball and avoiding it when necessary.

References

[1] Quanser, "Qdrone." https://www.quanser.com/products/qdrone/, 2023. [Accessed: April 17, 2024].

[2] Vicon, "Award winning motion capture systems." https://www.vicon.com/, 2024. [Accessed: April 17, 2024].

[3] T. A. Johansen, "Introduction to nonlinear model predictive control and moving horizon estimation." https://folk.ntnu.no/torarnj/nonlinear.pdf, -. [Accessed: April 17, 2024].

[4] N. S. Mohd Mokhtar and K. Osman, "Modelling and pid control system integration for quadcopter dji f450 attitude stabilization," *Indonesian Journal of Electrical Engineering and Computer Science*, 09 2020. [Accessed: March 15, 2024].

[5] W. Selby, "Quadrotor system modeling - non-linear equations of motion." https://wilselby.com/research/arducopter/modeling/#:~:text=The%20quadrotor%20vehicle%20operates%20on,horizontal%20pair%20rotating%20counter%2Dclockwise., -. [Accessed: May 2, 2024].

[6] H. Asada, *Introduction to Robotics*. City: Massachusetts Institute of Technology, - ed., -.

[7] U. of Illinois, "Linearization." https://courses.engr.illinois.edu/ece486/fa2021/documentation/lectures/slides/Lecture25B_Linearization.pdf, 2012. [Accessed: May 2, 2024].

[8] LumenLearning, "Projectile motion — physics." https://courses.engr.illinois.edu/ece486/fa2021/documentation/lectures/slides/Lecture25B_Linearization.pdf, -. [Accessed: May 2, 2024].

[9] R. Roy, M. Islam, N. Sadman, M. A. P. Mahmud, K. D. Gupta, and M. M. Ahsan, "A review on comparative remarks, performance evaluation and improvement strategies of quadrotor controllers," *Technologies*, vol. 9, no. 2, 2021.

[10] N. Zohrabi, "Transient response." https://www.sciencedirect.com/topics/computer-science/transient-response, 2019. [Accessed: May 2, 2024].

[11] G. C. Kiam Heong Ang, "Pid control system analysis, design,and technology," *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, vol. 13, 2005.

[12] D. Peterson, "Integral windup method in pid control." https://control.com/technical-articles/intergral-windup-method-in-pid-control/, 2020. [Accessed: May 2, 2024].

[13] R. M. Murray, "Control and dynamical systems," *California Institute of Technology*, 2009.

[14] R. F. Francesco Nori, "Linear optimal control problems and quadratic cost functions estimation." https://www.researchgate.net/profile/Francesco-Nori/publication/254117963_Linear_Optimal_Control_Problems_and_Quadratic_Cost_Functions_Estimation/links/00b4953a1aa2b55414000000/Linear-Optimal-Control-Problems-and-Quadratic-Cost-Functions-Estimation.pdf, -. [Accessed: May 2, 2024].

[15] A. K. Mohamed H. Merzban and H. F. Hamed, "Comparison of various control techniques applied to a quadcopter." https://www.researchgate.net/publication/371972285_Comparison_of_various_Control_Techniques_Applied_to_a_Quadcopter/link/656dc653b86a1d521b307b58/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19, -. [Accessed: May 2, 2024].

[16] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11773–11780, 2014. 19th IFAC World Congress.

[17] Spring, "Three-dimensional rotation matrices," 2012.

[18] X. He, "Modeling and nonlinear control of a quadcopter for stabilization and trajectory tracking," *Journal of Engineering*, 2022.

[19] A. Gibiansky, "Quadcopter dynamics and simulation - andrew gibiansky." https://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/, 2012. [Accessed: April 20, 2024].

[20] MathWorks, "Control of quadrotor using nonlinear model predictive control." https://au.mathworks.com/help/mpc/ug/control-of-quadrotor-using-nonlinear-model-predictive-control.html, -. [Accessed: April 20, 2024].

[21] MathWorks, "Kalman filter." https://au.mathworks.com/help/control/ug/kalman-filtering.html, -. [Accessed: April 20, 2024].