Robust Quadrotor Control Schemes with Application to Mobile Sensor Networks in the Nuclear Industry

Aydin Joseph Efe Can, MEng

School of Engineering,
Lancaster University

Supervisor: Dr Allahyar Montazeri

Industry Supervisor: Joshua Price

A thesis submitted for the degree of

Doctor of Philosophy

August, 2025

Abstract

This thesis presents the development and validation of a number of robust algorithms for both single and multi-quadrotor control for use in the nuclear industry, to improve the safety and efficiency of environmental monitoring processes associated with nuclear decommissioning. The objectives include the development and validation of a robust and reliable control algorithm for a single quadrotor, the design, development and validation of a robust multi-quadrotor system, and validation of the efficacy of the proposed control systems for applications in the nuclear industry.

A novel finite-time integral sliding mode control system was developed for robust trajectory tracking of a single quadrotor. This control method demonstrated superior performance over existing techniques in simulations and real-world experimentation in the presence of parameter uncertainties. A discrete-time sliding mode control system was also introduced to handle for the discrete nature of onboard sensors when controlling the quadrotors in environments where GPS is unavailable, showing enhanced performance when sampling rates were slow.

A discrete-time sliding mode formation control system was designed and implemented to enable the robust formation control of multiple quadrotors around a dynamic virtual leader in the presence of external disturbances. The efficacy of the algorithm was validated experimentally, through implementation on the Crazyflie 2.1 micro-quadrotor platform.

Finally, the applications of the proposed control algorithms for the nuclear industry was demonstrated by adapting the discrete-time sliding mode formation control with a gradient-climbing virtual leader, for locating the source of a radiative sensor field. Additionally, the data collection capability of the multi-quadrotor system was verified experimentally, using Gaussian Process Regression to estimate the temperature distribution within an environment. This highlighted the potential for the system to safely and efficiently monitor hazardous nuclear environments.

Overall, this thesis advances the field of quadrotor control by delivering robust algorithms tailored for the nuclear industry.

Acknowledgements

I would like to express deep gratitude to all those who supported me through this project. Firstly, I would like to thank my supervisors, Dr. Allahyar Montazeri and Joshua Price, for their expertise and continued support throughout the project, as well as Prof. James Taylor and Dr. Stephen Monk for their valuable oversight as members of the supervisory panel.

I would also like to thank Lancaster University, as well as United Kingdom National Nuclear Laboratory, for the support and resources provided throughout the project. The support from the faculty and staff from both Lancaster University and United Kingdom National Nuclear Laboratory has been instrumental to this project's successes.

Next, I would like to express my appreciation to my family, for their encouragement during this project.

Finally, I would like to express my deepest gratitude for my wife Zoë, for her unwavering support, encouragement, and patience throughout this journey.

Funding

This work has been supported by the Centre for Innovative Nuclear Decommissioning (CINDe), which is led by United Kingdom National Nuclear Laboratory, in partnership with Sellafield Ltd. and a network of Universities that includes the University of Manchester, Lancaster University, the University of Liverpool and the University of Cumbria. The authors would also like to acknowledge the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/R02572X/1, and the National Centre for Nuclear Robotics (NCNR).

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words, including appendices and footnotes, but excluding the bibliography. A rough estimate of the word count is:

27991

Aydin Joseph Efe Can

Publications

A. Can, H. Efstathiades, and A. Montazeri, "Design of a chattering-free sliding mode control system for robust position control of a quadrotor," in *2020 International Conference Nonlinearity, Information and Robotics (NIR)*, 2020, pp. 1–6. DOI: 10.1109/NIR50484.2020.9290206

A. Montazeri and A. Can, "Unmanned aerial systems: Autonomy, cognition, and control," in Jan. 2021, pp. 47–80, ISBN: 9780128202760. DOI: 10.1016/B978-0-12-820276-0.00010-8

N. Sadeghzadeh Nokhodberiz, A. Can, R. Stolkin, and A. Montazeri, "Dynamics-based modified fast simultaneous localization and mapping for unmanned aerial vehicles with joint inertial sensor bias and drift estimation," *IEEE Access*, vol. PP, pp. 1–1, Aug. 2021. DOI: 10.1109/ACCESS.2021.3106864

A. Can, J. Price, and A. Montazeri, "A nonlinear discrete-time sliding mode controller for autonomous navigation of an aerial vehicle using hector slam," *IFAC-PapersOnLine*, vol. 55, pp. 2653–2658, Oct. 2022, DOI: 10.1016/j.ifacol.2022.10.110

A. Can, J. Price, and A. Montazeri, "Robust formation control and trajectory tracking of multiple quadrotors using a discrete-time sliding mode control technique," *IFAC-PapersOnLine*, vol. 55, pp. 2974–2979, Oct. 2022, DOI: 10.1016/j.ifacol.2022.10.184

I. H. Imran, A. Can, R. Stolkin, and A. Montazeri, "Real-time nonlinear parameter estimation and tracking control of unmanned aerial vehicles in closed-loop," *Scientific Reports*, vol. 13, Feb. 2023. DOI: 10.1038/s41598-023-29544-6

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Robotics in the Nuclear Industry

In history, industrial revolutions have signified giant leaps forward in industrial processes, from the use of steam-powered machines in the first industrial revolution to the use of robots and computer automation in the third. In 2011, the German government proposed a new term, "Industry 4.0", which was suggested as the fourth industrial revolution. Industry 4.0 (I4.0) refers to a new form of industry in which the Internet of Things (IoT), Artificial Intelligence (AI) and Cyber-Physical Systems (CPS) are utilised to create smart factories with capabilities such as interconnected supply chains, predictive maintenance, and human-machine interactions [2], [5].

Although the implementation of these technologies has brought about this revolution, questions remain about how increasing energy demands can be met. Wind and solar power provide sustainable and renewable energy. Despite successes of growing wind and solar capabilities in recent years, neither of these energy sources is expected to provide a consistent availability to produce a continuous stable source of power. Combining these energy sources with nuclear leverages the strength of both methods to provide a more robust and sustainable energy system, with reduced carbon emissions, enhanced grid stability, and reduced reliance on fossil fuels [5].

The nuclear industry in the UK is currently facing many new challenges. The

requirement to continue the production of nuclear energy through the construction and commissioning of new sites, while actively decommissioning existing nuclear legacy sites, present a unique set of problems. The decommissioning of the nuclear site Sellafield is expected to cost £121 billion, and take until the year 2120 [7]. Several programs, such as Integrated Innovation in Nuclear Decommissioning [8] and the Industrial Strategy Challenge Fund [9] have been introduced to tackle these issues through the use of innovative technologies such as robotics. Part of the Industrial Strategy Challenge Fund aimed to deliver robots for a safer world, to remove people from potentially harmful operations and improve the resilience and infrastructure of public services by leveraging unmanned systems [10]. Through funds such as these, technological demonstrations have shown how emerging technologies can be used by the nuclear industry to improve operations.

1.1.1 Plant Characterisation

One such challenge of nuclear decommissioning is plant characterisation and monitoring of inaccessible locations on nuclear sites. Currently, on Sellafield site, radiological protection monitoring is carried out manually using commercially available handheld radiation monitors. Monitoring through surveys is often undertaken by hand, can be lengthy, and place humans in hazardous environments. Adding to the autonomy of this process can improve safety and reduce the costs associated with the operation. In collaboration with Sellafield Ltd, the University of Manchester developed the Continuous Automated Radiation Monitoring Assistance (CARMA) platform. This is a mobile platform that offers autonomous and wireless radiometric floor mapping of a nuclear facility. After a demonstration to prove the effectiveness of the platform, improvements were made and a second platform, CARMA2, was later deployed on Sellafield site in 2018 to carry out complete radiological floor surveys [11]. In 1957, a fire at Sellafield's Windscale Pile 1 chimney left radioactive contamination in elevated, inaccessible areas. The location of the contamination meant it was unable to be manually monitored. In February 2015,

the RISER Unmanned Aerial Vehicle (UAV) system, equipped with various sensors, was flown remotely into a solvent recovery plant on Sellafield site. The initial test flight in this location demonstrated the capabilities of the technology. This was later followed by a second successful demonstration in Windscale Pile 1 Chimney on Sellafield site. The demonstrations proved that UAV technologies could be applied to these challenging nuclear environments [12]. This new system allowed the effective characterisation of a nuclear legacy facility to allow decommissioning to progress while reducing the radiation dose that operators are exposed to through using remote technology.

1.1.2 Condition Monitoring and Inspection

Alongside plant characterisation, condition monitoring and inspection of nuclear packages has been identified as a key challenge in the nuclear industry [13]. The nuclear industry has committed to the safe storage of low-level waste (LLW), intermediate-level waste (ILW), and high-level waste (HLW). Condition monitoring and inspection can play a particularly important role in ensuring the safe storage of ILW [14]. ILW is first encapsulated into different types of packages, including drums and large concrete boxes. These are then placed into a storage facility such as the Windscale Advanced Gas-cooled Reactor (WAGR) store on Sellafield site [15]. It's anticipated that bringing innovation to condition monitoring and inspection processes will significantly benefit Sellafield's decommissioning programme by improving the efficiency and safety of such operations [16]. Some potential areas for improvement involve the development of 'smart packages' that monitor themselves, technologies for visual observation and analysis, and large-area scanning [17].

1.1.3 Physical Asset Management

The commissioning of new nuclear sites, as well as the decommissioning of legacy sites such as Sellafield, involves procurement, storage, and installation of a wide variety of equipment. Currently, asset identification is undertaken manually by

personnel performing walks of a site [18]. Records are then updated manually to note any changes in the status, condition, or location of physical assets. This process is often timely and complicated, as physical assets come from a diverse number of manufacturers that implement unique numbering systems. Innovative physical asset management solutions will enable the nuclear industry, among other industries, to identify and track several physical assets from procurement to decommissioning. Some main challenge aims include assigning unique tracking codes to physical assets, visualisation of assets against a digital model, and real-time tracking of assets to aid informed decision-making [19]. Implementing UAVs as inspection aids for physical asset management can address issues associated with safety, accessibility, and time involved with much physical asset management. Automated systems with integrated intelligent UAVs could provide a potential solution to the challenges associated with regulation and report generation for the inspection of physical assets.

1.1.4 Waste Handling and Storage

The handling and storage of radioactive waste poses many challenges due to the hazardous properties of the waste. Types of waste are largely diverse, from wet slurries to metals, among other by products. Often, these diverse waste types are mixed and need to be segregated. Due to the hazardous properties of this waste, innovative remote techniques need to be developed to handle the waste more efficiently [20]. As a final disposal facility is not yet available in the UK, any waste produced by UK nuclear sites needs to be safely and securely stored for long periods. This requires innovative packaging and storage methods to be developed to allow the costs and efforts associated with waste storage to be reduced. There are several challenges associated with the handling and storage of nuclear waste. These include but are not limited to; restricted access to the waste that needs to be handled, a lack of direct line of sight to the waste, and assessment of the waste properties. One article suggests the inclusion of autonomy for the management of nuclear waste [21]. The article highlights trials undertaken at the United Kingdom National Nuclear

Laboratory Workington facility, in which a robotic system is being developed using a KUKA KR500 robotic arm for post-processing of nuclear waste. Currently, this system is teleoperated by operators, relying on static cameras, with pan, tilt and zoom (PTZ) control, for observation of the system. The framework for a rational agent-based robotic system is proposed for nuclear waste management that utilises a Kinect sensor with vision processing and an intelligent control system to separate and identify waste. This could reduce the workload of operators controlling robotic systems through the introduction of autonomy within the process [21].

1.2 Problems and Objectives

The complexities involved in decommissioning tasks on sites such as Sellafield are vast. Tasks such as plant characterization, condition monitoring, inspection, and waste handling are currently labour-intensive, potentially hazardous to employees, and costly. The existing methods for environmental monitoring, such as manual surveying and observing through static cameras and sensors, are inefficient in these extensive, high-risk environments.

Many of the challenges highlighted above require new technologies to be developed that can more safely and efficiently monitor, inspect, and characterise physical and environmental conditions. The nuclear industry is safety-critical, and as such, adopting new technologies can be difficult. High levels of radiation can cause degradation of sensor measurements and introduce noise and uncertainties into the system. Nuclear environments are often cluttered and uncertain, with objects such as trolleys or robotic arms causing dynamic hazards. Existing infrastructure, such as ventilation systems, can create unpredictable air dynamics, impacting the stability of aerial robots. Existing robotic solutions, such as the CARMA platform, still require a significant amount of oversight, and fully autonomous operation is limited by current technological capabilities. Due to the hazardous nature of nuclear environments, the nuclear industry adopts strict safety standards. The use

of novel robotic solutions must undergo rigorous testing to validate the proposed solutions. There is the potential for catastrophic failure, causing health risks and environmental damage. Therefore, the developed systems must be exceptionally reliable, introducing the need for novel systems to have robustness considered as a first priority.

The main aim of this research is to provide a robust formation controller for an autonomous or semi-autonomous formation of quadrotor agents to enhance the safety, efficiency and effectiveness of tasks involved in the nuclear decommissioning process. With this in mind, the project aspires to deliver a novel and robust system by addressing four research aims:

- Develop a highly robust control algorithm for trajectory tracking of a single quadrotor for the purpose of navigating within hazardous nuclear environments.
- Develop an algorithm to enable stable and reliable formation control of a group of quadrotors in hazardous nuclear environments.
- Validate the designed algorithms through extensive simulation and implement the algorithms in real-world experimental testing.
- Apply the designed algorithms to problems within the nuclear industry to validate the efficacy of the proposed system.

Achieving the aims identified in this section would provide a cyber-physical system capable of addressing a number of identified challenges within the nuclear industry. The field of robotic research in the nuclear industry provides a unique opportunity to address challenges involved with decommissioning large-scale legacy sites such as Sellafield. By advancing research in the field of autonomous and robust aerial robotics, the nuclear industry can enhance safety and improve overall efficiency, potentially reducing both the time and costs associated with decommissioning. The proposed system will provide a solution for navigating a

group of quadrotors in hazardous environments and enable the collection of real-time data, reducing the need for human intervention. The robust control algorithms will provide a solution capable of trajectory tracking and stable formation control in complex and dynamic nuclear sites. These algorithms will be fine-tuned through rigorous simulation and validated experimentally to improve their likelihood of adoption in safety-critical nuclear environments. Ultimately, this research aims to contribute to a safer and more efficient nuclear industry.

1.3 State of the Art

Many of the issues highlighted above require technologies to be developed that can more safely and efficiently monitor, inspect, and characterise physical and environmental conditions and properties. The proposed solution to the challenges identified relies on the concept of multi-agent systems. Multi-agent systems offer a promising and innovative way to understand, manage, and use distributed, large-scale, dynamic, open, and heterogeneous computing and information systems [22] . Here, an agent is an autonomous computational entity such as a software program or a robot that can perceive and act upon its environment.

This review draws upon the sensor network and heterogeneous multi-robot system as the components of such a multi-agent system to develop a system that not only interacts with internal agents but can interact with humans to achieve goal-oriented and task-oriented coordination, both cooperatively and competitively. The review pays special attention to the opportunities and challenges that exist in extreme environments, as well as the context of the challenges of the Game Changer within the nuclear industry.

As part of the work completed during this project, findings from a literature survey were initially published in part in the following book chapter.

- A. Montazeri and A. Can, "Unmanned aerial systems: Autonomy, cognition, and control," in Jan. 2021, pp. 47–80, ISBN: 9780128202760. DOI: 10.1016/

B978-0-12-820276-0.00010-8

This section provides an up-to-date investigation of the state of the art surrounding the field of single and multi-quadrotor control. A survey of the most recent literature can group quadrotor control methods:

- Linear control
- Nonlinear control methods

1.3.1 Linear Quadrotor Control Methods

Quadrotor systems are complex, nonlinear and under-actuated systems. Therefore, the control of these systems using linear methods relies heavily on the linearisation of the quadrotor dynamics about a hover set-point [23].

The main foundation of linear control in quadrotors can be grouped into three techniques:

- Proportional Integral Derivative
- Linear Quadratic techniques
- Robust linear techniques

Early implementations of quadrotor control systems relied on the use of proportional integral derivative (PID) controllers. One of the earliest works in the field of quadrotor control applied PID and linear quadrotor (LQ) control to the attitude system of a quadrotor [24]. The study showed successful control over the orientation of the quadrotor and provided a foundation for what would later become a hugely researched area. The study found that PID had advantages over LQ control due to its simplicity. Additionally, imperfections in the quadrotor model hindered the performance of the LQ controller, showing further research was required in this area. In the modern day, PID is considered the standard in quadrotor control due to its simplicity and historical precedence [25]–[28]. Following this, works began on PID

control in quadrotors, focusing on the 6 Degree of Freedom (DoF) trajectory tracking capabilities of the PID control system [29], [30]. A number of papers implement position control of the quadrotor through the introduction of a cascaded control structure, where an outer-loop PID controller provides position control inputs to an inner-loop attitude control system [31], [32].

Due to the short flight time of quadrotors, research works have focused on optimal control methods. Linear optimal control of quadrotors can be achieved with Linear Quadratic Regulator (LQR) and Linear Quadratic Gaussian (LQG) control systems. In one paper, LQR control was found to have acceptable performance when compared to PID [27]. It also found that the quadrotors displayed less steady-state error when compared to PID control when an accurate model of the quadrotor was provided. Another paper analyses the performance of a quaternion-based LQR controller [33]. The authors of one comparative study found the LQR displayed smoother trajectory tracking performance when compared to PID [30]. The results from the papers suggest that LQR is suitable for control when the parameters are known, suggesting that any parameter variation or disturbance could have detrimental effects on the system.

Robust linear control methods aim to provide acceptable performance in the presence of uncertainties and disturbances. One paper combines a linear controller with a robust compensator to allow robust tracking of a quadrotor's roll and pitch angles [34]. Another robust linear control method is provided by H_∞ control. The robust performance of H_∞ control is shown in a number of works [35]–[37] in the presence of disturbances and uncertainties. While H_∞ is a robust control method, it has more recently been criticised in the robust control of quadrotors due to poor handling of uncertainties [38].

1.3.2 Nonlinear Quadrotor Control Methods

Nonlinear controllers allow control over the quadrotor's nonlinear states rather than around a linearised operating point, allowing quadrotors to operate within their

full range of dynamics. In early work in the field, a nonlinear Adaptive Integral Backstepping controller is compared against both PID and Integral-LQR control to demonstrate the benefits of considering the nonlinear dynamics in the design of the control law [39]. This section explores the recent state of the art in nonlinear control techniques. The main foundation of nonlinear control in quadrotors is based on four techniques:

- Feedback Linearisation
- Backstepping
- Adaptive
- Model Predictive Control
- Sliding Mode Control

Feedback linearisation allows for exact state transformation of the quadrotor model, converting the nonlinear model into a full, or partially linear model for control, effectively cancelling the nonlinearities [40]. Early research towards this introduces feedback linearisation in combination with a linear LQR controller [41]. The paper finds that through feedback linearisation, the system was able to reject a bounded disturbance in both simulations and real-world tests. Another paper was able to demonstrate the effectiveness of feedback linearisation in combination with LQR by developing a controller with zero steady-state error and no overshoot [40]. Both of these papers address the attitude control of the quadrotor system. Full the purpose of trajectory tracking, a cascaded controller is developed in [42] that combines feedback linearisation with a simple PD controller. The paper shows desirable trajectory performance when compared to H_∞ control methods in the presence of disturbances. Feedback linearisation has also been shown to be applicable to fault-tolerant control, where the system can remain stable despite the loss of a rotor [43]. A more modern approach improves the robustness of feedback linearisation by combining the technique with a super-twisting algorithm alongside

a disturbance observer to estimate the disturbance in the system [44]. The control system demonstrates superior performance when compared to other modern, robust techniques.

One approach to the under-actuated control problem of quadrotors is Backstepping control. Backstepping control provides a good solution for the position and attitude control of a quadrotor due to the inherent cascaded nature of the backstepping control system. Early work in the field leverages this technique to provide control over the full quadrotor states [45], [46], where each subsystem is stabilised by designing control systems based on Lyapunov stability. Following on from these works, one paper suggests that general backstepping control is not sufficient for the control of micro-quadrotors due to the requirements of accurate parameter identification and is not sufficiently robust to external disturbances [47]. To solve this, the paper presents an adaptive integral backstepping control law that is capable of estimating system disturbance online. The designed controller shows improvements over general backstepping control in the presence of model uncertainties. In the current state of the art, backstepping is implemented as the foundational framework of more complex and robust methods for full trajectory tracking control of quadrotor systems [48]–[50].

Adaptive control techniques rely on the ability of a system to estimate parameters online. This parameter estimation can be used to augment the control algorithms to provide more robustness to parametric uncertainties in the system. Initial work implemented adaptive control to enable robustness to increases in payload [51]. The invariance to parameters through this technique is also demonstrated in [52], where the control system demonstrated robustness in the presence of a varying centre of gravity. Another paper implements adaptive control to enable robustness to varying inertial parameters on a quadrotor [53] and was successful in removing oscillations compared to integral techniques. Adaptive control has also been shown to be suitable for rejecting external disturbances applied to the system [54], as well as in cases of loss of thrust due to component damage [55]. Remarkably,

adaptive control techniques have even been shown to be robust against time-varying uncertainties [56]. The main disadvantage highlighted from the literature is the reduced computation efficiency of the algorithm due to the parameter estimation step.

Fuzzy logic control implements the idea of fuzzy rules to map inputs to control actions. Simple fuzzy logic control has been shown to have improvements over classical PID in [57]. Fuzzy logic controllers are often combined with other robust control techniques highlighted in this review to provide robust control for quadrotor systems [58]–[61]. The feasibility and superiority of fuzzy logic control are shown in [62], where strong robustness and fault tolerance is demonstrated through simulation and real-world testing.

Another approach to handling disturbances and uncertainties in dynamics is model predictive control (MPC). MPC relies on the idea that the future system states and control inputs can be predicted using an online model of the system [63]. In [64], a learning-based model predictive controller is implemented to allow a quadrotor to catch and throw a ball. The control system implements statistical learning techniques alongside control techniques to guarantee a level of robustness in the system. MPC can be implemented alongside feedback linearisation techniques in real-time on embedded hardware for onboard trajectory tracking of a quadrotor system [65]. MPC can also be combined with adaptive techniques, with online parameter identification, to improve the robustness of the system further [66]. In a more modern approach, MPC is combined with a deep-learning algorithm to achieve robust trajectory tracking performance of a quadrotor system [64]. MPC faces issues similar to those of the adaptive techniques, where additional computation is required to achieve stable results.

First presented in 2006, a popular robust technique in the robust control of quadrotor systems is sliding mode control (SMC) [67]. The technique was shown to be successful in robustly controlling the under-actuated system through the implementation of a cascaded sliding mode control system. Through the

implementation of control via a sliding surface, the stability of the system can be guaranteed, making it ideal for robust control in the presence of uncertainties and disturbances. Further to this, the order of the sliding surface can be extended to higher degrees to increase robustness [68]. Other work combines sliding mode control with a disturbance observer to achieve robust trajectory tracking [69]. More recently, work within the field has moved toward discrete-time sliding mode control due to the inherent discrete nature of the sensing systems on-board quadrotors [70]. More recently, discrete-time sliding mode control has shown to be robust against both rotational and translational disturbances in aerial robotics [71].

Due to the safety-critical nature of the nuclear industry, robust control is of utmost importance. The system can be impacted by varying sensor noise due to radiation, external wind disturbances from powerful ventilation systems, and model uncertainties from potential propeller damage. With this in mind, sliding mode control techniques should be used to control an aerial robot in these environments to guarantee the safe convergence of the system to the desired states in the presence of these uncertainties and disturbances. Additionally, due to the inherent discrete-time nature of the sensing and processing units onboard these mobile platforms, designing the control system in discrete-time could prove beneficial, as it guarantees the convergence of the system under lower sampling times, which cannot be achieved with equivalent continuous methods. Advancing the field of discrete-time sliding mode control for the control of UAVs could prove critical to the success and uptake of aerial vehicles in the nuclear industry. From the literature review, it is seen that the main disadvantage of sliding mode control techniques is the presence of chattering, where the sliding system states rapidly oscillate about the sliding surface. This is particularly present in discrete-time sliding mode control systems, as they are only marginally stable within a quasi-sliding mode band. Future work would need to analyse the impact of this on the system through real-world implementation to validate its efficacy.

1.3.3 Multi-Agent Control Problems

The extension of control algorithms to the multi-agent case is a popular emerging topic in the field of robotic control. Multi-agent control is a field of robotic control that aims to drive the states of a group of agents to some sort of agreement, be that synchronisation or a desired formation. The concept enables coordination between a number of separate agents to achieve complex tasks beyond the capabilities of any single agent. This literature survey aims to review works surrounding multi-agent control strategies that solve a number of problems that, when addressed, could enable the use of collaborative robots inside nuclear legacy sites. Specifically, the review investigates three topics that are relevant to collaborative robotics in the nuclear industry:

- The rendezvous problem
- The formation problem
- The source seeking problem

The rendezvous problem considers the task of a group of communicating agents achieving synchronisation of their states and is first posed in [72], and is modelled in [73]. Work towards the problem often implements a consensus control approach to achieve rendezvous of system states [74]–[76]. One solution to the rendezvous problem through consensus-based cooperative control of a multi-UAV system is proposed in [77]. The paper shows that the states of the system are able to converge to a single point as long as there is at least one directed spanning tree in the communication graph. More recently, research has moved towards robust consensus-based control of multi-UAV systems, where one paper demonstrates successful convergence of each UAV's states in the presence of disturbances [78]. One more recent paper focuses on the rendezvous problem for multi-agent systems with input constraints [79]. The paper considers constraints on the input amplitude and rate constraints. The authors achieve this using a velocity damping term within the

control algorithm. By considering these constraints, the developed control system is guaranteed to achieve rendezvous while satisfying the input amplitude and rate constraints.

An extension of the rendezvous problem that is more applicable to control over quadrotors is the formation problem. The formation problem in regards to multi-agent control proposes the idea of a group of agents achieving a desired geometric formation, with each agent converging towards a desired separation between each agent. One paper proposes a low-cost test bed for designing multi-quadrotor formation control algorithms and implements a consensus-based formation controller to achieve a desired formation [80]. A formation controller for a multi-quadrotor system based on model predictive control is designed in [81]. The paper shows that formation control can be achieved for quadrotors through a hybrid approach where a linear MPC handles the inner-loop states of the quadrotor, while a hybrid MPC handles the path planning strategy to bring the quadrotors into the desired formation. A novel formation controller for a multi-quadrotor system is proposed in [82] based on backstepping control. The results are found to improve the steady-state error in the formation when compared to a Laplace method and an MPC method. Sliding mode control is also a popular approach to solving the multi-quadrotor formation problem, with a number of papers showing desirable results in terms of steady-state error and robustness [83]–[85]. In order to combat communication delays and slow sampling rates, one paper implements a discrete-time sliding mode formation controller in [86]. The paper shows that when implementing the control law in discrete-time, poor performance can be introduced through slow sampling rates. The discrete-time sliding mode control approach is extended in [87] through the introduction of a super-twisting algorithm. However, the paper does not address the possibility of external disturbances in the system and infers the robust properties of the controller through simulation results. Work still remains on discrete-time sliding mode formation control of quadrotors, as there is a lack of research that implements the algorithms experimentally. An alternative approach to traditional

control algorithms for solving the formation control problem is Reinforcement Learning (RL). In [88], RL is used to achieve formation control of a heterogeneous network of quadrotors and Unmanned Ground Vehicles. By leveraging system data, the paper implements RL to produce a fault-tolerant controller without accurate knowledge of each vehicle's dynamics. This allows the system to achieve formation in the presence of communication and actuator faults. In other work, RL is used to achieve an optimal formation strategy for a group of quadrotors under switching topologies [89]. The paper demonstrates the effectiveness of the proposed optimal leader-follower formation control strategy in simulation. This paper also demonstrates how RL is able to use sampled vehicle data to achieve these results without information of vehicle dynamics.

Building on the formation control problem, source-seeking is a task in which a group of agents in formation navigate towards the peak of a field based on sensor data. It is clear that this would be extremely useful for condition monitoring and inspection in nuclear sites by allowing temperature or radiation hotspots to be rapidly located by a group of autonomous agents. One paper provides a solution to the two-dimensional source-seeking problem for a scalar field [90]. The paper observes that through circular formation, the agents are able to estimate the gradient of the field. The control system then uses this information to steer the formation towards the peak of the environmental field. The results show that the agents are successful in localising a source in a scalar field. Source-seeking control can be achieved with robust techniques such as sliding mode control, as shown in [91]. This paper implements a sliding mode control approach to steer the leader of a formation of quadrotors to navigate towards the peak of a radiative field. An alternative method to gradient estimation for the purpose of source seeking is provided in [92], where an approach using an Extended Kalman Filter (EKF) is used to estimate the location of the source of a radiative field. The paper shows that the EKF-based approach improved convergence time, which is critical in quadrotor systems where flight time is limited. One paper also works towards overall system

robustness by implementing a source-seeking algorithm in a network of agents with switching network topologies [93]. Interestingly, it has been shown that source-seeking algorithms can be adapted to achieve contour mapping [94]. In this paper, a network of agents locates the peak of a scalar field and then proceeds to map a contour of a set value. This could be particularly useful within the nuclear sector, as it would allow a group of agents to identify safe zones in an environment. For locating time-varying sources in three-dimensional scalar fields, one paper proposes a solution using Adaptive Navigation control techniques [95]. In this paper, a Cluster-Space formation control algorithm is used to achieve a desired three-dimensional formation. The Adaptive Navigation layer of the control architecture provides cluster-shape and mobility signals to the Cluster-Space controller. The gradient is estimated at the centre of the cluster using measurements from each UAV, and is used by the Adaptive Navigation layer to guide the UAVs towards the peak of a three-dimensional scalar field. In more recent work, one paper implements a novel control algorithm that combines a gradient-free optimisation algorithm with a consensus-based formation control system to control a formation of quadrotors towards the peak of a scalar field [96]. This paper demonstrates the effectiveness of the control system in both simulation and physical experiments. The results show that the control system is capable of locating the peak of a two-dimensional scalar field faster and more accurately that previous approaches, without estimating the gradient of the field.

While these works provide useful insights into source-seeking algorithms, they are mostly implemented in simulation and do not show the experimental performance of the proposed algorithms. Additionally, much of the literature does not consider the robustness problem that is inherent in tasks of this nature.

The remainder of this thesis is organised as follows. Chapter 2 proposes a novel continuous-time sliding mode control system for the robust and chattering-free trajectory tracking control of a single quadrotor. Chapter 3 applies a robust discrete-time sliding mode control to a single quadrotor and analyses the performance in the presence of varying sample time in simulation through the Robot Operating

System (ROS). Chapter 4 extends the discrete-time sliding mode controller to the multi-agent case to investigate the robust performance of discrete-time sliding mode formation control for a group of quadrotor agents. Chapter 5 demonstrates the effectiveness of the proposed control strategies for applications in source-seeking and characterisation of nuclear legacy sites. Chapter 6 provides a discussion of the findings of the thesis, alongside concluding remarks and some ideas for future research in the field of robotic sensing in the nuclear industry.

Chapter 2

Finite-Time Sliding Mode Controller for a Quadrotor System

2.1 Introduction

In recent years, Unmanned Aerial Vehicles have become increasingly popular in commercial sectors, such as agriculture [97], infrastructure monitoring [98], and emergency response. Due to the agile and manoeuvrable nature of quadrotor UAVs, their remote sensing capabilities, and the significant reduction in development costs recently, they provide an ideal solution to many challenges faced in hazardous environments unsuitable for human access [99]. One area currently exploring UAV technology is the nuclear industry. In February 2015, the RISER UAV system, equipped with visual and radiometric sensors, was flown remotely into a solvent recovery plant on the Sellafield site. The initial test flight in this location demonstrated the capabilities of the technology. This was later followed by a second successful demonstration in Windscale Pile 1 Chimney on Sellafield site. The demonstrations proved that UAV technologies could be applied to these challenging nuclear environments [100]. The nuclear industry has identified a number of challenges through the Game Changers programme [101]. Using emerging technologies, such as UAVs, to address these challenges can greatly impact the future

of the nuclear industry [2].

Safety is a critical concern when adopting new technology within the nuclear industry, as there are several hazards to consider. Uncertainties such as sensor degradation, radiation noise, and a dynamic environment with many obstacles can cause several issues for UAV technology. For autonomous UAVs to be implemented in the nuclear industry, they must be robust against these uncertainties and disturbances. This, combined with the nonlinear and under-actuated dynamics of quadrotor UAVs, requires the development of advanced control algorithms for full position and attitude control. Recent advances in control systems allow these UAVs to robustly track planned trajectories autonomously without human involvement. Classic Proportional-Integral-Derivative (PID) control has been widely used for the control of quadrotor UAVs due to its practicality and ease of implementation [32]. The PID control, among other linear control methods, relies on the linearisation of the quadrotor dynamics around a point of hover. While the system is robust around this hover point, large deviations in states can cause the nonlinear quadrotor system to become unstable.

Indoor nuclear environments are often cluttered and GPS-denied [102]. As GPS can not be used for positioning in these environments, simultaneous localisation and mapping (SLAM) must be used to localise these robots in 3D space. Various 2D and 3D SLAM methods have been developed in recent years for robotic applications, such as fastSLAM [3], OctoMapping [103], Gmapping [104], Orb-SLAM2 [105], and Hector SLAM [106]. These various SLAM techniques rely on sensor data from either LiDAR sensors, or visual sensors, such as single camera, stereo camera, or RGBD camera devices. As such, with changing payloads of the quadrotor, it is essential that control systems must handle variations in the quadrotor parameters.

Sliding mode control provides a robust nonlinear solution for full control of a quadrotor's dynamics. Studies have validated the robustness of classical sliding mode control and its ability to handle uncertainties [107]. One disadvantage of classical sliding mode control is the presence of a phenomenon known as chattering,

where the state of the system can experience high frequency oscillations of a finite frequency and magnitude, caused by unmodeled dynamics or discrete sensing [108]. The chattering phenomenon can cause poor controller accuracy, degradation of mechanical components, and power losses [109]. The design and development of a chattering-free sliding mode controller is reported in [110], [111] and [112] through the use of a continuous and time-varying sliding surface. An alternative method to design a chattering-free sliding mode control involves introducing a non-singular terminal sliding mode controller [113], [114]. More recent work has introduced the use of an Extended Kalman Filter for practical implementation of the controller [115]. In the case of quadrotors with unknown inertia parameters, an adaptive closed-loop identification and control method can be implemented [116]. In another paper, an event-triggered particle filter was implemented to improve the energy management of the state estimation procedure onboard the quadrotor [117].

Initial results from work towards the development of robust control algorithms for quadrotor UAVs in hazardous environments, completed as part of this PhD research, were presented at a conference and published.

- A. Can, H. Efstathiades, and A. Montazeri, "Design of a chattering-free sliding mode control system for robust position control of a quadrotor," in *2020 International Conference Nonlinearity, Information and Robotics (NIR)*, 2020, pp. 1–6. DOI: 10.1109/NIR50484.2020.9290206

In this research, a nested chattering-free sliding mode controller was developed for position and attitude control of a quadrotor UAV for use in hazardous environments with parametric uncertainties. This chapter further builds on the novelty of the published research through the introduction of an integral term within the sliding surface, as well as a new robust control term in order to guarantee system stability and improve overall performance. This chapter also provides a rigorous stability analysis to derive the robust stability condition of the under-actuated system and prove that the system is finite-time stable. The proposed control system is compared to a modern finite-time sliding mode controller [118]

in order to evaluate its performance in simulation. The controller is implemented experimentally to illustrate the efficacy of the proposed controller.

The remainder of this chapter is organised as follows: Section 2.2 describes the quadrotor dynamical model; Section 2.3 presents the derivation of the flight controller; Section 2.4 displays the simulation results; Section 2.5 displays the experimental results; Section 2.6 discusses the chapters findings and results.

2.2 Quadrotor Dynamic Model

This section describes the dynamical model of an x-frame quadrotor system. Figure 2.1 displays a diagram of the quadrotor system, the coordinate system used, and the propeller directions.

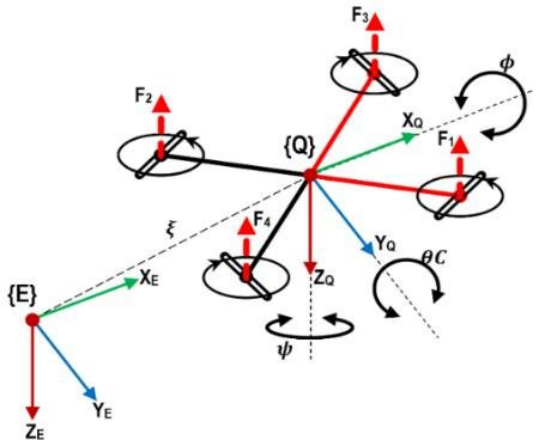


Figure 2.1: Diagram of an X-frame quadrotor with the coordinate systems and propeller directions [119].

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}. \tag{2.1}$$

Here, $\boldsymbol{\omega} \in \mathbb{R}^4$ is a vector that contains the four propeller velocities, each represented by ω_i.

The pose of the quadrotor can be described using two vectors $\mathbf{r}$ and $\boldsymbol{\alpha}$. Here vector $\mathbf{r}$ represents the position of the quadrotor in 3D space and $\boldsymbol{\alpha}$ contains the Euler angles of the system, representing the attitude of the quadrotor.

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \boldsymbol{\alpha} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}, \tag{2.2}$$

where φ, θ, and ψ are the roll, pitch, and yaw angles, respectively, and x, y and z describe the 3D coordinates of the quadrotor. The Euler angles of the quadrotor can be used to determine how the rigid body of the quadrotor is oriented in space with the rotational matrix $\mathbf{R}$ for a ZYX rotation order, shown in (2.3).

$$\mathbf{R} = \begin{bmatrix} c\theta c\psi & c\psi s\varphi s\theta - c\varphi s\psi & s\varphi s\psi + c\varphi c\psi s\theta \\ c\theta s\psi & c\varphi c\psi + s\varphi s\theta s\psi & c\varphi s\theta s\psi - c\psi s\varphi \\ -s\theta & c\theta s\varphi & c\varphi c\theta \end{bmatrix}, \tag{2.3}$$

where $c\varphi = \cos(\varphi)$ and $s\varphi = \sin(\varphi)$.

In order to define the equations of motion, we can use Newton-Euler formulae for rigid body dynamics.

$$\frac{d^2\mathbf{r}}{dt^2} = g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \frac{\mathbf{R}K_T}{m} \sum_{i=1}^{4} \omega_i^2 - \frac{\dot{\mathbf{r}}}{m} \begin{bmatrix} k_{dx} \\ k_{dy} \\ k_{dz} \end{bmatrix}, \tag{2.4}$$

where K_T is the coefficient of thrust for the propellers,k_{dx}, k_{dy} and k_{dz} are the drag terms in the x y and z directions, m is the mass of the quadrotor, and g represents the acceleration of the system due to gravity. The rotational motion of the system can be defined as a set of torques around each axis of the quadrotor system. Equations (2.5)-(2.7) describe the torques.

$$\tau_\varphi = K_T l(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2), \tag{2.5}$$

$$\tau_\theta = K_T l(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2), \tag{2.6}$$

$$\tau_\psi = K_D(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2), \tag{2.7}$$

where τ_φ, τ_θ, and τ_ψ are the roll, pitch and yaw torques respectively, K_D is the drag coefficient due to the propellers and l is the quadrotor's arm length. From Euler's second law of motion for rigid body dynamics, we get (2.8).

$$\dot{\boldsymbol{\omega}}_{\boldsymbol{\alpha}} \times \mathbf{I} = \boldsymbol{\tau}_{\boldsymbol{\alpha}} - (\boldsymbol{\omega}_{\boldsymbol{\alpha}} \times \mathbf{I}\boldsymbol{\omega}_{\boldsymbol{\alpha}}) - \boldsymbol{\tau}_{\mathbf{g}}, \tag{2.8}$$

where $\boldsymbol{\tau}_{\mathbf{g}}$ is the gyroscopic effect torque due to the rotors. As quadrotors operating in nuclear environments are not expected to perform aggressive manoeuvres and will be operating near-hover, it is safe to use the small-angle assumption when developing flight control systems for these environments. Control systems designed using the small-angle assumption can increase computational efficiency while providing control algorithms that are more easily tuned and implemented in real-world scenarios. Using the small-angle assumption,

$$\boldsymbol{\omega}_{\boldsymbol{\alpha}} = \begin{bmatrix} \omega_\varphi \\ \omega_\theta \\ \omega_\psi \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \quad \boldsymbol{\tau}_{\boldsymbol{\alpha}} = \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}, \quad \mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}.$$

The quadrotor system is an under-actuated system with 6 degrees of freedom and only 4 actuators to control them. The force applied on the quadrotor system from the four actuators can be converted into the overall thrust and torques acting on the system. Equation (2.9) shows this conversion.

The inputs of the system are defined as

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} T \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ lK_T & -lK_T & -lK_T & lK_T \\ lK_T & lK_T & -lK_T & -lK_T \\ K_D & -K_D & K_D & -K_D \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{2.9}$$

where $\mathbf{u}$ is a vector containing the system inputs, u_1 is the total thrust T of the UAV and u_2, u_3, and u_4 are the rotational torques for roll, pitch, and yaw.

Using equations (2.4),(2.8), and (2.9), the full dynamical model of the quadrotor can be defined. Equations (2.10) - (2.15) describe the 6 nonlinear equations of motion for the quadrotor.

$$\ddot{x} = (-c\varphi c\psi s\theta - s\varphi s\psi)\frac{u_1}{m} - \frac{k_{dx}}{m}\dot{x} \tag{2.10}$$

$$\ddot{y} = (c\psi s\varphi - c\varphi s\theta s\psi)\frac{u_1}{m} - \frac{k_{dy}}{m}\dot{y} \tag{2.11}$$

$$\ddot{z} = g - c\varphi c\theta\frac{u_1}{m} - \frac{k_{dz}}{m}\dot{z} \tag{2.12}$$

$$\ddot{\varphi} = \frac{(I_{yy} - I_{zz}) \times \dot{\theta} \times \dot{\psi}}{I_{xx}} - \frac{J_r\dot{\theta}}{I_{xx}}(\omega_1 - \omega_2 + \omega_3 - \omega_4) + \frac{u_2}{I_{xx}}, \tag{2.13}$$

$$\ddot{\theta} = \frac{(I_{zz} - I_{xx}) \times \dot{\varphi} \times \dot{\psi}}{I_{yy}} + \frac{J_r\dot{\varphi}}{I_{yy}}(\omega_1 - \omega_2 + \omega_3 - \omega_4) + \frac{u_3}{I_{yy}}, \tag{2.14}$$

$$\ddot{\psi} = \frac{(I_{xx} - I_{yy}) \times \dot{\varphi} \times \dot{\theta}}{I_{zz}} + \frac{u_4}{I_{zz}}, \tag{2.15}$$

2.3 Flight Controller Design

The problem of flight control in a quadrotor system is complex due to the under-actuated and nonlinear dynamics of the quadrotor system. Robustness is a particularly important property for quadrotors within a nuclear setting, due to the safety-critical nature of nuclear processes. Disturbances such as wind, changing payloads for different characterisation tasks, and sensor degradation from nuclear radiation can all affect the performance of a quadrotor. Ventilation systems inside of nuclear legacy sites can create areas of wind gusts that the quadrotor must fly through. For this reason, robust control methods, such as sliding mode control, are desirable for application in the nuclear industry due to their inherent ability to reject both external and internal disturbances.

A full robust trajectory tracking controller for the quadrotor is designed in a nested control form such that the position control of x and y, and the attitude control of φ and θ are separated into two distinct control loops, known as the outer-loop and the inner-loop. The outer-loop handles position control of the quadrotor,

while the inner-loop handles the attitude of the quadrotor. Altitude control of z and the yaw control of ψ are considered separate channels and are included within the inner-loop control system. The altitude controller produces the desired overall thrust value u_1. The position controller then outputs the desired roll and pitch values φ_d and θ_d. The desired attitude is handled with an attitude controller to provide the system inputs u_2, u_3 and u_4. The system inputs are then converted to individual motor speeds using equation (2.9). A flow diagram demonstrating the structure of the nested controller is displayed in Figure 2.2.

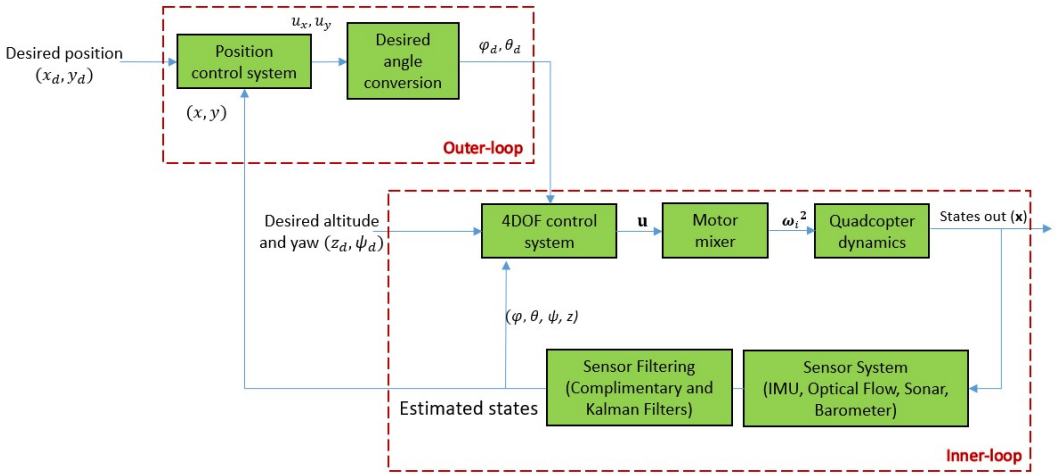


Figure 2.2: Flow diagram describing the structure of the nested control system.

2.3.1 Problem Formulation

The quadrotor model derived in Section 2.2 can be redefined in state space form as a nonlinear dynamical system.

$$\begin{cases} \dot{\mathbf{x}}_1 = \mathbf{x}_2 \\ \dot{\mathbf{x}}_2 = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} + \mathbf{d}(t). \end{cases} \tag{2.16}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \tag{2.17}$$

where $\mathbf{x_1} = [x\ y\ z\ \varphi\ \theta\ \psi]^T$, $\mathbf{x_1} \in \mathbb{R}^6$ represents the state vector of the quadrotor states. The vector $\mathbf{x_2} = [\dot{x}\ \dot{y}\ \dot{z}\ \dot{\varphi}\ \dot{\theta}\ \dot{\psi}]^T$, $\mathbf{x_2} \in \mathbb{R}^6$ is a vector of the time derivative of the states. The states, and their derivatives are concatenated into vector $\mathbf{x} \in \mathbb{R}^{12}$. The vector $\mathbf{u} = [u_1\ u_2\ u_3\ u_4]^T, \mathbf{u} \in \mathbb{R}^4$ denotes the control inputs in equation (2.9). The terms $\mathbf{f} : \mathbb{R}^{12} \rightarrow \mathbb{R}^6$ and $\mathbf{g} : \mathbb{R}^{12} \rightarrow \mathbb{R}^{6\times4}$ are nonlinear functions that describe the system dynamics represented in equations (2.10)-(2.15) and $\mathbf{d}(t) \in \mathbb{R}^6$ represents an external matched wind disturbance acting on the system. It is assumed that $\mathbf{d}(t)$ is bounded such that $\|\mathbf{d}(t)\|_\infty \leq d_{max}, \forall t \in \mathbb{R}$. Due to the time-varying nature of the quadrotor parameters, it is also important to consider their effect as dynamic perturbation on the functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$. Both nonlinear functions can be separated into their nominal terms and the disturbance as

$$\mathbf{f}(\mathbf{x}) = \mathbf{f_0}(\mathbf{x}) + \Delta\mathbf{f}(\mathbf{x}), \tag{2.18}$$

$$\mathbf{g}(\mathbf{x}) = \mathbf{g_0}(\mathbf{x}) + \Delta\mathbf{g}(\mathbf{x}), \tag{2.19}$$

where $\mathbf{f_0} : \mathbb{R}^{12} \rightarrow \mathbb{R}^6$ and $\mathbf{g_0} : \mathbb{R}^{12} \rightarrow \mathbb{R}^{6\times4}$ represent the nominal terms shown in (2.10)-(2.15), and $\Delta\mathbf{f} : \mathbb{R}^{12} \rightarrow \mathbb{R}^6$ and $\Delta\mathbf{g} : \mathbb{R}^{12} \rightarrow \mathbb{R}^{6\times4}$ represent the system dynamic uncertainties resulting from uncertainties in parameters I_{xx}, I_{yy}, I_{zz} and m. It is also assumed that $\Delta\mathbf{f}(\mathbf{x})$ and $\Delta\mathbf{g}(\mathbf{x})$ are norm bounded such that $\|\Delta\mathbf{f}(\mathbf{x})\|_\infty \leq f_{max}$ and $\|\Delta\mathbf{g}(\mathbf{x})\|_1 \leq g_{max}$ for all $\mathbf{x} \in \mathbb{R}^{12}$.

Remark 1: To illustrate how $\Delta\mathbf{f}(\mathbf{x})$ and $\Delta\mathbf{g}(\mathbf{x})$ can be derived for the quadrotor dynamics presented in (2.10)-(2.15), we perturb I_{xx}, I_{yy}, I_{zz} and m by ΔI_{xx}, ΔI_{yy}, ΔI_{zz} and Δm respectively. As such, the perturbed dynamics of the system in (2.18) and (2.3.1), i.e. $\Delta\mathbf{f}(\mathbf{x})$ and $\Delta\mathbf{g}(\mathbf{x})$ become

$$\Delta\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \frac{k_{dx}\dot{x}\Delta m}{m(m+\Delta m)} \\ \frac{k_{dy}\dot{y}\Delta m}{m(m+\Delta m)} \\ \frac{k_{dz}\dot{z}\Delta m}{m(m+\Delta m)} \\ \frac{(I_{xx}(\Delta I_{yy}-\Delta I_{zz})-\Delta I_{xx}(I_{yy}-I_{zz}))\dot{\theta}\dot{\psi}+\Delta I_{xx}J_r\dot{\theta}\Omega_r}{I_{xx}(I_{xx}+\Delta I_{xx})} \\ \frac{(I_{yy}(\Delta I_{zz}-\Delta I_{xx})-\Delta I_{yy}(I_{zz}-I_{xx}))\dot{\varphi}\dot{\psi}-\Delta I_{yy}J_r\dot{\varphi}\Omega_r}{I_{yy}(I_{yy}+\Delta I_{yy})} \\ \frac{(I_{zz}(\Delta I_{xx}-\Delta I_{yy})-\Delta I_{zz}(I_{xx}-I_{yy}))\dot{\varphi}\dot{\theta}}{I_{zz}(I_{zz}+\Delta I_{zz})} \end{bmatrix},$$

$$\Delta \mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\Delta m(c\varphi c\psi s\theta + s\varphi s\psi)}{m(m+\Delta m)} & 0 & 0 & 0 \\ \frac{\Delta m(c\varphi s\theta s\psi - c\psi s\varphi)}{m(m+\Delta m)} & 0 & 0 & 0 \\ \frac{\Delta m(c\varphi c\theta)}{m(m+\Delta m)} & 0 & 0 & 0 \\ 0 & -\frac{\Delta I_{xx}}{I_{xx}(I_{xx}+\Delta I_{xx})} & 0 & 0 \\ 0 & 0 & -\frac{\Delta I_{yy}}{I_{yy}(I_{yy}+\Delta I_{yy})} & 0 \\ 0 & 0 & 0 & -\frac{\Delta I_{zz}}{I_{zz}(I_{zz}+\Delta I_{zz})} \end{bmatrix},$$

where $\Omega_r = (\omega_1 - \omega_2 + \omega_3 - \omega_4) \in \mathbb{R}$.

The desired values of the system can be represented by $\mathbf{x_{1_d}} = [x_d \ y_d \ z_d \ \varphi_d \ \theta_d \ \psi_d]^T, \mathbf{x_{1_d}} \in \mathbb{R}^6$ and $\mathbf{x_{2_d}} = [\dot{x}_d \ \dot{y}_d \ \dot{z}_d \ \dot{\varphi}_d \ \dot{\theta}_d \ \dot{\psi}_d]^T, \mathbf{x_{2_d}} \in \mathbb{R}^6$. The tracking error of the system can be represented by $\mathbf{e_1} = \mathbf{x_1} - \mathbf{x_{1_d}} = [e_{1_x} \ e_{1_y} \ e_{1_z} \ e_{1_\varphi} \ e_{1_\theta} \ e_{1_\psi}]^T, \mathbf{e_1} \in \mathbb{R}^6$ and $\mathbf{e_2} = \mathbf{x_2} - \mathbf{x_{2_d}} = [e_{2_x} \ e_{2_y} \ e_{2_z} \ e_{2_\varphi} \ e_{2_\theta} \ e_{2_\psi}]^T$, $\mathbf{e_2} \in \mathbb{R}^6$. Furthermore, we take the time derivatives of error as

$$\begin{cases} \dot{\mathbf{e}}_\mathbf{1} = \mathbf{e_2} \\ \dot{\mathbf{e}}_\mathbf{2} = (\mathbf{f}(\mathbf{x}) - \dot{\mathbf{x}}_{\mathbf{2_d}}) + \mathbf{g}(\mathbf{x})\mathbf{u} + \mathbf{d}(t). \end{cases} \tag{2.20}$$

2.3.2 Finite-Time Integral Sliding Mode Control

From [120], it can be seen that the inclusion of an integral term within the sliding surface can help guarantee the invariance of the system to disturbance and uncertainties in the system in the reaching phase of the sliding mode control. Nevertheless, this can increase the chattering phenomenon found in the sliding phase of the controller. The conventional approach to reduce the effect of chattering in the sliding phase is to design a low-pass filter using the equivalent control technique or integrate it with higher-order SMC. This effect is reduced by implementing a finite-time sliding mode control approach by following [1].

First it is necessary to introduce some lemmas.

Lemma 1. *Suppose that there exists a continuously differentiable function* $V(x)$ $\mathbb{R}^n \to \mathbb{R}$, *and number* $c > 0, c \in \mathbb{R}$ *and* $0 < a < 1, a \in \mathbb{R}$ *such that*

$$V(0) = 0,$$

$$V(x) \text{ is positive definite on } \mathbb{R}^n,$$

$$\dot{V}(x) \leq -cV^a(x), \forall x \in \mathbb{R}^n.$$

The settling time T of the system can be defined as

$$T(x(0)) \leq \frac{V(x(0))^{1-a}}{c(1-a)}, \tag{2.21}$$

where $V(x(0))$ is the initial value of $V(x)$.

proof. See [121].

Lemma 2. *For real numbers $l_i, i = 1, 2, ..., n$ and for every $a \in (0, 1)$, the following inequality holds:*

$$(|l_1| + \ ... \ + |l_n|)^a \leq |l_1|^a + \ ... \ + |l_n|^a. \tag{2.22}$$

proof. See [122].

First, we take the nominal sliding surface as

$$\boldsymbol{\sigma_0} = \mathbf{e_2} + \mathbf{\Lambda e_1}, \tag{2.23}$$

where $\boldsymbol{\sigma_0} \in \mathbb{R}^6$ and $\mathbf{\Lambda} = diag[\lambda_x,\ \lambda_y,\ \lambda_z,\ \lambda_\varphi,\ \lambda_\theta,\ \lambda_\psi] \in \mathbb{R}^{6\times6}$ is a vector of tuneable parameters. Next, an integral term is introduced and a new sliding surface is defined as

$$\boldsymbol{\sigma_I} = \boldsymbol{\sigma_0} - \boldsymbol{\sigma_0(0)} + \mathbf{k_c} \int_0^t \boldsymbol{\sigma_0}\ d\tau, \tag{2.24}$$

where $\boldsymbol{\sigma_I} \in \mathbb{R}^6$ is the new integral sliding surface, $\boldsymbol{\sigma_0(0)} \in \mathbb{R}^6$ is the initial condition of the sliding surface at $t = 0$ and $\mathbf{k_c} =$
$diag[k_{c_x},\ k_{c_y},\ k_{c_z},\ k_{c_\varphi},\ k_{c_\theta},\ k_{c_\psi}], \mathbf{k_c} \in \mathbb{R}^{6\times6}$. Next, a Lyapunov candidate function is defined as

$$V = \sum_{i=1}^{n} V_i, \tag{2.25}$$

where each V_i is an individual Lyapunov candidate for each channel of the quadrotor system. Each separate channel can be defined as

$$V_i = \eta_i |\sigma_{I_i}|^{\gamma_i}, \tag{2.26}$$

where $\gamma_i = \frac{a_i}{b_i}$, $0 < a_i < b_i$, $a_i, b_i \in \mathbb{N}$, and a_i and b_i are odd. A sufficient condition for the stability of the system is each component $\dot{V}_i$ must be negative.

$$\dot{V}_i = \eta_i \gamma_i {\sigma_{I_i}}^{(\gamma_i - 1)} \dot{\sigma}_{I_i} sign(\sigma_{I_i}) < 0. \tag{2.27}$$

We can achieve this by choosing $\dot{V}_i$ such that

$$\dot{V}_i = -\mu_i |\sigma_{I_i}|^{\delta_i} < 0, \tag{2.28}$$

where $\delta_i = \frac{p_i}{q_i}$, $0 < p_i < q_i$, $p_i, q_i \in \mathbb{N}$ and p_i and q_i are odd.

From taking the time derivative of the Lyapunov candidate function (2.27), and inserting it into (2.28) the stability condition can be written as

$$\dot{\sigma_{I_i}} = -\frac{\mu_i}{\eta_i \gamma_i} {\sigma_{I_i}}^{(\delta_i - \gamma_i + 1)}. \tag{2.29}$$

Here, the power of σ_{I_i} can be re-written as ζ_i where $\zeta_i = \frac{\alpha_i}{\beta_i}$, $0 < \alpha_i < \beta_i$, $\alpha_i, \beta_i \in \mathbb{N}$ and α_i and β_i are odd. By designing the stability condition in this way, singularities often present in Terminal Sliding Mode Control design can be avoided, as here the final fractional power is strictly a positive, odd fractional power. This ensures the control law is non-singular. Furthermore, the coefficients can be collected into a single term identified as $\bar{\mu}_i$.

$$\dot{\sigma_{I_i}} = -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i}. \tag{2.30}$$

Next, we take the derivative of the integral sliding surface in (2.24)

$$\dot{\boldsymbol{\sigma}}_{\mathbf{I}} = \dot{\boldsymbol{\sigma}}_{\mathbf{0}} + \mathbf{k_c}\boldsymbol{\sigma}_{\mathbf{0}}, \tag{2.31}$$

$$\dot{\boldsymbol{\sigma}}_{\mathbf{I}} = \dot{\mathbf{e}}_{\mathbf{2}} + \boldsymbol{\Lambda}\dot{\mathbf{e}}_{\mathbf{1}} + \mathbf{k_c}\boldsymbol{\sigma}_{\mathbf{0}}, \tag{2.32}$$

$$\dot{\boldsymbol{\sigma}}_{\mathbf{I}} = (\mathbf{f}(\mathbf{x}) - \dot{\mathbf{x}}_{\mathbf{2_d}}) + \mathbf{g}(\mathbf{x})\mathbf{u} + \mathbf{d}(t) + \boldsymbol{\Lambda}\dot{\mathbf{e}}_{\mathbf{1}} + \mathbf{k_c}\boldsymbol{\sigma}_{\mathbf{0}}. \tag{2.33}$$

Equation (2.33) can be simplified by lumping all disturbances into a single term $\mathbf{h}(\mathbf{x}, \mathbf{u}, t)$. This gives

$$\dot{\boldsymbol{\sigma}}_{\mathbf{I}} = (\mathbf{f_0}(\mathbf{x}) - \dot{\mathbf{x}}_{\mathbf{2_d}}) + \mathbf{g_0}(\mathbf{x})\mathbf{u} \ + \boldsymbol{\Lambda}\dot{\mathbf{e}}_{\mathbf{1}} + \mathbf{k_c}\boldsymbol{\sigma}_{\mathbf{0}} + \mathbf{h}(\mathbf{x}, \mathbf{u}, t), \tag{2.34}$$

where $\mathbf{h}(\mathbf{x}, \mathbf{u}, t) = \Delta\mathbf{f}(\mathbf{x}) + \Delta\mathbf{g}(\mathbf{x})\mathbf{u} + \mathbf{d}(t)$. In order to assess the stability of the system, $\mathbf{u}$ can be split into the nominal and robust parts, given by

$$\mathbf{u} = -\mathbf{g_0}^{\dagger}(\mathbf{u_n} + \mathbf{u_r}), \tag{2.35}$$

where $\mathbf{u_n} \in \mathbb{R}^6$ is a control vector for the nominal system, $\mathbf{u_r} \in \mathbb{R}^6$ is the extra robustness term, and $\mathbf{g_0}^{\dagger} \in \mathbb{R}^{4\times6}$ is the generalized inverse of matrix $\mathbf{g_0}$. The nominal control vector $\mathbf{u_n}$ is designed such that

$$\mathbf{u_n} = (\mathbf{f_0}(\mathbf{x}) - \dot{\mathbf{x}}_{\mathbf{2_d}}) + \mathbf{\Lambda}\dot{\mathbf{e_1}} + \mathbf{k_c}\boldsymbol{\sigma_0}. \tag{2.36}$$

By substituting equations (2.35) and (2.36) into equation (2.34), we get the following equation:

$$\dot{\boldsymbol{\sigma}}_{\mathbf{I}} = -\mathbf{u_r} + \mathbf{h}(\mathbf{x}, \mathbf{u}, t). \tag{2.37}$$

Equation (2.37) can be split into its vector components. By equating this to (2.30) we get

$$\dot{\sigma_{I_i}} = -u_{r_i} + h_i(\mathbf{x}, \mathbf{u}, t) = -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i}. \tag{2.38}$$

However, using the subadditivity properties of the infinite norm, we can write

$$\begin{aligned} -\mathbf{u_r} + \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \leq -\mathbf{u_r} + \|\mathbf{h}(\mathbf{x}, \mathbf{u}, t)\|_\infty \leq -\mathbf{u_r} + \\ (\|\mathbf{\Delta f}(\mathbf{x})\|_\infty + \|\mathbf{\Delta g}(\mathbf{x})\mathbf{u}\|_\infty + \|\mathbf{d}(t)\|_\infty). \end{aligned} \tag{2.39}$$

Assuming that the infinity norm of the uncertainty and disturbance terms are known, a sufficient condition for robust stability of the closed-loop system can be achieved if u_{r_i} satisfies the inequality below

$$-u_{r_i} + (|f_{max}| + |g_{max}\bar{u}| + |d_{max}|) \leq -\bar{\mu}_i |\sigma_{I_i}|^{\zeta_i} \leq -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i}, \tag{2.40}$$

In (2.40), $\bar{u}$ is the norm bound of the control input vector $\mathbf{u}$ such that $\|\mathbf{u}\|_\infty \leq \bar{u}$. Here, we can lump the maximum perturbations into two new terms P_{pos} and P_{neg}, where $P_{pos} = f_{max} + g_{max}\bar{u} + d_{max}, P_{pos} \in \mathbb{R}^+$ and $P_{neg} = -(f_{max} + g_{max}\bar{u} + d_{max}), P_{neg} \in \mathbb{R}^-$. Therefore, the condition for robust stability of the closed-loop system (2.40) can be written for each case.

Case 1: $\sigma_{I_i} > 0$

$$-u_{r_i} + P_{pos} \leq -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i}, \tag{2.41}$$

Case 2: $\sigma_{I_i} < 0$

$$-u_{r_i} + P_{neg} \geq -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i}, \tag{2.42}$$

By choosing $u_{r_i} = K_i {\sigma_{I_i}}^{\zeta_i}$, as will be proved in Theorem 1, it is possible to show that the closed-loop system remains finite-time stable despite the uncertainties. Additionally, through the use of a fractional power in the robustness term, the negative effects of chattering should be alleviated due to the continuous nature of the robustness term about the sliding surface. Next, the robust stability of the control equation will be proven.

Theorem 1. *For the uncertain dynamic system defined in* (2.16)-(2.19) *and the control law defined by*

$$\mathbf{u} = \mathbf{g_0}^{\dagger}\left(\left((\mathbf{f_0(x)} - \dot{\mathbf{x}}_{\mathbf{2_d}}) + \mathbf{\Lambda \dot{e}_1} + \mathbf{k_c}\boldsymbol{\sigma}_{\mathbf{0}}\right) + \mathbf{u_r}\right), \tag{2.43}$$

the closed-loop system will be finite-time stable and the trajectories of the closed-loop system converge to the desired trajectory if the following sufficient condition is satisfied for each channel i and the control input u_{r_i} is selected such that

$$u_{r_i} = K_i {\sigma_{I_i}}^{\zeta_i}. \tag{2.44}$$

where, $K_i = \bar{\mu}_i + \bar{K}_i, \quad K_i, \bar{K}_i \in \mathbb{R}$, is a constant where $\bar{K}_i$ is a constant that is designed to dominate the maximum disturbances P_{pos} and P_{neg}.

Case 1: $\sigma_{I_i} > 0$

$$\bar{K}_i {\sigma_{I_i}}^{\zeta_i} \geq P_{pos}, \tag{2.45}$$

Case 2: $\sigma_{I_i} < 0$

$$\bar{K}_i {\sigma_{I_i}}^{\zeta_i} \leq P_{neg}, \tag{2.46}$$

proof. We start the proof by considering two cases for the sign of sliding surface.

Case 1: $\sigma_{I_i} > 0$

From equation (2.41) the condition for stability for each case is

$$-u_{r_i} \leq -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i} - P_{pos}. \tag{2.47}$$

Therefore,

$$u_{r_i} \geq \bar{\mu}_i {\sigma_{I_i}}^{\zeta_i} + P_{pos}. \tag{2.48}$$

Substituting equation (2.44) into (2.48) gives

$$K_i {\sigma_{I_i}}^{\zeta_i} \geq \bar{\mu}_i {\sigma_{I_i}}^{\zeta_i} + P_{pos}. \tag{2.49}$$

Case 2: $\sigma_{I_i} < 0$

From equation (2.42) the condition for stability for each case is

$$-u_{r_i} \geq -\bar{\mu}_i {\sigma_{I_i}}^{\zeta_i} - P_{neg}. \tag{2.50}$$

Therefore,

$$u_{r_i} \leq \bar{\mu}_i {\sigma_{I_i}}^{\zeta_i} + P_{neg}. \tag{2.51}$$

Substituting equation (2.44) into (2.51) gives

$$K_i {\sigma_{I_i}}^{\zeta_i} \leq \bar{\mu}_i {\sigma_{I_i}}^{\zeta_i} + P_{neg}. \tag{2.52}$$

Expanding the terms in equations (2.49) and (2.52) and cancelling gives equations (2.45) and (2.46). Therefore, the system is stable outside of the following bounds

$${\sigma_{I_i}}^{\zeta_i} \geq \frac{P_{pos}}{\bar{K}_i}, \ \forall \ \sigma_{I_i} > 0 \tag{2.53}$$

$${\sigma_{I_i}}^{\zeta_i} \leq \frac{P_{neg}}{\bar{K}_i}, \ \forall \ \sigma_{I_i} < 0 \tag{2.54}$$

In the context of this proof, the region defined by the conditions in equations (2.53) and (2.54) represents the boundary layer of the system. The system's trajectories converge to this boundary layer but not necessarily to zero. The boundary layer serves as a cushion around the desired sliding surface, ensuring that

the closed-loop system remains stable and converges towards the boundary layer while mitigating chattering effects that may occur in the presence of uncertainties and disturbances.

In this way, the control law u_{r_i} dominates the maximum disturbance and uncertainties in (2.34), and thus the system is robustly stable. To prove the finite-time convergence of the system, from lemma 2, we choose l_i as $\eta_i {\sigma_{I_i}}^{\gamma_i}$, defined in (2.27), and substitute this into equation (2.22) to get the following inequality

$$\left(\sum_{i=1}^{6} |\eta_i {\sigma_i}^{\gamma_i}| \right)^a \leq \sum_{i=1}^{6} |\eta_i {\sigma_i}^{\gamma_i}|^a. \tag{2.55}$$

Let $a = r/s < 1$ and r and s are positive odd integer numbers and $\frac{ra_i}{sb_i} = \delta_i$. Equation (2.55) becomes

$$\left(\sum_{i=1}^{6} |\eta_i {\sigma_i}^{\gamma_i}| \right)^a \leq \sum_{i=1}^{6} |\eta_i|^a |\sigma_i|^{\delta_i}. \tag{2.56}$$

Taking $|\eta_i|^a$ as μ_i and multiplying both sides of (2.57) by minus one gives

$$-\sum_{i=1}^{6} \mu_i |\sigma_i|^{\delta_i} \leq -\left(\sum_{i=1}^{6} \eta_i |\sigma_i|^{\gamma_i} \right)^a. \tag{2.57}$$

Substituting equations (2.25), (2.26) and (2.27) into (2.57) gives

$$\dot{V} \leq -V^a. \tag{2.58}$$

Thus, according to lemma 1, it can be seen that the closed-loop system is finite-time stable. The settling time of the system can be calculated as

$$T(\mathbf{x}(0)) \leq \frac{V(\mathbf{x_0})^{1-a}}{(1-a)}, \tag{2.59}$$

where $V(\mathbf{x_0})$ is the initial condition of V in equation (2.25).

2.3.3 Translational Control

The equations of translation motion in the x and y directions in equations (2.10) and (2.11) can be simplified with the introduction of two further control gains u_x and u_y to give the equations

$$\ddot{x} = u_x \frac{u_1}{m} - \frac{k_{dx}}{m} \dot{x}, \tag{2.60}$$

$$\ddot{y} = u_y \frac{u_1}{m} - \frac{k_{dy}}{m}\dot{y}, \tag{2.61}$$

where $u_x = (-c\varphi c\psi s\theta - s\varphi s\psi)$ and $u_y = (c\psi s\varphi - c\varphi s\theta s\psi)$. Using equations (2.60) and (2.61) and substituting them into equation (2.43) the outer-loop control laws for the x and y positions of the quadrotor can be derived.

$$u_x = \frac{m}{u_1}\left(\left(-K_x\sigma_{I_x}{}^{(\zeta_x)}\right) - \lambda_x(\dot{x}-\dot{x}_d) - k_{c_x}(\dot{x}-\dot{x}_d+\lambda_x(x-x_d)) + \frac{k_{dx}}{m}\dot{x}+\ddot{x}_d\right), \tag{2.62}$$

$$u_y = \frac{m}{u_1}\left(\left(-K_y\sigma_{I_y}{}^{(\zeta_y)}\right) - \lambda_y(\dot{y}-\dot{y}_d) - k_{c_y}(\dot{y}-\dot{y}_d+\lambda_y(y-y_d)) + \frac{k_{dy}}{m}\dot{y}+\ddot{y}_d\right). \tag{2.63}$$

Following this, using equations (2.64) and (2.65), the desired roll φ_d and pitch θ_d values are determined as

$$\varphi_d = \arcsin(-u_x\sin(\psi_d) + u_y\cos(\psi_d)), \tag{2.64}$$

$$\theta_d = \arcsin\left(\frac{u_x\cos(\psi_d) + u_y\sin(\psi_d)}{\cos(\varphi_d)}\right). \tag{2.65}$$

Values φ_d and θ_d are passed to the inner-loop attitude controller to allow for full trajectory tracking control of the quadrotor.

In order to control the z position of the quadrotor, equation (2.43) is expanded and substituted into the altitude dynamics of the quadrotor system in equation (2.12). This gives the control equation for u_1.

$$u_1 = \frac{m}{c\varphi c\theta}\left(\left(K_z\sigma_{I_z}{}^{(\zeta_z)}\right) + \lambda_z(\dot{z}-\dot{z}_d) + k_{c_z}(\dot{z}-\dot{z}_d+\lambda_z(z-z_d)) + g - \frac{k_{dz}}{m}\dot{z}-\ddot{z}_d\right). \tag{2.66}$$

2.3.4 Attitude Control

The attitude control of the quadrotor uses the dynamic system equations (2.13)-(2.15) and equation (2.43) to derive equations for u_2, u_3 and u_4

$$u_2 = I_{xx}\left(\left(-K_\varphi\sigma_{I_\varphi}{}^{(\zeta_\varphi)}\right) - \lambda_\varphi(\dot{\varphi}-\dot{\varphi}_d) + \ddot{\varphi}_d - k_{c_\varphi}(\dot{\varphi}-\dot{\varphi}_d+\lambda_\varphi(\varphi-\varphi_d)) - \left(\frac{(I_{yy}-I_{zz})\dot{\theta}\dot{\psi}}{I_{xx}}\right) + \frac{J_r\dot{\theta}}{I_{xx}}(\omega_1-\omega_2+\omega_3-\omega_4)\right), \tag{2.67}$$

$$u_3 = I_{yy}\Bigg(\Bigg(- K_\theta {\sigma_{I_\theta}}^{(\zeta_\theta)}\Bigg) - \lambda_\theta(\dot{\theta} - \dot{\theta}_d) + \ddot{\theta}_d - k_{c_\theta}(\dot{\theta} - \dot{\theta}_d + \lambda_\theta(\theta - \theta_d)) \\ -\Bigg(\frac{(I_{zz} - I_{xx})\dot{\varphi}\dot{\psi}}{I_{yy}}\Bigg) - \frac{J_r\dot{\varphi}}{I_{yy}}(\omega_1 - \omega_2 + \omega_3 - \omega_4)\Bigg), \quad (2.68)$$

$$u_4 = I_{zz}\Bigg(\Bigg(- K_\psi {\sigma_{I_\psi}}^{(\zeta_\psi)}\Bigg) - \lambda_\psi(\dot{\psi} - \dot{\psi}_d) + \ddot{\psi}_d - k_{c_\psi}(\dot{\psi} - \dot{\psi}_d + \lambda_\psi(\psi - \psi_d)) \\ -\Bigg(\frac{(I_{xx} - I_{yy})\dot{\varphi}\dot{\theta}}{I_{zz}}\Bigg)\Bigg). \quad (2.69)$$

Using equations (2.60)-(2.69), and implementing them as shown in Figure 2.2 each channel of the quadrotor system can be controlled, and full robust trajectory tracking control can be achieved for a quadrotor system.

2.4 Simulation Results

The FTI-SMC control system developed in Section 2.3 was simulated in MATLAB and Simulink using the "Parrot Minidrones Support from Simulink" toolbox. To evaluate the performance of the controller, it was compared against a modern finite-time sliding mode controller [118]. The FTI-SMC controller was also compared against a modified PID controller provided by the minidrone toolbox, as well as a classical sliding mode controller. The robustness of each controller was evaluated by introducing parametric uncertainties in the mass and inertia of the quadrotor. This section presents and compares the results from all three control systems.

The quadrotor parameters in equations (2.10)-(2.15) were set to match those of a Parrot Mambo Minidrone. Due to the small size of the Parrot Mambo Minidrone, the drag terms in the x, y, and z directions were considered to be negligible. These parameters are displayed in Table 2.1 and were used to collect the simulation results.

Table 2.1: Simulation parameters.

Variable	**Value**
m	$0.0630\ kg$
I_{xx}	$5.8286 \times 10^{-5}\ \ kg.m^2$
I_{yy}	$7.1691 \times 10^{-5}\ \ kg.m^2$
I_{zz}	$1.0000{\times}10^{-4}$ kg.m^2
J_r	$1.0209{\times}10^{-7}$ kg.m^2
l	$4.4100{\times}10^{-2}$ m
K_T	$4.7200{\times}10^{-8}$
K_D	$1.1393{\times}10^{-10}$
K_{dx}	0
K_{dy}	0
K_{dz}	0

Table 2.2: Control parameters used in simulation for (a) PID, (b) SMC, (c) FTSMC[118], and (d) FTI-SMC.

(a)

Kp_x	0.005
Kp_y	0.005
Kp_z	0.6
Kd_x	0.1
Kd_y	0.1
Kd_z	0.3
Ki_x	0
Ki_y	0
Ki_z	0
Kp_φ	0.013
Kp_θ	0.01
Kp_ψ	0.004
Kd_φ	0.02
Kd_θ	0.028
Kd_ψ	0.012
Ki_φ	0.01
Ki_θ	0.01
Ki_ψ	0

(b)

μ_x	0.4
μ_y	0.4
μ_z	7
λ_x	1
λ_y	1
λ_z	10
μ_φ	18
μ_θ	18
μ_ψ	8
λ_φ	12
λ_θ	12
λ_ψ	9

(c)

$\alpha_{1,x}$	0.8	$\alpha_{1,\varphi}$	15
$\alpha_{1,y}$	0.8	$\alpha_{1,\theta}$	15
$\alpha_{1,z}$	11	$\alpha_{1,\psi}$	10
$\alpha_{2,x}$	0.2	$\alpha_{2,\varphi}$	10
$\alpha_{2,y}$	0.2	$\alpha_{2,\theta}$	10
$\alpha_{2,z}$	5	$\alpha_{2,\psi}$	0.01
ρ_x	0.1	ρ_φ	1
ρ_y	0.1	ρ_θ	1
ρ_z	0.5	ρ_ψ	1
Θ_x	0.5	Θ_φ	5
Θ_y	0.5	Θ_θ	5
Θ_z	5	Θ_ψ	5
v_x	0.5	v_φ	0.5
v_y	0.5	v_θ	0.5
v_z	0.5	v_ψ	0.5
Φ_x	7/9	Φ_φ	11/17
Φ_y	7/9	Φ_θ	11/17
Φ_z	7/11	Φ_ψ	11/17
γ_x	7/9	γ_φ	11/13
γ_y	7/9	γ_θ	11/13
γ_z	7/13	γ_ψ	11/13

(d)

μ_x	0.4	μ_φ	10
μ_y	0.4	μ_θ	10
μ_z	10	μ_ψ	10
λ_x	1	λ_φ	10
λ_y	1	λ_θ	10
λ_z	1.5	λ_ψ	10
η_x	1	η_φ	1
η_y	1	η_θ	1
η_z	1	η_ψ	1
ζ_x	27/31	ζ_φ	31/33
ζ_y	27/31	ζ_θ	31/33
ζ_z	21/25	ζ_ψ	31/33
k_{cx}	0.05	$k_{c\varphi}$	0
k_{cy}	0.05	$k_{c\theta}$	0
k_{cz}	2.5	$k_{c\psi}$	0

2.4.1 Numerical Analysis of the Nominal Performance

To assess the performance of the nominal controllers, the system was first simulated with no disturbance applied to the mass or inertial parameters of the quadrotor. In order to assess the trajectory tracking capability of the quadrotor system, it was tasked to follow a circular trajectory in 3D space at the z position of -1. The integral of absolute error (IAE) was measured and the results are displayed in Table 2.3. The total IAE value is based on the euclidean distance between the desired

trajectory and the actual position of the drone. The total IAE is calculated using the equation

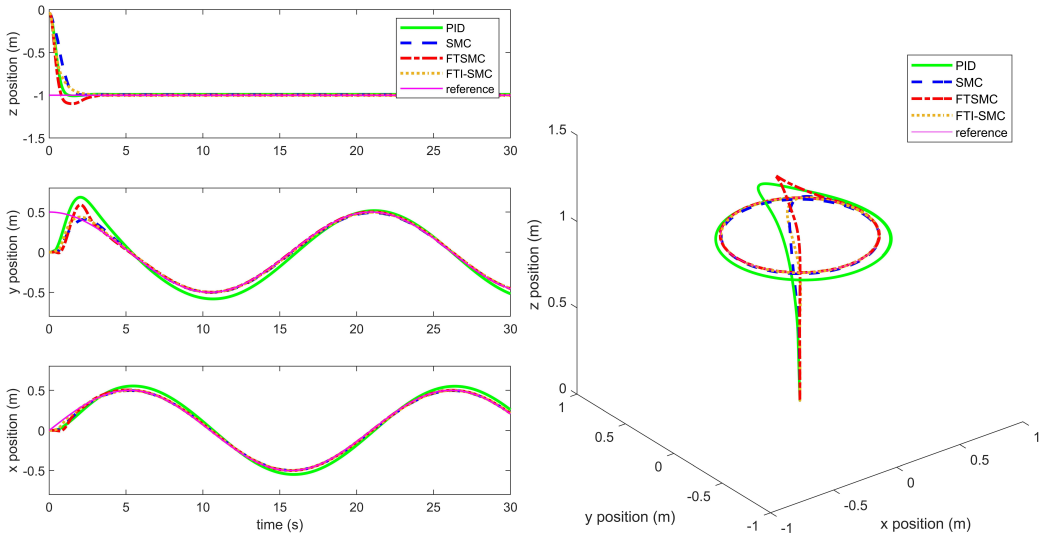
$$\text{Total IAE} = \sqrt{{IAE_x}^2 + {IAE_y}^2 + {IAE_z}^2}. \tag{2.70}$$

Table 2.3: IAE for quadrotor positions with no disturbance.

Integral of Absolute Error	**Control Systems**			
	PID	***SMC***	***FTSMC***	***FTI-SMC***
X Position (m)	1.2493	0.2767	0.2656	0.1835
Y Position (m)	2.0778	0.8736	0.8790	0.7631
Z Position (m)	0.7647	0.8697	0.5256	0.6046
Total (m)	2.5422	1.2634	1.0581	0.9908

From the simulation results of the nominal system displayed in Table 2.3, it can be seen that FTI-SMC displays a superior response in the x and y positions of the quadrotor when compared to FTSMC, SMC, and PID. As expected, all sliding mode control systems display more accurate performance for trajectory tracking of a quadrotor when compared to PID.

Figure 2.3 displays the position response of each control system when tasked with following the desired circular trajectory.



(a) (b)

Figure 2.3: Transient response of three control techniques following a circular trajectory, with (a) displaying the x, y, and z response of the control systems, and (b) displaying the three-dimensional trajectory of each control system.

From Figure 2.3, it can be seen that PID and FTSMC display an overshoot in the y position of the quadrotor, with PID displaying the largest overshoot. All three sliding mode controllers have good tracking, closely following the desired trajectory once they have settled. FTSMC displays oscillatory behaviour while settling to the desired trajectory on the y axis despite rigorous tuning of the control system. The y axis demonstrates the worst overall performance for all control systems, which may be due to the non-zero value of the desired y_d position of the quadrotor. From Table 2.3 and Figure 2.3 it can be seen that FTI-SMC displays the best overall tracking, most closely following the desired trajectory.

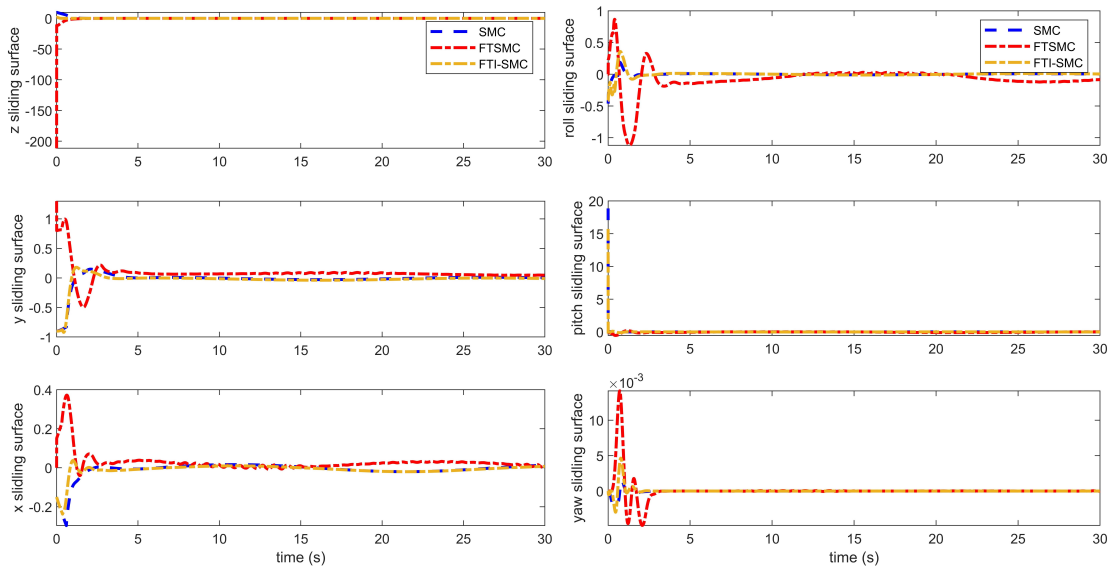
In order to assess the transient response of the control systems, FTI-SMC is also compared to SMC and FTSMC by observing their sliding surfaces and control inputs. The sliding surfaces for the position and attitude subsystems are shown in figure 2.4. The control inputs for u_1, u_2, u_3 and u_4 are displayed in Figure 2.5.

When observing the sliding surface for SMC, FTSMC, and FTI-SMC, it can be seen that both SMC and FTI-SMC control systems track a value of zero for the

sliding surface for the attitude subsystem, while FTSMC shows weaker tracking of a zero sliding surface value for the roll sliding surface. For the pitch and altitude subsystems, a large initial spike is present in the sliding surfaces, which is quickly corrected. This is due to the large difference between the initial values of y and z, and the non-zero initial values of y_d and z_d. When observing the position subsystem, oscillations are present in the sliding surface of FTSMC.

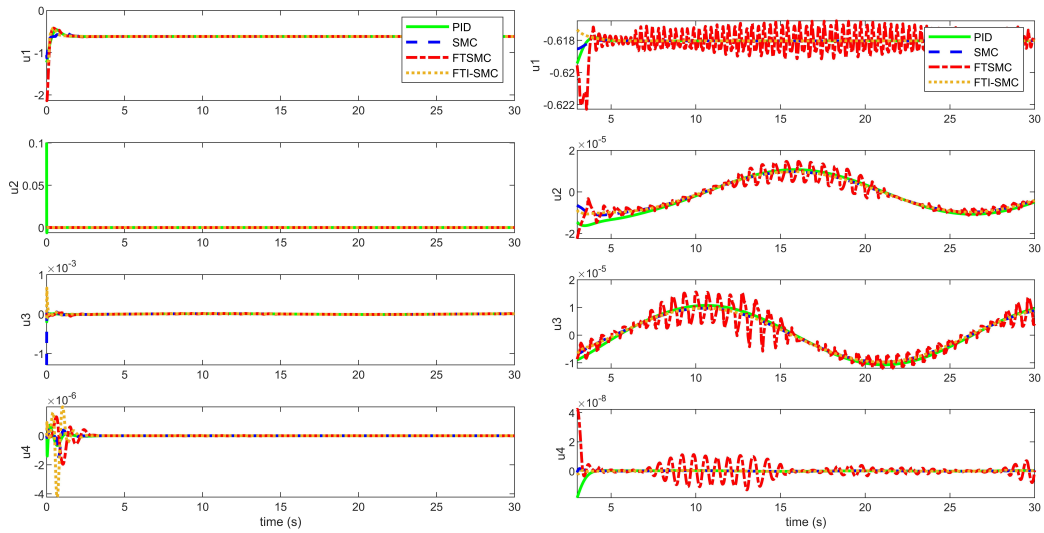
When observing the control inputs for each control system in Figure 2.5, it can be seen that the u_1, u_2 and u_3 control inputs display discontinuity with large oscillations when using FTSMC, with no such discontinuity present in classical SMC or FTI-SMC. Control input u_4 displays similar performance between both PID, SMC, and FTI-SMC, however, FTSMC displays oscillations once again. The discontinuities in the control inputs of FTSMC may be present due to the implementation of a switching function in the control law that contains the *signum* function. To help alleviate the chattering phenomenon when implementing classical SMC, *tanh* was used as an approximation for the *signum* function. While implementing FTI-SMC, the use of an odd fractional power allows the removal of the *signum* function in the control equations. Neither classical SMC nor FTI-SMC displays any discontinuity in the sliding surfaces.

These results clearly show the advantage of FTI-SMC when compared to the other control systems. FTI-SMC displays the best accuracy between the controllers while also guaranteeing convergence in finite time without the introduction of chattering, which is present in FTSMC.



(a) (b)

Figure 2.4: Sliding surfaces for the nominal system, with (a) displaying the sliding x, y, and z sliding surfaces, and (b) displaying the roll, pitch and yaw sliding surfaces.



(a) (b)

Figure 2.5: Control inputs for the nominal system. (a) displays all inputs from $t = 0s$ to $t = 30s$. (b) displays the transient control inputs from $t = 3s$ to $t = 30s$.

Table 2.4: Control efforts for each control system for the nominal system.

Control Effort	**Control Systems**			
	PID	***SMC***	***FTSMC***	***FTI-SMC***
u1	47.9907	47.9169	48.6062	48.0110
u2	0.1010	0.0012	0.0013	0.0013
u3	7.3499×10^{-4}	0.0014	7.4370×10^{-4}	0.0097
u4	7.5136×10^{-6}	8.9649×10^{-6}	1.7683×10^{-5}	2.9381×10^{-5}
Total	48.0925	47.9196	48.6083	48.0220

2.4.2 Numerical Analysis of the Robust Performance

The control systems were compared in the second scenario, where parametric uncertainty and sensor noise were introduced. An increase was applied to the mass m with $\Delta m = 0.0315kg$, and an increase was applied to the I_{xx} and I_{yy} values where $\Delta I_{xx} = 0.0874 \times 10^{-3} kg.m^2$ and $\Delta I_{yy} = 0.1075 \times 10^{-3} kg.m^2$. The performance of each controller was assessed while attempting to track the same desired trajectory as in the nominal case.

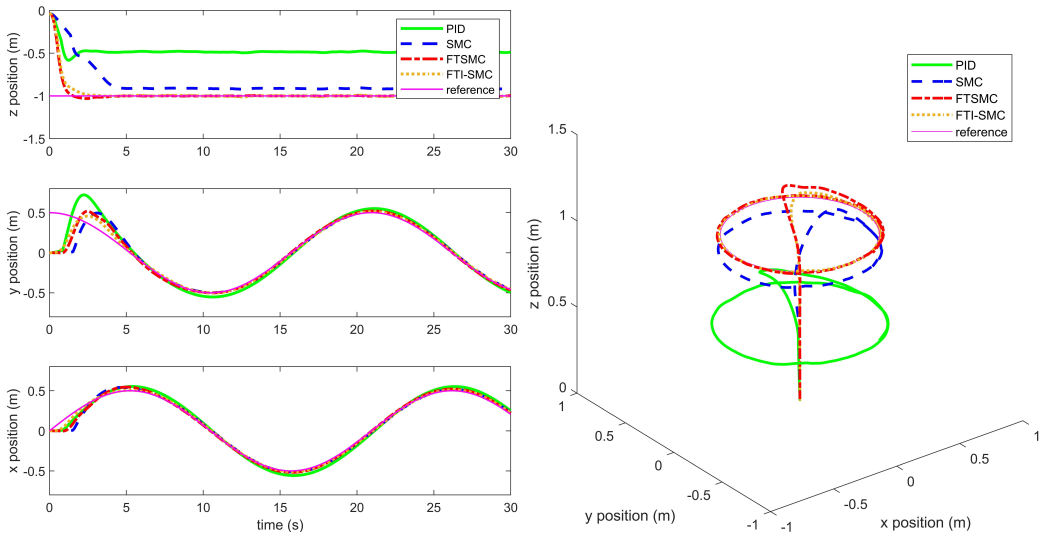
Table 2.5: IAE for quadrotor positions with parametric disturbances and sensor noise.

Integral of Absolute Error	**Control Systems**			
	PID	***SMC***	***FTSMC***	***FTI-SMC***
X Position (m)	1.2556	0.7264	0.6439	0.3723
Y Position (m)	2.1535	1.6580	1.3938	1.1020
Z Position (m)	15.6592	4.3996	0.6832	0.8216
Total IAE (m)	15.8564	4.7574	1.6805	1.4241

Here, the advantage of both finite time sliding mode controllers becomes

apparent, as the IAE for both has very little increase in the x and y positions of the quadrotor, while the performance of PID and classical SMC degrades. Furthermore, PID and SMC develop steady-state error in altitude control of the quadrotor, while FTSMC and FTI-SMC remain stable with no steady-state error. FTI-SMC shows the least increase in total IAE, as well as the best overall performance in the presence of uncertainties and sensor noise. Results in Figure 2.6 display how the quadrotor handles these uncertainties while tracking the circular trajectory.

Here FTI-SMC once again shows its advantages, displaying improved overall robustness when compared to the other control systems, while maintaining guaranteed convergence in finite time, with no introduction of chattering.



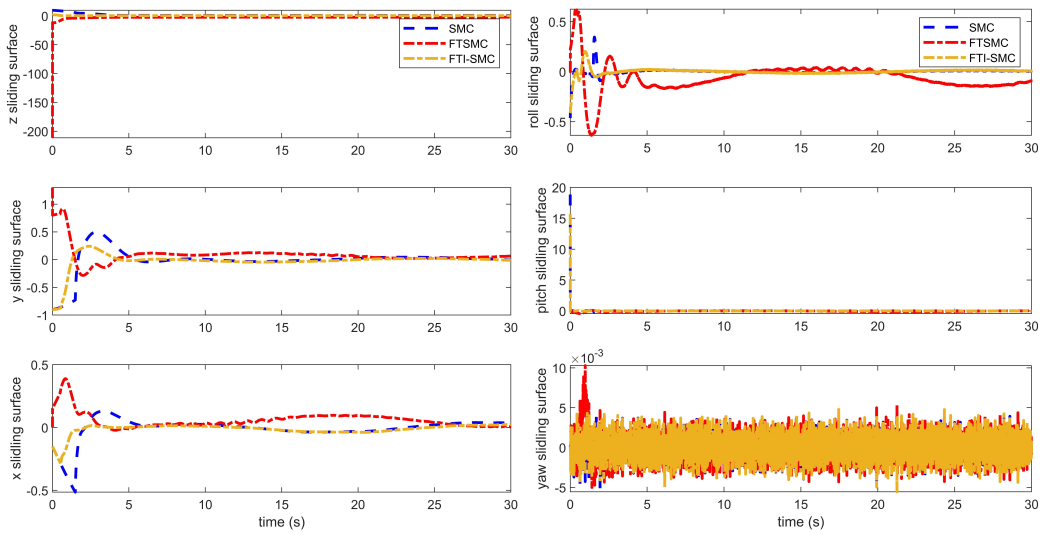
(a) (b)

Figure 2.6: Transient response of three control techniques following a circular trajectory with parametric disturbances and sensor noise, with (a) displaying the x, y, and z response of the control systems, and (b) displaying the three-dimensional trajectory of each control system.

It can be seen from Figure 2.6 that both PID and SMC lose tracking performance in the z position of the quadrotor, displaying steady state error. FTSMC and FTI-SMC remain capable of tracking the desired altitude. The tracking performance of

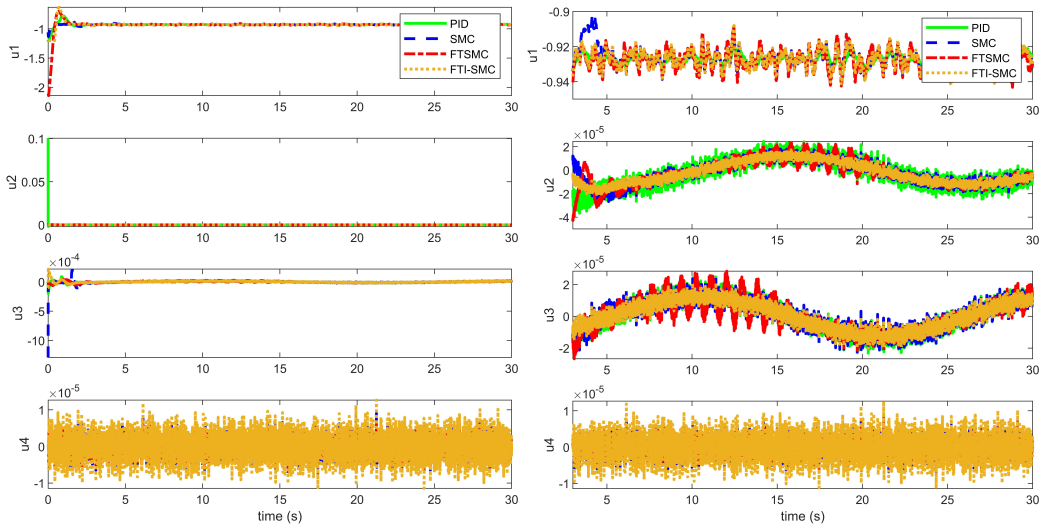
PID, SMC and FTSMC in the x and y axes degrades, with worse overall tracking, and a slower settling time than the nominal case. When observing the graphs there is very little change in the performance of FTI-SMC in the disturbed case when compared to the nominal case, displaying improved robust performance over FTSMC.

In order to assess the chattering response of the control systems in the presence of parametric uncertainties, once again FTI-SMC is compared to SMC and FTSMC by observing their sliding surfaces and control inputs. The sliding surfaces for the position and attitude subsystems are shown in Figure 2.7. The control inputs for u_1, u_2, u_3 and u_4 are displayed in Figure 2.8.



(a) (b)

Figure 2.7: Sliding surfaces with parametric disturbances and sensor noise, with (a) displaying the sliding x, y, and z sliding surfaces, and (b) displaying the roll, pitch and yaw sliding surfaces.



(a) (b)

Figure 2.8: Control inputs with parametric disturbances and sensor noise. (a) displays all inputs from $t = 0s$ to $t = 30s$. (b) displays the transient control inputs from $t = 3s$ to $t = 30s$.

Table 2.6: Control efforts for each control system with parametric uncertainties and sensor noise.

Control Effort	**Control Systems**			
	PID	***SMC***	***FTSMC***	***FTI-SMC***
u1	71.8612	71.8274	72.6131	71.8980
u2	0.1006	0.0023	0.0019	0.0014
u3	0.0011	0.0019	0.0011	0.0096
u4	1.2429×10^{-4}	1.7852×10^{-4}	1.3897×10^{-4}	2.6520×10^{-4}
Total	71.9630	71.8317	72.6162	71.9093

The sliding surfaces for FTI-SMC and SMC remain similar to the nominal case, with slightly larger deviations around a zero value. The chattering present

in FTSMC degrades further, with more severe oscillations present in the x, y and roll sliding surfaces, and worse tracking of a zero value. When observing the control inputs, the introduced sensor noise has a large impact on the smoothness of each controller, and chattering becomes less apparent. However, clear oscillations are still present in u_3 when implementing FTSMC.

2.5 Experimental Results

In order to further test the validity of the developed control methods, the FTI-SMC control system was implemented onto a Parrot Mambo Minidrone using the Parrot Simulink Support Package. The Minidrone is equipped with an IMU, an Optical Flow sensor, a sonar sensor, and a barometer. These sensors allow the quadrotor to estimate the various states of the system. PID was once again used as a baseline for comparison. To assess how finite-time convergence affects the quadrotor in a real scenario, it was also compared against classical SMC.

Experimental testing of the quadrotor proved to be more difficult due to the Parrot Mambo Minidrone sensor uncertainties. As the aim of this section is to provides an analysis of the control system itself, and not the onboard estimation system, the estimated states will be used as the ground truth values. Additionally, there are many unmodeled uncertainties in the system, such as motor dynamics, propeller dynamics, and power delivery inconsistencies. For these reasons, performance for all controllers degrades when compared to the simulation.

It should be noted that the results in this section will also be affected by the localisation performance of the Parrot Mambo Minidrone. By controlling states that may not be accurately updated by the on-board localisation system, the system can experience undesired behaviour. For example, latency in updating the measured states can introduce oscillatory or sluggish behaviour. Additionally, these potential delays and uncertainties are un-modelled in simulation. This can lead to large gaps between simulation and real-world performance. Future work can aim to address

this by repeating these experiments using more accurate ground truth methods such as external motion capture systems.

Each controller was evaluated using a sequence of three step inputs applied to the desired positions of the quadrotor. The following inputs were used as step inputs to evaluate the performance of each controller:

$$\begin{cases} t = 0, & (x_d, y_d, z_d) = (0, 0, -1.1). \\ t = 10, & (x_d, y_d, z_d) = (0.4, -1.1). \\ t = 20, & (x_d, y_d, z_d) = (0.4, 1, -1.1). \end{cases} \tag{2.71}$$

The results were recorded for the nominal system as well as in the presence of parametric uncertainties. It should be noted that as these are experimental results on a real quadrotor drone, sensor noise was present for both scenarios.

2.5.1 Experimental Results for the Nominal System

First, the experimental results were collected for the nominal system without any parametric disturbance. The position response, attitude response and a 3D plot of the quadrotor's trajectory are shown in Figure 2.9, 2.10 and Figure 2.11. Table 2.7 displays the IAE for the x, y and z positions of the quadrotor following this trajectory in experimentation.

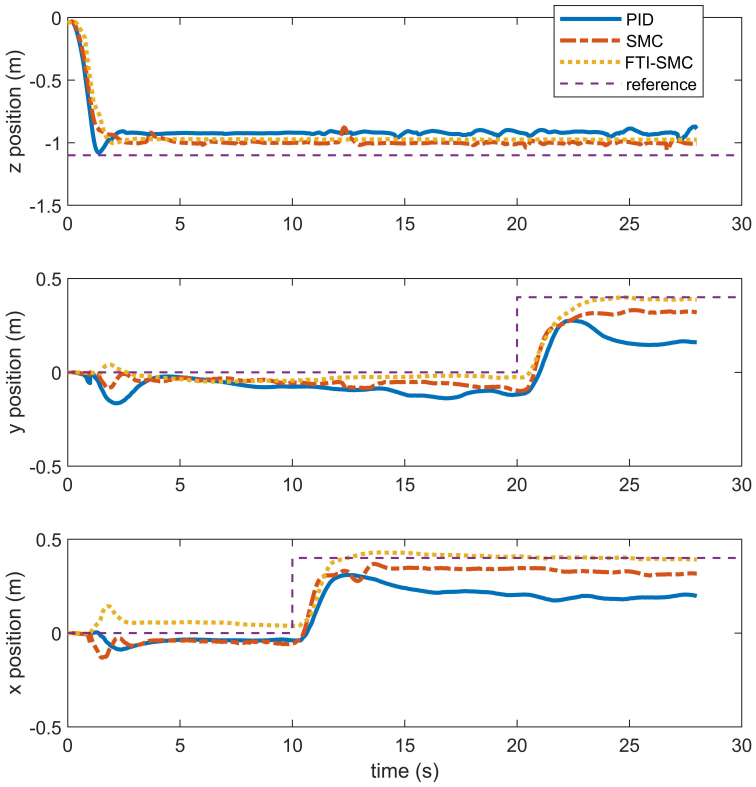


Figure 2.9: Position response of three control techniques in experimentation with a sequence of step inputs.

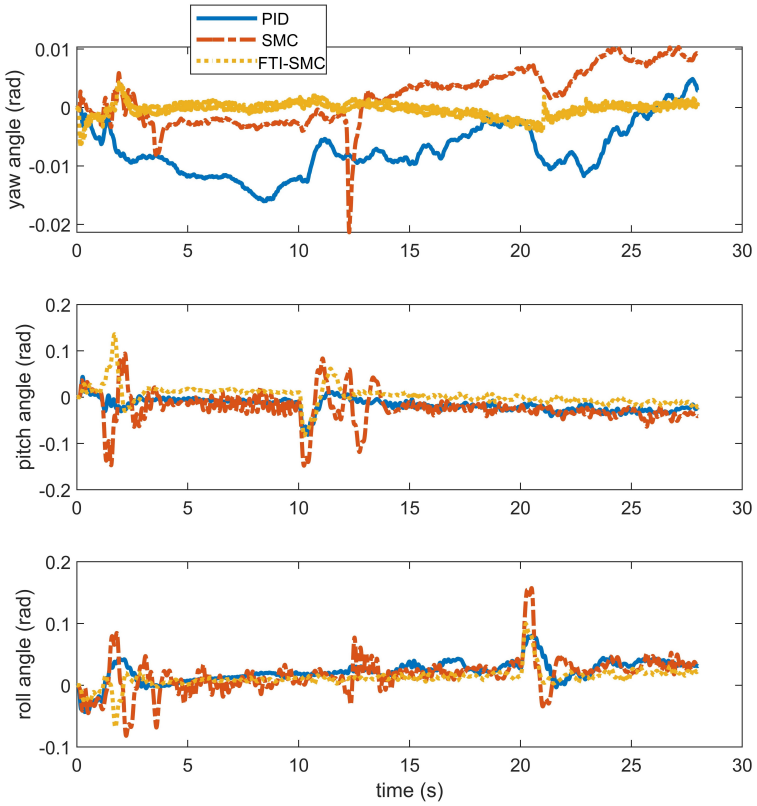


Figure 2.10: Attitude response of three control techniques in experimentation with a sequence of step inputs.

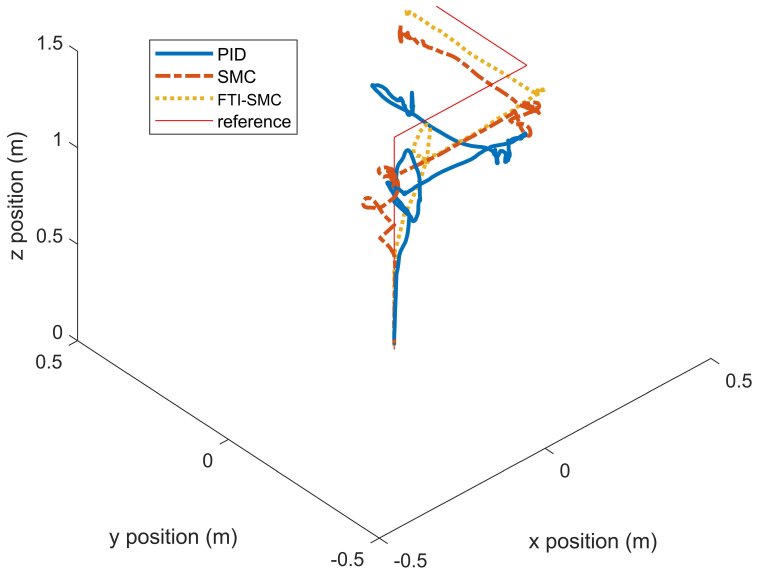


Figure 2.11: 3D position plot of three control techniques in experimentation with a sequence of step inputs.

Table 2.7: IAE for quadrotor positions in experimentation.

Integral of Absolute Error	**Control Systems**		
	PID	***SMC***	***FTI-SMC***
X Position (m)	3.9052	2.0150	1.1053
Y Position (m)	3.6566	2.1657	1.2623
Z Position (m)	5.5930	3.7717	4.5257
Total (m)	7.7397	4.7934	4.8267

From these results, we can see that in experimentation, both sliding mode controllers have improved performance over PID for the nominal system. FTI-SMC shows the best performance in control of the x and y positions of the quadrotor, with good tracking ability. Looking at the response of the attitude subsystem, we also see that FTI-SMC shows the best response for the control of yaw. When compared to SMC, FTI-SMC also displays the least oscillatory behaviour when observing the

roll and pitch response of the quadrotor. Finally, looking at the 3D plot in Figure 2.11, we can see that FTI-SMC displays the best overall trajectory tracking results in experimentation.

2.5.2 Experimental Results with Parametric Uncertainties

As the next step, the same inputs are applied, however, now a small mass was attached off-centre to the frame of the quadrotor. This mass offsets the balance of the Minidrone, introducing Δm, ΔI_{xx}, ΔI_{yy} and ΔI_{zz} as unknown parametric uncertainties. Results for the position and attitude response for each controller of the quadrotor are displayed in Figure 2.12 and Figure 2.13 for each controller. Figure 2.14 displays the 3D trajectory of each controller under this disturbance. Table 2.8 displays the IAE for the x, y and z positions of the quadrotor.

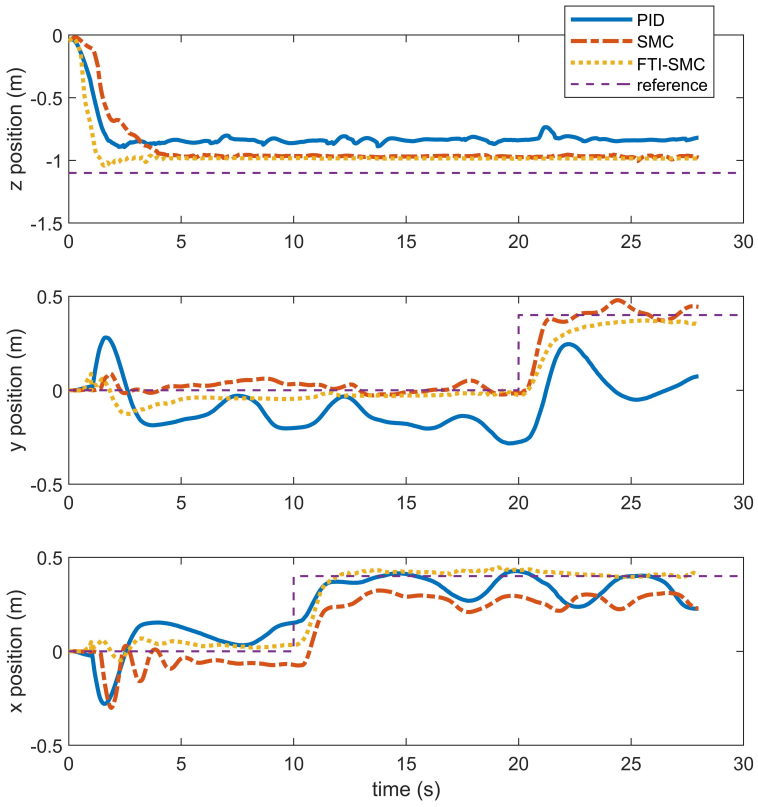


Figure 2.12: Position response of three control techniques in experimentation with a sequence of step inputs with parameter disturbance.

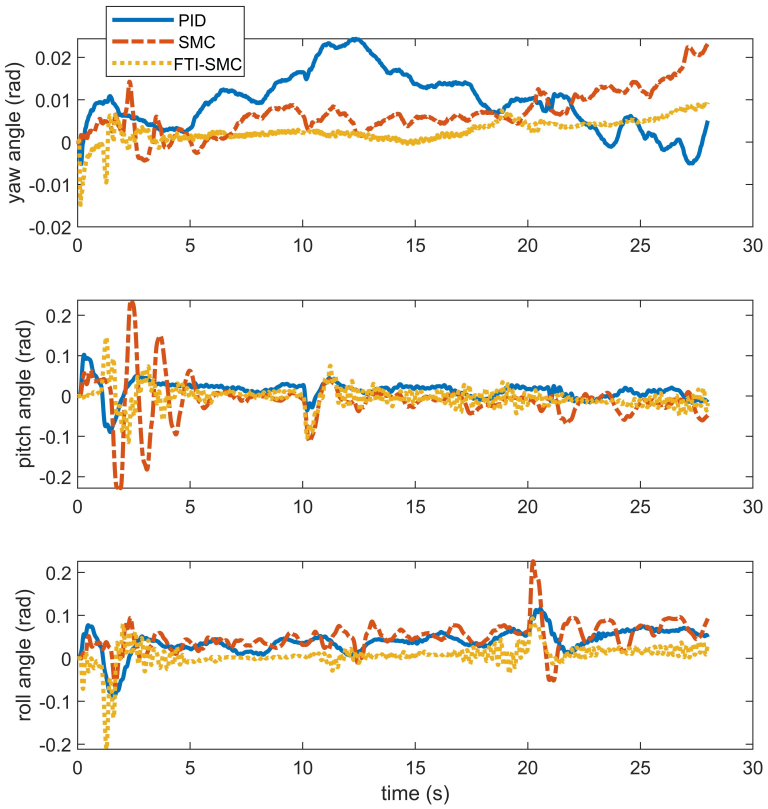


Figure 2.13: Attitude response of three control techniques in experimentation with a sequence of step inputs with parameter disturbance.

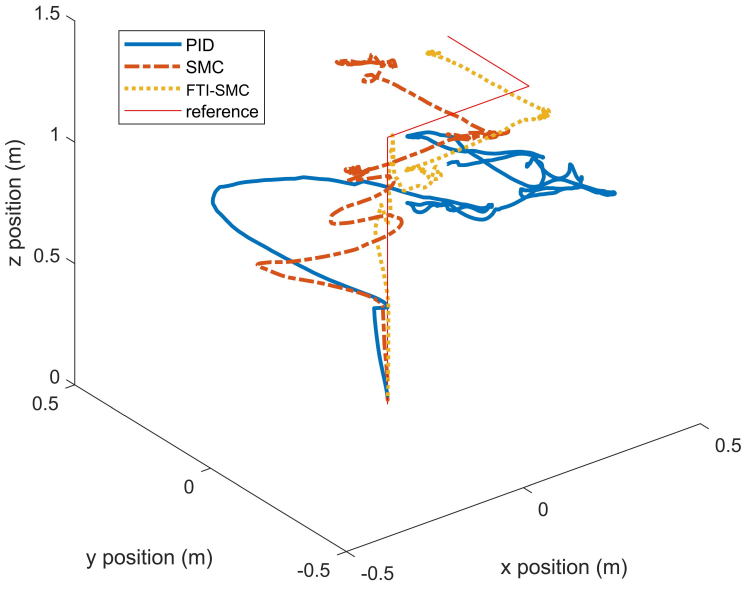


Figure 2.14: 3D position plot of three control techniques in experimentation with a sequence of step inputs with parameter disturbance.

Figures 2.12 and 2.14 display a clear problem with PID when parametric uncertainties are applied to the quadrotor. After take-off, the y position of the quadrotor experiences a large offset due to the off-centre mass attached to the quadrotor. This is likely due to the integral term of the inner-loop control system taking a substantial amount of time to reject the unknown disturbance, as the PID controller was tuned to display good performance in the nominal case. This behaviour is unsuitable for safety-critical industries such as the nuclear industry as there can often be unpredictable disturbances that cause the quadrotor to operate outside of its narrow operating range. Classical SMC also displays an offset after take-off but more quickly rejects the disturbance. FTI-SMC displays improved robust performance compared to both PID and classical SMC and is able to quickly reject the parametric disturbance.

Table 2.8: IAE for quadrotor positions in experimentation with a disturbance in parameters.

Integral of Absolute Error	**Control Systems**		
	PID	***SMC***	***FTI-SMC***
X Position (m)	2.1726	3.2935	0.9579
Y Position (m)	5.8951	1.0049	1.5846
Z Position (m)	8.1491	5.5025	2.9664
Total (m)	10.2898	6.4911	3.4969

When parametric uncertainties are applied, it becomes clear that PID has less robust performance when compared to SMC and FTI-SMC. Figure 2.12 and 2.14 display the unstable nature of PID in this scenario. SMC remains relatively stable, although large oscillations become present. FTI-SMC displays the best performance here, with stable transient response and good trajectory tracking capabilities. The attitude response of the system under parametric uncertainties in Figure 2.13 displays larger oscillations in the roll and pitch angles of the quadrotor. Overall, these results verify that FTI-SMC offers strong performance for applications in quadrotor control, with improved performance over PID and SMC.

2.6 Conclusion

This chapter has provided a novel approach for full trajectory tracking and position control of a quadrotor UAV in 3D space in the presence of nonlinear uncertainties and disturbances using sliding mode control. The controller is designed on a basis of integral sliding mode control to improve the robustness of the controller against disturbance and nonlinear uncertainties in the reaching phase. To alleviate the effect of chattering, a finite-time approach is proposed using an odd fractional power in conjunction with the integral sliding mode control. Further to numerical simulation,

the controller was implemented onto a Parrot Mambo Minidrone to evaluate the effectiveness of the controller against PID and SMC control in practice. Both simulation and experimental results validate the performance of this new controller, with FTI-SMC showing more accuracy and robustness over other controllers while avoiding chattering. This robust, accurate, and chattering-free qualities of the proposed control system make it an ideal controller for quadrotor control in safety-critical nuclear environments.

The FTI-SMC control system designed in this chapter is formulated in continuous time. While this is ideal for theoretical analysis, it is unrealistic, as real-world systems use sensors which have discrete sampling rates. Discrete sampling can reintroduce issues such as chattering that this chapter specifically aims to solve. Additionally, nuclear legacy sites are often GPS-denied and require localisation solutions that provide discrete measurements of the quadrotor's pose. To address these issues, the next chapter aims to design and investigate the use of a sliding mode control system developed in the discrete domain, and provide a comparison against continuous methods.

Chapter 3

Discrete-time Sliding Mode Control for Quadrotor Systems

3.1 Introduction

In the nuclear industry, many of the applications of UAV technology are aimed at indoor use. Therefore, implemented UAV systems must be able to operate without the use of a Global Positioning System (GPS). For this reason, other positioning methods using visual data must be implemented. One example of a system used is Simultaneous Localisation and Mapping (SLAM), in which different visual sensors, such as LiDAR and cameras, are used to position the quadrotor in space while mapping the environment [3], [103], [105]. Most notably, Hector SLAM uses a 2D LiDAR and on-board computational technology to position a quadrotor in space [106].

With the advent of cyber-physical systems and networked control technologies in the design of industrial autonomous systems in general and in the nuclear industry in particular, the use of sampled-data control methods has gained increasing popularity. In the field of robotic control for real-time implementation of such systems, implementing a fixed sampling time is necessary for periodic transmission of sensors' data and execution of the control signals through the actuators. For

example, in SLAM-based control systems, the publishing rate of the position estimate can vary greatly depending on the type of sensors. Intel Realsense depth cameras have the ability to operate between 6 Hz and 90 Hz, while various 2D LiDAR scanners used for SLAM can vary between 5 Hz and 50 Hz [123]–[125]. In these scenarios, when the update rate of the estimated position of a robot is slower, discretisation of the designed continuous-time SMC will not show the same performance as expected due to the long sampling period. Instead, implementing the controller in the discrete domain from the beginning should allow for improved robust stability and performance compared to the continuous-time design. A discrete-time multi-channel SMC is developed in another study for position and attitude control of a quadrotor for a time-invariant set-point [70]. Another study combines a discrete-time SMC with a disturbance observer. This control system provides excellent tracking and robustness in the presence of disturbances and uncertainties [126]. One alternative approach to discrete-time methods is an event-triggered approach, which is primarily used due to the resource-constrained nature of robotic applications [117], [127], [128].

This chapter builds on the previous chapter and related paper on the development of robust control algorithms for quadrotor UAVs in hazardous environments, completed during the project [1]. In this paper, a nested chattering-free sliding mode controller (CFSMC) was applied for position and attitude control of a quadrotor UAV for use in hazardous environments with parametric uncertainties. In the present chapter, a nested discrete-time sliding mode controller is designed and developed for a quadrotor UAV for complete trajectory tracking control in indoor hazardous environments. The results are compared with continuous-time control methods, including classical SMC as well as chattering-free SMC to highlight practical issues of implementing these controllers along with SLAM algorithms for low sampling rates in a closed-loop system. The results confirm the improved robustness of the proposed discrete-time method for both inner and outer-loop control with hector SLAM in the loop when compared with the other two methods.

This chapter presents work that has been completed during the PhD project. The work has been peer-reviewed and was published following presentation of the findings at a conference:

- A. Can, J. Price, and A. Montazeri, "A nonlinear discrete-time sliding mode controller for autonomous navigation of an aerial vehicle using hector slam," *IFAC-PapersOnLine*, vol. 55, pp. 2653–2658, Oct. 2022, DOI: 10.1016/j.ifacol.2022.10.110

This chapter slightly modifies some sections of the publication to fit the context of the overall thesis.

The remainder of this chapter is organised as follows: The quadrotor model is derived in Section 3.2, and the derivation of the proposed discrete-time sliding mode controller is presented in Section 3.3; Section 3.4 discusses details surrounding the implementation of Hector SLAM; and the proposed method compared with the previous ones is studied in Section 3.5; finally, Section 3.6 will conclude the chapter.

3.2 Quadrotor Model

The design of a control system for the quadrotor first requires the derivation of the mathematical model of the quadrotor. A simplified form of the continuous-time quadrotor model, first shown in equations (2.10)-(2.15), is given in equation (3.1) [1]. Variables $[x, y, z]$ represent the position of the quadrotor in the world frame, while $[\varphi, \theta, \psi]$ represent the roll, pitch and yaw angles of the quadrotor in the body frame. The mass of the quadrotor is denoted by m. The moments of inertia are represented by $[I_{xx}, I_{yy}, Izz]$, while control inputs are represented by $[u_1, u_2, u_3, u_4]$.

$$\begin{cases} \ddot{x} = (-c\varphi c\psi s\theta - s\varphi s\psi)\frac{u_1}{m}, \\ \ddot{y} = (-c\varphi s\theta s\psi - c\psi s\varphi)\frac{u_1}{m}, \\ \ddot{z} = g - c\varphi c\theta\frac{u_1}{m}, \\ \ddot{\varphi} = \frac{(I_{yy}-I_{zz})\times\dot{\theta}\ \times\dot{\psi}}{I_{xx}} + \frac{u_2}{I_{xx}}, \\ \ddot{\theta} = \frac{(I_{zz}-I_{xx})\times\dot{\varphi}\ \times\dot{\psi}}{I_{yy}} + \frac{u_3}{I_{yy}}, \\ \ddot{\psi} = \frac{(I_{xx}-I_{yy})\times\dot{\varphi}\ \times\dot{\theta}}{I_{zz}} + \frac{u_4}{I_{zz}}, \end{cases} \tag{3.1}$$

where $c\varphi = \cos(\varphi)$, $s\varphi = \sin(\varphi)$ and $t\varphi = \tan(\varphi)$.

In order to design a discrete-time control system, the continuous model of the quadrotor must be converted into the discrete domain. Though a number of methods for discretisation exist, this chapter will use the forward-Euler method shown in equation (3.2).

$$\dot{x}_k = \frac{x_{k+1} - x_k}{T}. \tag{3.2}$$

Variable T denotes the sample time of the discrete system. The discrete-time quadrotor model is therefore calculated using equation (3.2) to give the set of system equations in (3.3) and (3.4).

$$\begin{cases} x_{k+1} = x_k + T\dot{x}_k, \\ \dot{x}_{k+1} = \dot{x}_k + T(-c\varphi_k c\psi_k s\theta_k - s\varphi_k s\psi_k)\frac{u_{1,k}}{m}, \\ y_{k+1} = y_k + T\dot{y}_k, \\ \dot{y}_{k+1} = \dot{y}_k + T(-c\varphi_k s\theta_k s\psi_k - c\psi_k s\varphi_k)\frac{u_{1,k}}{m}, \\ z_{k+1} = z_k + T\dot{z}_k, \\ \dot{z}_{k+1} = \dot{z}_k + T\big(g - (c\varphi_k c\theta_k\frac{u_{1,k}}{m})\big). \end{cases} \tag{3.3}$$

$$\begin{cases} \varphi_{k+1} = \varphi_k + T\dot{\varphi}_k, \\ \dot{\varphi}_{k+1} = \dot{\varphi}_k + T(\frac{(I_{yy}-I_{zz})\dot{\theta}_k\dot{\psi}_k+u_{2,k}}{I_{xx}}), \\ \varphi_{k+1} = \varphi_k + T\dot{\varphi}_k, \\ \dot{\theta}_{k+1} = \dot{\theta}_k + T(\frac{(I_{yy}-I_{zz})\dot{\varphi}_k\dot{\psi}_k+u_{3,k}}{I_{xx}}), \\ \psi_{k+1} = \psi_k + T\dot{\psi}_k, \\ \dot{\psi}_{k+1} = \dot{\psi}_k + T(\frac{(I_{yy}-I_{zz})\dot{\varphi}_k\dot{\theta}_k+u_{4,k}}{I_{xx}}), \end{cases} \tag{3.4}$$

where x_k represents the variable x at time step k and x_{k+1} represents the variable x one time step in the future. Using this set of discrete-time equations, a discrete-time sliding mode controller can be developed [4].

3.3 Discrete-Time Sliding Mode Control

The task of control system design for quadrotor UAVs is challenging due to the nonlinear, under-actuated, and coupled quadrotor dynamics [4]. In this section, a discrete-time sliding mode controller (DTSMC) is designed.

Control systems used in modern robotics rely on sensor data for feedback of the state variables. However, these sensors provide data that is inherently continuous-time, but is discretised with a specific sampling period. The sensors therefore feed discrete-domain data to the control system. The natural progression for these control systems is, therefore, for them to be developed and implemented entirely in the discrete domain. The control system developed in this section uses a nested, multi-channel structure to allow for full position and attitude control of the quadrotor. An inner-loop attitude control subsystem controls the roll, pitch and yaw of the quadrotor. An outer-loop control system controls the position of the quadrotor by generating desired angles for the attitude controller based on the position error. A separate subsystem controls the altitude of the quadrotor.

3.3.1 Attitude and Altitude Subsystems

The attitude subsystem controls the roll, pitch and yaw of the quadrotor, while the altitude control system controls the height of the quadrotor. To derive these, first, the sliding surface for each channel σ_s is defined at time step k in the discrete domain.

$$\sigma_{s,k} = a_s(s_k^d - s_k) + (\dot{s}_d^k - \dot{s}_k) \tag{3.5}$$

where s_k and $\dot{s}_k$ represent any variable and the rate of change of that variable at time step k. s_k^d and $\dot{s}_k^d$ represent the desired value and rate of change of variables s and $\dot{s}_k$ at time step k. a_s is a tuning parameter. The sliding surface is then defined at time step $k+1$.

$$\sigma_{s,k+1} = a_s(s_{k+1}^d - s_{k+1}) - (\dot{s}_{k+1}^d - \dot{s}_{k+1}) \tag{3.6}$$

Next, a discrete-time reaching law proposed by [129] is implemented to force the system to slide along sliding surface σ.

$$\begin{aligned} \sigma_{s,k+1} - \sigma_{s,k} &= -\eta_s T\sigma_{s,k} - \epsilon_s T sgn(\sigma_{s,k}) \\ \eta_s > 0,\ \sigma_s > 0,\ 1 - \eta_s T > 0, \end{aligned} \tag{3.7}$$

where η_s and ϵ_s are tuning parameters. sgn is the signum function. By substituting equations (3.5) and (3.6) into (3.7), while using the discrete model in (3.3) and (3.4), the control laws for inputs u_1 to u_4 at time step k can be derived.

$$\begin{aligned} u_{1,k} = \left(\frac{m}{c\varphi_k c\theta_k T}\right)\big(&Tg + a_z z_{k+1}^d - a_z T\dot{z}_k + \dot{z}_{k+1}^d \\ &-a_z z_k^d - \dot{z}_k^d + (\eta_z \sigma_{z,k} T) + (\epsilon_z T sgn(\sigma_{z,k}))\big) \end{aligned} \tag{3.8}$$

$$\begin{aligned} u_{2,k} = \left(\frac{I_{xx}}{T}\right)\Bigg(&-T\left(\frac{I_{yy} - I_{zz}}{I_{xx}}\right)\dot{\theta}_k\dot{\psi}_k + a_\varphi\varphi_{k+1}^d \\ &-a_\varphi T\dot{\varphi}_k - \dot{\varphi}_{k+1}^d - a_\varphi\varphi_k^d - \dot{\varphi}_k^d + (\eta_\varphi\sigma_{\varphi,k}T) \\ &+(\epsilon_\varphi T sgn(\sigma_{\varphi,k}))\Bigg) \end{aligned} \tag{3.9}$$

$$u_{3,k} = \left(\frac{I_{yy}}{T}\right)\Bigg(-T\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right)\dot{\varphi}_k\dot{\psi}_k + a_\theta\theta^d_{k+1} - a_\theta T\dot{\theta}_k - \dot{\theta}^d_{k+1} - a_\theta\theta^d_k - \dot{\theta}^d_k + (\eta_\theta\sigma_{\theta,k}T) + (\epsilon_\theta T sgn(\sigma_{\theta,k}))\Bigg) \tag{3.10}$$

$$u_{4,k} = \left(\frac{I_{zz}}{T}\right)\Bigg(-T\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right)\dot{\varphi}_k\dot{\theta}_k + a_\psi\psi^d_{k+1} - a_\psi T\dot{\psi}_k - \dot{\psi}^d_{k+1} - a_\psi\psi^d_k - \dot{\psi}^d_k + (\eta_\psi\sigma_{\psi,k}T) + (\epsilon_\psi T sgn(\sigma_{\psi,k}))\Bigg) \tag{3.11}$$

From equation (3.8) it can be observed that the control law breaks down when $c\varphi_k c\theta_k = 0$. This can occur when either $|\varphi_k|$ or $|\theta_k|$ is equal to $\frac{\pi}{2}\ rad$. As $|\varphi_k|$ or $|\theta_k|$ approach $\frac{\pi}{2}\ rad$, the control law $u_{1,k}$ approaches ∞. For this reason, the remainder of this chapter assumes that the quadrotor is controlled within a reasonable angular range of $|\varphi_k| < \frac{\pi}{2}\ rad,\ \ |\vartheta_k| < \frac{\pi}{2}\ rad$.

3.3.2 Position Control Subsystem

The inner-loop subsystem allows control over the attitude and the altitude of the quadrotor. In order to control the position of the quadrotor to allow control over all six degrees of freedom, an outer-loop controller is developed that generates desired angles based on the position error.

In the discrete-time position quadrotor model in (3.3), two new controller gains, u_x and u_y, are substituted in.

$$u_{x,k} = s\varphi_k s\psi_k + c\varphi_k s\theta_k c\psi_k \tag{3.12}$$

$$u_{y,k} = -s\varphi_k c\psi_k + c\varphi_k s\theta_k s\psi_k \tag{3.13}$$

This gives two new equations for the discrete time quadrotor model presented in (3.3) .

$$\begin{cases} x_{k+1} = x_k + T\dot{x}_k \\ \dot{x}_{k+1} = \dot{x}_k + u_{x_k}\frac{u_{1,k}T}{m} \\ y_{k+1} = y_k + T\dot{y}_k \\ \dot{y}_{k+1} = \dot{y}_k + u_{y,k}\frac{u_{1,k}T}{m} \end{cases} \tag{3.14}$$

By substituting equations (3.5) and (3.6) into (3.7), while using the new model shown in (3.14), two new control laws for u_x and u_y can be derived.

$$\begin{aligned} u_{x,k} = \left(\frac{m}{T}\right)\big(&a_x x^d_{k+1} - a_x T\dot{x}_k + \dot{x}^d_{k+1} - a_x x^d_k \\ &-\dot{x}^d_k + (\eta_x \sigma_{x,k} T) + (\epsilon_x T sgn(\sigma_{x,k}))\big) \end{aligned} \tag{3.15}$$

$$\begin{aligned} u_{y,k} = \left(\frac{m}{T}\right)\big(&a_y y^d_{k+1} - a_y T\dot{y}_k + \dot{y}^d_{k+1} - a_y y^d_k \\ &-\dot{y}^d_k + (\eta_y \sigma_{y,k} T) + (\epsilon_y T sgn(\sigma_{y,k}))\big) \end{aligned} \tag{3.16}$$

Values $u_{x,k}$ and $u_{y,k}$ can then be converted into φ^d_k and θ^d_k using equations (3.17) and (3.18). This can then be fed into the attitude control subsystem to allow full control over the quadrotor's position and attitude in 3D space.

$$\varphi^d_k = \arcsin\left(u_{x,k}\sin(\psi_k) - u_{y,k}\cos(\psi_k)\right) \tag{3.17}$$

$$\theta^d_k = \arcsin\left(\frac{u_{x,k}\cos(\psi_k) + u_{y,k}\sin(\psi_k)}{\cos(\varphi_k)}\right) \tag{3.18}$$

The diagram in Figure 3.1 displays the control diagram for the DTSMC system.

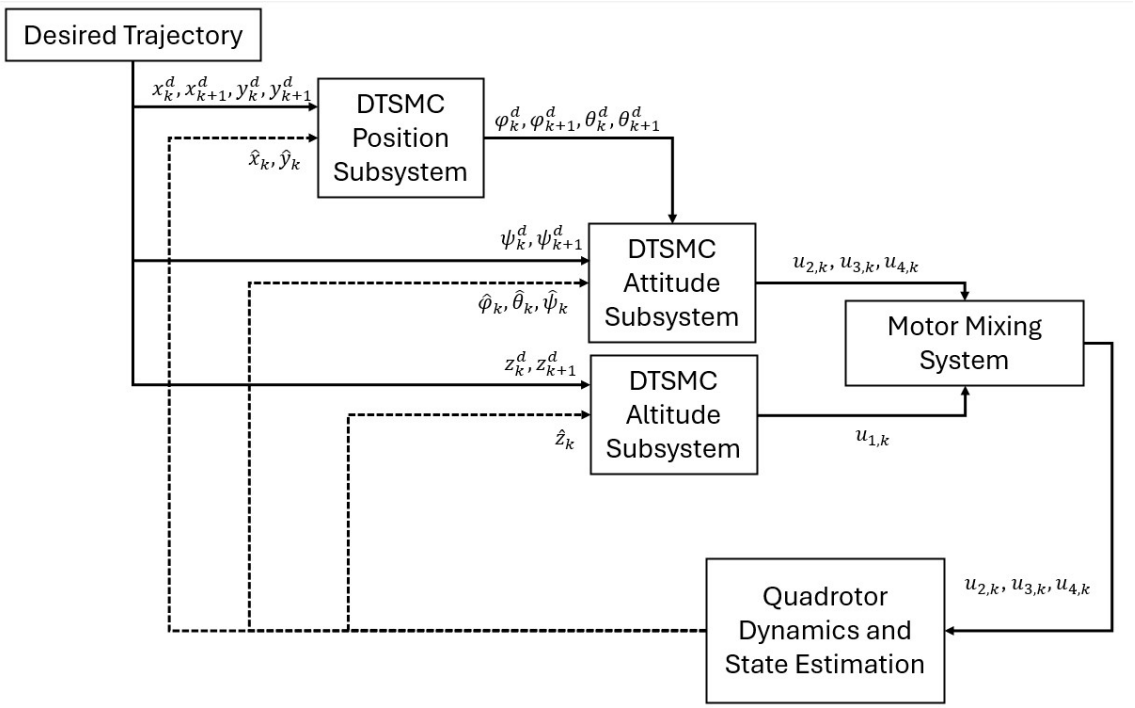


Figure 3.1: Discrete time sliding mode control diagram for 6DOF control of a quadrotor.

3.4 Hector SLAM

As this chapter does not contain experimental results, in an attempt to close the gap between real-world and simulation, a SLAM state estimation system is used inside of the Gazebo simulator. This allows the control systems to be tested with sensor noise, and uncertainties in state-estimation, to provide more rigorous evidence of the efficacy of the DTSMC system. Hector SLAM is a 2D SLAM method that relies on laser scan data from a LiDAR sensor mounted to a robot [106]. Hector SLAM is often used in aerial robotic applications, as it does not rely on odometry data from the turning of wheels for localisation. Instead, it depends on LiDAR data and scan-matching techniques to create a map and localise itself within that map. It achieves this by creating an occupancy grid map of its environment from LiDAR data. It

then finds the optimal robot pose that aligns the latest LiDAR scan with the stored map using Gauss-Newton optimisation. Each cell in the occupancy grid stores a probability value that the cell is occupied, allowing the map to be improved over several iterations. Other benefits of Hector SLAM are its 360° capabilities due to implementing LiDAR data over visual data, its increased computational efficiency, and its increased range compared to 3D visual methods. As it inherently runs on depth information, the map data can also be leveraged for obstacle avoidance and path planning algorithms.

For positioning and localisation of the quadrotor, a 2D LiDAR was mounted to the quadrotor frame in ROS and Hector SLAM was used. Rather than using ground truth values for control, SLAM provides estimated values for the quadrotor's position in space [106]. A scan-matching technique matches LiDAR points to a previously generated map. This is estimated using (3.19), where $\xi = (\hat{x}, \hat{y}, \hat{\psi})^T$, and $\Delta\xi$ is calculated once per revolution of the 2D LiDAR when the map is updated.

$$\xi_k = \xi_{k-1} + \Delta\xi \tag{3.19}$$

where $\Delta\xi$ is calculated by optimizing the error between the current LiDAR points and the generated map. The variables $\hat{x}$, $\hat{y}$, and $\hat{\psi}$ represent the estimated values of x, y, and ψ respectively. In the following simulation section, the estimated position values provided by hector SLAM are fed into each control system.

3.5 Simulation Results

3.5.1 Simulation Setup

Simulink and MATLAB were used to implement each control system, while the Robot Operating System was used to implement the Hector SLAM algorithm. Simulink was also used to read the quadrotor's estimated states and communicate control signals to ROS. A flow diagram demonstrating the communication between ROS and Simulink is shown in Figure 3.2.

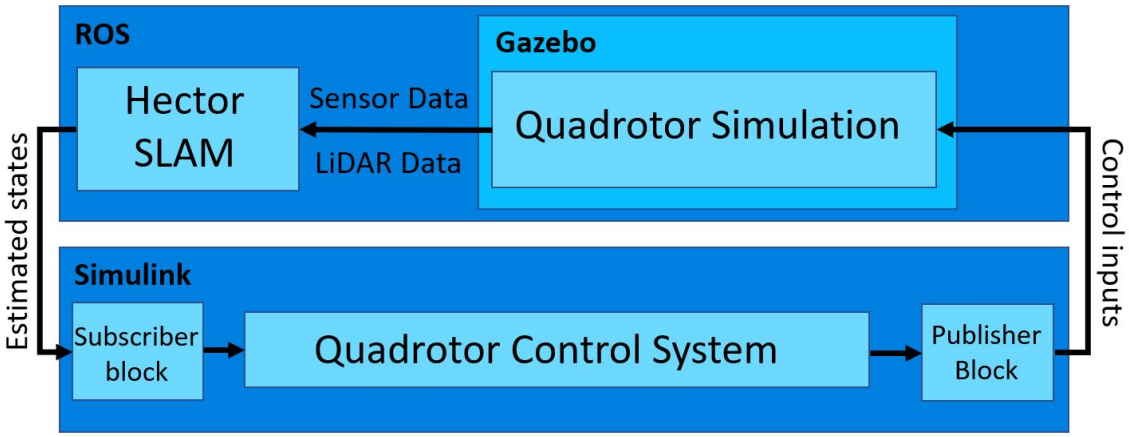


Figure 3.2: Flow chart showing communication between ROS and Simulink.

ROS was used to simulate the quadrotor system to ensure that the testing of the control systems was as accurate to life as possible. A quadrotor frame, equipped with various sensors, was simulated inside an indoor world environment. Figure 3.3 shows the simulation of the quadrotor, as well as the visualised LiDAR data and the map in the ROS software RViz.

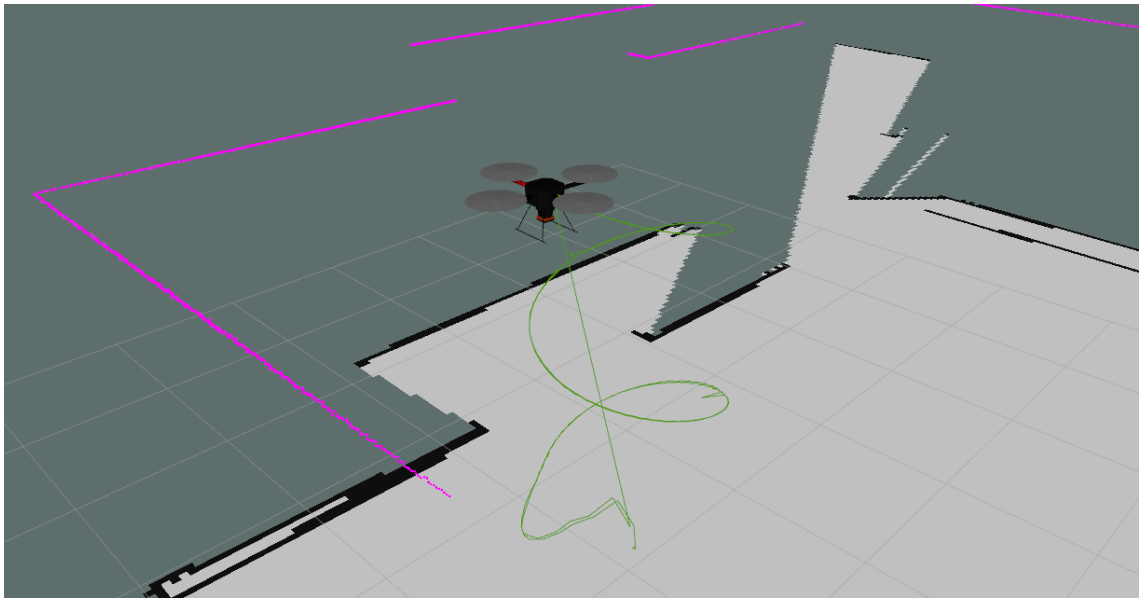


Figure 3.3: RViz software displaying quadrotor trajectory, laser scan data, and the developed map.

In order to evaluate the performance of the discrete-time controller, it will be compared against classical SMC as well as CFSMC in three scenarios. Firstly it will be tested with a smaller sampling period of $T = 0.01$ seconds, then a larger sampling time of $T = 0.05$ seconds, and finally a sampling time of $T = 0.1$ seconds. In each scenario, the quadrotor was tasked to track a helical trajectory, with $x^d = 0.5\sin(t)$, $y^d = 0.5\cos(t)$, and $z^d = 0.3 + \frac{t}{25}$, where t represents the current simulation time.

Table 3.1: Control parameters used in simulation for (a) SMC, (b) CFMSC, and (c) DTSMC control systems.

(a)

μ_x	0.7
μ_y	0.7
μ_z	16
λ_x	2.8
λ_y	2.8
λ_z	10
μ_φ	35
μ_θ	35
μ_ψ	2.8
λ_φ	1.6
λ_θ	1.6
λ_ψ	0.7

(b)

μ_x	0.9	μ_φ	3
μ_y	0.9	μ_θ	3
μ_z	10	μ_ψ	3
λ_x	1.2	λ_φ	3
λ_y	1.2	λ_θ	3
λ_z	20	λ_ψ	3
η_x	0.9	η_φ	0.8
η_y	0.9	η_θ	0.8
η_z	10	η_ψ	0.8
α_x/β_x	11/15	$\alpha_\varphi/\beta_\varphi$	15/17
α_y/β_y	11/15	$\alpha_\theta/\beta_\theta$	15/17
α_z/β_z	13/17	α_ψ/β_ψ	15/17

(c)

α_x	1.2	α_φ	5
α_y	1.2	α_θ	5
α_z	6	α_ψ	5
η_x	1.5	η_φ	8
η_y	1.5	η_θ	8
η_z	15	η_ψ	8
ϵ_x	0.01	ϵ_φ	0.01
ϵ_y	0.01	ϵ_θ	0.01
ϵ_z	0.01	ϵ_ψ	0.01

3.5.2 Numerical Results

The sampling rate of the sensors within ROS, the control system in Simulink, and the rate at which commands were published from Simulink to ROS were each set to the desired sampling rate in order to test each control system[4]. The integral of absolute error (IAE) was measured to evaluate and compare the performance of each controller. Figures 3.4 to 3.7 demonstrate the performance of the quadrotor following the trajectory, with each control system implemented at three sample times, from $T = 0.01$ to $T = 0.05$ seconds respectively. Table 3.2 displays and compares the integral of absolute error for each control system at each sample time.

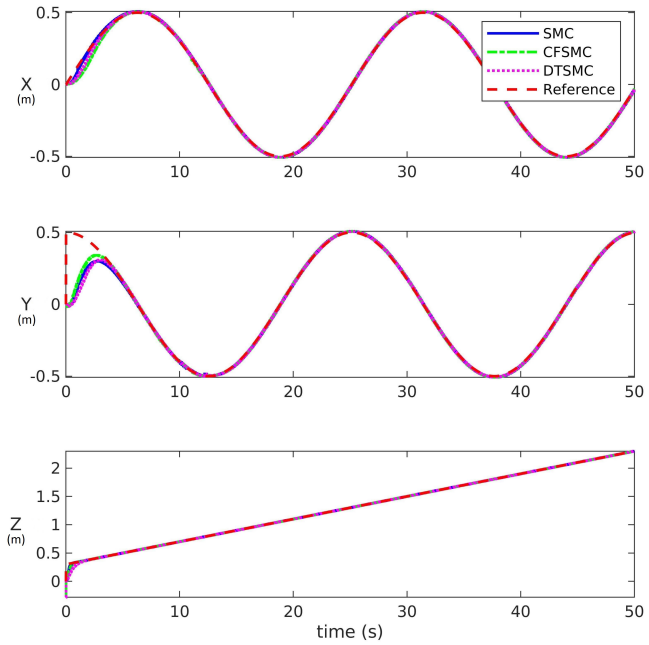


Figure 3.4: Position response of all control systems with a sampling time of 0.01s.

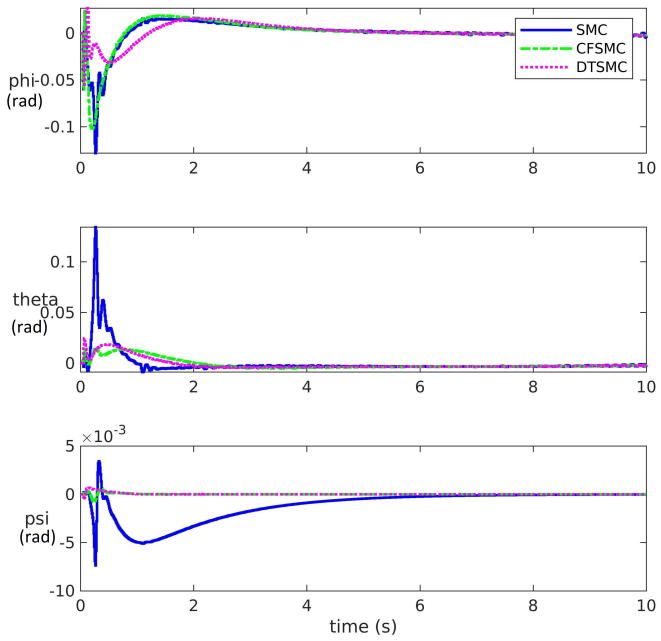


Figure 3.5: Attitude response of all control systems with a sampling time of 0.01s.

Due to the robust nature of sliding mode control, all three control systems display strong performance with a smaller sampling time of 0.01 seconds. DTSMC offers

comparable performance to classical SMC for control of the X and Y position of the quadrotor. DTSMC also displays slightly worse performance in the control of the Z position of the quadrotor. CFSMC displays the best control over the Y position of the quadrotor, while performing slightly worse than SMC and DTSMC in the control over the X position.

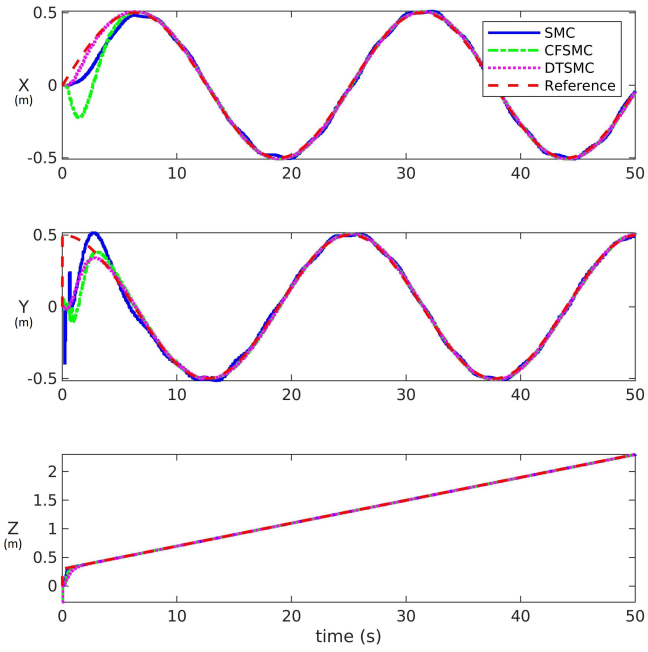


Figure 3.6: Position response of all control systems with a sampling time of 0.05s.

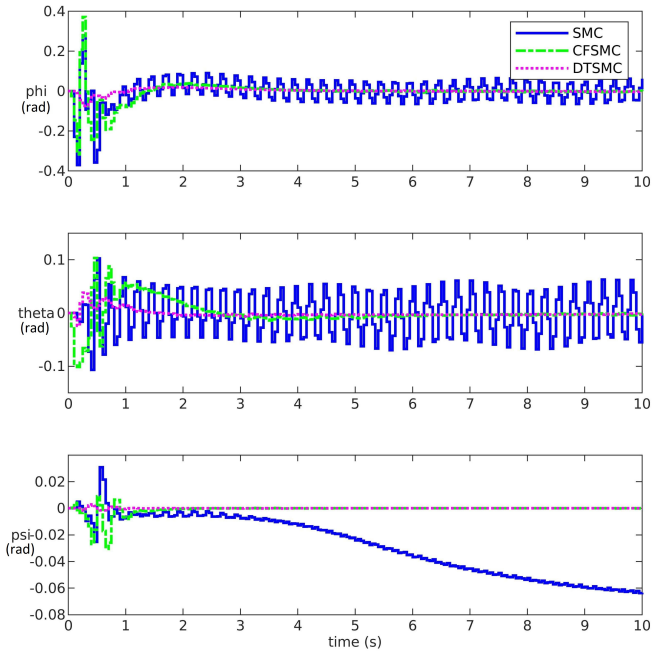


Figure 3.7: Attitude response of all control systems with a sampling time of 0.05s.

Table 3.2: IAE for quadrotor positions for each sample time T.

Integral of Absolute Error	**Control Systems**		
	SMC	***CFSMC***	***DTSMC***
T = 0.01 seconds			
X Position	0.3147	0.4382	0.3457
Y Position	1.1046	0.9495	1.1204
Z Position	0.0759	0.0935	0.1630
Total	1.4952	1.4812	1.6291
T = 0.05 seconds			
X Position	1.0590	1.3213	0.3944
Y Position	1.4730	1.3058	1.0573
Z Position	0.0942	0.1328	0.1827
Total	2.6262	2.7599	1.6344
T = 0.1 seconds			
X Position	Fail	Fail	0.8995
Y Position	Fail	Fail	1.1565
Z Position	Fail	Fail	0.0931
Total	-	-	2.1491

When the sampling time is increased to $T = 0.05$ seconds, both classical SMC and CFSMC display a deterioration in controller performance. The mean increase in IAE between $T = 0.01$ and $T = 0.05$ for position error of the quadrotor using SMC and CFSMC is 75.6% and 86.3% respectively. The mean increase in position error of the quadrotor using DTSMC is 0.33%. These results suggest that continuous sliding mode control methods such as classical SMC and CFSMC are not robust against changes in sample time on discrete systems. Meanwhile, the discrete-time sliding mode controller developed in this study demonstrates robustness against an increase in sample time, and shows far better performance than both continuous methods at

a larger sample time of 0.05 seconds. Both SMC and CFSMC displayed overshoot when the sampling time was increased, while DTSMC did not. Furthermore, increasing the sample time to $T = 0.1$ seconds caused both continuous control methods to fail, while DTSMC remained stable.

In the case of attitude response, SMC and CFSMC both display oscillatory behaviour when the sampling time is increased, while DTSMC has a stable transient response. SMC also displays steady-state error for the quadrotor's yaw.

3.6 Conclusion

In this chapter, a robust control system was developed in the discrete domain for complete trajectory tracking control of a quadrotor in 3D space. All three controllers displayed robust and accurate trajectory tracking performance in indoor environments when the sampling time was negligible. However, discrete-time sliding mode control provides a far more suitable option for scenarios where the sampling time of a system is larger, such as when autonomous SLAM is used for on-board localisation in GPS-denied nuclear legacy sites.

As the control system designed in this chapter implements a cascaded approach to the control system design, the inner and outer loops are modular. The following chapter aims to build on the work completed in this chapter by replacing the outer loop control system with a robust multi-agent control system based on DTSMC and consensus control. By extending the DTSMC algorithm developed in this chapter to the multi-agent case, the following chapter aims to provide a robust control strategy for the formation control of multiple quadrotors. This is particularly beneficial for the nuclear industry, as multi-quadrotor systems could be deploying for more efficient mapping and characterisation of hazardous and inaccessible areas of nuclear legacy sites.

Chapter 4

Discrete-time Sliding Mode Formation Control for Multi-Quadrotor Systems

4.1 Introduction

A Cyber-Physical System (CPS) based on a multi-quadrotor platform must be capable of autonomous movement around unknown areas. Modern research has provided insight into various methods to control these multi-quadrotor systems. Multi-agent control problems include the rendezvous problem, where a number of agents meet at a common location [130], the formation problem, where a multi-agent network must maintain a desired formation [131], and the flocking problem, where multiple agents must mimic the flocking behaviour displayed in nature in birds [132].

This chapter addresses the formation problem for a multi-agent system of networked quadrotors. Several methods have been proposed to solve the formation problem in quadrotors. In one paper, the authors propose a linear PID controller alongside a leader-follower network structure to solve the formation problem [133]. While PID can be robust, quadrotors are complex systems comprised of nonlinear and under-actuated coupled dynamics that can be exposed to parametric

uncertainties and disturbances, in which case a more robust control algorithm may be necessary. Another paper proposes a robust control approach to solve the formation problem based on sliding-mode control [134]. In the paper, a group of quadrotors could maintain formation while a load is suspended from the quadrotors in simulation. This was compared to linear methods, such as a linear control system using LQR-PID control and was shown to be superior. In another paper, adaptive control is proposed to solve the formation problem in the presence of parametric uncertainties [114]. The paper successfully developed an algorithm that estimates the parameters of each quadrotor while controlling the formation of quadrotors.

The aim of this chapter is to extend current research by developing a discrete-time sliding mode control system for the formation control of a number of quadrotor agents in a network and validate its efficacy for implementation in hazardous, complex, and dynamic indoor nuclear environments. By designing the controller in the discrete domain, the chapter aims to reduce the negative effects present when sampling at lower sampling rates.

This chapter extends published work completed during the PhD project. The work was peer-reviewed and published following a presentation of the findings at a conference:

- A. Can, J. Price, and A. Montazeri, "Robust formation control and trajectory tracking of multiple quadrotors using a discrete-time sliding mode control technique," *IFAC-PapersOnLine*, vol. 55, pp. 2974–2979, Oct. 2022, DOI: 10.1016/j.ifacol.2022.10.184

Parts of the published paper, such as the introduction and problem statement, have largely been reused in this chapter. The chapter significantly extends the published work by expanding the controller to the complete trajectory tracking case, with control over the x, y, and z axes. The work has also been updated to include a rigorous robustness proof of the multi-agent control system. Numerical results are also provided in the presence of simulated external disturbances. Most notably, the proposed discrete-time sliding mode formation control algorithm was implemented

experimentally to validate the robustness proof and the numerical results.

The structure of the chapter is organised as follows: the multi-agent control problem is formulated in Section 4.2; a robust discrete-time formation control algorithm is proposed in Section 4.3; Section 4.4 presents the simulation results for a nominal system and in the presence of simulated wind disturbance; Section 4.5 presents an experimental framework and validates the proposed algorithm in the real-world case; Section 4.6 provides a discussion of the findings of the chapter.

4.2 Problem Formulation

The following problem statement has been adapted from the original conference paper [5] and has been extended to include additional dynamics for distributed 6 DoF control over the quadrotors.

4.2.1 Discrete-time Quadrotor Model

The discrete-time quadrotor model in equations (3.3) to (3.4) is rearranged to provide the two following subsections of the quadrotor model that can be used for multi-agent control. The first block (4.1) provides the states that can controlled with an outer-loop multi-agent control system, while the second block (4.2) provides the states that can be controller by an inner-loop controller onboard each quadrotor.

$$\begin{cases} x_{k+1} = x_k + T\dot{x}_k, \\ \dot{x}_{k+1} = \dot{x}_k + T(c\varphi_k c\psi_k s\theta_k + s\varphi_k s\psi_k)\frac{u_{1,k}}{m}, \\ y_{k+1} = y_k + T\dot{y}_k \\ \dot{y}_{k+1} = \dot{y}_k + T(c\varphi_k s\theta_k s\psi_k - c\psi_k s\varphi_k)\frac{u_{1,k}}{m}, \\ z_{k+1} = z_k + T\dot{z}_k, \end{cases} \tag{4.1}$$

$$\begin{cases} \varphi_{k+1} = \varphi_k + T\dot{\varphi}_k, \\ \dot{\varphi}_{k+1} = \dot{\varphi}_k + T(\frac{(I_{yy}-I_{zz})\dot{\theta}_k\dot{\psi}_k+u_{2,k}}{I_{xx}}), \\ \varphi_{k+1} = \varphi_k + T\dot{\varphi}_k, \\ \dot{\theta}_{k+1} = \dot{\theta}_k + T(\frac{(I_{yy}-I_{zz})\dot{\varphi}_k\dot{\psi}_k+u_{3,k}}{I_{xx}}). \\ \dot{z}_{k+1} = \dot{z}_k + T\big(g - (c\varphi_k c\theta_k \frac{u_{1,k}}{m})\big), \\ \psi_{k+1} = \psi_k + T\dot{\psi}_k, \\ \dot{\psi}_{k+1} = \dot{\psi}_k + T(\frac{(I_{xx}-I_{yy})\dot{\varphi}_k\dot{\theta}_k+u_{4,k}}{I_{zz}}). \end{cases} \tag{4.2}$$

The outer-loop position subsystem is shown in equation (4.1) and presents the x, y, and z dynamics of a quadrotor. The inner-loop roll and pitch dynamics are presented in equation (4.2). The positions of the quadrotor in space at time step k are given by x_k, y_k, and z_k, while the velocities of the quadrotor are given by $\dot{x}_k$, $\dot{y}_k$, and $\dot{z}_k$. The roll, pitch and yaw of the quadrotor and its angular velocities are given by φ_k, θ_k, ψ_k, $\dot{\varphi}_k$, $\dot{\theta}_k$, and $\dot{\psi}_k$, respectively. The subscript $k+1$ is used to denote the values at the next time step T. The gravity acting on the quadrotor is given by g, while m represents the mass of the quadrotor. The diagonal components of the inertial matrix acting on the quadrotor are given by I_{xx}, I_{yy}, and I_{zz}. The control inputs to the system are thrust and torques around the x y and z axes of the quadrotor and are given by $u_{1,k}$, $u_{2,k}$, $u_{3,k}$, and $u_{4,k}$ respectively. For brevity, *cos* and *sin* are substituted with c and s.

Graph theory can be used to describe the communication topology between the agents. The following section provides preliminary information on graph theory and provides error terms used for deriving a control algorithm for the formation control of a multi-agent quadrotor system [5].

Definition 1. *A directed graph* $G = \{\mathbf{V}, \mathbf{E}, \mathbf{A}\}$ *contains a set of nodes* $\mathbf{V} \in \mathbb{R}^n$*, where each node represents an agent in the network,* n *is the total number of agents, and a set of directed edges* $\mathbf{E} \in \mathbf{V} \times \mathbf{V}$*, representing the communication link between*

two agents in the network. The matrix $\mathbf{A} = [a^{ij}] \in \mathbb{R}^{n\times n}$ *is the adjacency matrix of graph* G *such that*

$$\begin{cases} a^{ij} = 0 \quad if \;\; i = j \\ a^{ij} = 1 \quad V^i \; has \; a \; directed \; connection \; to \; V^j \\ a^{ij} = 0 \quad otherwise \end{cases}$$

Furthermore, as the communication topology can be time-varying, graph $G \in \hat{\boldsymbol{G}}$ *where* $\hat{\boldsymbol{G}} = [G^1, G^2, ...]$ *is a set of all possible graphs containing a spanning tree. At any time-step* k*, graph* G *can be one of any of the possible graphs in the set* $\hat{\boldsymbol{G}}$*.*

The Laplacian matrix of graph G *is a matrix representing the graph's structure and communication properties and is given by* $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $\mathbf{L}, \mathbf{D}, \mathbf{A} \in \mathbb{R}^{n\times n}$*, where* $\mathbf{D}$ *is the degree matrix of graph* G*.*

For the purpose of formation control, we take the dynamics of the outer-loop position subsystem only. The set of equations in (4.1) can be converted into the state-space form to describe the position dynamics of each agent i in the network.

$$\begin{cases} \boldsymbol{\eta}^i_{1,k+1} = \boldsymbol{\eta}^i_{1,k} + T\boldsymbol{\eta}^i_{2,k}, \\ \boldsymbol{\eta}^i_{2,k+1} = \boldsymbol{\eta}^i_{2,k} + T(\mathbf{f}^i(\boldsymbol{\eta}^i_k) + \mathbf{g}^i(\boldsymbol{\eta}^i_k)\mathbf{u}^i_k + \boldsymbol{\delta}^i_k). \end{cases} \tag{4.3}$$

$$\boldsymbol{\eta}^i_k = \begin{bmatrix} \boldsymbol{\eta}^i_{1,k} \\ \boldsymbol{\eta}^i_{2,k} \end{bmatrix}, \tag{4.4}$$

where $\boldsymbol{\eta}^i_{1,k} = [x^i_k \;\; y^i_k \;\; z^i_k]^\top$, $\boldsymbol{\eta}^i_{1,k} \in \mathbb{R}^3$ represents the state vector of the system indicating the position of the quadrotor for each agent i. The vector $\boldsymbol{\eta}^i_{2,k} = [\dot{x}^i_k \;\; \dot{y}^i_k \;\; \dot{z}^i_k]^\top$, $\boldsymbol{\eta}^i_{2,k} \in \mathbb{R}^3$ is the state vector of the system indicating the velocities of the quadrotor for agent i. The position states and their derivatives are concatenated into the vector $\boldsymbol{\eta}^i_k \in \mathbb{R}^6$.

The vector $\mathbf{u}^i_k = [u^i_{1,k} \;\; u^i_{2,k}, \;\; u^i_{3,k}, u^i_{4,k}]^\top, \mathbf{u}^i_k \in \mathbb{R}^4$. The functions $\mathbf{f}^i : \mathbb{R}^6 \to \mathbb{R}^3$ and $\mathbf{g}^i : \mathbb{R}^6 \to \mathbb{R}^{3\times 4}$ are nonlinear mappings that describe the system dynamics and $\boldsymbol{\delta}^i_k \in \mathbb{R}^3$ represents an external matched disturbance acting on the positional

subsystem of agent i. It is assumed that $\boldsymbol{\delta}_k^i$ is bounded such that $\|\boldsymbol{\delta}_k^i\|_\infty \leq \delta_{\max}, \forall k \in \mathbb{N}, \forall i$.

The compact form for the dynamics of the graph is therefore:

$$\begin{cases} \boldsymbol{\eta}_{1,k+1} = \boldsymbol{\eta}_{1,k} + T\boldsymbol{\eta}_{2,k}, \\ \boldsymbol{\eta}_{2,k+1} = \boldsymbol{\eta}_{2,k} + T(\mathbf{f}(\boldsymbol{\eta}_k) + \mathbf{g}(\boldsymbol{\eta}_k)\mathbf{u}_k + \boldsymbol{\delta}_k), \end{cases} \tag{4.5}$$

$$\boldsymbol{\eta}_k = \begin{bmatrix} \boldsymbol{\eta}_{1,k} \\ \boldsymbol{\eta}_{2,k} \end{bmatrix}, \tag{4.6}$$

where the vector $\boldsymbol{\eta}_k \in \mathbb{R}^{6n}$ contains the position and velocity components. The state vectors of the system in (4.5) are defined as $\boldsymbol{\eta}_{1,k} = [\boldsymbol{\eta}_{1,k}^{1^\top}, \boldsymbol{\eta}_{1,k}^{2^\top}, ..., \boldsymbol{\eta}_{1,k}^{n^\top}]^\top$, $\boldsymbol{\eta}_{1,k} \in \mathbb{R}^{3n}$ and $\boldsymbol{\eta}_{2,k} = [\boldsymbol{\eta}_{2,k}^{1^\top}, \boldsymbol{\eta}_{2,k}^{2^\top}, ..., \boldsymbol{\eta}_{2,k}^{n^\top}]^\top, \boldsymbol{\eta}_{2,k} \in \mathbb{R}^{3n}$. The input vector is defined as $\mathbf{u}_k = [\mathbf{u}_k^{1^\top}, \mathbf{u}_k^{2^\top}, ..., \mathbf{u}_k^{n^\top}]^\top, \mathbf{u}_k \in \mathbb{R}^{4n}$. The state transition vector $\mathbf{f}(\boldsymbol{\eta}_k) = [\mathbf{f}^1(\boldsymbol{\eta}_k^1)^\top, \mathbf{f}^2(\boldsymbol{\eta}_k^2)^\top, ..., \mathbf{f}^n(\boldsymbol{\eta}_k^n)^\top]^\top$, $\mathbf{f}(\boldsymbol{\eta}_k) \in \mathbb{R}^{3n}$ and the state input matrix $\mathbf{g}(\boldsymbol{\eta}_k) = diag[\mathbf{g}^1(\boldsymbol{\eta}_k^1)^\top, \mathbf{g}^2(\boldsymbol{\eta}_k^2)^\top, ..., \mathbf{g}^n(\boldsymbol{\eta}_k^n)^\top]^\top, \mathbf{g}(\boldsymbol{\eta}_k) \in \mathbb{R}^{4n\times 3n}$. Finally the matched disturbance vector is mapped using $\boldsymbol{\delta}_k = [\boldsymbol{\delta}_k^{1^\top}, \boldsymbol{\delta}_k^{2^\top}, ..., \boldsymbol{\delta}_k^{n^\top}]^\top, \boldsymbol{\delta}_k \in \mathbb{R}^{3n}$.

The dynamics of the virtual leader can be taken as the desired trajectory for the centre of the formation of the quadrotors and thus can be represented as

$$\begin{cases} \boldsymbol{\eta}_{1,k+1}^0 = \boldsymbol{\eta}_{1,k}^0 + T\boldsymbol{\eta}_{2,k}^0, \\ \boldsymbol{\eta}_{2,k+1}^0 = -c_1\boldsymbol{\eta}_{1,k}^0 - c_2T\boldsymbol{\eta}_{2,k}^0 + c_1\boldsymbol{\eta}_{1,k}^{d0} + c_2T\boldsymbol{\eta}_{2,k}^{d0}, \end{cases} \tag{4.7}$$

where agent 0 is considered the virtual leader of the system with the same states as the other agents in the system. Scalar values $c_1 \in \mathbb{R}$ and $c_2 \in \mathbb{R}$ are gains to be designed. $\boldsymbol{\eta}_{1,k}^{d0}$ is the desired position for $\boldsymbol{\eta}_{1,k}^0$, and $\boldsymbol{\eta}_{2,k}^{d0}$ is the desired velocity for $\boldsymbol{\eta}_{2,k}^0$.

Furthermore, for the purpose of formation control, the term $\Delta\boldsymbol{\eta}_{1,k}^i = [\Delta x_k^i\ \Delta y_k^i\ \Delta z_k^i]^\top$, $\Delta\boldsymbol{\eta}_{1,k}^i \in \mathbb{R}^3$ is introduced as the desired distance of agent i from the position of agent 0 or leader x^0. The term $\Delta\boldsymbol{\eta}_{2,k}^i = [\Delta\dot{x}_k^i\ \Delta\dot{y}_k^i\ \Delta\dot{z}_k^i]^\top$, $\Delta\boldsymbol{\eta}_{2,k}^i \in \mathbb{R}^3$ is the rate of change of positions in the formation. The formation is considered time-varying if any element in $\Delta\boldsymbol{\eta}_{2,k}^i$ is non-zero.

Definition 2. *Assuming that the virtual leader* 0 *is the centre of the formation, the quadrotor swarm has achieved formation if the following limits are satisfied:*

$$\begin{cases} \lim_{k\to\infty}(\mathbf{e}^i_{1,k}) = 0, \ \ \forall i, \\ \lim_{k\to\infty}(\mathbf{e}^i_{2,k}) = 0, \ \ \forall i, \end{cases} \tag{4.8}$$

where

$$\mathbf{e}^i_{1,k} = \sum_{j\in n_j} a^{ij}(\boldsymbol{\eta}^j_{1,k} - \Delta\boldsymbol{\eta}^j_{1,k} - \boldsymbol{\eta}^i_{1,k} + \Delta\boldsymbol{\eta}^i_{1,k}) + b^i(\boldsymbol{\eta}^0_{1,k} - \boldsymbol{\eta}^i_{1,k} + \Delta\boldsymbol{\eta}^i_{1,k}), \tag{4.9}$$

$$\mathbf{e}^i_{2,k} = \sum_{j\in n_j} a^{ij}(\boldsymbol{\eta}^j_{2,k} - \Delta\boldsymbol{\eta}^j_{2,k} - \boldsymbol{\eta}^i_{2,k} + \Delta\boldsymbol{\eta}^i_{2,k}) + b^i(\boldsymbol{\eta}^0_{2,k} - \boldsymbol{\eta}^i_{2,k} + \Delta\boldsymbol{\eta}^i_{2,k}). \tag{4.10}$$

Here, $\mathbf{e}^i_{1,k} \in \mathbb{R}^3$ *contains the* x*,* y *and* z *components of the position error dynamics, and* $\mathbf{e}^i_{2,k} \in \mathbb{R}^3$ *contains the* x*,* y *and* z *components of the velocity error dynamics. The term* b^i *is the scalar components of the diagonal matrix* $B \in \mathbb{R}^{n\times n}$ *for each agent, where* $b^i = 1$ *if agent* i *is directly connected to the virtual leader* 0*, otherwise* $b^i = 0$*.*

From equations (4.9) and (4.10), the compact form of the error dynamics can be derived for each position channel of the quadrotor agents.

$$\mathbf{e}_{1,k} = -\Big((L+B)\otimes\mathbf{I}_n\Big)(\boldsymbol{\eta}_{1,k} - \Delta\boldsymbol{\eta}_{1,k} - \underline{1}\otimes\boldsymbol{\eta}^0_{1,k}), \tag{4.11}$$

$$\mathbf{e}_{2,k} = -\Big((L+B)\otimes\mathbf{I}_n\Big)(\boldsymbol{\eta}_{2,k} - \Delta\boldsymbol{\eta}_{2,k} - \underline{1}\otimes\boldsymbol{\eta}^0_{2,k}), \tag{4.12}$$

where $\mathbf{e}_{1,k} \in \mathbb{R}^{3n}$ contains the error dynamics for x, y and z positions for each quadrotor agent and $\mathbf{e}_{2,k} \in \mathbb{R}^{3n}$ contains the error for each velocity in the x, y and z directions for each agent. The Kronecker product is denoted by $\otimes$ and $\mathbf{I}_N \in \mathbb{R}^{3\times3}$ is the identity matrix where N denotes the number of states.

Remark 1. *The position and velocity errors in equations* (4.11) *and* (4.12) *can be broken down into their individual components by eliminating the Kronecker product. For example, the compact form of the error dynamics for the x subsystem, represented by* $\mathbf{e}_{1,k,x} \in \mathbb{R}^n$ *and* $\mathbf{e}_{2,k,x} \in \mathbb{R}^n$*, can be expressed as*

$$\mathbf{e}_{1,k,x} = -(L+B)(\mathbf{x}_{1,k} - \Delta\mathbf{x}_{1,k} - \underline{1}x_k^0), \tag{4.13}$$

$$\mathbf{e}_{2,k,x} = -(L+B)(\mathbf{x}_{2,k} - \Delta\mathbf{x}_{2,k} - \underline{1}\dot{x}_k^0). \tag{4.14}$$

At time step k*,* $\mathbf{x}_{1,k} \in \mathbb{R}^n$ *holds the* x_k^i *position of each agent* i *in the network, while* $\mathbf{x}_{2,k} \in \mathbb{R}^n$ *contains the velocity* $\dot{x}_k^i$ *of each agent. Additionally,* $\Delta\mathbf{x}_{1,k} \in \mathbb{R}^n$ *and* $\Delta\mathbf{x}_{2,k} \in \mathbb{R}^n$ *store the desired separation* Δx_k^i *for each agent and the rate of change of that separation* $\Delta\dot{x}_k^i$ *respectively.*

4.3 Robust DTSMC Based Formation Control

The DTSMC formation control inputs (4.15), developed in this section, can provide outer loop control inputs for each agent i. The x and y axis control inputs can be converted to desired angles using methods presented in Section 3.3.2. Equations (3.17) and (3.18) are used to convert u_x^i and u_y^i to desired angles φ_d^i and θ_d^i for each agent i. Following this, a separate inner loop control system can be used to control the attitude of each agent. In this chapter, the discrete-time attitude control system designed in Section 3.3.1 is implemented as the inner loop controller. The altitude of each agent in the system is controlled directly using the u_z^i control input from (4.15).

Theorem 2. *A robust formation control algorithm is proposed for each agent* i*.*

$$\mathbf{u}_k^i = \left(\frac{1}{(d^i+b^i)T}\right)(\mathbf{g^i}(\boldsymbol{\eta}_k^i))^{-1}\Bigg[-(d^i+b^i)(\boldsymbol{\eta}_{2,k}^i + f(\boldsymbol{\eta}_k^i)T) + \sum_{j\in n_j} a^{ij}\Big(\boldsymbol{\eta}_{2,k+1}^j - \Delta\boldsymbol{\eta}_{2,k+1}^j + \Delta\boldsymbol{\eta}_{2,k+1}^i\Big) + b^i\Big(\boldsymbol{\eta}_{2,k+1}^0 + \Delta\boldsymbol{\eta}_{2,k+1}^i\Big) + \boldsymbol{\alpha}^i(\mathbf{e}_{1,k+1}^i) - (1-\boldsymbol{\mu}^i T)\boldsymbol{\sigma}_k^i + \boldsymbol{\epsilon}^i sgn(\boldsymbol{\sigma}_k^i)T\Bigg]. \tag{4.15}$$

Here, $\boldsymbol{\mu}_k^i \in \mathbb{R}^{3\times3}$ *is a diagonal matrix of design parameters* $\mu_{x,k}^i$, $\mu_{y,k}^i$ *and* $\mu_{z,k}^i$ *where each* $\mu_k^i > 0, \mu_k^i < 2$ *. Matrix* $\boldsymbol{\epsilon}_k^i \in \mathbb{R}^{3\times3}$ *is a diagonal matrix of design parameters* $\epsilon_{x,k}^i$, $\epsilon_{y,k}^i$ *and* $\epsilon_{z,k}^i$ *where each* $\epsilon_k^i > 0$*. The global form of the formation control protocol is given by*

$$\mathbf{u}_k = \frac{1}{T}\mathbf{g}(\boldsymbol{\eta}_\mathbf{k})^{-1}\Bigg(- \Big((L+B)\otimes\mathbf{I}_n\Big)^{-1}\Bigg[\Big((L+B)\otimes\mathbf{I}_n\Big)\Big(\boldsymbol{\eta}_{2,k} + T\mathbf{f}(\boldsymbol{\eta}_\mathbf{k}) - \Delta\boldsymbol{\eta}_{2,k+1} - \underline{1}\otimes\boldsymbol{\eta}_{2,k+1}^0\Big) + (1-\boldsymbol{\mu}T)\boldsymbol{\sigma}_k - \boldsymbol{\epsilon}Tsgn(\boldsymbol{\sigma}_k) - \boldsymbol{\alpha}(\mathbf{e}_{1,k+1})\Bigg]. \tag{4.16}$$

For the multi-UAV system presented by (4.5) *and* (4.6) *and the virtual leader presented by* (4.7)*, the formation control protocol* (4.16) *results in the convergence of the multi-UAV system to the desired formation determined by* (4.8) *to* (4.10) $\Delta\boldsymbol{\eta}_k$ *around the virtual leader* (4.7)*.*

Proof. A sliding surface is selected for the networked system.

$$\boldsymbol{\sigma}_k = \boldsymbol{\alpha}(\mathbf{e}_{1,k}) + \mathbf{e}_{2,k}, \tag{4.17}$$

where $\boldsymbol{\sigma}_k \in \mathbb{R}^{3n}$ contains the sliding surface $\boldsymbol{\sigma}_k^i$ for each agent i. $\boldsymbol{\sigma}_k^i = [\sigma_{x,k}^i, \sigma_{y,k}^i, \sigma_{z,k}^i]^\top \in \mathbb{R}^3$ contains the individual sliding surfaces for the x, y, and z axes of the quadrotor model. Matrix $\boldsymbol{\alpha} \in \mathbb{R}^{3n\times3n}$ is a diagonal matrix of control parameters for each agents x, y, and z terms. For this proof, each $\sigma_{x,k}^i$, $\sigma_{x,k}^i$ and $\sigma_{x,k}^i$ term can be represented by the term $\sigma_{s,k}^i$ for brevity. The same holds true for tuning parameters μ_s^i, ϵ_s^i, and disturbance term δ_s^i.

The stability of the networked system can be analysed using the following Lyapunov Candidate function.

$$V_k = \frac{1}{2}\boldsymbol{\sigma}_k^T\boldsymbol{\sigma}_k. \tag{4.18}$$

Separating this into the components for each agent gives

$$V_k = \frac{1}{2}\sum_{i=1}^{n}{\sigma_{s,k}^i}^2. \tag{4.19}$$

To guarantee robust stability, $\Delta V_k < 0$ must be satisfied for each discrete increment.

$$\Delta V_k = V_k - V_{k+1} < 0. \tag{4.20}$$

$$\Delta V_k = \frac{1}{2}\sum_{i=1}^{n}\Big({\sigma^i_{s,k+1}}^2 - {\sigma^i_{s,k}}^2\Big). \tag{4.21}$$

Using the difference of squares gives the following result.

$$\Delta V_k = \frac{1}{2}\sum_{i=1}^{n}\Big(\sigma^i_{s,k+1} + \sigma^i_{s,k}\Big)\Big(\sigma^i_{s,k+1} - \sigma^i_{s,k}\Big). \tag{4.22}$$

From equation (4.22), it can be seen that the system is stable in two cases.

Case 1:

$$\Big(\sigma^i_{s,k+1} - \sigma^i_{s,k}\Big) > 0, \quad \Big(\sigma^i_{s,k+1} + \sigma^i_{s,k}\Big) < 0, \quad \forall \quad i. \tag{4.23}$$

Case 2:

$$\Big(\sigma^i_{s,k+1} - \sigma^i_{s,k}\Big) < 0, \quad \Big(\sigma^i_{s,k+1} + \sigma^i_{s,k}\Big) > 0, \quad \forall \quad i. \tag{4.24}$$

The global sliding surface at time step $k+1$ is given by

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\alpha}(\mathbf{e}_{1,k+1}) + \mathbf{e}_{2,k+1}. \tag{4.25}$$

Substituting $\mathbf{e}_{2,k+1}$ with the propagated error dynamics from (4.12) gives

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\alpha}(\mathbf{e}_{1,k+1}) - \Big((L+B)\otimes\mathbf{I}_n\Big)(\boldsymbol{\eta}_{2,k+1} - \Delta\boldsymbol{\eta}_{2,k+1} - \underline{1}\otimes\boldsymbol{\eta}^0_{2,k+1}). \tag{4.26}$$

Substituting $\boldsymbol{\eta}_{2,k+1}$ with the multi-agent dynamics (4.5) gives

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\alpha}(\mathbf{e}_{1,k+1}) - \Big((L+B)\otimes\mathbf{I}_n\Big)(\boldsymbol{\eta}_{2,k} + T(\mathbf{f}(\boldsymbol{\eta}_k) + \mathbf{g}(\boldsymbol{\eta}_k)\mathbf{u}_k + \mathbf{d}_k) - \Delta\boldsymbol{\eta}_{2,k+1} - \underline{1}\otimes\boldsymbol{\eta}^0_{2,k+1}). \tag{4.27}$$

Substituting in global control protocol (4.16) and cancelling terms gives

$$\boldsymbol{\sigma}_{k+1} = (1 - \boldsymbol{\mu}T)\boldsymbol{\sigma}_k - \boldsymbol{\epsilon}Tsgn(\boldsymbol{\sigma}_k) - \Big((L+B)\otimes\mathbf{I}_n\Big)T\boldsymbol{\delta}_k. \tag{4.28}$$

The equivalent sliding surface for a single agent is given by

$$\sigma^i_{s,k+1} = (1 - \mu^i_s T)\sigma^i_{s,k} - \epsilon^i_s T sgn(\sigma^i_{s,k}) + \sum_{j\in n_j} a^{ij}(\delta^i_{s,k} - \delta^j_{s,k}) + b^i\delta^i_{s,k}). \tag{4.29}$$

For Case 1, substituting $\sigma^i_{s,k+1}$ into the first part of equation (4.23) gives

$$(1-\mu^i_s T)\sigma^i_{s,k} - \boldsymbol{\epsilon}^i T sgn(\sigma^i_{s,k}) + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) - \sigma^i_{s,k} > 0. \quad (4.30)$$

When $\sigma^i_{s,k} < 0$ the above stability condition can be rearranged as

$$-\mu^i_s T\sigma^i_{s,k} + \epsilon^i_s T + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) > 0. \quad (4.31)$$

The most extreme disturbance on the system in this case is when $\delta s, k^i = \delta_{\max}$ and $\delta^j_{s,k} = -\delta_{\max}$. Substituting the most extreme case for the disturbance into equation (4.31) gives

$$-\mu^i_s T\sigma^i_{s,k} + \epsilon^i_s T - (2d^i + b^i)\delta_{\max} > 0. \quad (4.32)$$

Inequality (4.32) always holds true as long as the gains are designed such that

$$\epsilon^i_s > (2d^i + b^i)\delta_{\max}. \quad (4.33)$$

Substituting $\sigma^i_{s,k+1}$ into the second part of equation (4.23) gives

$$(1-\mu^i_s T)\sigma^i_{s,k} - \epsilon^i_s T sgn(\sigma^i_{s,k}) + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) + \sigma^i_{s,k} < 0. \quad (4.34)$$

When $\sigma^i_{s,k} < 0$ the above stability condition can be rearranged as

$$(2-\mu^i_s T)\sigma^i_{s,k} + \epsilon^i_s T + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) < 0. \quad (4.35)$$

The most extreme disturbance on the system here is when $\delta s, k^i = -\delta_{\max}$ and $\delta^j_{s,k} = \delta_{\max}$. Substituting the most extreme case for the disturbance into equation (4.34) gives

$$(2-\mu^i_s T)\sigma^i_{s,k} + \epsilon^i_s T + (2d^i + b^i)\delta_{\max} < 0. \quad (4.36)$$

From equation (4.36), the lower bound of a Quasi-Sliding-Mode Band (QSMB) can be found as

$$\sigma^i_{s,k} < \frac{-\epsilon^i_s T - (2d^i + b^i)\delta_{\max}}{2-\mu^i_s T}. \quad (4.37)$$

For Case 2, substituting $\sigma^i_{s,k+1}$ into the first part of equation (4.24) gives

$$(1-\mu^i_s T)\sigma^i_{s,k} - \boldsymbol{\epsilon}^i T sgn(\sigma^i_{s,k}) + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) - \sigma^i_{s,k} < 0. \quad (4.38)$$

When $\sigma^i_{s,k} > 0$ the above stability condition can be rearranged as

$$-\mu^i_s T\sigma^i_{s,k} - \boldsymbol{\epsilon}^i T + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) < 0. \quad (4.39)$$

In this case, the most extreme disturbance is found when $\delta s, k^i = \delta_{\max}$ and $\delta^j_{s,k} = -\delta_{\max}$. Therefore, the inequality in equation (4.39) always holds true as long as

$$\epsilon^i_s > (2d^i + b^i)\delta_{\max}. \quad (4.40)$$

Substituting $\sigma^i_{s,k+1}$ into the second part of equation (4.24) gives

$$(2-\mu^i_s T)\sigma^i_{s,k} - \epsilon^i_s T sgn(\sigma^i_{s,k}) + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) > 0. \quad (4.41)$$

When $\sigma^i_{s,k} > 0$ the above stability condition can be rearranged as

$$(2-\mu^i_s T)\sigma^i_{s,k} - \epsilon^i_s T + \sum_{j\in n_j} a^{ij}(\delta s, k^i - \delta^j_{s,k}) + b^i(\delta s, k^i) > 0. \quad (4.42)$$

Here, the most extreme disturbance is found when $\delta s, k^i = -\delta_{\max}$ and $\delta^j_{s,k} = \delta_{\max}$. Rearranging (4.42) gives the upper bound of the QSMB as

$$\sigma^i_{s,k} > \frac{\epsilon^i_s T + (2d^i + b^i)\delta_{\max}}{2-\mu^i_s T}. \quad (4.43)$$

This proof demonstrates the stability of the networked system and shows that the system converges to the QSMB shown by bounds in equations (4.37) and (4.43). The robust stability of the system is guaranteed if control term ϵ dominates the disturbances of the networked system. This concludes the proof. □

Interestingly, the QSMB can be reduced by increasing the sampling rate of the discrete-time system. The stability proof also shows that the effects of disturbance on the system can be limited by reducing the number of neighbours j per agent i. Therefore, to ensure the best controller performance, the sampling rate should

be reduced and the number of neighbouring agents limited. Finally, by reducing the number of neighbours, (4.33) and (4.40) show that the magnitude of ϵ can be reduced. A smaller value for ϵ would reduce the chattering present in the system by decreasing the magnitude of the switching term.

4.4 Numerical Results

In this section the formation controller proposed in Theorem 2 is evaluated numerically using MATLAB/Simulink simulation platform.

4.4.1 Simulation Design

The numerical simulation platform developed in this section simulates three cooperative quadrotors. In this scenario, 3 quadrotors were tasked with tracking a virtual leader. Assuming that the heading angles of each agent are aligned at time step $t = 0s$, the desired yaw angle for each quadrotor is set to $\psi^i_{k,d} = 0$.

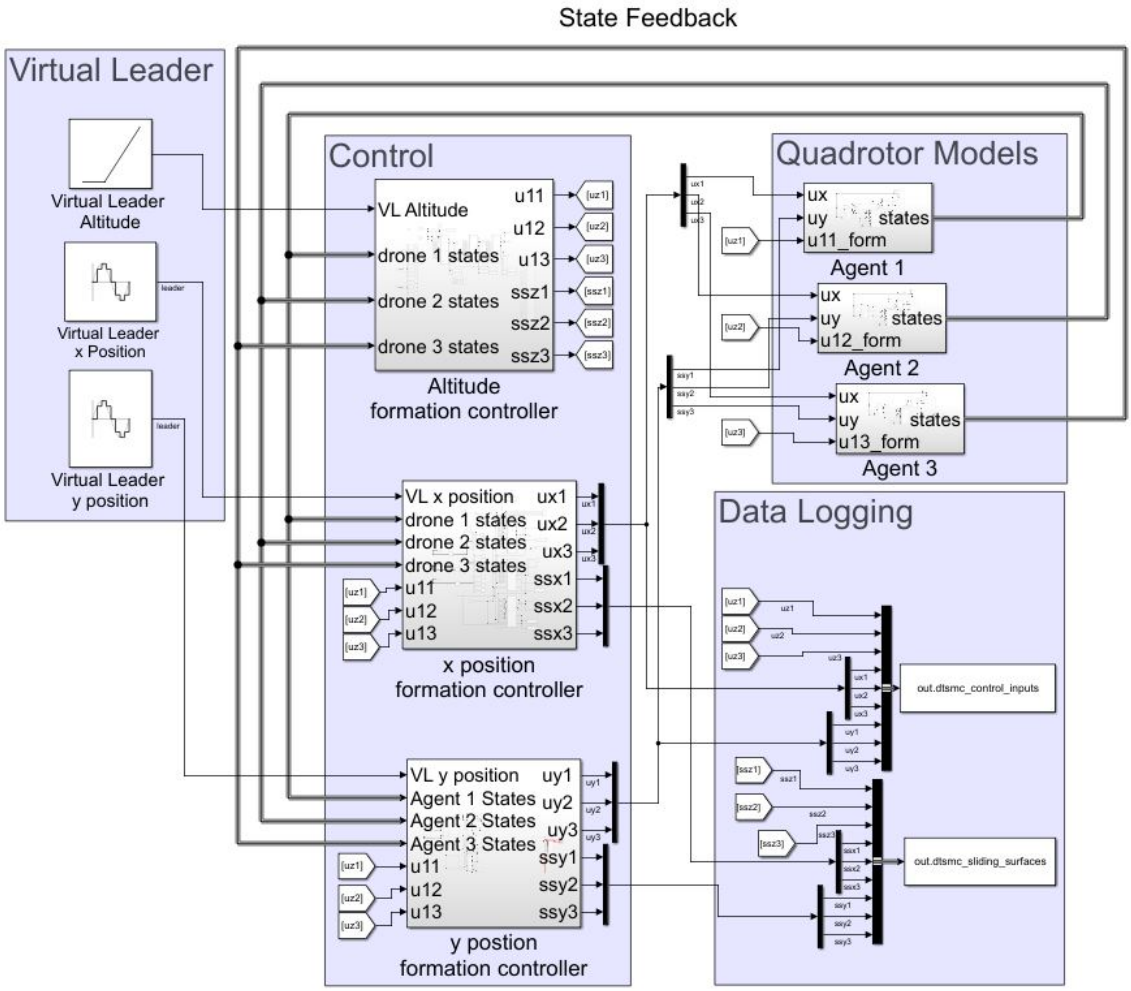


Figure 4.1: Simulink block diagram for simulation of the DTSMC formation controller.

A helical trajectory was selected to assess the performance of the multi-agent system in simulation. The trajectory of the leader is described as

$$\begin{cases} x_k^0 = 0.5\sin(0.1kT), \\ y_k^0 = 0.5\cos(0.1kT), \\ z_k^0 = 0.02kT + 0.5. \end{cases} \tag{4.44}$$

A helical trajectory was used for the input from the virtual leader to assess the

controller's robust performance under a complex and continuously differentiable trajectory. Additionally, the agents' initial positions are out of formation to assess the convergence performance of the multi-agent system. The initial position for each agent is shown in Table 4.1

Table 4.1: Table of initial positions for each agent.

Axis	**Initial Position (m)**		
	Agent 1	**Agent 2**	**Agent 3**
x_k	2.0	-2.0	1.5
y_k	-1.0	-1.0	1.5
z_k	0.0	0.0	0.0

An equilateral triangle was selected as a formation where the displacement from the leader for each agent is described in Table 4.2.

Table 4.2: Reference formation for each agent.

Axis	**Reference Formation (m)**		
	Agent 1	**Agent 2**	**Agent 3**
Δx_k	1.0	-0.5	-0.5
Δy_k	0.0	-0.866	0.866
Δz_k	0.0	0.1	-0.1

Only Agent 1 receives information from the virtual leader to provide a more realistic testing scenario. The communication topology implements a directed spanning tree and is displayed in Figure 4.2.

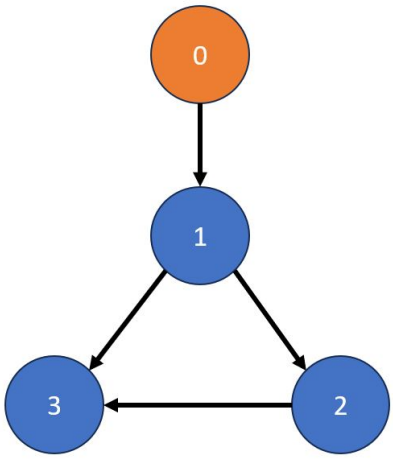


Figure 4.2: Graph showing communication topology between agents 1, 2, 3, and a virtual leader 0.

The mass and inertial parameters selected for the simulation, based on the parameters of a Crazyflie 2.0, are shown in Table 4.3 [135].

Table 4.3: Simulation parameters for each agent.

Parameter	**Values**		
	Agent 1	**Agent 2**	**Agent 3**
mass (kg)	3.3×10^{-2}	3.3×10^{-2}	3.3×10^{-2}
I_{xx} (kg/m^2)	1.40×10^{-5}	1.40×10^{-5}	1.40×10^{-5}
I_{yy} (kg/m^2)	1.44×10^{-5}	1.44×10^{-5}	1.44×10^{-5}
I_{zz} (kg/m^2)	2.17×10^{-5}	2.17×10^{-5}	2.17×10^{-5}

Table 4.4: DTSMC control parameters for each agent.

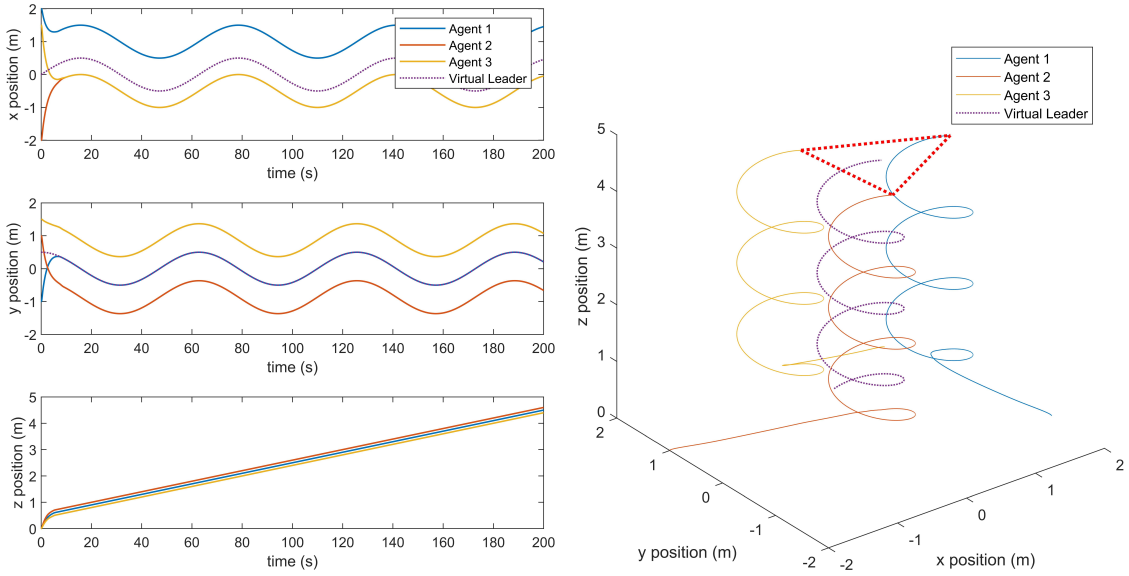
Parameter	**Values**		
	Agent 1	**Agent 2**	**Agent 3**
α_x	5	5	5
α_y	5	5	5
α_z	5	5	5
μ_x	0.5	0.5	0.5
μ_y	0.5	0.5	0.5
μ_z	0.5	0.5	0.5
ϵ_x	0.1	0.1	0.1
ϵ_y	0.1	0.1	0.1
ϵ_z	0.1	0.1	0.1
α_φ	16	16	16
α_θ	16	16	16
α_ψ	3	3	3
η_φ	30	30	30
η_θ	30	30	30
η_ψ	10	10	10
ϵ_φ	0.01	0.01	0.01
ϵ_θ	0.01	0.01	0.01
ϵ_ψ	0.01	0.01	0.01

The simulation sample time was set to $T = 0.01s$ for all simulations.

4.4.2 Simulation Results Under Nominal Conditions

The control system is first assessed under nominal conditions without any disturbance applied. The results were recorded in Simulink and plotted using MATLAB.

Figure 4.3 displays the x, y and z response of each quadrotor, as well as the trajectory of the virtual leader. The three-dimensional positions of each drone are plotted in Figure 4.3(b).



(a) (b)

Figure 4.3: Trajectory response of the networked system following a virtual leader with a helical trajectory, with (a) displaying the separate x, y, and z response, and (b) displaying the three-dimensional trajectories.

From Figure 4.3(a), it is clear that the agents converge to the desired formation around the virtual leader in under 10 seconds and track the virtual leader for the duration of the flight. Each agent maintains the formation around the virtual leader successfully, with no observed chattering in the position states. Agent 1 rapidly converges to the desired position around the virtual leader, while Agent 2 and Agent 3 experience a slight initial deviation on the y axis. This may be due to the communication topology of the system, as only Agent 1 has access to the position states of the virtual leader. Therefore, it is unlikely that this behaviour can be improved with additional tuning. The dotted red line in Figure 4.3(b) connects the final positions of each agent, showing the triangular formation.

To show the transient response of a single UAV in this system, Figure 4.4 shows the roll, pitch and yaw angles of Agent 1 over time.

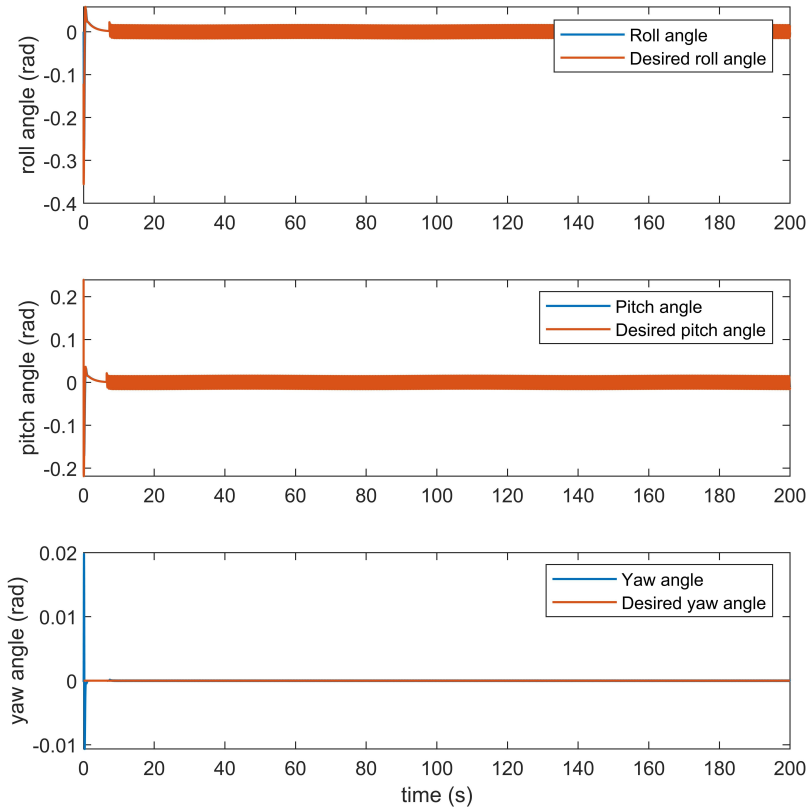
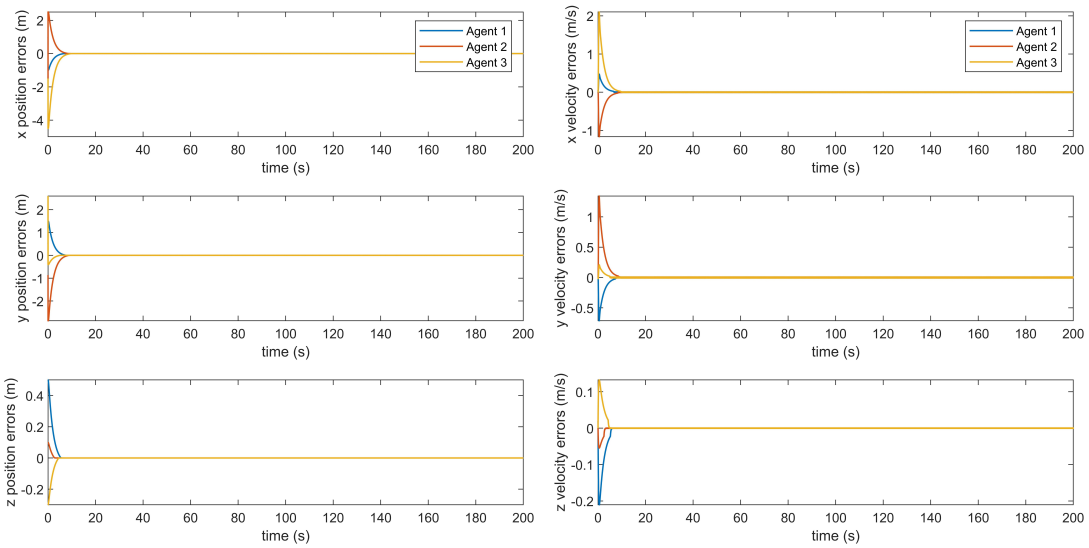


Figure 4.4: Plot of the actual and desired attitude of Agent 1.

The thick lines in Figure 4.4 suggest the presence of rapid oscillations in control input from the proposed control system. This could indicate the presence of chattering in the control system. While this allows the system to accurately track the virtual leader, this may be unfeasible for a quadrotor in the real world due to motor or power supply constraints.

To assess the system's stability, the error values in equations (4.11) and (4.12) were plotted against time.



(a) (b)

Figure 4.5: Position (a) and velocity (b) error plots for each agent for the x, y, and z.

According to Definition 2, the system is stable if the error dynamics approach 0 as t approaches infinity. From Figures 4.5(a) and 4.5(b), it can be seen that this condition is met for both the position and velocity error dynamics, $\mathbf{e}^i_{1,k}$ and $\mathbf{e}^i_{2,k}$ for each agent i in the system. The system errors converge to zero by time $t = 10s$. The error terms then remain around 0 for the duration of the flight, demonstrating the controller's performance in the nominal case.

The sliding surfaces can also observed to assess the stability of the networked system. The sliding surfaces of the system are shown in Figure 4.6.

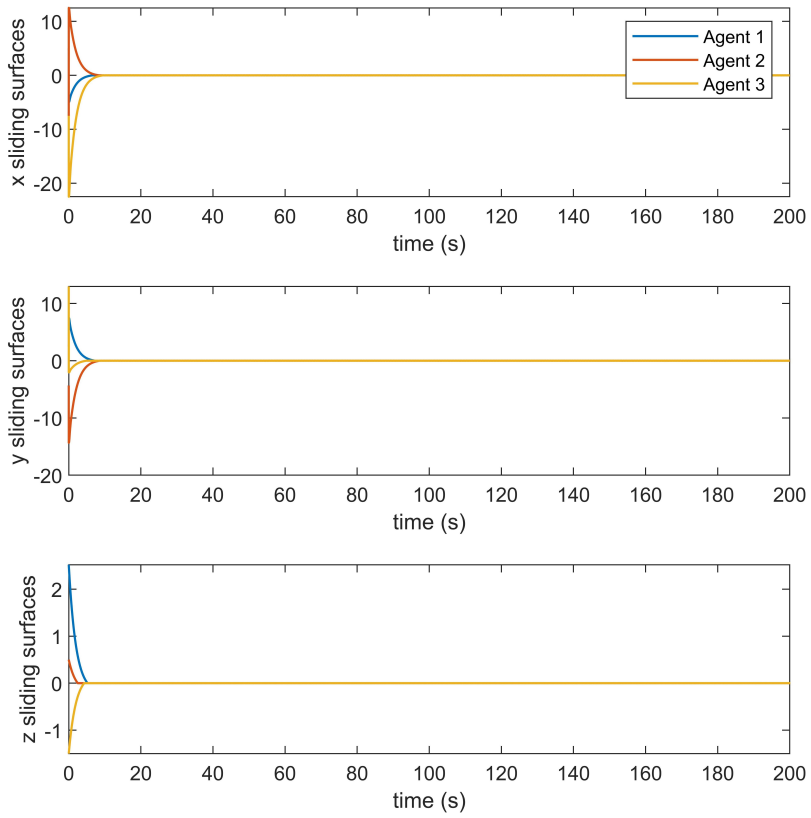


Figure 4.6: Sliding surfaces on the x, y, and z axes for each agent.

From Figure 4.6, it can be seen that all sliding surfaces converge to 0 by time $t = 10s$ and remain there for the duration of the simulation. The initial sliding surfaces at time $t = 0s$ are non-zero due to the initial conditions of the quadrotors being outside of the formation. To assess the chattering response, the transient behaviour of the sliding surfaces can be observed by plotting a small portion of the sliding surface after convergence. In Figure 4.7, the sliding surface is plotted over the final 5 seconds of the flight, from $t = 195s$ to $t = 200s$.

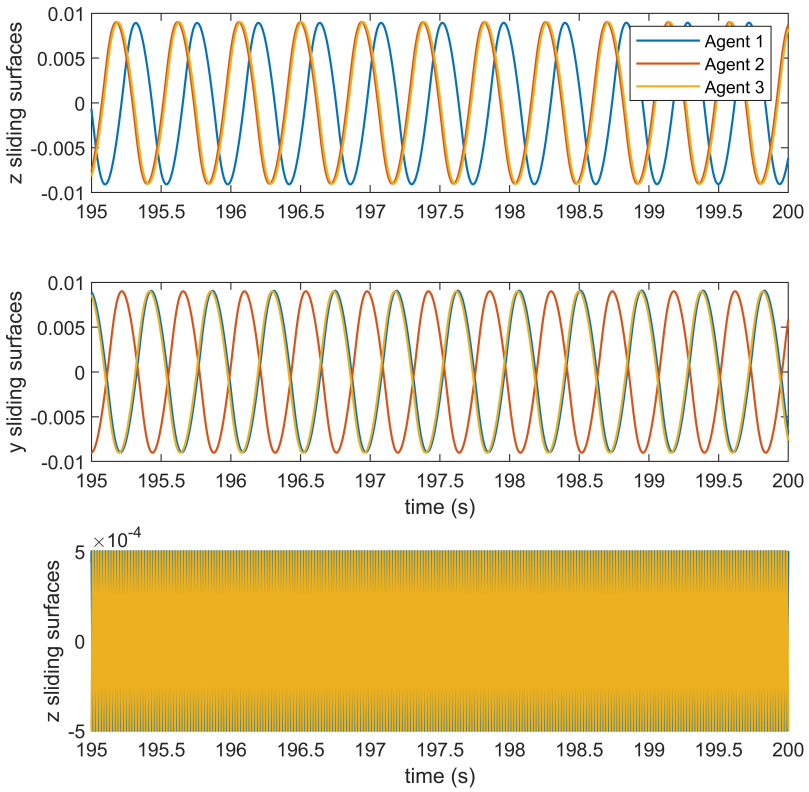


Figure 4.7: Sliding surfaces on the x, y, and z axes for each agent from time $t = 195s$ to $t = 200s$, displaying the transient behaviour of the system after convergence.

From Figure 4.7, it can be seen that under the designed DTSMC formation controller, the system experiences marginal stability after reaching the QSMB. After converging to around zero, the transient response of the controller shows rapid oscillations above and below the sliding surface, although the amplitude of these oscillations is small and is unlikely to cause dramatic chattering in the nominal case. To assess the feasibility of applying DTSMC formation control to a group of quadrotors experimentally, the control gains must also be analysed. Figure 4.8 shows the control inputs u_x^i, u_y^i, and u_z^i for each agent i in the system.

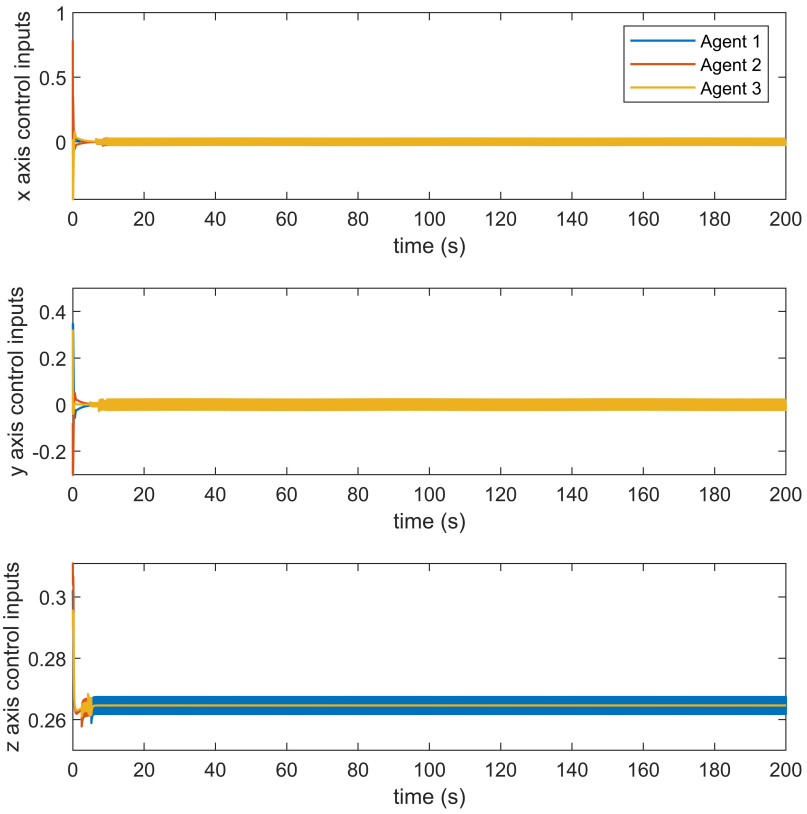


Figure 4.8: inputs u_x, u_y, and u_z for each agent.

From Figure 4.8, severe chattering can be seen again in the system. The oscillations here have a small amplitude and are within a reasonable range to be applied to an experimental system. However, it should be noted that these control inputs can could lead to damage to actuators of the quadrotors.

4.4.3 Simulation Results in the Presence of Time-Varying External Disturbances

The robust stability of the discrete-time sliding mode formation controller is further assessed by analysing its performance in the presence of simulated disturbances. The control system has been designed to reject matched disturbances applied to the system. In a real-world scenario, these matched disturbances can represent model uncertainties, load changes, unmodeled aerodynamic effects, or external wind

disturbances.

Wind disturbance is simulated as a matched time-varying input on the acceleration states in equation (4.5). Identical disturbance was applied to the x, y, and z axis for each individual agent. The disturbance applied to each quadrotor is calculated using

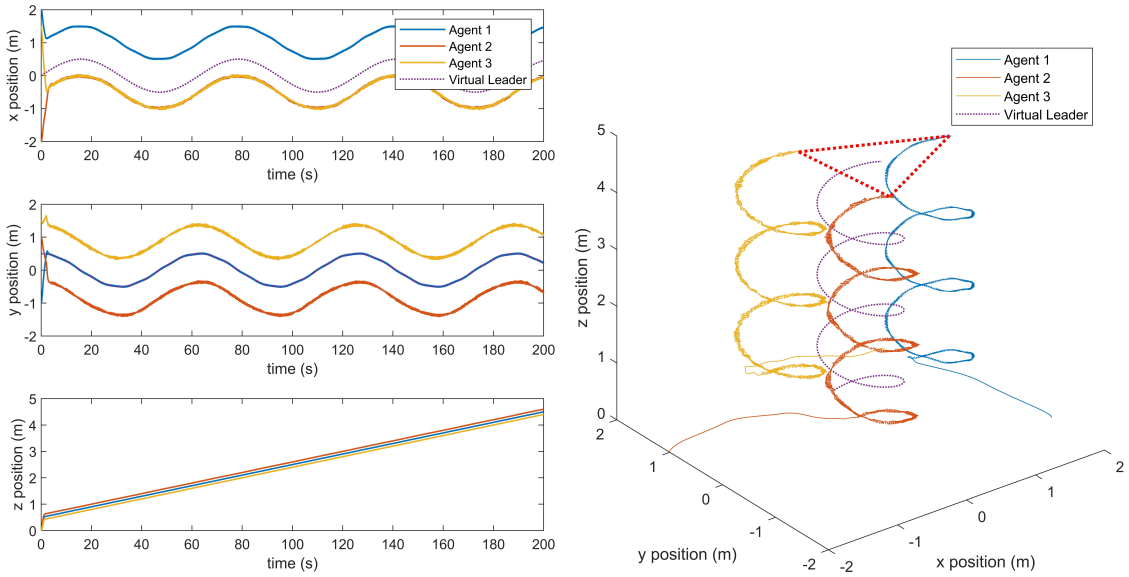
$$\delta_k = A\sin(\Omega_f kT), \tag{4.45}$$

where A is the amplitude of the disturbance, Ω_f is the frequency of the oscillations in rad/s, k is the sample, and T is the sample time in seconds.

Table 4.5: Disturbance parameters for each agent.

Agent	Amplitude A	Frequency Ω_f (rad/s)
Agent 1	0.5	0.7
Agent 2	1.0	0.3
Agent 3	2.0	0.5

The evolution of the positions of each drone over 200 seconds is provided in Figure 4.9.



(a) (b)

Figure 4.9: Trajectory response of the networked system following a virtual leader with a helical trajectory in the presence of simulated wind disturbance, with (a) displaying the separate x, y, and z response, and (b) displaying the three-dimensional trajectories.

Figure 4.9(a) demonstrates the effect of external disturbance on the position states of the quadrotors over the duration of the simulation. Each agent was able to maintain formation about the virtual leader in the presence of disturbances. This shows the efficacy of the control system for formation control in windy environments. Interestingly, from Figure 4.9(b), it can be seen that as the agents converge to the desired formation, the position states remain smooth. However, once the agents reach the desired formation, the position states appear noisy. This indicates the presence of chattering in the system. To further investigate the chattering present in the system, the desired and actual roll and pitch angles are plotted for Agent 1 in Figure 4.10.

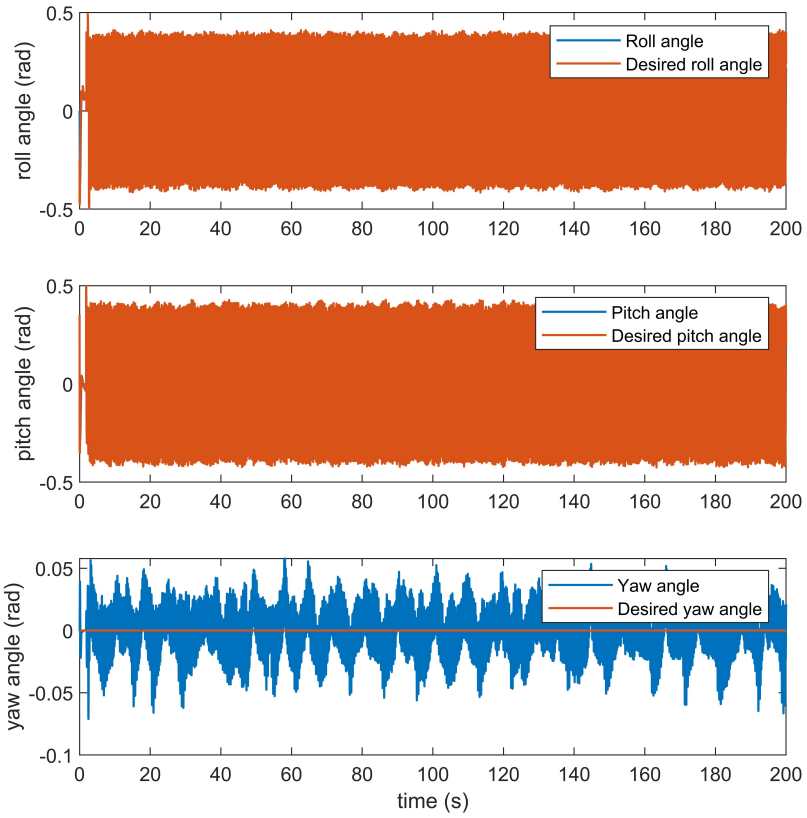
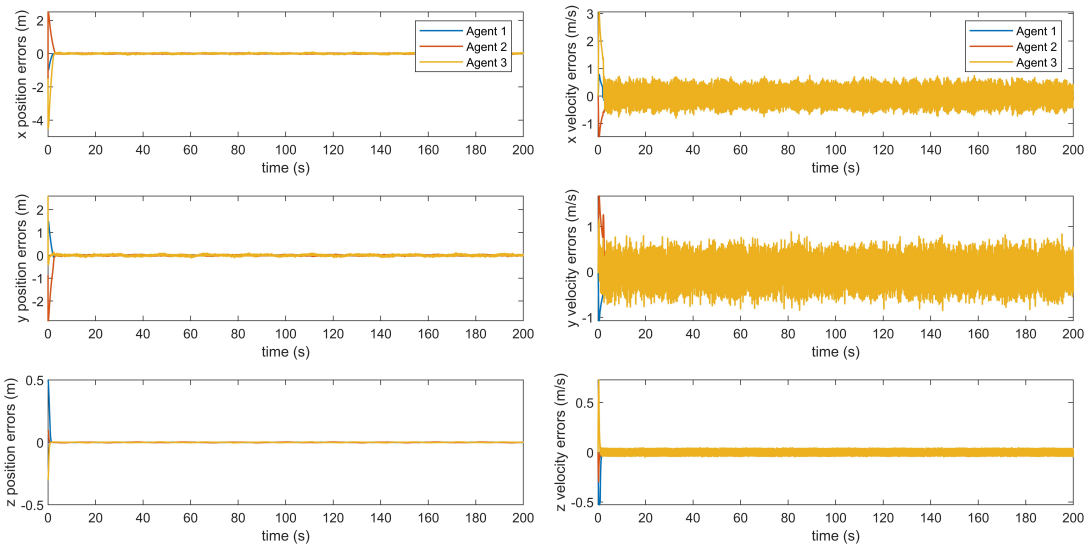


Figure 4.10: Plot of the actual and desired attitude of Agent 1 in the presence of simulated wind disturbance.

From Figure 4.10, excessive chattering is present in the control system, indicated by the thick band created by the desired roll angle. This larger band further shows the marginal stability of the system within the QSMB. Interestingly, although the yaw of each agent is not controlled by the DTSMC formation control system, it still shows degradation due to chattering. This may be impacted by the discrete-time implementation of the controller, as at each time step, a fixed control input is applied to each motor of each agent.

To assess the system's stability in the presence of external disturbances, the error values in equations (4.11) and (4.12) were plotted against time.



(a) (b)

Figure 4.11: Position (a) and velocity (b) error plots for each agent for the x, y, and z in the presence of simulated wind disturbance.

Despite the presence of disturbances, the error terms for positions and velocities are still able to converge towards zero, and remain around zero for the duration of the simulation. The position terms show some deviation in error, along with some oscillations. On the z axis, Agent 3 experiences a periodic divergence in the position error. This may be due to Agent 3 experiencing the largest amount of simulated disturbance. Furthermore, velocity errors experience large amounts of rapid oscillations, which are likely caused by chattering. Regardless, the errors do not diverge from zero for the duration of the simulation. From Definition 2, it can be seen that the system is stable if errors $\mathbf{e}_1$ and $\mathbf{e}_2$ converge to zero. This suggests that the system remains stable in the presence of external disturbances.

The sliding surfaces can also observed to assess the stability of the networked system in the presence of disturbance. The sliding surfaces of the system are shown in Figure 4.12.

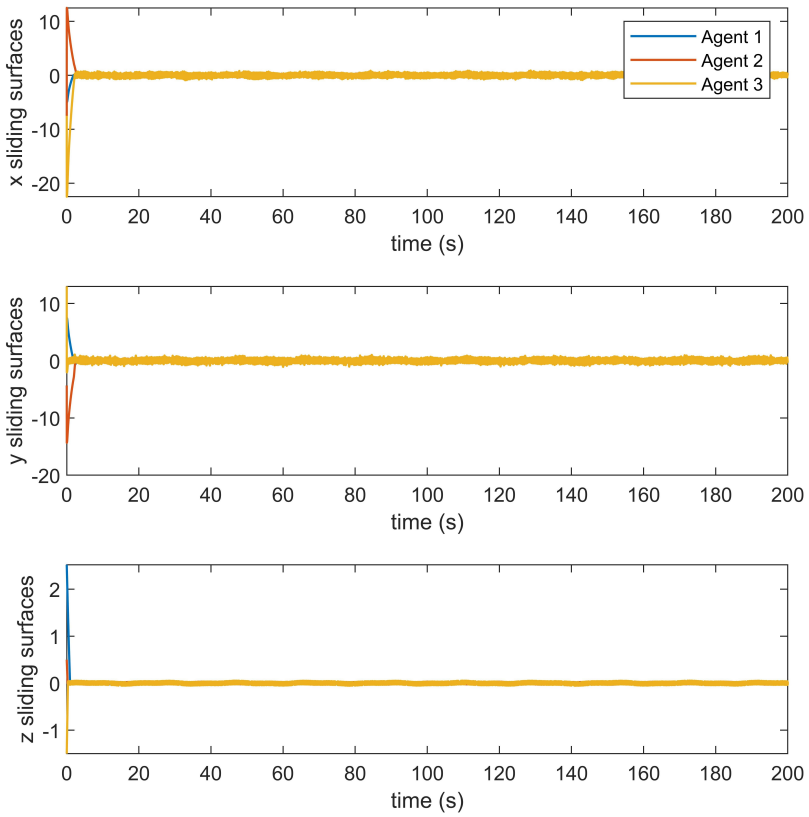


Figure 4.12: Sliding surfaces on the x, y, and z axes for each agent in the presence of simulated wind disturbance.

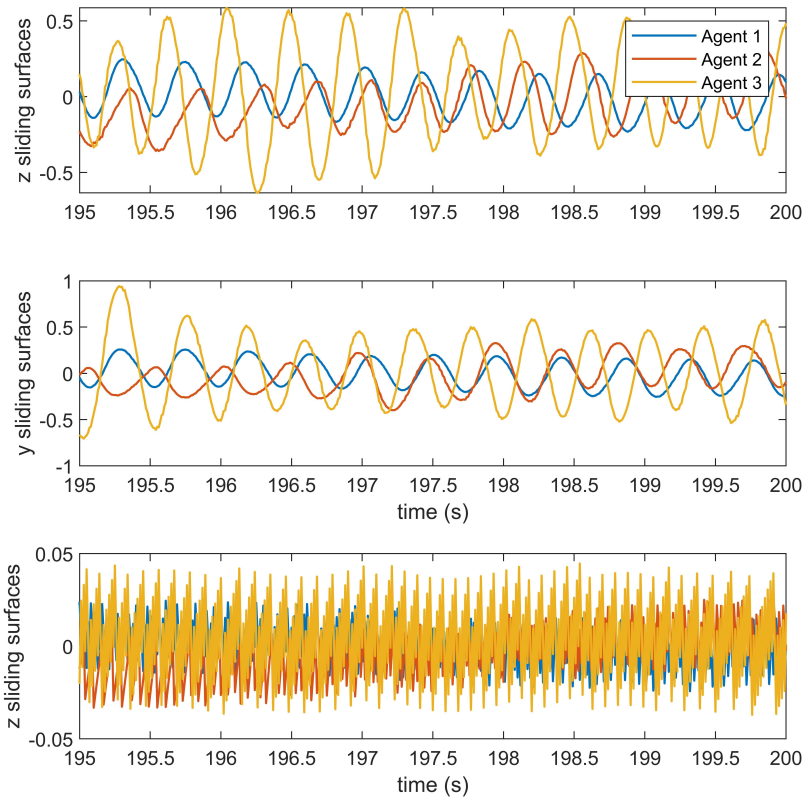


Figure 4.13: Sliding surfaces on the x, y, and z axes for each agent in the presence of simulated wind disturbance from time $t = 195s$ to $t = 200s$.

From Figures 4.12 and 4.13, the sliding surfaces converge to zero and rapidly oscillate within the QSMB. The plots show periodic divergences in Agent 3's z sliding surface due to the large disturbances applied. However, the controller returns to the sliding surface, rejecting the disturbance. This validates the claim that the designed DTSMC formation control system is robust and marginally stable in the presence of disturbances. The increased size of the QSMB is evident in Figure 4.13.

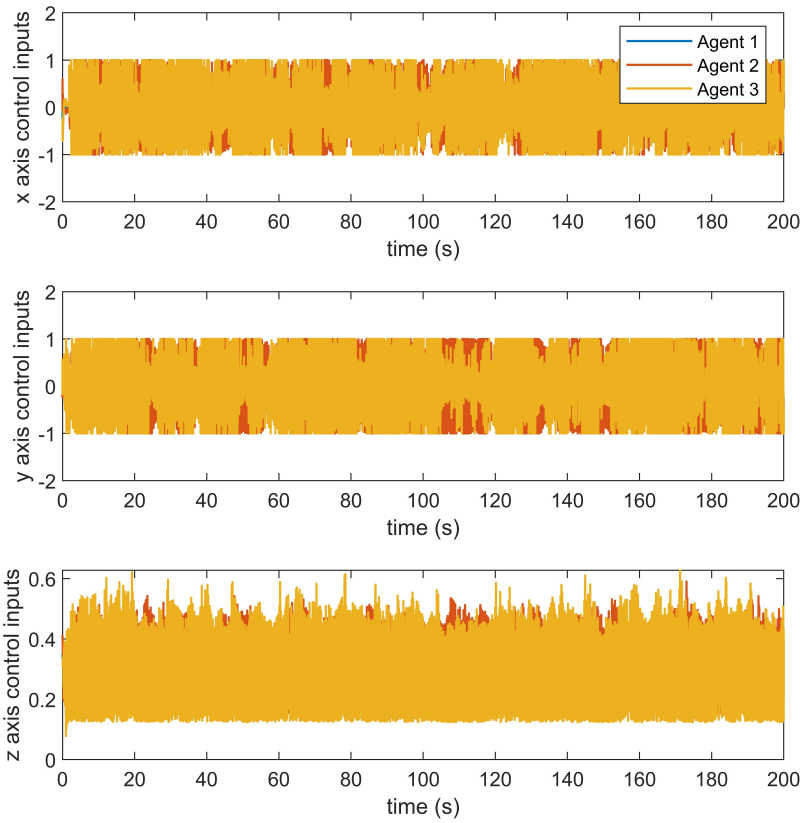


Figure 4.14: Control inputs u_x, u_y, and u_z for each agent in the presence of simulated wind disturbance.

From Figure 4.14, the increased chattering is clear when disturbances are applied to the system. While this works in simulation, real-world experimentation is required to further validate the designed formation control system's efficacy.

4.5 Experimental Validation

This chapter has shown that the designed DTSMC controller is robustly stable through a rigorous stability proof, as well as in simulation. However, the simulation raises some concerns surrounding the chattering phenomenon present in the control system. To further validate the designed control system, this section applies the controller to a real-world quadrotor swarm.

The Crazyflie 2.1 platform [136] was selected for real-world controller imple-

mentation due to the access to the open-source software available for programming the drones [137]. Additionally, an affordable external position system provides the swarm with the sensor data required to undergo formation control. For spatial positioning, each Crazyflie 2.1 was equipped with a Lighthouse Positioning Deck [138]. Each deck is equipped with 4 infra-red (IR) receivers. These allow each Crazyflie 2.1 to receive IR signals from 2 HTC Vive base stations to calculate their pose. Figure 4.15 shows the drones that were used to complete the real-world testing. Figure 4.16 shows one of the HTC Vive base stations placed in a corner of the test area. The lighthouse positioning system was found to be suitable for research applications, with a mean euclidean position error of 1cm during flight, with outliers up to 5cm [139].

Figure 4.15: Three Crazyflie 2.1 nano quadrotors, equipped with Lighthouse Positioning Decks, used for real-world testing.

Figure 4.16: Photograph of the HTC Vive IR base station place in the test environment.

A test environment was constructed at Lancaster University to fly the Crazyflie 2.1 network indoors safely. The testing environment consisted of an outdoor gazebo surrounded by safety netting. A small entrance was cut into the safety netting to allow access to the testing area. A ground station equipped with a radio transmitter and receiver was placed next to the testing area to allow communication with each Crazyflie 2.1 quadrotor. The test environment design is displayed in Figure 4.17, and the final setup is shown in Figure 4.18.

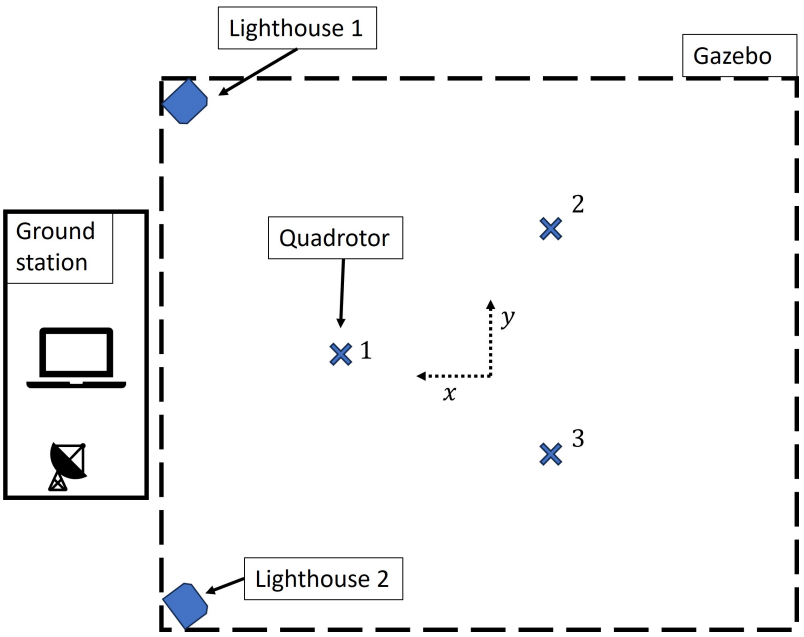


Figure 4.17: Diagram of formation control testing environment.

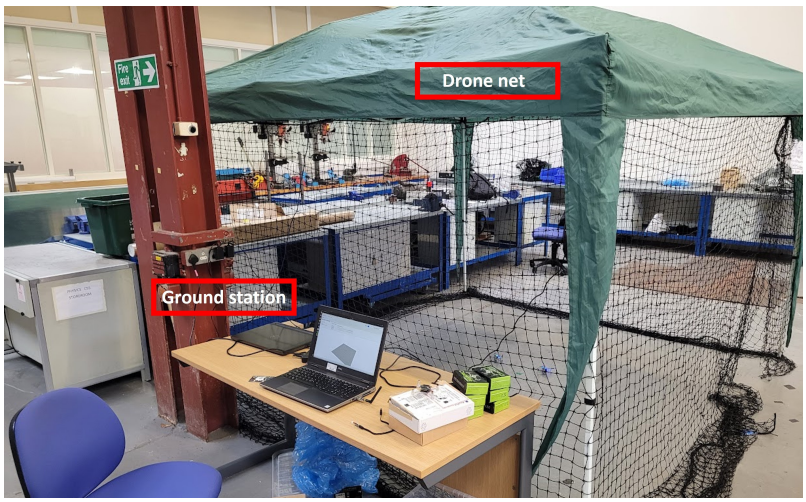


Figure 4.18: Formation control testing environment.

Due to radio communication constraints, only x and y position and velocity data were able to be shared between each Crazyflie 2.1 Agent. For this reason, the altitude of each drone was set to a fixed height during each test. The DTSMC formation control system was used to generate desired roll and pitch angles that were fed into the Crazyflie 2.1 internal PID controller.

4.5.1 Results Under Nominal Conditions

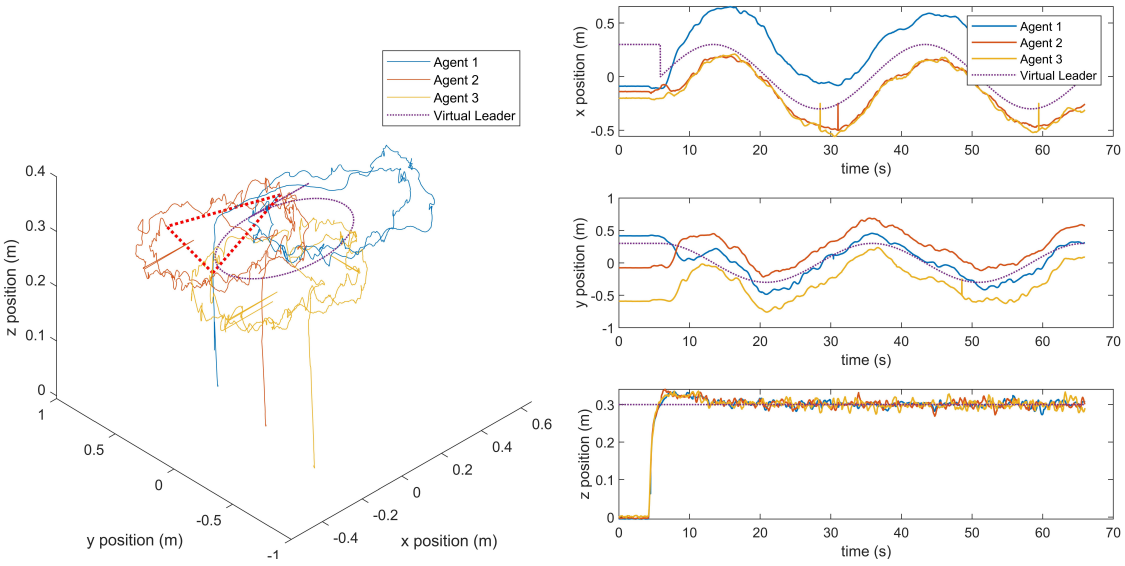
The system was first tested without the presence of external disturbances. However, the main difference between the simulation and the real-world implementation of the control system was the presence of sensor noise, unmodeled dynamics, parameter uncertainties, and external aerodynamic effects. These effects can significantly impact the system and can be considered external disturbances. Additionally, as the inner-loop attitude controller was not modified on the firmware, the default Crazyflie PID attitude controller was used to control the attitude of each agent.

The virtual leader was set to follow a circular trajectory about the origin at a fixed altitude. The trajectory is described as

$$\begin{cases} x_k^0 = 0.3\sin(4\pi(\frac{k-300}{3000})), \\ y_k^0 = 0.3\sin(4\pi(\frac{k-300}{3000})), \\ z_k^0 = 0.3, \end{cases} \tag{4.46}$$

where k is the current sample number. As we want to start the trajectory at sample $k = 300$, this is removed from the value of k. To retain consistency with the simulation, the same communication strategy from Figures 4.2 is implemented, where agents only pass information down the directed spanning tree.

Figure 4.19(a) shows the three-dimensional trajectory of the quadrotors in space, while Figure 4.19(b) shows the evolution of each agent's trajectory alongside the virtual leader over time t.



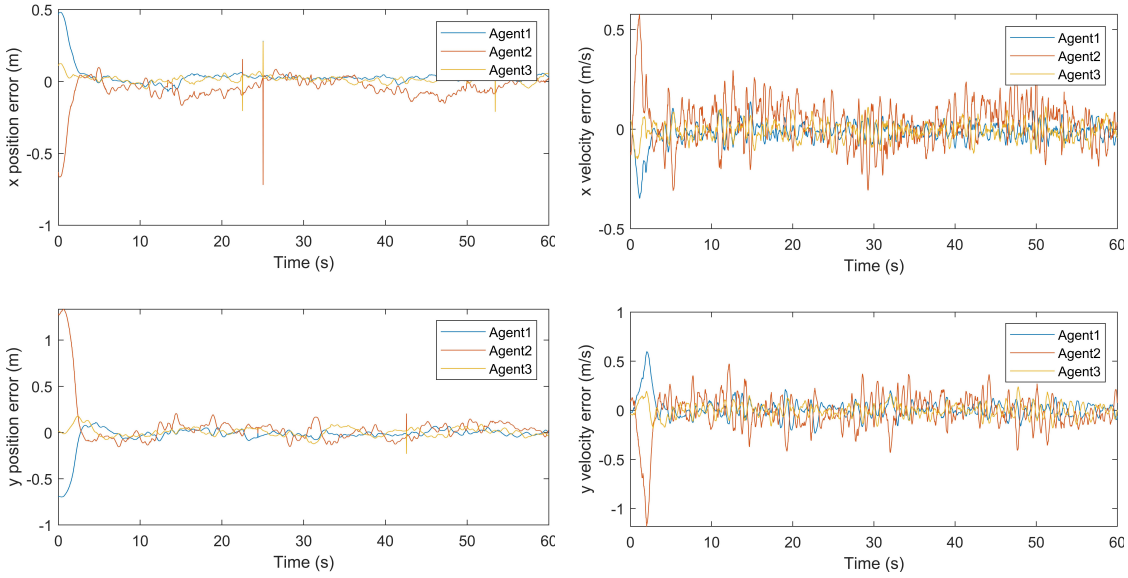
(a) (b)

Figure 4.19: (a) A plot showing the three-dimensional trajectory of each agent alongside the virtual leader and (b) a plot showing the evolution of each agent's trajectory over time t.

From Figure 4.19(a), the agents are shown to achieve formation after takeoff, and maintain this formation around the virtual leader. The figure displays each agent's path in space, clearly showing the successful navigation of the circular trajectory around the leader. The red line indicates the separation between each agent at the final sample time. While the trajectory shows larger deviations and more noise when compared with the simulation results, it is clear from the plot that the agents were able to maintain the formation and did not diverge.

Figure 4.19(b) displays the position response of each agent over the time of the flight. From the plots, it is clear that the agents converge to the desired formation and remain in formation for the duration of the flight. Interestingly, Agent 2 and Agent 3 maintain the desired separation from Agent 1. However, Agent 1 experiences deviations in the desired separation from the virtual leader. This suggests that while the desired controller can maintain the desired formation, further tuning may be required to improve the tracking of the virtual leader. It

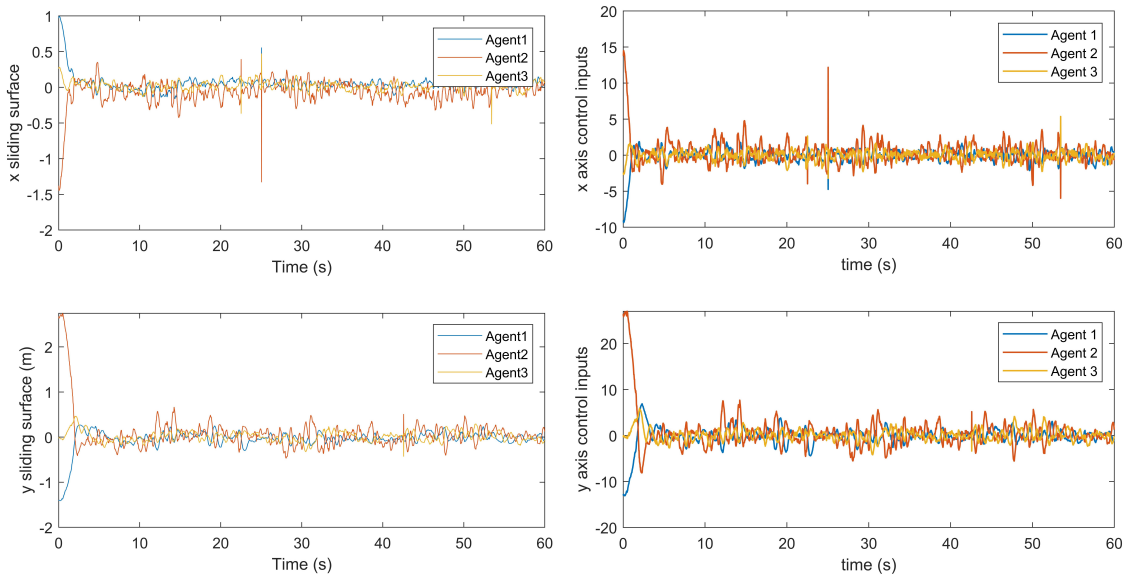
should also be noted that, as Agent 1 is the only agent connected to the virtual leader, the tracking of the formation may be further improved by experimenting with alternative communication topology.



(a) (b)

Figure 4.20: Position (a) and velocity (b) errors for each agent in the formation during real-world testing.

Figure 4.20 shows the position and velocity errors of each agent during the flight. The position and velocity errors converge towards zero before time $t = 5s$ and remain around zero for full flight duration, demonstrating the stability of the system in a real-world scenario. A sharp spike is present in the position error at time $t = 25s$ due to erroneous data from the positioning system. Omitting this data, the position error does not exceed 0.16m for any agent in the system after converging. When observing the velocity errors in Figure 4.20(b), the system displays larger oscillations in error. The behaviour here may be due to chattering. In order to investigate this further, the sliding surfaces and control inputs for each agent are plotted in Figure 4.21



(a) (b)

Figure 4.21: (a) A plot displaying the sliding surfaces and (b) a plot displaying the control inputs along each axis for each agent in the formation during real-world testing.

Figure 4.21(a) shows that each agent's sliding surface converges around the QSMB for the x, y and z axis. This demonstrated the marginal stability of the system in the real-world case. The sliding surfaces rapidly oscillate around a zero value, displaying the chattering behaviour present when implementing DTSMC. Regardless, this shows the stability of the system and further validates the presented simulation results. The u_x, u_y, and u_z control inputs for each agent in the system are shown in Figure 4.21(b). These further display the presence of potential chattering in the real-world implementation of the DTSMC formation controller. Despite the chattering, the quadrotors remained stable during the flight, and there was no obvious degradation in control due to the chattering present.

4.5.2 Results in the Presence of Disturbance with a Static Virtual Leader

This section presents the results obtained through real-world experimental testing of the robust discrete-time sliding mode formation control system in the presence of applied wind disturbance under a static leader position. The aim of these tests is to evaluate the efficacy and robustness of the control system under environmental conditions that simulate moderate to severe wind disturbances. The reason for first testing the formation under a static leader position is to introduce complexity into the system gradually. By initially keeping the virtual leader static, the effects of the applied external wind disturbances on the system can be analysed in isolation.

An oscillating fan was introduced into the test environment to provide the system with a time-varying external wind disturbance. A diagram for this is shown in Figure 4.22, and a photograph of the location of this fan is provided in Figure 4.23.

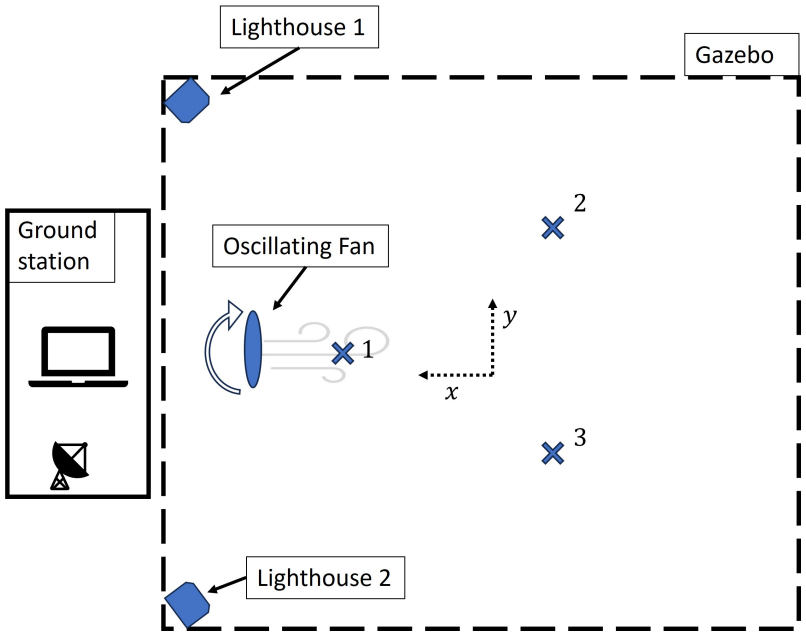
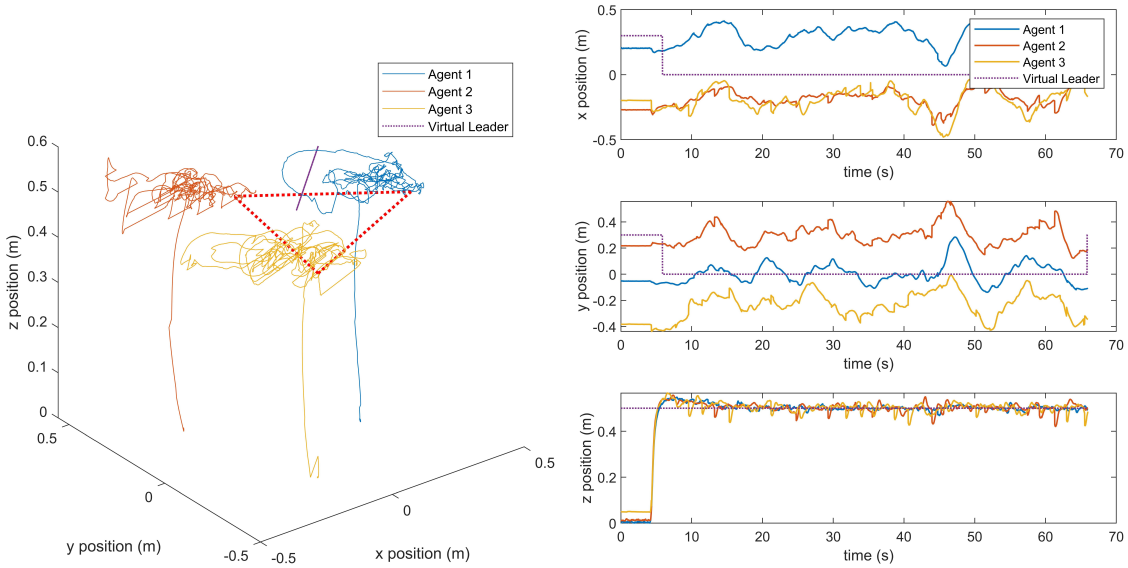


Figure 4.22: A modified diagram of formation control testing environment with the location of an oscillating fan.

Figure 4.23: A photograph of the test area with an oscillating fan.

Figure 4.24 displays the position response of each agent in the system under discrete-time sliding mode formation control.

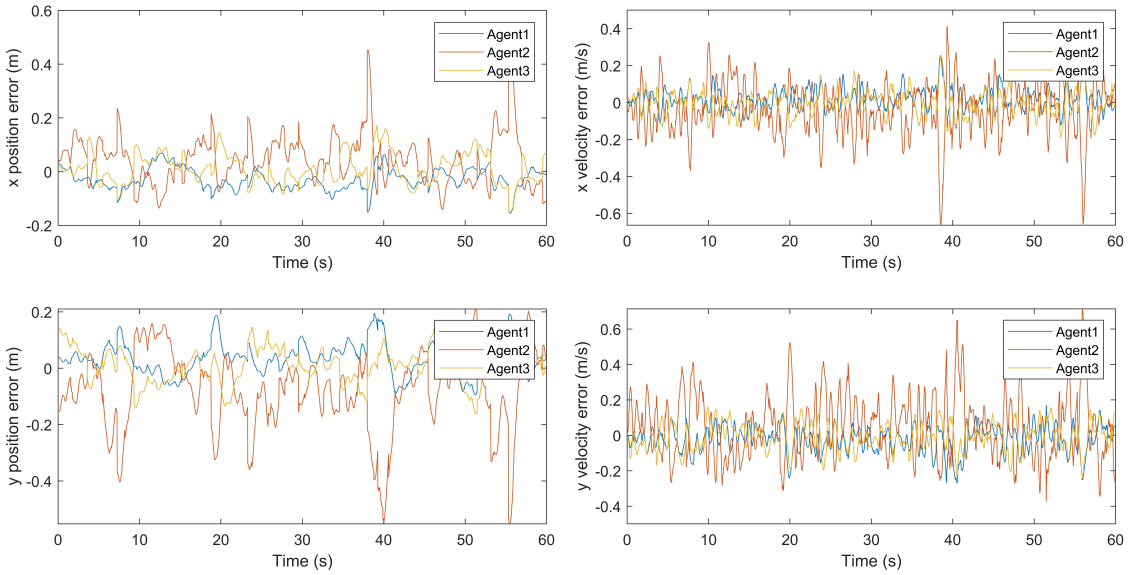


(a) (b)

Figure 4.24: (a) A plot showing the three-dimensional trajectory of each agent alongside the virtual leader and (b) a plot showing the evolution of each agent's trajectory over time t in the presence of disturbance with a static virtual leader.

Figure 4.24(a) illustrates the three-dimensional trajectory of the the quadrotors, while Figure 4.24(b) displays the evolution of each agent position on the x, y, and z axis over time. From the plots, it is evident that there is a deviation in positions relative to the virtual leader. Interestingly, despite the deviation from the virtual leaders, from Figure 4.24(b), it is evident that the agents maintain their separation. All agents diverge from the virtual leader in a synchronised manner. This suggests that while the formation is robust under external wind disturbances, further tuning of the controller may improve the system's tracking of the virtual leader. This may also be improved by providing more agents with access to the states of the virtual leader. These visualisations confirm the effectiveness of the control strategy in the case of a static leader and also identify areas that have potential for further improvement in future works. To further analyse the robustness of the system, the errors for each agent are shown in Figure 4.25 displays the evolution of the system

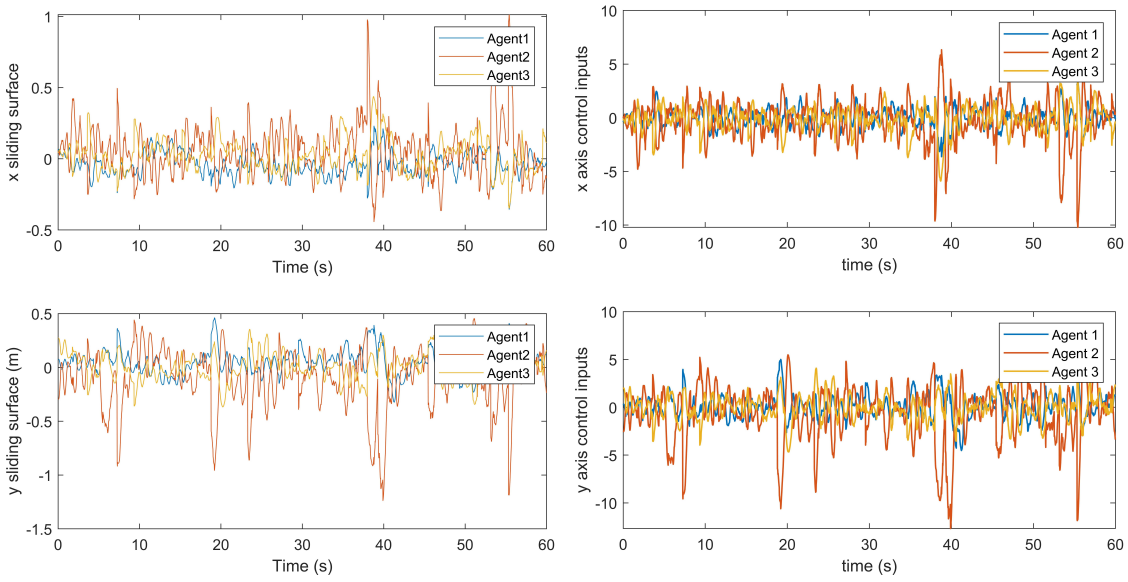
errors over time t.



(a) (b)

Figure 4.25: Position (a) and velocity (b) errors for each agent in the formation during real-world testing in the presence of wind disturbance with a static virtual leader.

The position errors shown in Figure 4.25(a) display good tracking of positions over time. The errors are shown to occasionally diverge from the virtual leader. After the original divergence under the disturbance, they once again converge towards zero error, further validating the robustness of the system. From the velocity errors, displayed in Figure 4.25(b), the presence of rapid oscillatory behaviour in the system becomes evident. This shows the continuation of the chattering behaviour found in the previous results. Figure 4.29 displays the system sliding surfaces and control inputs.



(a) (b)

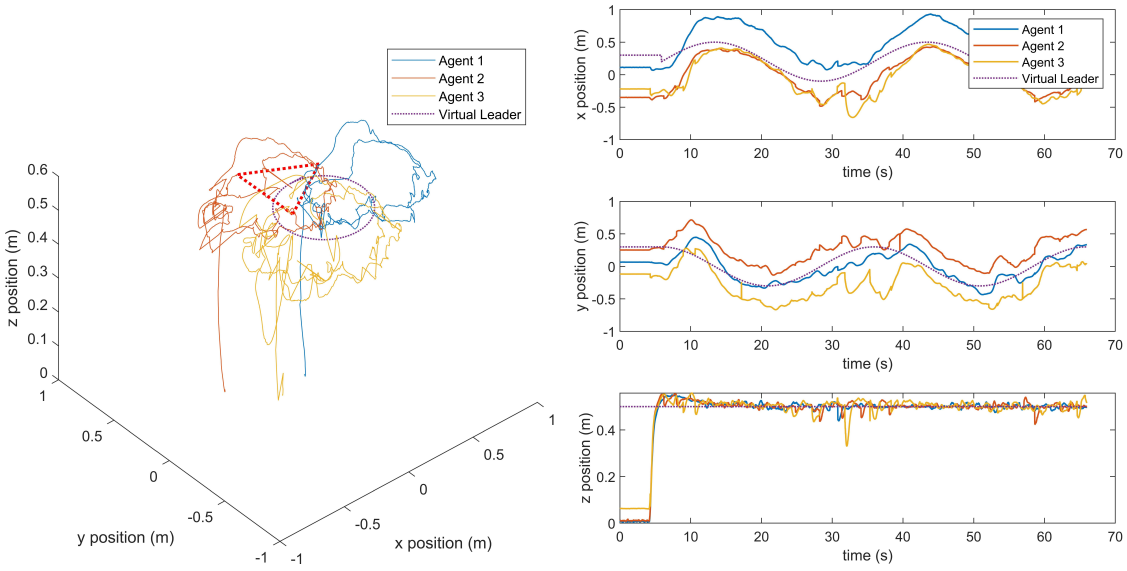
Figure 4.26: (a) A plot displaying the sliding surfaces and (b) a plot displaying the control inputs along each axis for each agent in the formation during real-world testing in the presence of wind disturbance with a static virtual leader.

The stability of the system becomes clear in Figure 4.26(a), with the sliding surfaces for the x and y rapidly oscillating around a zero value. The presence of disturbances causes the x and y sliding surfaces to occasionally diverge. After the divergence, they once again converge to the QSMB around the sliding surface. The chattering of the control input under these conditions is evident in the control inputs shown in Figure 4.26(b).

4.5.3 Results in the Presence of Disturbance with a Dynamic Virtual Leader

Following the tests implemented in the previous section, the agents were tasked with tracking a dynamic virtual leader in the presence of an applied wind disturbance. The aim of these tests was to validate the robust trajectory tracking performance of the multi-agent system. Validation of performance under this scenario would enable

a group of agents to navigate unanimously, enabling the system's data collection abilities for future work. Figure 4.27 displays the position response of each agent in the system when tracking a dynamic leader in the presence of applied external wind disturbances.

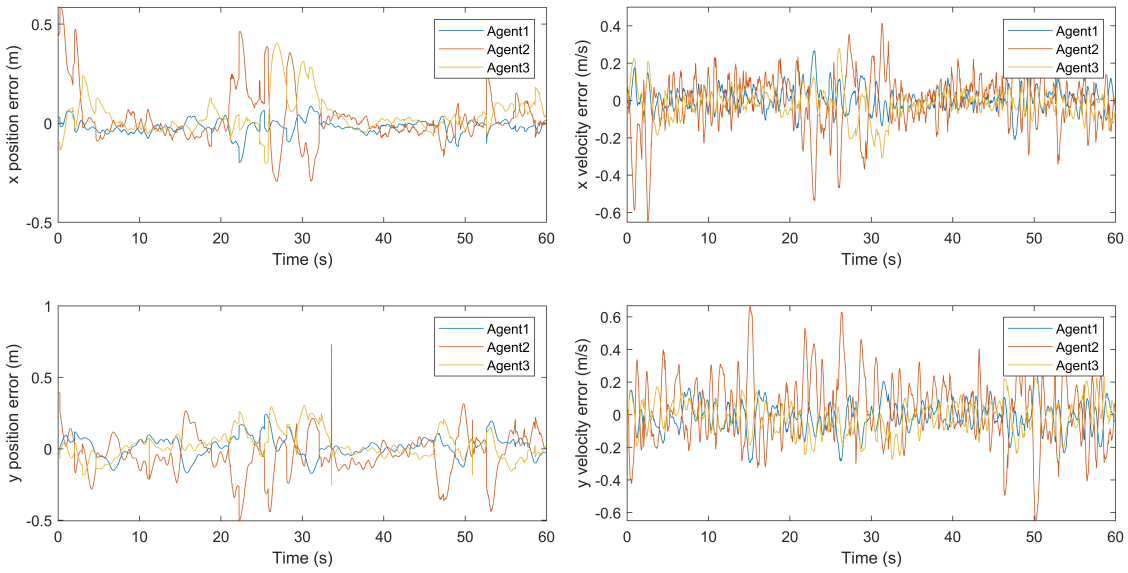


(a) (b)

Figure 4.27: (a) A plot showing the three-dimensional trajectory of each agent alongside the virtual leader and (b) a plot showing the evolution of each agent's trajectory over time t in the presence of wind disturbance with a dynamic virtual leader.

The agents achieve altitude and converge to the desired formation around the leader. From Figure 4.28(a) it is clear that there is a decrease in performance compared to Figure 4.19(a). This shows the negative effects of the external disturbance on the system. Regardless, the system maintains its formation around the virtual leader with no collisions. Similarly to Figure 4.24(b), Figure 4.28(b) demonstrates that the agents are able to maintain the desired separation despite the presence of the external disturbance. The main source of errors in the system is Agent 1's ability to track the virtual leader. This further enforces the belief that

additional tuning could improve the tracking capabilities of the system, in addition to the introduction of an alternative communication topology, where an increased number of agents is connected to the virtual leader.

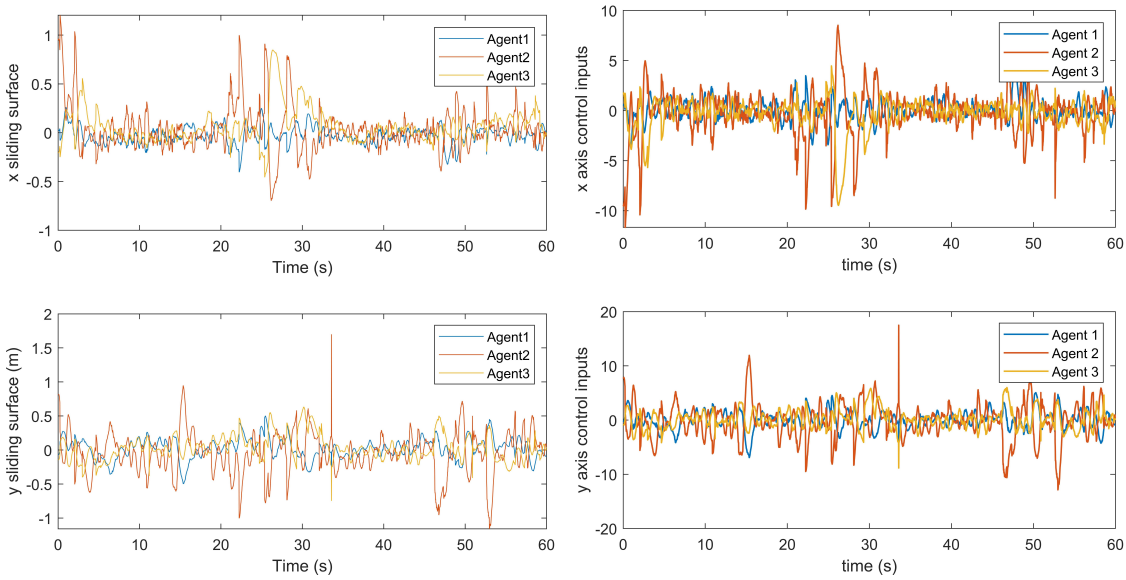


(a) (b)

Figure 4.28: Position (a) and velocity (b) errors for each agent in the formation during real-world testing in the presence of wind disturbance with a dynamic virtual leader.

Analysis of the system's position and velocity errors in Figure 4.28 display similar results to those shown in Figure 4.25. This suggests that the performance of the discrete-time sliding mode formation control system does not degrade further through the introduction of a dynamic leader. This is promising as it demonstrates the system's ability to track complex trajectories in the presence of external disturbances. The evidence from these tests provides additional validation of the rigorous proof provided in Section 4.3 and the simulation results in Section 4.4. According to definition 2, the real-world implementation of the system is considered robustly stable. Finally, in order to investigate the chattering response of the system when tracking a dynamic leader under external disturbances, the sliding surfaces and

control inputs are displayed in Figure 4.29.



(a) (b)

Figure 4.29: (a) A plot displaying the sliding surfaces and (b) a plot displaying the control inputs along each axis for each agent in the formation during real-world testing in the presence of wind disturbance with a dynamic virtual leader.

Compared to Figures 4.26(a) and 4.26(b), there is not a dramatic increase in the amplitude of the oscillations around a zero value. This evidences the efficacy of the designed multi-agent system when tracking a dynamic virtual leader in the presence of external unknown disturbances and uncertainties. The system's stability in these conditions is evidenced in Figure 4.26(a), as the sliding surfaces do not diverge dramatically from zero value and converge to the system's QSMB. Despite the rapid oscillatory behaviour of the system's control inputs, the system was able to maintain formation around the virtual leader, and the system's sliding surfaces did not diverge dramatically.

4.6 Conclusion

In this chapter, a robust control system was proposed for the formation control of a group of quadrotor agents. The proposed control system, based on discrete-time sliding mode control, allows a group of agents to obtain a desired formation around a virtual leader and track a dynamic virtual leader following a complex trajectory. The chapter provides a rigorous stability proof, theorising that the system is stable in the presence of unknown, time-varying bounded disturbances. The stability proof also shows that the system is capable of maintaining its formation under communication constraints, where agents only have access to a limited number of neighbouring agent's states. Additionally, it is shown that only a single agent in the system requires access to the states of the virtual leader, potentially making implementation with a ground station more accessible. Finally, the stability proof provides evidence that the system is stable under time-varying any communication graph that has a directed spanning tree. This allows reconfiguration of the communication topology in real-time, improving robustness to communication failures.

To validate the performance of the controller and support the stability proof, the chapter provides a numerical simulation displaying ideal performance in both the nominal case and under unknown disturbances. During the project, the simulations provided an environment to rapidly prototype and tune the proposed control system.

The simulation section also provides useful insights into the chattering behaviour of the system under discrete-time sliding mode formation control. From the results, it is clear that chattering can cause large issues in networked systems, particularly where control signals are shared between quadrotors over a network. The chattering becomes significantly worse when the quadrotors are placed under external matched disturbances, such as wind. In particular, the chattering in simulation under external disturbances causes the x, y control inputs to display large oscillations. If left unaddressed, this could lead to the mechanical failure of the quadrotor actuators over time. To help remedy this, the *signum* function in equation (4.15) can be replaced with the *tanh* as a continuous approximation. The chattering issues likely

contribute to the real-world performance degradation present in the experimental results. From the results, the altitude of the quadrotor experiences high frequency oscillations of a fixed magnitude, which can be caused by chattering control inputs.

It is also evident from the results that there exists a large gap between simulation and real-world experimentation. This is a common problem in control system design and there are many potential reasons for this. Primarily, the simulations provided in this chapter do not employ estimation methods for feeding back the state of each agent, while real-world results employ various sensors and filters for estimating the pose of each quadrotor. The control system can be greatly impacted by the localisation accuracy of the estimation system, as inaccuracies can lead to degradation in controller performance. Additionally, latency can be present in these estimation systems that is not accounted for in the control system design. Another factor present in real-world experimentation that is not captured by simulation is time-varying communication delay between each agent. Finally, the real-world system also contains many unmodeled dynamics, such as propeller characteristics, aerodynamic interference, motor dynamics, chassis vibrations, and sensor noise.

To further evidence the efficacy of the designed control system for use in complex and hazardous indoor nuclear environments, an environment was constructed for real-world testing of the system. This provided a controlled environment for integration and testing of the designed control system. Real-world testing of the multi-quadrotor system validated the efficacy of the system for use in these safety-critical nuclear environments, where errors could cause less robust control systems to fail, leading to potentially catastrophic consequences. Remarkably, the system was also able to achieve this formation under communication constraints, where each agent only had access to information from its direct neighbours. Furthermore, the application of an external disturbance source during real-world testing demonstrates the robustness of the system. The real-world experimentation also provides insight into the potential improvements that can be made to the control system by adjusting the communication topology or introducing additional gains into the system to

improve tracking of the virtual leader.

To summarise, this chapter has successfully demonstrated the design, proof, validation, and experimental verification of a discrete-time sliding mode controller for the robust formation control of a multi-quadrotor system. The chapter provides a promising solution for use in a mobile sensor network in safety-critical complex nuclear environments by demonstrating robustness to communication constraints and external wind disturbances. Future works could aim to improve on the results in this chapter through optimisation of the control gains, experimentation with alternative communication topologies, and additional real-world testing in larger environments. While this chapter proposes a robust control law for controlling a formation of quadrotors, the following chapter applies the principles developed in this chapter to more complex tasks within the nuclear industry. Taking the concept of multi-agent systems, and applying them to tasks such as source seeking and environmental characterisation within the nuclear industry, could allow for far more efficient characterisation of nuclear legacy sites.

Chapter 5

Source Seeking with Applications in the Nuclear Industry

5.1 Introduction

The previous chapters in this thesis have developed and validated the efficacy of robust methods for controlling single quadrotors and networked multi-quadrotor systems. The thesis has also identified several problems in the nuclear industry, including condition monitoring, inspection, and plant characterisation. Building upon this work, this chapter investigates the practical applications of such control algorithms in the nuclear industry.

This chapter aims to apply the previously developed robust control algorithms to address the outlined challenges, specifically in the context of locating the source of an environmental scalar field, such as temperature or radiation. By applying these robust control algorithms, we aim to show their efficacy for applications in the nuclear industry. One case study in which this is particularly applicable is radiation monitoring within high-dose areas of nuclear legacy sites. This has been addressed with a single quadrotor in [12], where a remotely operated quadrotor, equipped with a radiation sensor and radiation mapping software was used to acquire accurate contamination distributions in a historical nuclear facility. This allowed

decommissioning to progress while reducing the dose to operators.

The implementation of a gradient climbing virtual leader for source-seeking is also investigated. Specifically, this chapter explores the discrete-time sliding mode formation control algorithms to gradient climbing applications for source-seeking in the nuclear industry. The technique steers a group of agents towards the field's peak to find the highest radiation intensity levels in an environment. The motivation behind this is to allow an operator of a multi-quadrotor network to pinpoint the location of a source in temperature or radiation sensor fields. Traditional methods rely on the involvement of human workers and can be time-consuming and costly. More recently, ground robots have been used to map radiation within nuclear environments [140], although these face issues where robots cannot access parts of a facility due to obstacles. Implementing a source-seeking algorithm for the multi-quadrotor system would dramatically improve the plant characterisation process by increasing the safety and efficiency of the operation.

Radiation sources are known to have an inversely proportional relationship with the distance from the radiation source [141], [142]. The dose rate from the distance of the source is defined as

$$I(r) = \frac{I^0}{r^2}. \tag{5.1}$$

Here, $r \in \mathbb{R}^+$ is the radius from the source in meters. The function $I(r)$, returns the intensity of the radiation at a distance r from the source. $I^0 \in \mathbb{R}^+$ is the intensity of the radiation at the source. Additionally, radiation may be blocked or attenuated by obstacles presence in the environment [141], [142]. However, this chapter does not consider the obstacles in the environment and assumes a direct line of sight to the radiation source. According to the Office for Nuclear Regulations, the maximum permitted dose rate at the surface for transporting a nuclear package is 10 mSvhr^{-1} [143]. This will be used as the maximum value at the source of the scalar field I^0.

In addition to this, background radiation in nuclear sites for the years 2022/23 was found to be approximately 2×10^{-5} mSvhr^{-1} according to the Sellafield annual

review [144]. Adjusting equation (5.1) to account for this gives

$$I(r) = \frac{I^0}{r^2} + I^b \tag{5.2}$$

where $I^b \in \mathbf{R}^+$ represents the levels of background radiation present.

The remainder of this chapter presents a robust source-seeking formation control algorithm by building on the discrete-time formation control algorithm presented in Chapter 4. The source-seeking algorithm is presented for locating the peak of a radiative sensor field in the presence of wind disturbance. In addition to this, to provide an operator with an appropriate radiation map, the application of Gaussian Process Regression is validated in experimental work using a temperature source in place of a source of gamma radiation, as thermal radiation similarly follows the inverse-square law [145].

5.1.1 Source-seeking in the Presence of External Disturbances

The discrete-time sliding mode control law presented in Chapter 4 implements a virtual leader at the centre of the formation to allow the quadrotor swarm to navigate towards a set point or to follow a predefined trajectory. To build on this work, this chapter proposes a solution to the source-seeking problem in which the position of the virtual leader is updated to steer the formation of quadrotors towards the peak of an environmental field, such as a radiative or temperature sensor field.

To estimate the environmental field's gradient at the formation's centre based on information from i agents, the virtual leader can combine the measurements and positions from each agent. The position of each agent at time step k is x_k^i and y_k^i. A measurement from each agent is denoted by ζ_k^i for $i = 1, 2, \ldots, n$. The centre of the circular formation is represented by the virtual leader position and is given by (x_k^0, y_k^0). The gradient of the field at the centre, $\nabla\zeta_k = \left(\frac{\partial\zeta}{\partial x}, \frac{\partial\zeta}{\partial y}\right)$, can be estimated using linear algebra. An assumption is made that the gradient is approximately linear in the formation region. The change in the field value ζ between the centre

and each agent i can be approximated by the dot product of the gradient and the difference in position.

$$\zeta_k^i - \hat{\zeta}_k^0 \approx \nabla\zeta \cdot ((x_k^i - x_k^0), (y_k^i - y_k^0)), \tag{5.3}$$

where $\hat{\zeta}_k^0$ is the estimated measurement value at position of the virtual leader ζ_k^0 and can be approximated using

$$\hat{\zeta}_k^0 = \frac{1}{n}\sum_i^n (\zeta_k^i). \tag{5.4}$$

Expanding (5.3) for n agents gives

$$\boldsymbol{\Delta\eta_k} \begin{bmatrix} \frac{\partial\zeta}{\partial x} \\ \frac{\partial\zeta}{\partial y} \end{bmatrix} = \begin{bmatrix} \zeta_k^1 - \zeta_k^0 \\ \zeta_k^2 - \zeta_k^0 \\ \vdots \\ \zeta_k^n - \zeta_k^0 \end{bmatrix}, \tag{5.5}$$

where

$$\boldsymbol{\Delta\eta_k} = \begin{bmatrix} (x_k^1 - x_k^0) & (y_k^1 - y_k^0) \\ (x_k^2 - x_k^0 & (y_k^2 - y_k^0) \\ \vdots & \vdots \\ (x_k^n - x_k^0) & (y_k^n - y_k^0). \end{bmatrix} \tag{5.6}$$

By taking the pseudo inverse of $\boldsymbol{\Delta\eta_k}$ in equation (5.5), $\nabla\zeta_k$ can be isolated.

$$\nabla\zeta_k = (\boldsymbol{\Delta\eta_k}^\top \boldsymbol{\Delta\eta_k})^{-1} \boldsymbol{\Delta\eta_k}^\top \begin{bmatrix} \zeta_k^1 - \zeta_k^0 \\ \zeta_k^2 - \zeta_k^0 \\ \vdots \\ \zeta_k^n - \zeta_k^0 \end{bmatrix} \tag{5.7}$$

To steer the virtual leader's position towards the field's peak, the leader can navigate toward an increasing gradient. To achieve this in practice, the virtual leader's position is updated using a value $K_v \in \mathbb{R}^+$, which is a set velocity in ms^{-1}, and the unit vector of the gradient $\nabla\zeta_k$.

$$\boldsymbol{\eta}_{k+1}^0 = \boldsymbol{\eta}_k^0 + TK_v \frac{\nabla\zeta_k}{|\nabla\zeta_k|}, \tag{5.8}$$

where $\boldsymbol{\eta}_k^0 = [x_k^0 y_k^0]^\top$, $\boldsymbol{\eta}_k^0 \in \mathbb{R}^2$ is a vector containing the x and y positions of the virtual leader at sample k. $\boldsymbol{\eta}_k^0$ is the position of the virtual leader at sample $k+1$ and $T \in \mathbb{R}^+$ is the sample time of the discrete source-seeking control algorithm. The initial position of each quadrotor and the position of the simulated gamma source in the environment are shown in Table 5.1.

Table 5.1: Table showing the initial positions of each agent and the position of a simulated gamma radiation source.

	x position (m)	y position (m)
Gamma radiation source	15.0	7.5
Initial Agent 1 position	2.0	-1.0
Initial Agent 2 position	-2.0	1.0
Initial Agent 3 position	1.5	1.5

A disturbance representing wind is applied to each quadrotor using equation (4.45) to validate the system's robustness further. The amplitude and frequency of the disturbances are provided in Table 5.2.

Table 5.2: Table showing the amplitude and frequency of the disturbance applied to each agent.

Agent	Amplitude A	Frequency Ω_f (rad/s)
Agent 1	0.3	0.7
Agent 2	0.7	0.3
Agent 3	0.5	0.5

From Table 5.2, it can be seen that external disturbances with different amplitudes and frequencies are applied to each agent in the system. This allows many disturbance combinations to more rigorously test the system's robustness.

In the simulations, an assumption is made that the virtual leader can access the sensor samples from a location from each quadrotor in the system. This can be achieved by implementing an undirected graph network topology, where the x and y positions, along with a sensor measurement from each quadrotor, are passed back to the virtual leader. Figure 5.1 shows the topology implemented in the simulations.

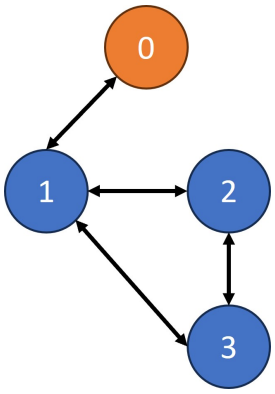


Figure 5.1: Graph representing the communication topology between each agent and the virtual leader used for the simulations.

While the graph shows that each agent can communicate in both directions, only the state information required for discrete-time sliding mode formation control is passed in one direction through a subgraph containing a directed spanning tree. Only sensor measurements and position states are returned to the virtual leader in the opposite direction. Therefore, the system works in any network graph containing at least one subgraph with a directed spanning tree, similar to the one shown in 4.2. It should also be noted that only Agent 1 maintains a connection with the virtual leader in the simulations. This provides a system capable of handling switching communication topologies, increasing the system's robustness.

Through this approach, the virtual leader can access the required position states and sensor data to compute the gradient's direction at the formation's centre. To evaluate the performance of the navigation algorithm developed in section 5.1.1 a gamma radiation source is placed in a simulation environment illustrated in Fig. 5.1 with a value of 10 mSvhr^{-1}.

Figure 5.2 displays an example block diagram of the data communicated in the source-seeking system. In this diagram, the virtual leader can be one of the agents in the system or an external ground station. In this diagram, each agent runs the DTSMC formation control algorithm to create the source-seeking formation. The temperature data is passed to the virtual leader. The virtual leader then updates its own position information to allow the formation to navigate towards the peak of the sensor field.

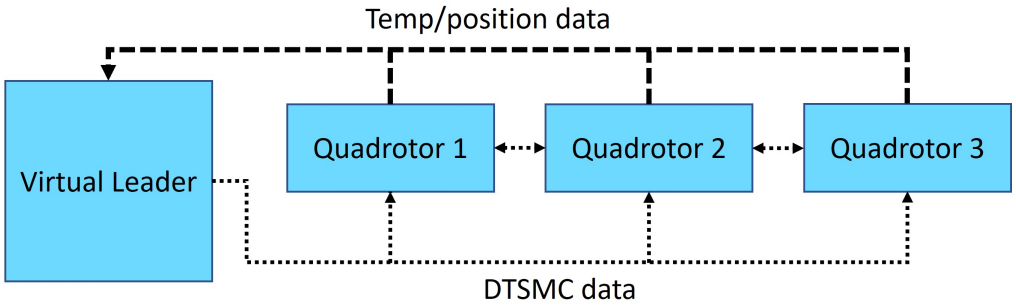


Figure 5.2: Block diagram of the source-seeking system.

5.1.2 Simulation Results

Figure 5.3 shows the trajectory of the agents and the virtual leader as they converge towards the centre of the formation.

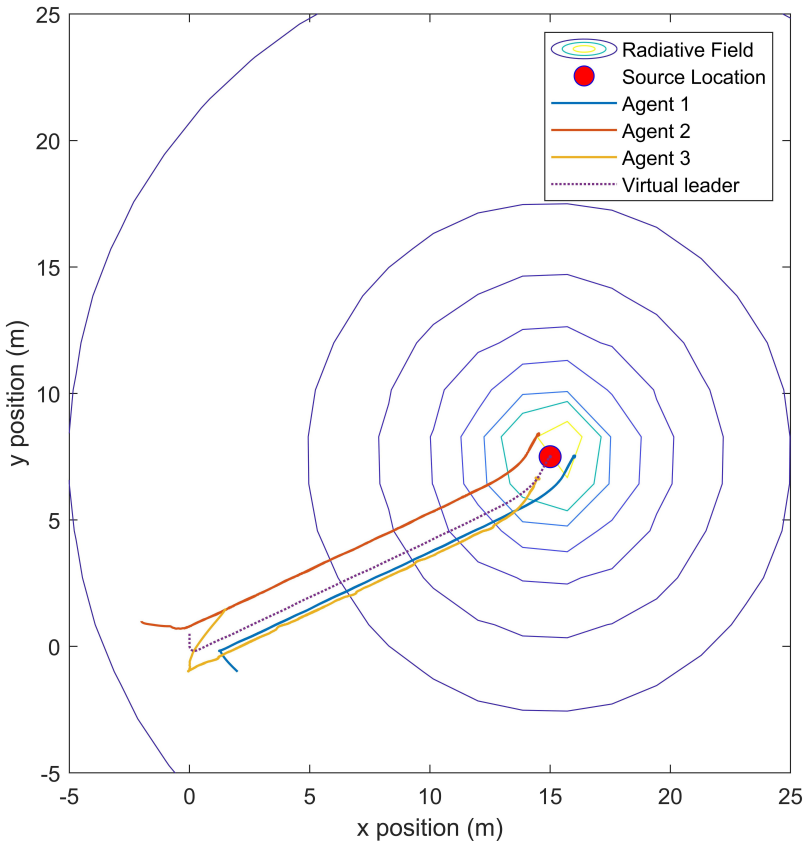


Figure 5.3: Plot of the xy plane in simulation, showing each agent's trajectory alongside the virtual leader and the location of a gamma source, with a contour plot of the produced radiative sensor field from equation (5.2).

From Figure 5.3, it is demonstrated that the agents achieve formation and navigate towards the peak of the radiative sensor field. Due to the circular nature of the sensor field, the gradient is constant within the field. This causes the virtual leader to navigate directly to the field's peak and remain at the source. The virtual leader then moves back and forth over the peak of the field with an amplitude of K_v/T, where the amplitude is expressed in meters.

To investigate the response of the source-seeking control algorithm over time, Figure 5.4 displays the position response of the algorithm over time. From the figure, it is clear that the agents approach the source of the radiation at a velocity of K_v ms^{-1}. Once the agents locate the peak of the field, the virtual leader, along

with all agents in the formation, remains around the source for the remainder of the simulation.

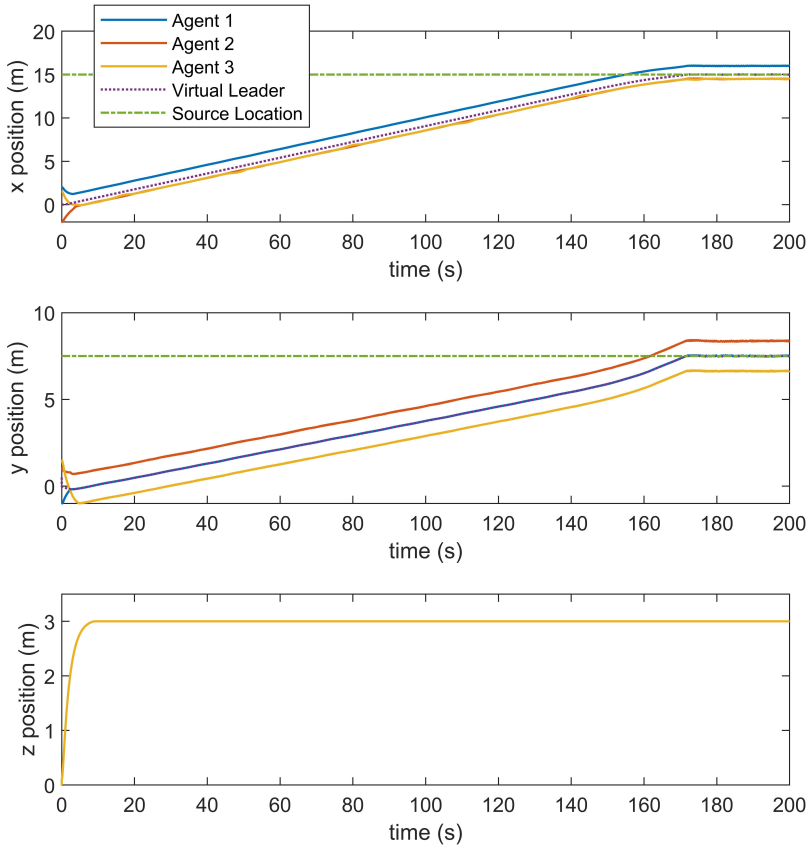


Figure 5.4: x, y, and z position plots of each agent alongside the virtual leader, and the x and y position of the gamma radiation source over time t.

Figure 5.5 shows a surface plot of the radiative field, clearly showing how the inverse-square law in equation (5.1) affects the distribution of the field. Here, the figure displays the efficacy of the source-seeking ability of the proposed control solution as the agents reach their desired altitude and navigate towards the source in the presence of external disturbances. The surface plot, combined with the colour bar on the graph, shows the distribution dose rate across the field.

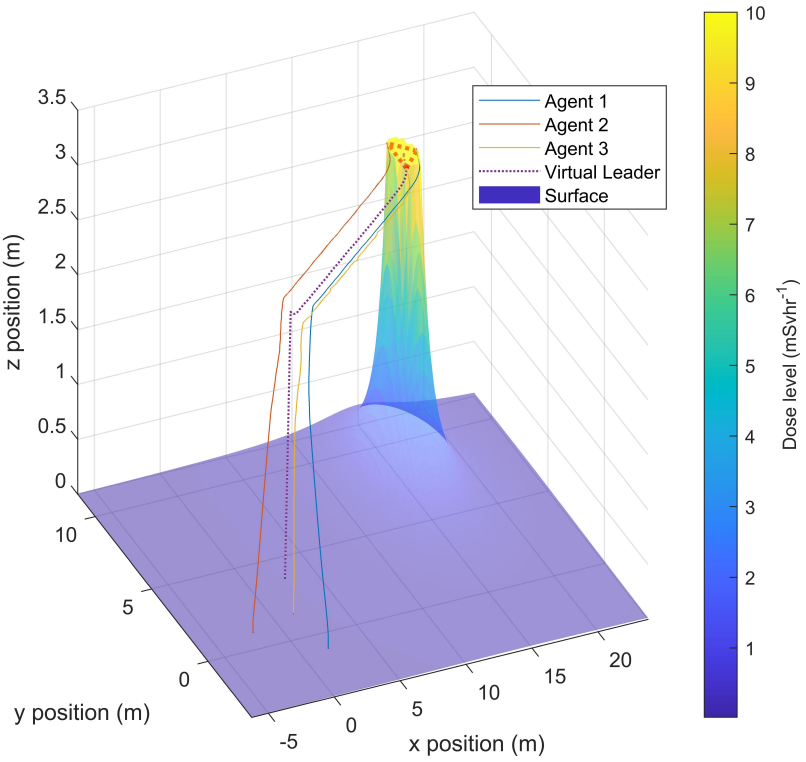


Figure 5.5: Three-dimensional trajectory of each agent alongside the virtual leader, with a surface plot of the gamma dose measurements from radiative sensor field in mSvhr^{-1}.

Position error can be investigated to analyse the performance of the underlying discrete-time sliding mode formation control algorithms as the agents track the time-varying position of the virtual leader as it approaches the field's peak. The errors are shown in Figure 5.6. From the figure, the errors for the x, y and z positions of each agent in the system rapidly converge to zero as they achieve formation. The errors remain in the region of zero for the entire simulation duration, demonstrating that the agents can accurately and effectively track the virtual leader in the presence of external disturbances while it navigates towards the peak of the radiative sensor field.

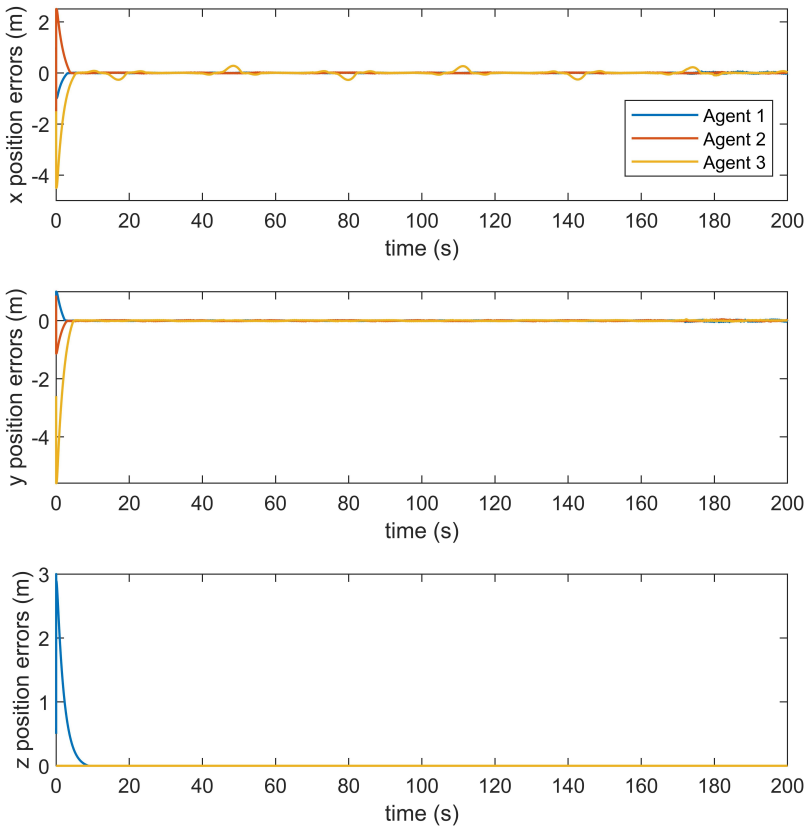


Figure 5.6: Plot showing the position errors from equation(4.9) in meters, as the agents track the virtual leader towards the peak of the radiative sensor field.

From the above figures, the efficacy of the control algorithm for source-seeking purposes is evident. This is extremely useful for applications in the nuclear industry, as it allows environments to be rapidly characterised. Additionally, by quickly locating the peak of the field, this could enable a system to be implemented that is capable of routine monitoring of nuclear sites. The system could identify potential leaks effectively, without putting unnecessary risk on operators completing the task manually. It could also significantly increase the efficiency of the waste monitoring and plant characterisation process, potentially reducing the costs and time required.

Table 5.3: Integral of Absolute Error for each quadrotor agent tracking a virtual leader towards the peak of a radiative scalar field.

Integral of Absolute Error	**Agent**		
	Agent 1	***Agent 2***	***Agent 3***
X Position (m)	2.662	5.626	16.402
Y Position (m)	2.508	3.224	11.354
Z Position (m)	6.125	0.001	0.001

Table 5.3 above shows the IAE for each quadrotor in the networked system. From the table, Agent 1 displays significant error in altitude. This is due to the communication topology of the system, as Agent 1 is the only agent connected to the virtual leader in this simulation. The results demonstrate that Agent 2 and Agent 3 accurately track the altitude of Agent 1 throughout take-off. Table 5.3 also shows that Agent 3 has the worst performance on the x and y axis, despite not having the highest amplitude of disturbance applied to it. This can also be seen in the error plot, as periodic bumps appear in the error signal for Agent 3. The minima of these bumps occurs at approximately $t = 18s$, $t = 80s$, $t = 142s$. The frequencies for the disturbance applied each agent are $0.3rad/s$, $0.5rad/s$, and $0.7rad/s$. The Lowest Common Multiple (LCM) of the periods of each of these frequencies is $T_{common} = 20\pi$, meaning that the three frequencies enter phase every $62.83s$. This aligns with the periodic bumps present in the plot and shows how disturbances applied to other quadrotors can be propagated throughout the network. To avoid this, the control system should be tuned to reject the maximum disturbance applied all quadrotors.

While the agents successfully track the virtual leader as it navigates towards the peak of the radiative sensor field to locate the area of highest dosage, it is not sufficient to simulate the system with the simple model in equation (5.2). A more complex radiation field needs to be tested in simulation to investigate the

system's capability further. Therefore, an alternative function is provided to show a more realistic distribution of the radiative sensor field. However, the process can be applied to any environmental field with a single peak, such as a temperature or humidity sensor field.

The function to calculate intensity at a point in the field $I(x, y)$ at a given point (x, y) is described as follows:

$$I(x,y) = I_0 \cdot \left(\frac{I_1(x,y) + I_2(x,y) + I_3(x,y)}{3} \right), \tag{5.9}$$

where

$$I_l(x,y) = \exp\left(- \begin{bmatrix} (x - x_s) & (y - y_s) \end{bmatrix} \mathbf{R}_i \mathbf{S} \mathbf{R}_i^\top \begin{bmatrix} x - x_s \\ y - y_s \end{bmatrix} \right) \qquad l \in 1, 2, 3,$$

$$\mathbf{S} = \begin{bmatrix} \frac{1}{30} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}, \in \mathbb{R}^{2\times 2}$$

$$\mathbf{R}_1 = 0.25 \begin{bmatrix} \cos\left(-\frac{\pi}{6}\right) & -\sin\left(-\frac{\pi}{6}\right) \\ \sin\left(-\frac{\pi}{6}\right) & \cos\left(-\frac{\pi}{6}\right) \end{bmatrix}, \in \mathbb{R}^{2\times 2}$$

$$\mathbf{R}_2 = 0.25 \begin{bmatrix} \cos\left(\frac{11\pi}{15}\right) & -\sin\left(\frac{11\pi}{15}\right) \\ \sin\left(\frac{11\pi}{15}\right) & \cos\left(\frac{11\pi}{15}\right) \end{bmatrix}, \in \mathbb{R}^{2\times 2}$$

$$\mathbf{R}_3 = 0.25 \begin{bmatrix} \cos(\pi) & -\sin(\pi) \\ \sin(\pi) & \cos(\pi) \end{bmatrix} \in \mathbb{R}^{2\times 2}.$$

In this equation, (x, y) are the coordinates of the point where the field value is being calculated. The term

$$\begin{bmatrix} (x - x_s) & (y - y_s) \end{bmatrix} \mathbf{R}_i \mathbf{S} \mathbf{R}_i^\top \begin{bmatrix} x - x_s \\ y - y_s \end{bmatrix} \tag{5.10}$$

represents the transformed and scaled coordinates, which shifts the origin of the coordinate system to (x_s, y_s). The matrix

$$\mathbf{S} = \begin{bmatrix} \frac{1}{30} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \tag{5.11}$$

is the scaling matrix, where $\mathbf{S}_{11}$ controls the width of the ellipses in the sensor field and $\mathbf{S}_{22}$ controls height of the ellipses.

$\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ are the rotation matrices that rotate each ellipse by a selected angle to create three rotated ellipses. Specifically, $\mathbf{R}_1$ rotates by $-\frac{\pi}{6}$ radians, $\mathbf{R}_2$ rotates by $\frac{11\pi}{15}$ radians, and $\mathbf{R}_3$ rotates by π radians.

Each term

$$I_l(x, y) = \exp\left(-\begin{bmatrix}(x - x_s) & (y - y_s)\end{bmatrix}\mathbf{R}_i\mathbf{S}\mathbf{R}_i^\top\begin{bmatrix}x - x_s \\ y - y_s\end{bmatrix}\right) \tag{5.12}$$

represents the sensor field value contributed by one of the rotated ellipses at the point (x, y). The final field value $I(x, y)$ is obtained by averaging the contributions from the three ellipses and scaling the result to make the peak value 10 mSvhr^{-1}. In this case, the source of gamma radiation was placed further away from the agents' initial positions to assess the source-seeking algorithm's convergence time.

Table 5.4: Table showing the initial positions of each agent and the position of a simulated gamma radiation source for the complex radiative sensor field simulations.

	x position (m)	y position (m)
Gamma radiation source	51.0	-20.0
Initial Agent 1 position	2.0	-1.0
Initial Agent 2 position	-2.0	1.0
Initial Agent 3 position	1.5	1.5

Figure 5.7 shows the contour plot of the complex field from equation (5.9). The effect of combining three ellipses is evident in creating a complex gradient is shown to develop a complex gradient distribution across the xy plane. The plot shows that despite this, the formation of agents could track the virtual leader to the sensor field's peak and stop at the location of the radiation's source. In this figure, the trajectory that the leader takes towards the location of the source is not linear due to the complex gradient of the sensor field.

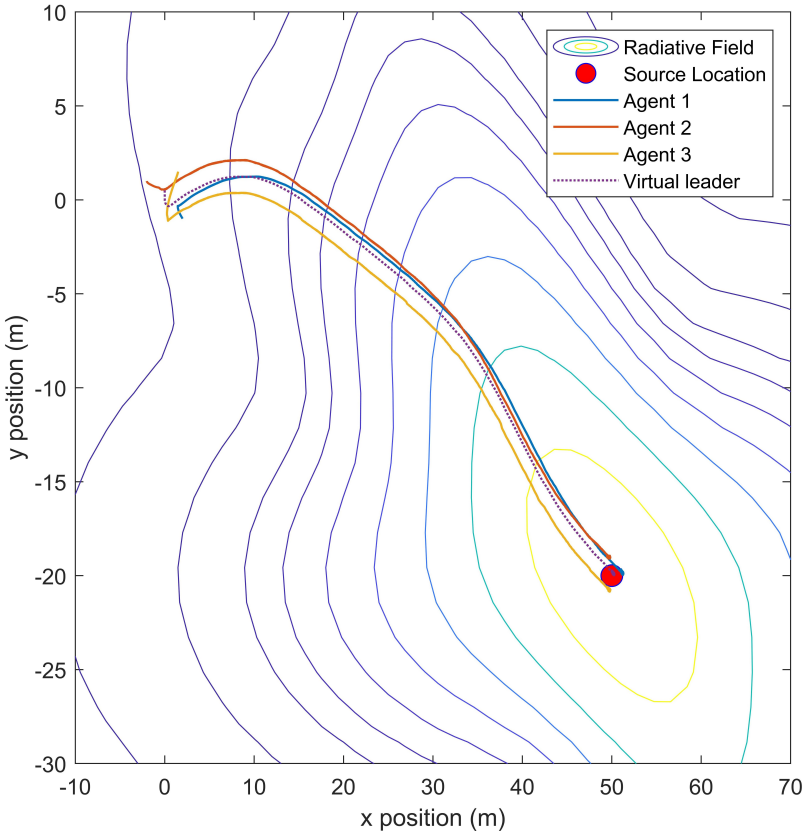


Figure 5.7: Plot of the xy plane in simulation, showing each agent's trajectory alongside the virtual leader and the location of a gamma source, with a contour plot of the produced radiative sensor field from equation (5.9).

To assess the response of the system over time, Figure 5.8 displays the x, y, and z position response of each agent, along with the position response of the virtual leader in the system. The figure also displays the x and y locations of the gamma radiation source. From the figures, it is clear that the formation converges towards the field's peak in finite time in the presence of external disturbances applied to the system. The nonlinear nature of the trajectory taken is clear, as the y position initially diverges away from the location of the source before approaching it following time $t = 55s$. Meanwhile, the x positions of the agents in the formation approach the location of the source rapidly, locating a source 54.78m away within 300 seconds, making it feasible for implementation on UAVs with limited flight time. It is seen

that on both axes, the formation converges to the location of the gamma radiation source.

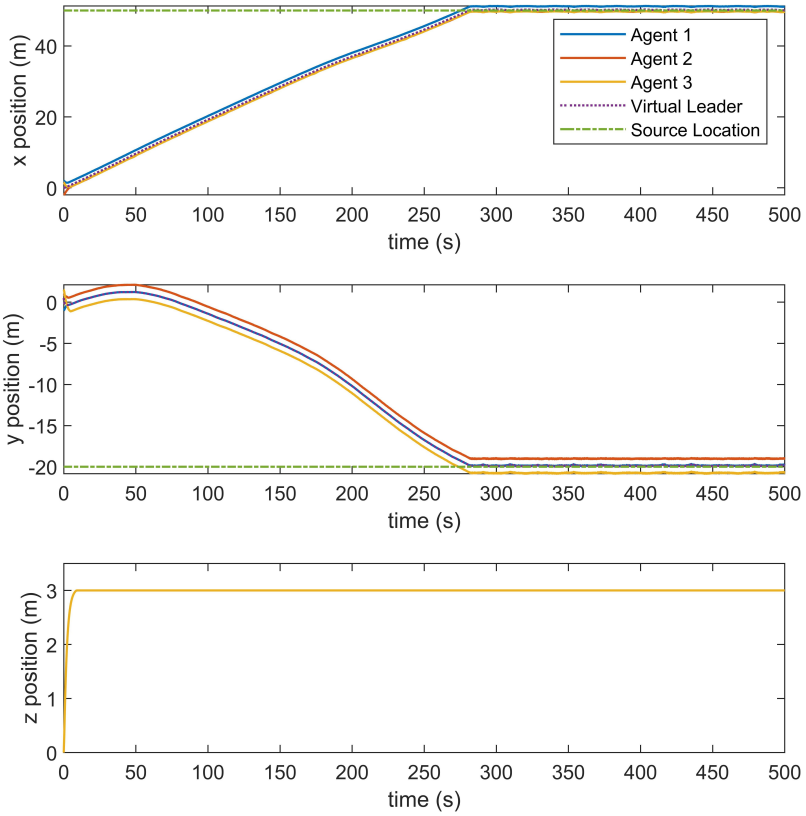


Figure 5.8: x, y, and z position plots of each agent alongside the virtual leader, and the x and y position of the gamma radiation source over time t while source-seeking within a complex sensor field.

To further analyse the system's performance, the three-dimensional trajectory plot of each agent, alongside the virtual leader, is shown in Figure 5.9. The figure also shows a surface plot of the field, displaying the complex distribution of the sensor field. The plots show that after attaining formation at the desired altitude, the agents accurately track the virtual leader towards the field's peak and stop at the field's peak.

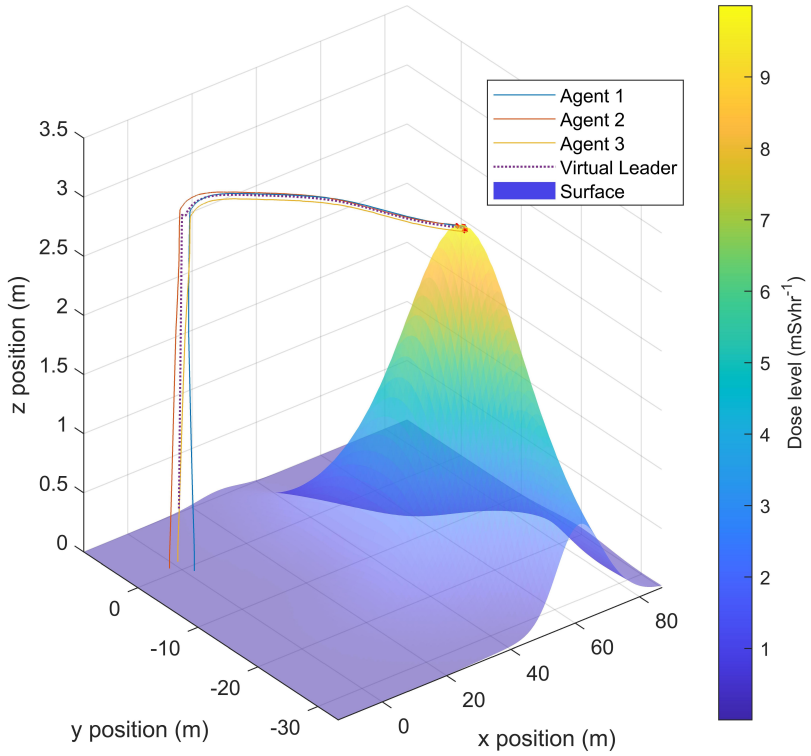


Figure 5.9: Three-dimensional trajectory of each agent alongside the virtual leader, with a surface plot of the gamma dose measurements from radiative sensor field in mSvhr^{-1} while source-seeking within a complex sensor-field.

The discrete-time sliding mode formation control algorithm (4.16) that provides the foundation for this method can be analysed again by assessing the position error response of the system in Figure 5.10. From the figure, the errors on each of the x, y, and z axis converge to zero by time $t = 10s$, showing that formation is achieved. The errors then remain around a zero error value for the remainder of the simulation until time $t = 500s$. This further evidences the robustness of the underlying algorithm, as the agents can track the dynamic virtual leader for the duration of the simulation.

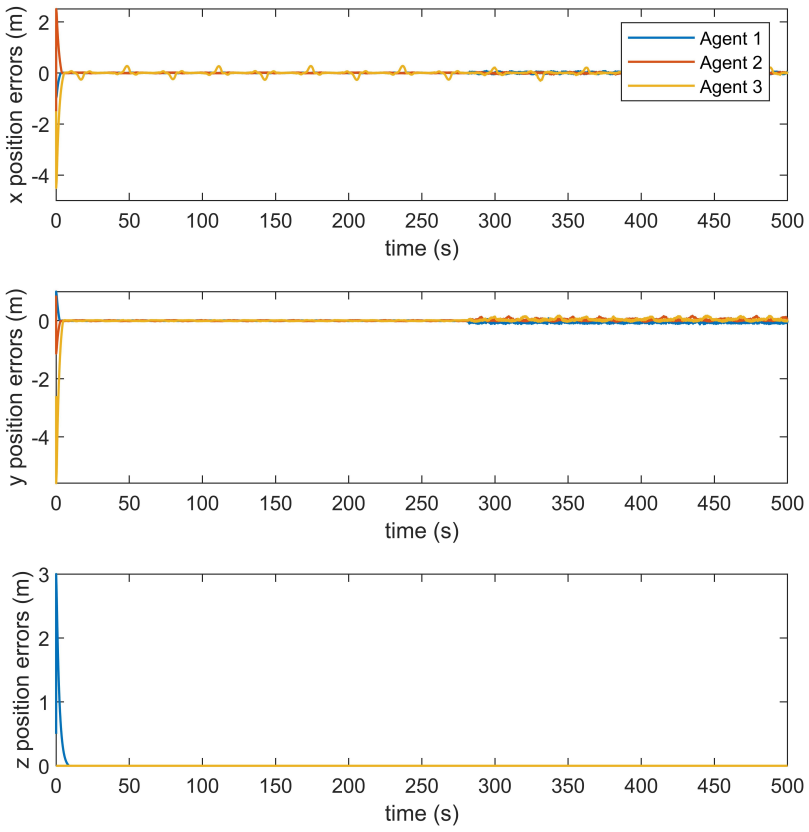


Figure 5.10: Plot of the position errors of each agent in the system.

Figure 5.10 shows jumps in the error signal with a period of approximately $62s$. At $t = 270s$ the impact of the oscillations of the source seeking virtual leader around the scalar peak also become more visible on the y axis.

Table 5.5: Integral of Absolute Error for each quadrotor agent tracking a virtual leader towards the peak of a complex radiative scalar field.

Integral of Absolute Error	**Agent**		
	Agent 1	***Agent 2***	***Agent 3***
X Position (m)	7.618	9.241	29.470
Y Position (m)	13.854	13.063	22.470
Z Position (m)	6.126	0.002	0.002

Table 5.5 further demonstrates the impact of the combined disturbance signal on the multi-agent system. The large IAE for Agent 3 is impacted by its initial distance from the formation. To investigate this further, the complex field simulation was repeated with a fully connected network of three agents.

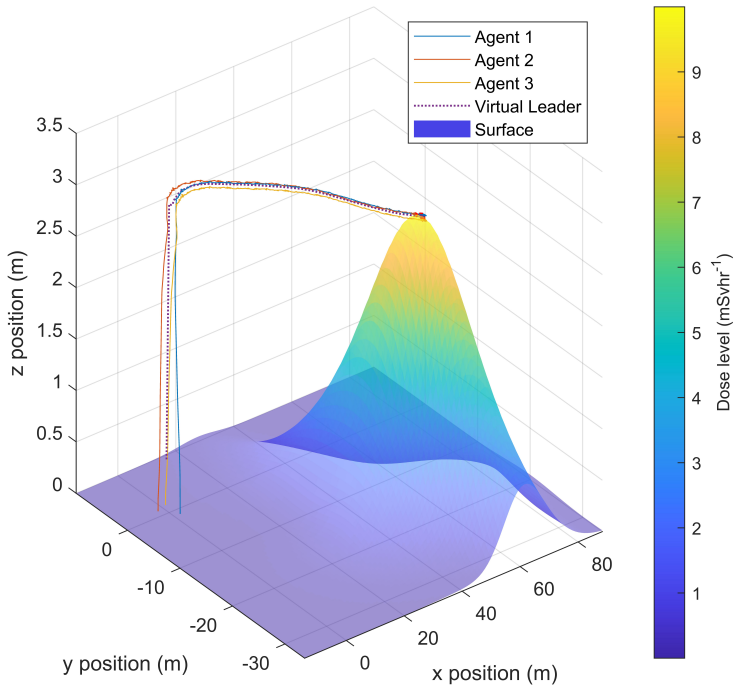


Figure 5.11: Three-dimensional trajectory of each agent alongside the virtual leader, with a surface plot of the gamma dose measurements from radiative sensor field in mSvhr^{-1} while source-seeking within a complex sensor-field with a fully connected graph.

Table 5.6: Integral of Absolute Error for each quadrotor agent tracking a virtual leader towards the peak of a complex radiative scalar field using a fully connected network topology.

Integral of Absolute Error	**Agent**		
	Agent 1	***Agent 2***	***Agent 3***
X Position (m)	29.295	49.482	31.936
Y Position (m)	148.807	55.123	39.158
Z Position (m)	6.126	0.002	0.002

From Table 5.6, it is clear that there are now significant errors present for the x and y positions of all three agents. This demonstrates further that disturbances propagate through the networked system. From the previous chapter, equations (4.40), (4.37), and (4.43) show that the size of the Quasi-Sliding-Mode Band becomes larger with an increased number of connected agents, demonstrating one of the limitations of the control system design. For real-world applications within the nuclear industry, it is therefore advisable to ensure that there is a limited number of interconnected agents within the multi-agent system, while maintaining at least one directed spanning tree.

From the results provided in this section, the efficacy of the discrete-time sliding mode formation control system has been validated in simulation to application in source-seeking within radiative sensor fields. The algorithm's robustness is demonstrated by introducing external wind disturbance in the simulation. The source-seeking strategy is effective in cases where a radiative sensor field follows the inverse-square law. Also, the strategy shows strong performance in the presence of more complex environmental sensor fields. Finally, by implementing the algorithm in an undirected communication network, the system can handle switching network topologies, increasing robustness in the case of communication drop-outs between agents.

5.2 Environmental Field Estimation

While Section 5.1.1 presents a robust source-seeking strategy, operators require more information about the distribution of an environmental field to characterise nuclear sites fully. Towards this, the current section aims to validate a common environmental field estimation technique, i.e Gaussian Process Regression (GPR), experimentally [146], [147]. Performing GPR while post-processing data collected can provide an operator with an estimate of the distribution of a sensor field. This is particularly useful in the nuclear industry, as it would allow rapid characterisation of nuclear legacy sites and provide informative and easy-to-read data from any flights conducted.

5.2.1 Experimental Setup

To assess the data collection abilities of the distributed quadrotor system, the environment presented in Section 4.5 was adapted by introducing a space heater. This allowed for the generation of a temperature gradient within the test environment to validate the use of Gaussian Process Regression for post-processing data collected onboard the quadrotors to produce and estimate an environmental sensor field. Due to safety restrictions, it was not possible to validate the approach using a source of gamma radiation; hence a heat source was used to produce a scalar sensor field. Figure 5.12 shows a diagram of the constructed test environment with the placement of two heat sources.

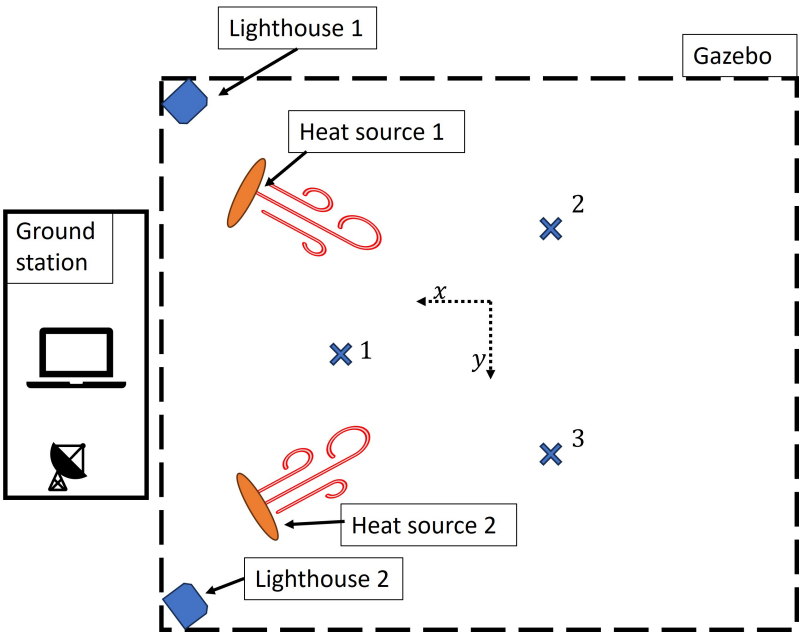


Figure 5.12: Diagram of the experimental test area with the location of two heat sources.

Figure 5.13: Test environment for data collection with two space heaters.

For temperature sensing, LM35 sensors **lm35** were attached to three Crazyflie 2.1 micro-quadrotors. Power was delivered to the LM35 sensors through the

onboard 3.3V power pins. GPIO was used to pass data from the LM35 sensors to the Crazyflie2.1 embedded firmware. The firmware, written in embedded C, was modified to accept the data through the GPIO pin and broadcast this data to the ground station, where it was logged. A photograph of the sensor attached to a Crazyflie2.1 micro-quadrotor is shown in Figure 5.14.

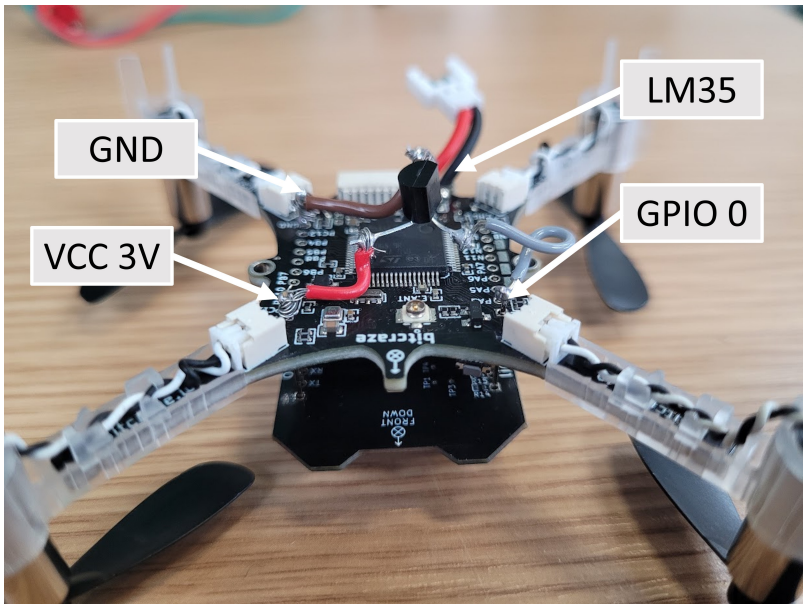


Figure 5.14: Photograph showing the LM35 temperature sensor connected to a Crazyflie2.1.

Similar to Chapter 3, the Lighthouse Positioning Deck was used to localise each Crazyflie2.1 in the environment. The Python code used to collect the data is provided in Appendix A

5.2.2 Experimental Results

To create a simple testing scenario, only one heater was powered in the area to create a smooth temperature distribution over the environments. The ambient temperature in the room was measured with a mercury thermometer as 22.1°C. The peak temperature in front of the heat source was 31.1°C.

The temperature and position data were concatenated into one CSV file. The Gaussian Process package from scikit-learn was used in Python to process the data and produce an estimate of the environmental field [148]. The code for this is provided in Appendix B. In the first scenario, the agents were tasked to achieve a formation and tasked with following a virtual leader with a circular trajectory. The same trajectory presented in Section 4.5.1, with the altitude of each agent set to $z = 0.5m$ to ensure immersion within the temperature field. The estimated field is plotted in Figures 5.15.

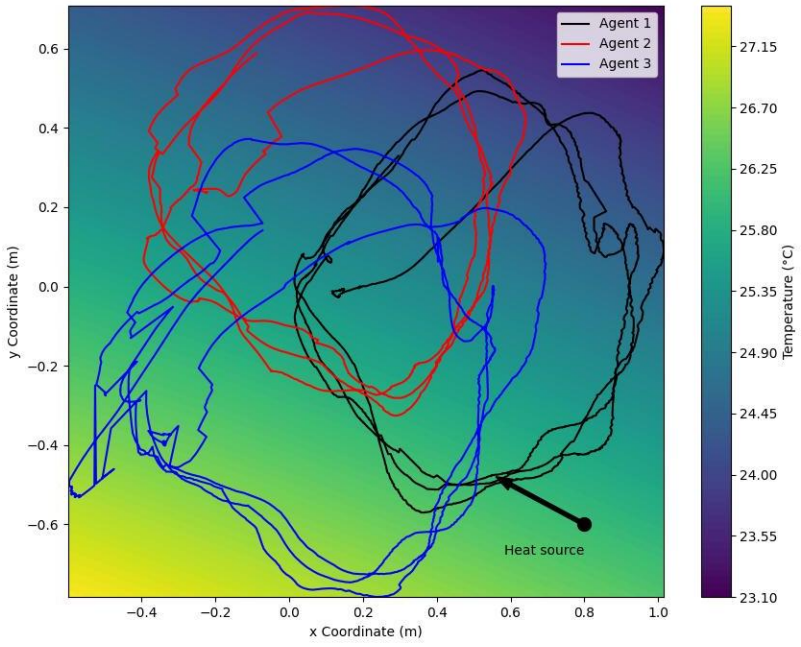


Figure 5.15: Estimated temperature distribution across the area from data collected using LM35 sensors onboard Crazyflie2.1 micro-quadrotors.

From Figure 5.15, a temperature gradient is measured across the field. The

temperature gradient appears to have a slight clockwise rotation compared to the expected hotspot. It is believed that this was due to temperature lag present in the system. The LM35 temperature sensors do not measure the temperature instantaneously at a location. The sensors themselves heat up over time while exposed to warm air. Additionally, they also require time to cool after heating. This lag may be causing the unwanted rotation that affects the estimated temperature field. Despite this, the network accurately predicts temperature readings close to the ambient air temperature and the maximum air temperature recorded next to the heat source before the test. The temperature lag can potentially be compensated with a time and space dependent temperature model, though this lies out of the scope of this work as it is believed that when implementing the system using a radiation monitor such as a scintillator, negative effects similar to temperature lag will not be present.

To further investigate the efficacy of the GPR estimation algorithm, the second heat source in diagram 5.12 was enabled. The temperature distribution within the environment was expected to have two hotspots. In an attempt to improve the environmental field estimation of the swarm, the virtual leader's trajectory was modified to execute a spiral trajectory within the environments. This would allow a larger surface area to be covered on the xy plane, providing the algorithm with more valuable data. Figure 5.16 displays the estimated field.

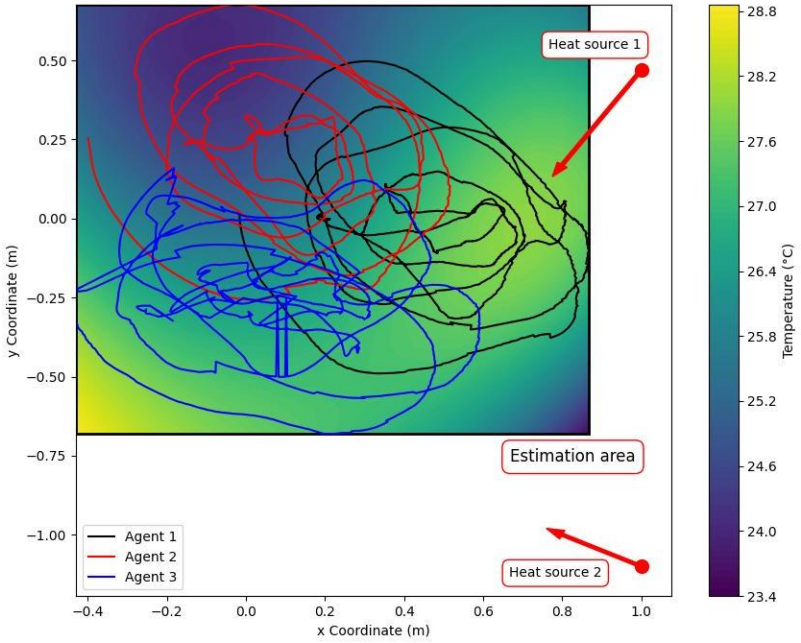


Figure 5.16: Estimated temperature distribution across the area from data collected using LM35 sensors onboard Crazyflie2.1 micro-quadrotors with two active heat sources.

In Figures 5.16, it is clear that the unwanted rotation effect due to sensor lag is still present when using a spiral trajectory with two heat sources. Regardless, the estimated field displays two clear hotpots and identifies a distribution consistent with the expected distribution based on the fan sensors.

Using GPR combined with the proposed source-seeking algorithm could significantly improve the plant characterisation and continuous monitoring process in nuclear sites. Implementing these emerging techniques at the industry level could increase the safety and efficiency of nuclear monitoring processes while reducing the time and costs associated with such activities.

5.3 Conclusion

In this chapter, applications of a multi-quadrotor formation are validated at the industry level. A novel, robust approach for the source-seeking problem is proposed

based on robust discrete-time sliding mode formation control. The proposed solution is robust to switching network topologies and external disturbances introduced through robust ventilation systems. The algorithm proves capable of steering a formation of quadrotors towards the peak of a radiative sensor field. The source-seeking algorithm was also tested with a more complex environmental field, based on the superposition of three rotated elliptical sensor fields, to validate the approach for a non-circular field.

In addition, Gaussian Process Regression was used to estimate the distribution of the field from data collected by a multi-quadrotor sensing system, which was tested experimentally. The experimental results demonstrate a reasonable estimation of the environmental field, with an estimation of the minimum and maximum temperature values within the field. Due to lag in the temperature sensors, the quadrotors estimated a shift in the environmental field. The results presented in this chapter indicate that, through novel and emerging technologies, the nuclear industry can significantly improve its processes while maintaining safety by implementing robust techniques such as sliding mode control.

Chapter 6

Conclusion

In this chapter, conclusions are made surrounding the key findings of this thesis, and the potential impact of the research is discussed. This chapter also aims to summarise the work completed during the thesis, discuss how the aims of the project have been met, discuss the limitations of the project, and finally propose ideas for future work.

The aims of the project are stated below:

- Develop a robust control algorithm for trajectory tracking of a single quadrotor for the purpose of navigating within hazardous nuclear environments.
- Develop an algorithm to enable stable and reliable formation control of a group of quadrotors in hazardous nuclear environments.
- Validate the designed algorithms through extensive simulation and implement the algorithms in real-world experimental testing.
- Apply the designed algorithms to problems within the nuclear industry to validate the efficacy of the proposed system.

The overall goal of the project was to deliver a robust system of quadrotors that addressed these four research aims. As stated in the introduction of this thesis, achieving these aims would provide a cyber-physical system capable of addressing a

number of key challenges within the nuclear industry, such as plant characterisation and condition monitoring and inspection. The ultimate motivation for this was to contribute to improving the safety and efficiency across the nuclear industry.

To investigate the success of this thesis in achieving these aims, the findings from each chapter will be summarised. In Chapter 2 a novel finite-time integral sliding mode control system is provided for trajectory tracking of a single quadrotor. The control algorithm provided robust, accurate, and chattering-free trajectory tracking performance for a continuous-time system. The robustness of the system was analysed through a stability proof, and was shown to be stable in the presence of uncertainties in parameters, such as the mass, or inertial properties of a quadrotor. The proof also shows the finite-time properties of the controller. The proposed algorithm was then validated through numerical simulations, which also enabled iterative tuning of the proposed control system. In order to assess the efficacy of the system for the real-world case, FTI-SMC was implemented on a Mambo Minidrone. The results were compared to another modern finite-time sliding mode technique, as well as PID, and displayed superior performance to the other control systems. The chapter proposed and validated a control system suitable for robust trajectory tracking inside of nuclear environments, in the presence of parameter uncertainties, addressing the first aim of the project. Despite this, the chapter does not consider the discrete-time behaviour of the system.

Due to the discrete-time nature of sensors and compute systems available onboard UAVs, continuous control methods may not be suitable for the control of these systems in safety-critical environments. This is particularly important in indoor nuclear sites, where environments are hazardous, cluttered, and GPS denied. Because of this, localisation systems such as SLAM are required to allow UAVs and other robots to navigate safely within these environments. SLAM systems can have slow update rates, making the control of UAVs with SLAM-based localisation particularly difficult for continuous-time control methods. To address this, Chapter 3 implements a discrete-time sliding mode control system for a quadrotor. The

controller was tested using ROS, and investigated using a co-simulation environment, where the control system was implemented in Simulink, and the simulation of the quadrotor used the Gazebo simulation environment to provide a more realistic testing scenario for the system. The Hector SLAM package was implemented for localisation of the quadrotor system. The discrete-time update rates from the system were varied to assess the performance of the discrete-time control system compared to continuous methods. The results showed that the discrete-time system had superior performance compared to continuous methods in cases where sampling rates were slower. This demonstrated the potential requirement for discrete-time control systems for trajectory tracking of a quadrotor in indoor nuclear environments.

To address the second research aim above, Chapter 4 designs and implements a discrete-time sliding mode formation control system for a network of quadrotors. The use of discrete-time sliding mode control allowed a robust system of quadrotors to be developed, capable of holding a formation about a dynamic virtual leader in the presence of external disturbances. A robustness analysis shows that the control system demonstrated quasi-sliding mode behaviour. The marginal stability of the control system was further demonstrated with the sliding surfaces remaining within a quasi-sliding mode band in a Simulink simulation. To further validate the efficacy of the control system for a network of quadrotors, the system was implemented experimentally using Crazyflie 2.1 micro-quadrotors. Through real-world experimental testing, strong performance in the nominal case was demonstrated, as well as reasonable performance when in the presence of an external wind disturbance. Results from experimental testing showed that, while the system successfully retained its formation for the duration of the flights, the system temporarily diverged from the position of the virtual-leader when a disturbance was applied. This can potentially be mitigated, and future work should aim to investigate alternative network topologies, or tune the system further to improve the performance of the system. To summarise, Chapter 4 addresses the second and third aims of the project, by developing and experimentally testing a discrete-time

formation control algorithm.

To address the final research aim, Chapter 5 investigates the efficacy of the discrete-time sliding mode formation control for application to challenges in the nuclear industry. First, a system capable of source-seeking in a radiative sensor field is developed by implementing a gradient-climbing virtual leader. By supplying the virtual leader with temperature measurements, and position readings from each quadrotor, through an undirected graph network topology, it was able to navigate up a gradient towards the source of the field. Implementing this in combination with the discrete-time sliding mode formation controller in Chapter 4, a system was developed capable of locating the peaks of these sensor fields, while rejecting external wind disturbance. The proposed algorithm was investigated in simulation with a simple gamma radiation source, following the inverse-square law, placed in an environment in Simulink. The proposed system allowed the agents to navigate towards the source and remain at the source for the duration of the simulation. To further investigate the efficacy of the control system for more complicated sensor fields, an environmental field was simulated with superposition of three ellipses. This simulation made the gradient climbing performance of the system evident, as the plots in the chapter show the quadrotor swarm navigating up the gradient towards the peak, locating the source within 300 seconds. To further investigate the applications of the formation of quadrotors, the data-collection capability of the quadrotors was combined with a technique known as Gaussian Process Regression, to provide an operator with a two-dimensional map of a temperature sensor-field. Data recorded by three quadrotors, equipped with LM35 temperature sensors and placed in an environment with a varying temperature gradient, was processed using Gaussian Process Regression in Python. The post-processed data displayed the temperature gradient in the environment, in the presence of both a single hotspot, and two separate hotspots. Combining these techniques, effectively allowing a network of quadrotor to locate the peak of a radiative sensor field, while collecting data to provide an operator with an environmental map of the area covered,

could greatly improve the efficiency and safety of monitoring and characterisation processes within the nuclear industry. This chapter validated the efficacy of the designed control algorithms for applications in the nuclear industry.

The development of a robust source-seeking swarm of quadrotors has several implications for the nuclear industry. Firstly, the developed algorithms provides a novel robust method for rapid localisation of a radiation source. This could allow a group of quadrotors to locate potential radiation hotspots within a nuclear facility, for example, locating the source of radiation within a cell in a nuclear facility undergoing decommissioning, or the exact location of a radiation hostpot in the event of an incident. The system is also capable of providing an operator with a two-dimensional map of the environmental field over the covered location using Gaussian Process Regression. This could dramatically increase the efficiency of monitoring approaches in these facilities. In addition to improved efficiency, the system also provides improved safety through the reduction of human exposure in these processes. Furthermore, the proposed system is scalable and adaptable. While the thesis focuses on the use of a small network of three quadrotors, the algorithms proposed are capable of achieving formation of a larger number of quadrotors. The adaptability of the approach comes from the invariance to network topology, as the system will work in any communication graph displaying at least one directed spanning tree, improving system redundancy and robustness. Using the system for rapid environmental monitoring can also supply timely and accurate data collection of radiation levels in various environments. This information could be used for regulatory compliance and potentially improve public confidence in safety within the nuclear industry. Ultimately, this thesis provides advancements in the fields of both single and multi-quadrotor control, improving and validating the robustness of quadrotor control systems for use in the nuclear industry.

The work in this thesis has therefore addressed the four research aims outlined by the project, and has provided the components of a cyber-physical system capable of addressing a number of challenges identified in the nuclear industry. Despite this,

there is still remaining work to be undertaken were the system to be implemented on Sellafield site, or another nuclear site. This thesis has proposed and initially validated a number of robust algorithms for the accurate and robust control of quadrotors. Work remains on the further validation of these techniques for use in the nuclear industry, to ensure safety of the system and regulatory compliance. One of the limitation of the project was the inability to verify the robustness of the formation control algorithm, in combination with the previously designed inner-loop attitude and altitude control systems. Future work could aim to further validate the efficacy of the control systems by combining these control techniques, and implementing them on a larger quadrotor systems in the presence of disturbances. A combination of different control algorithms could be considered, where an outer-loop formation control system, based on discrete-time sliding mode formation control is implemented alongside an attitude and altitude controller based on finite-time integral sliding mode control. This could potential combine the chattering-free capability of FTI-SMC, with the outer loop discrete-time sliding mode formation controller, handling the discontinuous update rate of the networked system.

In addition to the validation of the the underlying control systems, the applications proposed in this thesis can also be further investigated in future work. In particular, future works should aim to investigate the efficacy of the system for locating of a radioactive source in the real-world case, potentially trialing the system on Sellafield site. An alternative avenue for future research is the application of the developed algorithms to other fields, such as environmental monitoring and agriculture.

This concluding chapter has provided a summary of the work completed in this thesis, demonstrating the success of the project in addressing the research aims outlined in Chapter 1. By advancing the understanding of quadrotor control algorithms, the thesis has provided a number of robust control strategies for safe and accurate control of quadrotors in the nuclear industry. The research demonstrates the applications of robust mutli-quadrotor control in the nuclear industry, for

problems such as radiation source localisation. This thesis presents the critical foundational work required for the future implementation of autonomous multi-quadrotor systems within the nuclear industry.

References

[1] A. Can, H. Efstathiades, and A. Montazeri, "Design of a chattering-free sliding mode control system for robust position control of a quadrotor," in *2020 International Conference Nonlinearity, Information and Robotics (NIR)*, 2020, pp. 1–6. DOI: 10.1109/NIR50484.2020.9290206.

[2] A. Montazeri and A. Can, "Unmanned aerial systems: Autonomy, cognition, and control," in Jan. 2021, pp. 47–80, ISBN: 9780128202760. DOI: 10.1016/B978-0-12-820276-0.00010-8.

[3] N. Sadeghzadeh Nokhodberiz, A. Can, R. Stolkin, and A. Montazeri, "Dynamics-based modified fast simultaneous localization and mapping for unmanned aerial vehicles with joint inertial sensor bias and drift estimation," *IEEE Access*, vol. PP, pp. 1–1, Aug. 2021. DOI: 10.1109/ACCESS.2021.3106864.

[4] A. Can, J. Price, and A. Montazeri, "A nonlinear discrete-time sliding mode controller for autonomous navigation of an aerial vehicle using hector slam," *IFAC-PapersOnLine*, vol. 55, pp. 2653–2658, Oct. 2022, © 2022 IFAC. CC BY-NC-ND 4.0. DOI: 10.1016/j.ifacol.2022.10.110.

[5] A. Can, J. Price, and A. Montazeri, "Robust formation control and trajectory tracking of multiple quadrotors using a discrete-time sliding mode control technique," *IFAC-PapersOnLine*, vol. 55, pp. 2974–2979, Oct. 2022, © 2022 IFAC. CC BY-NC-ND 4.0. DOI: 10.1016/j.ifacol.2022.10.184.

[6] I. H. Imran, A. Can, R. Stolkin, and A. Montazeri, "Real-time nonlinear parameter estimation and tracking control of unmanned aerial vehicles in closed-loop," *Scientific Reports*, vol. 13, Feb. 2023. DOI: 10.1038/s41598-023-29544-6.

[7] *Sellafield nuclear decommissioning work 'significantly' delayed and millions over budget, report reveals — independent.co.uk*, https://www.independent.co.uk/climate-change/news/sellafield-nuclear-power-plant-decommission-overspend-delays-budget-report-a8609671.html, [Accessed 30-06-2024].

[8] *Innovation — gov.uk*, https://www.gov.uk/government/case-studies/innovation.

[9] *UKRI Challenge Fund: For research and innovation — gov.uk*, https://www.gov.uk/government/collections/industrial-strategy-challenge-fund-joint-research-and-innovation.

[10] *[Withdrawn] Robots for a safer world: Industrial Strategy Challenge Fund — gov.uk*, https://www.gov.uk/government/collections/robots-for-a-safer-world-industrial-strategy-challenge-fund.

[11] B. Nouri Rahmat Abadi, A. West, H. Peel, *et al.*, "Carma ii: A ground vehicle for autonomous surveying of alpha, beta and gamma radiation," *Frontiers in Robotics and AI*, vol. 10, p. 1 137 750, 2023.

[12] *Sellafield: Remotely operated unmanned aerial vehicle combined with radiation mapping software — gov.uk*, https://www.gov.uk/government/case-studies/sellafield-remotely-operated-unmanned-aerial-vehicle-combined-with-radiation-mapping-software, [Accessed 25-06-2024].

[13] *Game Changers - Our challenges — gamechangers.technology*, https://www.gamechangers.technology/challenges/, [Accessed 25-06-2024].

[14] J. Morris, T. Hicks, S. Wickham, *et al.*, "Development of a methodology for determining ilw package monitoring and inspection requirements," in *International Conference on Radioactive Waste Management and Environmental Remediation*, vol. 54983, 2011, pp. 1361–1370.

[15] C. Halliwell, "The windscale advanced gas cooled reactor (wagr) decommissioning project a close out report for wagr decommissioning campaigns 1 to 10-12474," in *WM2012 Conference, February*, 2012.

[16] S. Watson, B. Lennox, and A. Stancu, "Robotic systems for remote characterization and decommissioning–15102," WM Symposia, Inc., PO Box 27646, 85285-7646 Tempe, AZ (United States), 2015.

[17] *Game Changers - Our challenges — gamechangers.technology*, `https://www.gamechangers.technology/challenges/condition-monitoring-inspection`.

[18] Game Changers, *Asset management trial success — gamechangers.technology*, `https://www.gamechangers.technology/news/Asset_management_trial_success`, [Accessed 30-06-2024].

[19] *Game Changers - Our challenges — gamechangers.technology*, `https://www.gamechangers.technology/challenges/physical-asset-management/`.

[20] *Game Changers - Our challenges — gamechangers.technology*, `https://www.gamechangers.technology/challenges/waste-handling-and-storage/`.

[21] J. M. Aitken, S. M. Veres, A. Shaukat, *et al.*, "Autonomous nuclear waste management," *IEEE Intelligent Systems*, vol. 33, no. 6, pp. 47–55, 2018.

[22] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[23] L. Li, L. Sun, and J. Jin, "Survey of advances in control algorithms of quadrotor unmanned aerial vehicle," in *2015 IEEE 16th international conference on communication technology (ICCT)*, IEEE, 2015, pp. 107–111.

[24] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2451–2456 vol.3. DOI: 10.1109/IROS.2004.1389776.

[25] A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. S. Gaeid, "Modelling and pid controller design for a quadrotor unmanned air vehicle," in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, vol. 1, 2010, pp. 1–5. DOI: 10.1109/AQTR.2010.5520914.

[26] A. Salih, M. Moghavvemi, and H. A. Mohamed, "Flight pid controller design for a uav quadrotor.," *Scientific Research and Essays (SRE)*, vol. 5, no. 23, pp. 3660–3667, 2010.

[27] S. Khatoon, D. Gupta, and L. Das, "Pid & lqr control for a quadrotor: Modeling and simulation," in *2014 international conference on advances in computing, communications and informatics (ICACCI)*, IEEE, 2014, pp. 796–802.

[28] J. C. V. Junior, J. C. De Paula, G. V. Leandro, and M. C. Bonfim, "Stability control of a quad-rotor using a pid controller," *Journal of Applied Instrumentation and Control*, vol. 1, no. 1, pp. 15–20, 2013.

[29] S. Abdelhay and A. Zakriti, "Modeling of a quadcopter trajectory tracking system using pid controller," *Procedia Manufacturing*, vol. 32, pp. 564–571, 2019.

[30] C. Liu, J. Pan, and Y. Chang, "Pid and lqr trajectory tracking control for an unmanned quadrotor helicopter: Experimental studies," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 10 845–10 850. DOI: 10.1109/ChiCC.2016.7555074.

[31] H. S. Khan and M. B. Kadri, "Position control of quadrotor by embedded pid control with hardware in loop simulation," in *17th IEEE international multi topic conference 2014*, IEEE, 2014, pp. 395–400.

[32] B. Pratama, A. Muis, A. Subiantoro, M. Djemai, and R. B. Atitallah, "Quadcopter trajectory tracking and attitude control based on euler angle limitation," in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, IEEE, 2018, pp. 1–6.

[33] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, "Lqr control for a quadrotor using unit quaternions: Modeling and simulation," in *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*, IEEE, 2013, pp. 172–178.

[34] Y. Bai, H. Liu, Z. Shi, and Y. Zhong, "Robust control of quadrotor unmanned air vehicles," in *Proceedings of the 31st Chinese Control Conference*, 2012, pp. 4462–4467.

[35] S. B. Monfared, A. Kalhor, and M. A. Atashgah, "Robust h control for path tracking of a quadrotor through estimation of system parameters," in *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, 2016, pp. 337–342. DOI: 10.1109/ICRoM.2016.7886761.

[36] A. Astudillo, B. Bacca, and E. Rosero, "Optimal and robust controllers design for a smartphone-based quadrotor," in *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)*, 2017, pp. 1–6. DOI: 10.1109/CCAC.2017.8276392.

[37] S. B. Monfared, A. Kalhor, and M. A. Atashgah, "Robust adaptive h control for attitude control of a quadrotor," in *2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM)*, 2018, pp. 560–565. DOI: 10.1109/ICRoM.2018.8657497.

[38] D.-A. Pham and S.-H. Han, "Critically leveraging theory for optimal control of quadrotor unmanned aircraft systems," *Applied Sciences*, vol. 14, no. 6, 2024, ISSN: 2076-3417. DOI: 10.3390/app14062414.

[39] Y. M. Al-Younes, M. A. Al-Jarrah, and A. A. Jhemi, "Linear vs. nonlinear control techniques for a quadrotor vehicle," in *7th International Symposium on Mechatronics and its Applications*, 2010, pp. 1–10.

[40] A. Ermeydan and A. Kaba, "Feedback linearization control of a quadrotor," in *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2021, pp. 287–290. DOI: 10.1109/ISMSIT52890.2021.9604617.

[41] Z. Shulong, A. Honglei, Z. Daibing, and S. Lincheng, "A new feedback linearization lqr control for attitude of quadrotor," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, 2014, pp. 1593–1597. DOI: 10.1109/ICARCV.2014.7064553.

[42] F. Yousefi and S. B. Monfared, "Cascade feedback linearization controller with actuators dynamic for trajectory tracking of flying robot," in *2018 8th Conference of AI Robotics and 10th RoboCup Iranopen International Symposium (IRANOPEN)*, 2018, pp. 46–51. DOI: 10.1109/RIOS.2018.8406630.

[43] A. Jebelli, M. C. E. Yagoub, and B. S. Dhillon, "Feedback linearization approach to fault tolerance for a micro quadrotor," in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 165–168. DOI: 10.1109/ICIT.2018.8352170.

[44] M. Sadiq, R. Hayat, K. Zeb, A. Al-Durra, and Z. Ullah, "Robust feedback linearization based disturbance observer control of quadrotor uav," *IEEE Access*, vol. 12, pp. 17 966–17 981, 2024. DOI: 10.1109/ACCESS.2024.3360333.

[45] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3255–3260. DOI: 10.1109/IROS.2006.282433.

[46] A. Das, F. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1, pp. 127–151, Sep. 2009, ISSN: 1573-0409. DOI: 10.1007/s10846-009-9331-0.

[47] Z. Fang and W. Gao, "Adaptive integral backstepping control of a micro-quadrotor," in *2011 2nd International Conference on Intelligent Control and Information Processing*, vol. 2, 2011, pp. 910–915. DOI: 10.1109/ICICIP.2011.6008382.

[48] H. Hassani, A. Mansouri, and A. Ahaitouf, "Design and real-time implementation of a robust backstepping non-singular integral terminal sliding mode attitude control for a quadrotor aircraft," *Journal of Vibration and Control*, p. 10 775 463 241 240 641, 2024.

[49] A. Kapnopoulos, C. Kazakidis, and A. Alexandridis, "Quadrotor trajectory tracking based on backstepping control and radial basis function neural networks," *Results in Control and Optimization*, vol. 14, p. 100 335, 2024.

[50] A. Belmouhoub, Y. Bouzid, S. Medjmadj, S. Hocine Derrouaoui, and M. Guiatni, "Backstepping control merged with disturbances observer for quadrotor with rotating arms," *Unmanned Systems*, vol. 12, no. 01, pp. 61–74, 2024.

[51] C. Nicol, C. Macnab, and A. Ramirez-Serrano, "Robust adaptive control of a quadrotor helicopter," *Mechatronics*, vol. 21, no. 6, pp. 927–938, 2011, ISSN: 0957-4158. DOI: https://doi.org/10.1016/j.mechatronics.2011.02.007.

[52] I. Palunko and R. Fierro, "Adaptive control of a quadrotor with dynamic changes in the center of gravity," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 2626–2631, 2011, 18th IFAC World Congress, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20110828-6-IT-1002.02564.

[53] M. Schreier, "Modeling and adaptive control of a quadrotor," in *2012 IEEE International Conference on Mechatronics and Automation*, 2012, pp. 383–390. DOI: 10.1109/ICMA.2012.6282874.

[54] S. Islam, P. X. Liu, and A. El Saddik, "Nonlinear adaptive control for quadrotor flying vehicle," *Nonlinear Dynamics*, vol. 78, no. 1, pp. 117–133, Oct. 2014, ISSN: 1573-269X. DOI: 10.1007/s11071-014-1425-y. [Online]. Available: https://doi.org/10.1007/s11071-014-1425-y.

[55] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive control of quadrotor uavs: A design trade study with flight evaluations," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1400–1406, 2013. DOI: 10.1109/TCST.2012.2200104.

[56] I. H. Imran, R. Stolkin, and A. Montazeri, "Adaptive control of quadrotor unmanned aerial vehicle with time-varying uncertainties," *IEEE Access*, vol. 11, pp. 19 710–19 724, 2023. DOI: 10.1109/ACCESS.2023.3243835.

[57] K. S. Kumar, M. Rasheed, and R. M. M. Kumar, "Design and implementation of fuzzy logic controller for quad rotor uav," in *2nd international conference on research in science, engineering and technology (ICRSET'2014). Dubai*, 2014, pp. 114–120.

[58] M. Santos, V. Lopez, and F. Morata, "Intelligent fuzzy controller of a quadrotor," in *2010 IEEE international conference on intelligent systems and knowledge engineering*, IEEE, 2010, pp. 141–146.

[59] Y. Deia, M. Kidouche, and A. Ahriche, "Fully decentralized fuzzy sliding mode control with chattering elimination for a quadrotor attitude," in *2015*

4th International Conference on Electrical Engineering (ICEE), IEEE, 2015, pp. 1–6.

[60] C. Huang, "Design of decoupling fuzzy logic controller for quadrotor uav," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1684, 2020, p. 012 155.

[61] A. S. H. Loayza and C. G. P. Zuñiga, "Design of a fuzzy sliding mode controller for the autonomous path-following of a quadrotor," *IEEE Latin America Transactions*, vol. 17, no. 06, pp. 962–971, 2019.

[62] K. Liu, P. Yang, R. Wang, L. Jiao, T. Li, and J. Zhang, "Observer-based adaptive fuzzy finite-time attitude control for quadrotor uavs," *IEEE Transactions on Aerospace and Electronic Systems*, 2023.

[63] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle III, *Process dynamics and control*. John Wiley & Sons, 2016.

[64] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, 2023.

[65] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.

[66] P. N. Chikasha and C. Dube, "Adaptive model predictive control of a quadrotor," *IFAC-PapersOnLine*, vol. 50, no. 2, pp. 157–162, 2017.

[67] R. Xu and U. Ozguner, "Sliding mode control of a quadrotor helicopter," in *Proceedings of the 45th IEEE Conference on Decision and Control*, IEEE, 2006, pp. 4957–4962.

[68] E.-H. Zheng, J.-J. Xiong, and J.-L. Luo, "Second order sliding mode control for a quadrotor uav," *ISA transactions*, vol. 53, no. 4, pp. 1350–1356, 2014.

[69] H. Ríos, R. Falcón, O. A. González, and A. Dzul, "Continuous sliding-mode control strategies for quadrotor robust tracking: Real-time application," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1264–1272, 2019. DOI: 10.1109/TIE.2018.2831191.

[70] J.-J. Xiong and G. Zhang, "Discrete-time sliding mode control for a quadrotor uav," *Optik*, vol. 127, no. 8, pp. 3718–3722, 2016.

[71] I. H. Imran, A. M. Memon, D. F. Kurtulus, S. Goli, and L. M. Alhems, "Extended discrete-time quasi-sliding mode control for vtol uav in the presence of uncertain disturbances," *IEEE Access*, vol. 11, pp. 55 630–55 643, 2023. DOI: 10.1109/ACCESS.2023.3280543.

[72] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999. DOI: 10.1109/70.795787.

[73] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem," in *42nd ieee international conference on decision and control (ieee cat. no. 03ch37475)*, IEEE, vol. 2, 2003, pp. 1508–1513.

[74] W. Ren, H. Chao, W. Bourgeous, N. Sorensen, and Y. Chen, "Experimental validation of consensus algorithms for multivehicle cooperative control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 745–752, 2008. DOI: 10.1109/TCST.2007.912239.

[75] X. Wei, D. Fengyang, Z. Qingjie, Z. Bing, and S. Hongchang, "A new fast consensus algorithm applied in rendezvous of multi-uav," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 2015, pp. 55–60. DOI: 10.1109/CCDC.2015.7161666.

[76] X. Peng and Z. Geng, "Distributed rendezvous and consensus control of multiple unicycle-type vehicles under directed graphs," in *2019 IEEE Inter-*

national Conference on Industrial Technology (ICIT), 2019, pp. 1436–1441. DOI: 10.1109/ICIT.2019.8755110.

[77] T. Bento da Silva, M. V. M. Souza e Silva, and A. S. Brandão, "Consensus-based navigation of a uav formation," in *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2015, pp. 219–224. DOI: 10.1109/RED-UAS.2015.7441010.

[78] E. Restrepo, A. Loría, I. Sarras, and J. Marzat, "Robust consensus of high-order systems under output constraints: Application to rendezvous of underactuated uavs," *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 329–342, 2023. DOI: 10.1109/TAC.2022.3144107.

[79] G. Wang, Z. Zuo, P. Li, and Y. Shen, "Controllers for multiagent systems with input amplitude and rate constraints and their application to quadrotor rendezvous," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 6354–6364, 2025. DOI: 10.1109/TASE.2024.3443402.

[80] A. S. Jaimes B and M. Jamshidi, "Consensus-based and network control of uavs," in *2010 5th International Conference on System of Systems Engineering*, 2010, pp. 1–6. DOI: 10.1109/SYSOSE.2010.5544106.

[81] A. Bemporad and C. Rocchi, "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 900–11 906, 2011, 18th IFAC World Congress, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20110828-6-IT-1002.00942.

[82] J. Zhang, J. Yan, and P. Zhang, "Multi-uav formation control based on a novel back-stepping approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 2437–2448, 2020.

[83] Z. Wang, Q. Fei, and B. Wang, "Distributed adaptive sliding mode formation control for multiple unmanned aerial vehicles," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 2105–2110. DOI: 10.1109/CCDC49329.2020.9164781.

[84] W. Shang, G. Jing, D. Zhang, T. Chen, and Q. Liang, "Adaptive fixed time nonsingular terminal sliding-mode control for quadrotor formation with obstacle and inter-quadrotor avoidance," *IEEE Access*, vol. 9, pp. 60 640–60 657, 2021.

[85] Y. Xin, H. Ni, Z. Huang, Y. He, and S. Xia, "Anti-disturbance formation control of quadrotor uavs based on terminal sliding mode control," in *2023 China Automation Congress (CAC)*, 2023, pp. 5381–5385. DOI: 10.1109/CAC59555.2023.10450878.

[86] J. Wang, L. Han, X. Dong, Q. Li, and Z. Ren, "Distributed sliding mode control for time-varying formation tracking of multi-uav system with a dynamic leader," *Aerospace Science and Technology*, vol. 111, p. 106 549, 2021.

[87] A. Modi, N. Joshi, and A. Mehta, "Robust discrete-time super twisting formation protocol for a 6-dof quadcopter swarm," *ISA transactions*, vol. 143, pp. 177–187, 2023.

[88] W. Zhao, H. Liu, K. P. Valavanis, and F. L. Lewis, "Fault-tolerant formation control for heterogeneous vehicles via reinforcement learning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 4, pp. 2796–2806, 2022. DOI: 10.1109/TAES.2021.3139260.

[89] W. Zhao, H. Liu, J. Lü, Q. Gao, and G. Feng, "Data-driven optimal formation control for multiple nonlinear quadrotors with switching topologies," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 2538–2548, 2025. DOI: 10.1109/TVT.2023.3278701.

[90] L. Briñón-Arranz, L. Schenato, and A. Seuret, "Distributed source seeking via a circular formation of agents under communication constraints," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 104–115, 2016. DOI: 10.1109/TCNS.2015.2428391.

[91] S. Zhuo and C. Li, "Source seeking of multi-uav with obstacle avoidance," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 6955–6959. DOI: 10.23919/ChiCC.2018.8483618.

[92] J. Han, "Small unmanned aircraft systems for cooperative source seeking with fractional order potential fields," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 6677–6683. DOI: 10.1109/CCDC.2018.8408307.

[93] M. Sahal, T. Agustinah, and A. Jazidie, "Switching formation and topology in cooperative multi-agent source seeking using gradient estimation," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019, pp. 151–156. DOI: 10.1109/ICAIIT.2019.8834525.

[94] J. Han and Y. Chen, "Cooperative source seeking and contour mapping of a diffusive signal field by formations of multiple uavs," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013, pp. 35–40. DOI: 10.1109/ICUAS.2013.6564671.

[95] R. K. Lee, C. A. Kitts, M. A. Neumann, and R. T. Mcdonald, "Multiple uav adaptive navigation for three-dimensional scalar fields," *IEEE Access*, vol. 9, pp. 122 626–122 654, 2021. DOI: 10.1109/ACCESS.2021.3107854.

[96] Z. Jin, H. Li, Z. Qin, and Z. Wang, "Gradient-free cooperative source-seeking of quadrotor under disturbances and communication constraints," *IEEE Transactions on Industrial Electronics*, vol. 72, no. 2, pp. 1969–1979, 2025. DOI: 10.1109/TIE.2024.3423337.

[97] J. Barbedo, "A review on the use of unmanned aerial vehicles and imaging sensors for monitoring and assessing plant stresses," *Drones*, vol. 3, no. 2, p. 40, Apr. 2019.

[98] S. S. Congress, A. J. Puppala, and C. L. Lundberg, "Total system error analysis of UAV-CRP technology for monitoring transportation infrastructure assets," *Engineering Geology*, vol. 247, pp. 104–116, Dec. 2018.

[99] T. Burrell, C. West, S. D. Monk, A. Montezeri, and C. J. Taylor, "Towards a Cooperative Robotic System for Autonomous Pipe Cutting in Nuclear Decommissioning," in *2018 UKACC 12th International Conference on Control, CONTROL 2018*, Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 283–288, ISBN: 9781538628645. DOI: 10.1109/CONTROL.2018.8516841.

[100] A. Taylor, C. Hope, N. Smart, D. Morris, K. Warren, and A. Banford, "A programme of technology demonstrations for decomissioning at Sellafield," *Nuclear Future*, vol. 12, no. 2, pp. 30–34, 2017.

[101] Game Changers, *Challenges*, n.d. [Online]. Available: https://www.gamechangers.technology/challenges/ (visited on 08/20/2019).

[102] F. Mascarich, T. Wilson, C. Papachristos, and K. Alexis, "Radiation source localization in gps-denied environments using aerial robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6537–6544. DOI: 10.1109/ICRA.2018.8460760.

[103] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013, ISSN: 09295593. DOI: 10.1007/s10514-012-9321-0.

[104] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[105] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions*

on Robotics, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, ISSN: 15523098. DOI: 10.1109/TRO.2017.2705103. arXiv: 1610.06475.

[106] K. Kamarudin, S. M. Mamduh, A. Y. Md Shakaff, and A. Zakaria, "Performance analysis of the Microsoft Kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques," *Sensors (Switzerland)*, vol. 14, no. 12, pp. 23 365–23 387, Dec. 2014.

[107] J. J. Xiong and G. Zhang, "Sliding mode control for a quadrotor UAV with parameter uncertainties," in *Proceedings - 2016 the 2nd International Conference on Control, Automation and Robotics, ICCAR 2016*, Institute of Electrical and Electronics Engineers Inc., Jun. 2016, pp. 207–212.

[108] H. Lee and V. I. Utkin, "Chattering suppression methods in sliding mode control systems," *Annual Reviews in Control*, vol. 31, no. 2, pp. 179–188, 2007, ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2007.08.001.

[109] V. Utkin and H. Lee, "Chattering problem in sliding mode control systems," *IFAC Proceedings Volumes*, vol. 39, no. 5, p. 1, Jan. 2006.

[110] H. Nemati and A. Montazeri, "Analysis and design of a multi-channel time-varying sliding mode controller and its application in unmanned aerial vehicles," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 244–249, Jan. 2018.

[111] H. Nemati, M. Oishi, N. Kobayashi, R. Nakata, and S. Hokamoto, *New continuous sliding mode design for spacecraft attitude control*, Presented at the 24th Workshop on JAXA Astrodynamics and Flight Mechanics, Japan, 2014.

[112] H. Nemati and A. Montazeri, "Design and development of a novel controller for robust attitude stabilisation of an unmanned air vehicle for nuclear environments," in *2018 UKACC 12th International Conference on Control*, 2018, pp. 373–378. DOI: 10.1109/CONTROL.2018.8516729.

[113] Y. Li, Y. Qin, F. Wang, F. Guo, and J. T. Yeow, "Global fast terminal sliding mode control for a quadrotor UAV," in *Proceedings of the 15th IEEE Conference on Industrial Electronics and Applications, ICIEA 2020*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020, pp. 1820–1825.

[114] W. Wang and X. Yu, "Chattering free and nonsingular terminal sliding mode control for attitude tracking of a quadrotor," in *Proceedings of the 29th Chinese Control and Decision Conference, CCDC 2017*, Institute of Electrical and Electronics Engineers Inc., Jul. 2017, pp. 719–723.

[115] H. Nemati and A. Montazeri, "Output Feedback Sliding Mode Control of Quadcopter Using IMU Navigation," in *Proceedings - 2019 IEEE International Conference on Mechatronics, ICM 2019*, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 634–639, ISBN: 9781538669594. DOI: 10.1109/ICMECH.2019.8722899.

[116] I. Imran, R. Stolkin, and A. Montazeri, "Adaptive Closed-Loop Identification and Tracking Control of an Aerial Vehicle with Unknown Inertia Parameters," in *19th IFAC Symposium on System Identification: learning models for decision and control, SYSID 2021*, Dec. 2021.

[117] N. S. Nokhodberiz, H. Nemati, and A. Montazeri, "Event-triggered based state estimation for autonomous operation of an aerial robotic vehicle," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2348–2353, Sep. 2019.

[118] H. Ghadiri and A. Montazeri, "Adaptive integral terminal sliding mode control for the nonlinear active vehicle suspension system under external disturbances and uncertainties," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 2665–2670, 2022, 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.10.112.

[119] A. Noordin, A. Basri, and Z. Mohamed, "Sensor fusion for attitude estimation and pid control of quadrotor uav," *International Journal of Electrical and*

Electronic Engineering Telecommunications., vol. 7, pp. 183–189, Oct. 2018. DOI: 10.18178/ijeetc.7.4.183-189.

[120] Y. Pan, C. Yang, L. Pan, and H. Yu, "Integral Sliding Mode Control: Performance, Modification, and Improvement," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3087–3096, Jul. 2018. DOI: 10.1109/TII.2017.2761389.

[121] S. P. Bhat and D. S. Bernstein, "Finite-time stability of continuous autonomous systems," *SIAM Journal on Control and Optimization*, vol. 38, no. 3, pp. 751–766, 2000. DOI: 10.1137/S0363012997321358. eprint: https://doi.org/10.1137/S0363012997321358.

[122] S. Yu, X. Yu, B. Shirinzadeh, and Z. Man, "Continuous finite-time control for robotic manipulators with terminal sliding mode," *Automatica*, vol. 41, no. 11, pp. 1957–1964, 2005, ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2005.07.001.

[123] H. Liu, B. Gao, Y. Shen, F. Hussain, D. Addis, and C. Kai Pan, "Comparison of Sick and Hokuyo UTM-30LX laser sensors in canopy detection for variable-rate sprayer," *Information Processing in Agriculture*, vol. 5, no. 4, pp. 504–515, Dec. 2018.

[124] Slamtec, *RPLIDAR-A1 360°Laser Range Scanner*, n.d. [Online]. Available: https://www.slamtec.com/en/Lidar/A1 (visited on 04/25/2021).

[125] Intel, *External synchronization of Intel RealSense™ depth cameras*, 2020. [Online]. Available: https://dev.intelrealsense.com/docs/external-synchronization-of-intel-realsense-depth-cameras (visited on 04/25/2021).

[126] J. S. Han, T. I. Kim, T. H. Oh, *et al.*, "Error-dynamics-based performance shaping methodology for discrete-time sliding mode control with disturbance observer," 15, vol. 52, Elsevier B.V., Sep. 2019, pp. 460–464.

[127] M. K. Sarkar, Arkdev, and S. S. K. Singh, "Sliding mode control: A higher order and event triggered based approach for nonlinear uncertain systems," in *2017 8th Industrial Automation and Electromechanical Engineering Conference, IEMECON 2017*, Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 208–211.

[128] B. Tian, J. Cui, H. Lu, L. Liu, and Q. Zong, "Attitude control of UAVs based on event-triggered supertwisting algorithm," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1029–1038, Feb. 2021.

[129] W. Gao, Y. Wang, and A. Homaifa, "Discrete-time variable structure control systems," *IEEE Transactions on Industrial Electronics*, vol. 42, no. 2, pp. 117–122, 1995. DOI: 10.1109/41.370376.

[130] Y. Wang, Q. Wu, Y. Wang, and D. Yu, "Consensus algorithm for multiple quadrotor systems under fixed and switching topologies," *Journal of Systems Engineering and Electronics*, vol. 24, no. 5, pp. 818–827, 2013, ISSN: 10044132.

[131] T. Liu and Z. P. Jiang, "Distributed formation control of nonholonomic mobile robots without global position measurements," *Automatica*, vol. 49, no. 2, pp. 592–600, Feb. 2013, ISSN: 0005-1098.

[132] G. Wen, Z. Duan, Z. Li, and G. Chen, "Flocking of multi-agent dynamical systems with intermittent nonlinear velocity measurements," *International Journal of Robust and Nonlinear Control*, vol. 22, no. 16, pp. 1790–1805, Nov. 2012, ISSN: 1099-1239.

[133] F. Wu, J. Chen, and Y. Liang, "Leader-Follower Formation Control for Quadrotors," *IOP Conference Series: Materials Science and Engineering*, vol. 187, no. 1, p. 012 016, Mar. 2017, ISSN: 1757-899X.

[134] H. M. Abdoli, M. Najafi, I. Izadi, and F. Sheikholeslam, "Sliding mode approach for formation control of multi-agent systems with unknown nonlinear interactions," *ISA Transactions*, vol. 80, pp. 65–72, Sep. 2018, ISSN: 0019-0578.

[135] C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," *arXiv preprint arXiv:1608.05786*, 2016.

[136] Bitcraze, *Crazyflie 2.1*, https://www.bitcraze.io/products/crazyflie-2-1/, Accessed: 2024-06-25.

[137] Bitcraze, *GitHub - bitcraze/crazyflie-lib-python: Python library to communicate with Crazyflie — github.com*, https://github.com/bitcraze/crazyflie-lib-python.

[138] Bitcraze, *Lighthouse positioning deck*, https://www.bitcraze.io/products/lighthouse-positioning-deck/, Accessed: 2024-06-25, 2024.

[139] A. Taffanel, B. Rousselot, J. Danielsson, *et al.*, *Lighthouse positioning system: Dataset, accuracy, and precision for uav research*, Apr. 2021. [Online]. Available: https://arxiv.org/abs/2104.11523.

[140] B. Bird, M. Nancekievill, A. West, *et al.*, "Vega—a small, low cost, ground robot for nuclear decommissioning," *Journal of Field Robotics*, vol. 39, no. 3, pp. 232–245, 2022.

[141] S. N. Ahmed, *Physics and engineering of radiation detection*. Academic Press, 2007.

[142] S. Schraml, M. Hubner, P. Taupe, M. Hofstätter, P. Amon, and D. Rothbacher, "Real-time gamma radioactive source localization by data fusion of 3d-lidar terrain scan and radiation data from semi-autonomous uav flights," *Sensors*, vol. 22, no. 23, p. 9198, 2022.

[143] Office for Nuclear Regulation, *Transport guidance on dose rate monitoring*, Accessed: 2024-06-24, 2024. [Online]. Available: https://www.onr.org.uk/media/4u5pxzkr/transport-guidance-monitoring-ti.pdf.

[144] Sellafield Ltd, *Annual review of environmental performance 2022/23*, https://www.gov.uk/government/publications/sellafield-ltd-annual-review-of-environmental-performance-202223/sellafield-ltd-

annual-review-of-environmental-performance-202223, Accessed: 2024-06-25, 2023.

[145] S. K. Foad Marashi, F. Farhadi, and N. Fallahpour, "A lab-scale set up for thermal radiation experiments with cold junction compensation," *Education for Chemical Engineers*, vol. 7, no. 4, e203–e209, 2012, ISSN: 1749-7728. DOI: https://doi.org/10.1016/j.ece.2012.08.002.

[146] A. West, I. Tsitsimpelis, M. Licata, *et al.*, "Use of gaussian process regression for radiation mapping of a nuclear reactor with a mobile robot," *Scientific Reports*, vol. 11, no. 1, p. 13 975, Jul. 2021, ISSN: 2045-2322. DOI: 10.1038/s41598-021-93474-4. [Online]. Available: https://doi.org/10.1038/s41598-021-93474-4.

[147] D. Mansfield and A. Montazeri, "A survey on autonomous environmental monitoring approaches: Towards unifying active sensing and reinforcement learning," *Frontiers in Robotics and AI*, vol. 11, 2024, ISSN: 2296-9144. DOI: 10.3389/frobt.2024.1336612.

[148] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, *Gaussian processes — scikit-learn 1.2.2 documentation*, https://scikit-learn.org/stable/modules/gaussian_process.html, Accessed: 2024-06-25, 2023.

Appendix A

Discrete-Time Sliding Mode Swarm Python Code

```
import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
from cflib.crazyflie.log import LogConfig
from cflib.crazyflie.swarm import CachedCfFactory
import time
import math
import csv
import pandas as pd
import numpy as np

import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.swarm import CachedCfFactory
from cflib.crazyflie.log import LogConfig
from cflib.crazyflie.swarm import Swarm
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
from cflib.crazyflie.syncLogger import SyncLogger
```

```
from cflib.crazyflie.syncLogger import SyncLogger


URI1 = 'radio://0/80/2M/E7E7E7E701'
URI2 = 'radio://0/80/2M/E7E7E7E702'
URI3 = 'radio://0/80/2M/E7E7E7E703'

uris = [
   'radio://0/80/2M/E7E7E7E701',
   'radio://0/80/2M/E7E7E7E702',
   'radio://0/80/2M/E7E7E7E703',
    ]

sequence1 = [

    (1, 0.25, 0.75, 3.0),
    (-0.25, 0.25, 0.5, 3.0),
    (-0.0, -0.0, 0.5, 3.0),]

sequence2 = [
    (2, 0.25, 0.75, 3.0),
    (-0.25, -0.25, 0.75, 3.0),
    (0.25, -0.25, 0.1, 3.0),
    (0.25, 0.25, 0.5, 3.0),]

sequence3 = [
    (3, -0.25, 1, 3.0),
    (0.25, -0.25, 1, 3.0),
    (0.25, 0.25, 1, 3.0),
    (-0.25, 0.25, 1, 3.0),]
```

```
seq_args = {
    'radio://0/80/2M/E7E7E7E701': [sequence1],
    'radio://0/80/2M/E7E7E7E702': [sequence2],
    'radio://0/80/2M/E7E7E7E703': [sequence3],}



#Sample Time
T = 0.02

xd_ref = [0,0,0,0]
yd_ref = [0,0,0,0]
pi = 3.142
radius = 0.3
spin_duration = 100
vlpx = 0
vlpy = 0
vldx = 0
vldy = 0
vlpx_old = 0
vlpy_old = 0
#initial velocities of leader

vlpz = 0.3
leaderpos = (0,0,0.3)
deltax1 = radius*math.cos(0) #distance of drone 0 to leader x
deltay1 = radius*math.sin(0) # distance of drone 0 to leader
   y
deltax2 = radius*math.cos(2*pi*1/3) # distance of drone 1 to
   leader x
deltay2 = radius*math.sin(2*pi*1/3) # disctance of drone 1 to
```

```
    leader y
deltax3 = radius*math.cos(2*pi*2/3) # distance of drone 2 to
   the leader x
deltay3 = radius*math.sin(2*pi*2/3) #distance of drone 2 to
   the leader y
gain = 1.7 #gain for desired velocity

#desired formation vectors
delta_x = np.array([deltax1, deltax2, deltax3])
delta_y = np.array([deltay1, deltay2, deltay3])

#gains
ax = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] )  # alpha x gain

ay = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] ) # alpha x gain

mux = 4*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

muy = 4*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

etax = 1*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )
```

```
etay = 1*np.array([[1, 0, 0],
                   [0, 1, 0],
                   [0, 0, 1]] )

'''


BEST GAINS SO FAR

ax = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] )  # alpha x gain

ay = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] ) # alpha x gain

mux = 5*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

muy = 5*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

etax = 1*np.array([[1, 0, 0],
                   [0, 1, 0],
                   [0, 0, 1]] )

etay = 1*np.array([[1, 0, 0],
```

```
                    [0, 1, 0],
                    [0, 0, 1]] )



'''


#Graph Connectivity

# Identity vector
eyebar = np.array([1, 1, 1])

A = np.array([[0,1,0],
              [1,0,1],
              [0,1,0]])

D = np.array([[1,0,0],
              [0,2,0],
              [0,0,1]])

L = D-A

# B matrix (connections to the leader)
B = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]])

#assuming just enough thrust for hover
m1, m2, m3 = 1, 1, 1
u11, u12, u13 = 9.81,9.81,9.81

u1mat = np.array([[u11, 0, 0], [0, u12,0], [0, 0, u13]])
```

```
mVect = np.array([[1/m1, 0, 0], [0, 1/m2, 0], [0, 0, 1/m3]])

u1m = T*u1mat@mVect

#lists for logging data to csv
dronexyz1_list = []
dronexyz2_list = []
dronexyz3_list = []
dronedxyz1_list = []
dronedxyz2_list = []
dronedxyz3_list = []
dronerpy1_list = []
dronerpy2_list = []
dronerpy3_list = []
dronepqr1_list = []
dronepqr2_list = []
dronepqr3_list = []
leaderxyz_list = []
errors_list = []
ss_list = []
controlInputs_list = []

def log_stab_callback(uri, timestamp, data, log_conf):
        x = float(data['stateEstimate.x'])
        y = float(data['stateEstimate.y'])
        z = float(data['stateEstimate.z'])
        vx = float(data['stateEstimate.vx'])
        vy = float(data['stateEstimate.vy'])
        vz = float(data['stateEstimate.vz'])

def land(cf, position):
```

```
    landing_time = 1.0
    sleep_time = 0.1
    steps = int(landing_time / sleep_time)
    vz = position / landing_time

    print(vz)

    for _ in range(steps):
        cf.commander.send_velocity_world_setpoint(0, 0, vz,
           0)
        time.sleep(sleep_time)

    cf.commander.send_stop_setpoint()
    # Make sure that the last packet leaves before the link
       is closed
    # since the message queue is not flushed before closing
    time.sleep(0.1)

def simple_log_async(scf,droneNum):
    lg_vars = {
        'stateEstimate.x': 'FP16',
        'stateEstimate.y': 'FP16',
        'stateEstimate.z': 'FP16',
        'stateEstimate.vx': 'FP16',
        'stateEstimate.vy': 'FP16',
        'stateEstimate.vz': 'FP16',
        'stabilizer.roll': 'FP16',
        'stabilizer.pitch': 'FP16',
        'stabilizer.yaw': 'FP16',
    }
```

```
lg_stab = LogConfig(name='stateEstimate', period_in_ms
   =20)
for key in lg_vars:
    lg_stab.add_variable(key, lg_vars[key])

cf = scf.cf
cf.log.add_config(lg_stab)
idx = 0

with SyncLogger(scf, lg_stab) as logger:
    for entry in logger:
        idx += 1
        lg_stab.data_received_cb.add_callback(lambda t, d
           , l: log_stab_callback(cf.link_uri, t, d, l))
        x = entry[1]['stateEstimate.x']
        y = entry[1]['stateEstimate.y']
        z = entry[1]['stateEstimate.z']
        vx = entry[1]['stateEstimate.vx']
        vy = entry[1]['stateEstimate.vy']
        vz = entry[1]['stateEstimate.vz']
        phi = entry[1]['stabilizer.roll']
        theta = entry[1]['stabilizer.pitch']
        psi = entry[1]['stabilizer.yaw']

        #saving variables to list for plotting later:

        posdrones[scf.cf.link_uri] = (x,y,z)
        veldrones[scf.cf.link_uri] = (vx,vy,vz)
        attdrones[scf.cf.link_uri] = (phi,theta,psi)
```

```
uri = scf.cf.link_uri

if uri == URI1:
    dronexyz1_list.append(posdrones[URI1])
    dronedxyz1_list.append(veldrones[URI1])
    dronerpy1_list.append(attdrones[URI1])

elif uri == URI2:
    dronexyz2_list.append(posdrones[URI2])
    dronedxyz2_list.append(veldrones[URI2])
    dronerpy2_list.append(attdrones[URI2])

elif uri == URI3:
    dronexyz3_list.append(posdrones[URI3])
    dronedxyz3_list.append(veldrones[URI3])
    dronerpy3_list.append(attdrones[URI3])



if idx > 5:
####Here is where we run the main control loop
    #print("")
    formation_control(scf,posdrones,veldrones,idx
       ,droneNum)
if abs(x) > 0.9:
    idx = 3300
elif abs(y) > 0.9:
    idx = 3300
if idx == 3300:
    if uri == URI1:
        print("saving drone 1 data")
```

```
            dp1=pd.DataFrame(dronexyz1_list)
            dp1.to_csv('dronexyz1_list.csv')
            dv1=pd.DataFrame(dronedxyz1_list)
            dv1.to_csv('dronedxyz1_list.csv')
            da1=pd.DataFrame(dronerpy1_list)
            da1.to_csv('dronerpy1_list.csv')


            print("CSVs saved!")
        elif uri == URI2:
            print("saving to dronexyz2_list.csv")

            dp2=pd.DataFrame(dronexyz2_list)
            dp2.to_csv('dronexyz2_list.csv')
            dv2=pd.DataFrame(dronedxyz2_list)
            dv2.to_csv('dronedxyz2_list.csv')
            da2=pd.DataFrame(dronerpy2_list)
            da2.to_csv('dronerpy2_list.csv')


            print("CSVs saved!")
        elif uri == URI3:
            print("saving to dronexyz3_list.csv")

            dp3=pd.DataFrame(dronexyz3_list)
            dp3.to_csv('dronexyz3_list.csv')
            dv3=pd.DataFrame(dronedxyz3_list)
            dv3.to_csv('dronedxyz3_list.csv')
            da3=pd.DataFrame(dronerpy3_list)
            da3.to_csv('dronerpy3_list.csv')
```

```
                print("CSVs saved!")

def get_control_inputs(uri,xyz1,xyz2,xyz3,dxyz1,dxyz2,dxyz3,
    vlpx,vlpy,vldx, vldy, idx):

    x1,y1,z1 = xyz1[0],xyz1[1],xyz1[2]
    x2,y2,z2 = xyz2[0],xyz2[1],xyz2[2]
    x3,y3,z3 = xyz3[0],xyz3[1],xyz3[2]

    dx1,dy1,dz1 = dxyz1[0],dxyz1[1],dxyz1[2]
    dx2,dy2,dz2 = dxyz2[0],dxyz2[1],dxyz2[2]
    dx3,dy3,dz3 = dxyz3[0],dxyz3[1],dxyz3[2]

    x0x = vlpx
    x0y = vlpy
    dx0x = vldx
    dx0y = vldy


    #dx1, dx2, dy1, dy2, dx3, dy3 = dx1*T, dx2*T, dy1*T, dy2*
       T, dx3*T, dy3*T
    # Definitions of the graph
    xVect = np.array([x1, x2, x3])  # vector of agents x
       positions
    yVect = np.array([y1, y2, y3])  # y positions
    dxVect = np.array([dx1, dx2, dx3])  # x velocities
    dyVect = np.array([dy1, dy2, dy3])  # y velocities

    # Define errors
    e1x = -(L+B) @ (xVect - delta_x - x0x*eyebar)  # x pos
```

```
    error
e2x = -(L+B) @ (dxVect - eyebar*dx0x)  # x vel error
e1y = -(L+B) @ (yVect - delta_y - eyebar*x0y) # y pos
    error
e2y = -(L+B) @ (dyVect - eyebar*dx0y)  # y vel

e1x_plus = -(L+B) @ ((xVect + T*dxVect) - (delta_x) - (
    eyebar*x0x + T*eyebar*dx0x))
e1y_plus = -(L+B) @ ((yVect + T*dyVect) - (delta_y) - (
    eyebar*x0y + T*eyebar*dx0y))

ssx = ax@(e1x) + e2x
ssy = ay@(e1y) + e2y

sgnssx = np.array([np.sign(ssx[0]),
                   np.sign(ssx[1]),
                   np.sign(ssx[2])])

sgnssy = np.array([np.sign(ssy[0]),
                   np.sign(ssy[1]),
                   np.sign(ssy[2])])


# dxplus = np.linalg.inv(-(L+B))@(T*(-mux*ssx) + (L+B)@(
    ax@((xVect+T*dxVect)-delta_x-(eyebar*x0x+T*eyebar*dx0x
    ))-eyebar*dx0x-ax@(xVect-delta_x-eyebar*x0x)-(dxVect-
    eyebar*dx0x)))
# dyplus = np.linalg.inv(-(L+B))@(T*(-muy*ssy) + T*
    etay@sgnssy + (L+B)@(ay@((yVect+T*dyVect)-delta_y-(
    eyebar*x0y+T*eyebar*dx0y))-eyebar*dx0y-ay@(yVect-
    delta_y-eyebar*x0y)-(dyVect-eyebar*dx0y)))
```

```
# uxmat = np.linalg.inv(u1m)@(dxplus - dxVect)
# uymat = np.linalg.inv(u1m)@(dyplus - dyVect)

uxmat = np.linalg.inv(u1m) @ (T*(dxVect - eyebar*dx0x) -
   np.linalg.inv(L+B) @ ((T*mux*ssx) + T*etax@sgnssx + ax
   *e1x_plus - ssx))
uymat = np.linalg.inv(u1m) @ (T*(dyVect - eyebar*dx0y) -
   np.linalg.inv(L+B) @ ((T*muy*ssy) + T*etay@sgnssy + ay
   *e1y_plus - ssy))


ux1, ux2, ux3 = uxmat
uy1, uy2, uy3 = uymat

e_all = (e1x,e1y,e2x,e2y)
ss_all = (ssx, ssy)
controlInputsAll = (-ux1[0], uy1[0],-ux2[1], uy2[1], -ux3
   [2], uy3[2])

if uri == URI1:
    controlInputs_list.append(controlInputsAll)
    ss_list.append(ss_all)
    errors_list.append(e_all)


if uri == URI1:
    return -ux1[0], uy1[0]
elif uri == URI2:
    return -ux2[1], uy2[1]
```

```
    elif uri == URI3:
        return -ux3[2], uy3[2]



def formation_control(scf,posdrones,veldrones,idx,droneNum):
    global vlpx_old
    global vlpy_old
    cf = scf.cf
    uri = scf.cf.link_uri



    xyz1 = posdrones['radio://0/80/2M/E7E7E7E701']
    xyz2 = posdrones['radio://0/80/2M/E7E7E7E702']
    xyz3 = posdrones['radio://0/80/2M/E7E7E7E703']

    dxyz1 = veldrones['radio://0/80/2M/E7E7E7E701']
    dxyz2 = veldrones['radio://0/80/2M/E7E7E7E702']
    dxyz3 = veldrones['radio://0/80/2M/E7E7E7E703']

    #xyz1 = (deltax1,deltax2,0.3)
    #xyz2 = (deltax2, deltax2, 0.3) #when drones are off,
       assume they are at their desired positions
    #xyz3 = (deltax3, deltay3, 0.3) #when drones are off,
       assume they are at their desired postision
    #dxyz1 = (0,0,0)
    #dxyz2 = (0,0,0) #when drones are off, assume they are at
        their desired veloctities
    #dxyz3 = (0,0,0)
```

```
if 300 <= idx < 3300 :
    vlpx = 0.3*math.sin(4*pi*((idx-300))/3000)
    vlpy = 0.3*math.cos(4*pi*((idx-300))/3000)
    vlpz = 0.3

    vldx = 0.3*math.sin(2*pi*((idx-300))/3000) - 0.3*math
       .sin(2*pi*((idx-1-300))/3000)
    vldy = 0.3*math.cos(2*pi*((idx-300))/3000) - 0.3*math
       .cos(2*pi*((idx-1-300))/3000)



else:
    vlpx = 0.3
    vlpy = 0.3
    vlpz = 0.3
    vldx = 0
    vldy = 0
    vldz = 0

leaderpos = (vlpx, vlpy, vlpz,vldx, vldy, 0)


print(vldx, vldy)
if uri == URI1:
    leaderxyz_list.append(leaderpos)
```

```
if 0<= idx <200:

    a = 1
    #can run some initialisation here if needed


#Takeoff Sequence, each drone goes to its desired
   position
elif 200 <= idx < 300 :
    if uri == 'radio://0/80/2M/E7E7E7E701':
        a = 1
        cf.commander.send_position_setpoint(xyz1[0],xyz1
           [1],0.3,0)   #set position to 0,0,1
        #if drone is drone 1
    elif uri == 'radio://0/80/2M/E7E7E7E702':
        cf.commander.send_position_setpoint(xyz2[0],xyz2
           [1],0.3,0)   #set position to 0,0,1
        a = 1
    elif uri == 'radio://0/80/2M/E7E7E7E703':
        a = 1
        cf.commander.send_position_setpoint(xyz3[0],xyz3
           [1],0.3,0)   #set position to 0,0,1


#Main loop where drones are controlled with formation
   controller
elif 300 <= idx < 3300 :
    if uri == URI1:
        ux,uy= get_control_inputs(uri,xyz1,xyz2,xyz3,
           dxyz1,dxyz2,dxyz3,vlpx,vlpy,vldx,vldy,idx)
        phi_d1, theta_d1 = uxuy2phidthetad(ux, uy)
```

```
            cf.commander.send_zdistance_setpoint(1*phi_d1,1*
                theta_d1, 0,0.3)

        elif uri == URI2:
            ux,uy= get_control_inputs(uri,xyz1,xyz2,xyz3,
                dxyz1,dxyz2,dxyz3,vlpx,vlpy,vldx,vldy,idx)
            phi_d2, theta_d2 = uxuy2phidthetad(ux, uy)
            cf.commander.send_zdistance_setpoint(1*phi_d2,1*
                theta_d2, 0,0.3)

        elif uri == URI3:
            ux,uy= get_control_inputs(uri,xyz1,xyz2,xyz3,
                dxyz1,dxyz2,dxyz3,vlpx,vlpy,vldx,vldy,idx)
            phi_d3, theta_d3 = uxuy2phidthetad(ux, uy)
            cf.commander.send_zdistance_setpoint(1*phi_d3,1*
                theta_d3, 0,0.3)


    #landing sequence, each drone goes to final position
    elif 3300 <= idx < 3400 :
        #print("landing")
        if uri == 'radio://0/80/2M/E7E7E7E701':
            cf.commander.send_position_setpoint(deltax1,
                deltay1,0.1,0)   #set position to 0,0,1
            a = 1
                #if drone is drone 1
        elif uri == 'radio://0/80/2M/E7E7E7E702':
            cf.commander.send_position_setpoint(deltax2,
                deltay2,0.1,0)   #set position to 0,0,1
            a = 1
```

```
        elif uri == 'radio://0/80/2M/E7E7E7E703':
            cf.commander.send_position_setpoint(deltax3,
               deltay3,0.1,0)    #set position to 0,0,1
            a = 1

    #run final landing sequence and shurdown motors
    elif 3400 == idx:
        print('powering down and saving logs')
        land(cf,-0.15)
        if uri == URI1:
            lp=pd.DataFrame(leaderxyz_list)
            lp.to_csv('leaderxyz_list.csv')
            print("adding errors and sliding surface to CSV")
            el=pd.DataFrame(errors_list)
            el.to_csv('errors_list.csv')
            ss=pd.DataFrame(ss_list)
            ss.to_csv('ss_list.csv')
            il = pd.DataFrame(controlInputs_list)
            il.to_csv('controlInputs_list.csv')


    else:
        quit()

def uxuy2phidthetad(ux, uy):

    psi = 0
    sinofphi = ux * math.sin(psi) - uy * math.cos(psi)
    #print(sinofphi)
    sinofphi = max(min(sinofphi, 1), -1)  # clamp value
       between -1 and 1
```

```
    phi_d = math.asin(sinofphi)

    sinoftheta = (ux * math.cos(psi) + uy * math.sin(psi)) /
       abs(math.cos(0))
    sinoftheta = max(min(sinoftheta, 1), -1)  # clamp value
       between -1 and 1
    theta_d = math.asin(sinoftheta)
    return phi_d, theta_d

if __name__ == '__main__':
    posdrones = dict()
    veldrones = dict()
    attdrones = dict()
    attrates = dict()
    initlog = 0

    cflib.crtp.init_drivers(enable_debug_driver=False) #
       initialize drivers
    factory = CachedCfFactory(rw_cache='./cache')

    with Swarm(uris, factory=factory) as swarm:
        swarm.parallel_safe(simple_log_async, args_dict=
           seq_args)

        # while True:
        #     #swarm.parallel_safe(simple_log_async,
           args_dict=seq_args)
        #     a = 1
```

Appendix B

Temperature Data Collection Python Code

```
import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
from cflib.crazyflie.log import LogConfig
from cflib.crazyflie.swarm import CachedCfFactory
import time
import math
import csv
import pandas as pd
import numpy as np

import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.swarm import CachedCfFactory
from cflib.crazyflie.log import LogConfig
from cflib.crazyflie.swarm import Swarm
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
from cflib.crazyflie.syncLogger import SyncLogger
```

```
from cflib.crazyflie.syncLogger import SyncLogger


URI1 = 'radio://0/80/2M/E7E7E7E703'
URI2 = 'radio://0/80/2M/E7E7E7E702'
URI3 = 'radio://0/80/2M/E7E7E7E701'

uris = [
    'radio://0/80/2M/E7E7E7E703',
    'radio://0/80/2M/E7E7E7E702',
    'radio://0/80/2M/E7E7E7E701',
     ]

sequence1 = [

     (1, 0.25, 0.75, 3.0),
     (-0.25, 0.25, 0.5, 3.0),
     (-0.0, -0.0, 0.5, 3.0),]

sequence2 = [
     (2, 0.25, 0.75, 3.0),
     (-0.25, -0.25, 0.75, 3.0),
     (0.25, -0.25, 0.1, 3.0),
     (0.25, 0.25, 0.5, 3.0),]

sequence3 = [
     (3, -0.25, 1, 3.0),
     (0.25, -0.25, 1, 3.0),
     (0.25, 0.25, 1, 3.0),
     (-0.25, 0.25, 1, 3.0),]
```

```
seq_args = {
    'radio://0/80/2M/E7E7E7E703': [sequence1],
    'radio://0/80/2M/E7E7E7E702': [sequence2],
    'radio://0/80/2M/E7E7E7E701': [sequence3],}



#Sample Time
T = 0.02

xd_ref = [0,0,0,0]
yd_ref = [0,0,0,0]
pi = 3.142
radius = 0.25
spin_duration = 100
vlpx = 0
vlpy = 0
vldx = 0
vldy = 0
vlpx_old = 0
vlpy_old = 0
#initial velocities of leader

vlpz = 0.3
leaderpos = (0,0,0.3)
deltax1 = radius*math.cos(0) #distance of drone 0 to leader x
deltay1 = radius*math.sin(0) # distance of drone 0 to leader
   y
deltax2 = radius*math.cos(2*pi*1/3) # distance of drone 1 to
   leader x
deltay2 = radius*math.sin(2*pi*1/3) # disctance of drone 1 to
```

```
    leader y
deltax3 = radius*math.cos(2*pi*2/3) # distance of drone 2 to
   the leader x
deltay3 = radius*math.sin(2*pi*2/3) #distance of drone 2 to
   the leader y
gain = 1.7 #gain for desired velocity

#desired formation vectors
delta_x = np.array([deltax1, deltax2, deltax3])
delta_y = np.array([deltay1, deltay2, deltay3])

#gains
ax = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] )  # alpha x gain

ay = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] ) # alpha x gain

mux = 4*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

muy = 4*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

etax = 1*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )
```

```
etay = 1*np.array([[1, 0, 0],
                   [0, 1, 0],
                   [0, 0, 1]] )

'''


BEST GAINS SO FAR

ax = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] )  # alpha x gain

ay = 2*np.array([[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]] ) # alpha x gain

mux = 5*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

muy = 5*np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]] )  # alpha x gain

etax = 1*np.array([[1, 0, 0],
                   [0, 1, 0],
                   [0, 0, 1]] )

etay = 1*np.array([[1, 0, 0],
```

```
                    [0, 1, 0],
                    [0, 0, 1]] )



'''


#Graph Connectivity

# Identity vector
#eyebar = np.array([1, 1, 1])

#A = np.array([[0,1,0],
#              [1,0,1],
#              [0,1,0]])

#D = np.array([[1,0,0],
#              [0,2,0],
#              [0,0,1]])

#L = D-A

# B matrix (connections to the leader)
#B = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]])

# Identity vector
eyebar = np.array([1, 1, 1])

A = np.array([[0,1,1],
              [1,0,1],
```

```
                [1,1,1]])

D = np.array([[3,0,0],
                [0,3,0],
                [0,0,3]])

L = D-A

# B matrix (connections to the leader)
B = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])

#assuming just enough thrust for hover
m1, m2, m3 = 1, 1, 1
u11, u12, u13 = 9.81,9.81,9.81

u1mat = np.array([[u11, 0, 0], [0, u12,0], [0, 0, u13]])
mVect = np.array([[1/m1, 0, 0], [0, 1/m2, 0], [0, 0, 1/m3]])

u1m = T*u1mat@mVect

#lists for logging data to csv
dronexyz1_list = []
dronexyz2_list = []
dronexyz3_list = []
dronetemp1_list = []
dronetemp2_list = []
dronetemp3_list = []
dronedxyz1_list = []
dronedxyz2_list = []
dronedxyz3_list = []
dronerpy1_list = []
```

```
dronerpy2_list = []
dronerpy3_list = []
dronepqr1_list = []
dronepqr2_list = []
dronepqr3_list = []
leaderxyz_list = []
errors_list = []
ss_list = []
controlInputs_list = []

def log_stab_callback(uri, timestamp, data, log_conf):
        x = float(data['stateEstimate.x'])
        y = float(data['stateEstimate.y'])
        z = float(data['stateEstimate.z'])
        vx = float(data['stateEstimate.vx'])
        vy = float(data['stateEstimate.vy'])
        vz = float(data['stateEstimate.vz'])

def land(cf, position):
    landing_time = 1.0
    sleep_time = 0.1
    steps = int(landing_time / sleep_time)
    vz = position / landing_time

    #print(vz)

    for _ in range(steps):
        cf.commander.send_velocity_world_setpoint(0, 0, vz,
           0)
        time.sleep(sleep_time)
```

```
    cf.commander.send_stop_setpoint()
    # Make sure that the last packet leaves before the link
       is closed
    # since the message queue is not flushed before closing
    time.sleep(0.1)

def simple_log_async(scf,droneNum):
    lg_vars = {
        'stateEstimate.x': 'FP16',
        'stateEstimate.y': 'FP16',
        'stateEstimate.z': 'FP16',
        'stateEstimate.vx': 'FP16',
        'stateEstimate.vy': 'FP16',
        'stateEstimate.vz': 'FP16',
        'stabilizer.roll': 'FP16',
        'stabilizer.pitch': 'FP16',
        'stabilizer.yaw': 'FP16',
        'baro.temp': 'float',
    }

    lg_stab = LogConfig(name='stateEstimate', period_in_ms
       =20)
    for key in lg_vars:
        lg_stab.add_variable(key, lg_vars[key])

    cf = scf.cf
    cf.log.add_config(lg_stab)
    idx = 0

    with SyncLogger(scf, lg_stab) as logger:
        for entry in logger:
```

```
idx += 1
lg_stab.data_received_cb.add_callback(lambda t, d
    , l: log_stab_callback(cf.link_uri, t, d, l))
x = entry[1]['stateEstimate.x']
y = entry[1]['stateEstimate.y']
z = entry[1]['stateEstimate.z']
vx = entry[1]['stateEstimate.vx']
vy = entry[1]['stateEstimate.vy']
vz = entry[1]['stateEstimate.vz']
phi = entry[1]['stabilizer.roll']
theta = entry[1]['stabilizer.pitch']
psi = entry[1]['stabilizer.yaw']
temp = entry[1]['baro.temp']

#saving variables to list for plotting later:

posdrones[scf.cf.link_uri] = (x,y,z, temp)
veldrones[scf.cf.link_uri] = (vx,vy,vz)
attdrones[scf.cf.link_uri] = (phi,theta,psi)
tempdrones[scf.cf.link_uri] = (temp)


uri = scf.cf.link_uri

if uri == URI1:
    dronexyz1_list.append(posdrones[URI1])
    dronedxyz1_list.append(veldrones[URI1])
    dronerpy1_list.append(attdrones[URI1])
    dronetemp1_list.append(tempdrones[URI1])

elif uri == URI2:
```

```
            dronexyz2_list.append(posdrones[URI2])
            dronedxyz2_list.append(veldrones[URI2])
            dronerpy2_list.append(attdrones[URI2])
            dronetemp2_list.append(tempdrones[URI2])

        elif uri == URI3:
            dronexyz3_list.append(posdrones[URI3])
            dronedxyz3_list.append(veldrones[URI3])
            dronerpy3_list.append(attdrones[URI3])
            dronetemp3_list.append(tempdrones[URI3])



        if idx > 5:
        ####Here is where we run the main control loop
            #print("")
            formation_control(scf,posdrones,veldrones,idx
               ,droneNum)
        if abs(x) > 1.5:
            idx = 3300
        elif abs(y) > 1.5:
            idx = 3300
        if idx == 3300:
            if uri == URI1:
                print("saving drone 1 data")


                dp1=pd.DataFrame(dronexyz1_list)
                dp1.to_csv('dronexyz1_list.csv')
                dv1=pd.DataFrame(dronedxyz1_list)
                dv1.to_csv('dronedxyz1_list.csv')
                da1=pd.DataFrame(dronerpy1_list)
                da1.to_csv('dronerpy1_list.csv')
```

```
                print("CSVs saved!")
            elif uri == URI2:
                print("saving to dronexyz2_list.csv")

                dp2=pd.DataFrame(dronexyz2_list)
                dp2.to_csv('dronexyz2_list.csv')
                dv2=pd.DataFrame(dronedxyz2_list)
                dv2.to_csv('dronedxyz2_list.csv')
                da2=pd.DataFrame(dronerpy2_list)
                da2.to_csv('dronerpy2_list.csv')


                print("CSVs saved!")
            elif uri == URI3:
                print("saving to dronexyz3_list.csv")

                dp3=pd.DataFrame(dronexyz3_list)
                dp3.to_csv('dronexyz3_list.csv')
                dv3=pd.DataFrame(dronedxyz3_list)
                dv3.to_csv('dronedxyz3_list.csv')
                da3=pd.DataFrame(dronerpy3_list)
                da3.to_csv('dronerpy3_list.csv')

                print("CSVs saved!")

def get_control_inputs(uri,xyz1,xyz2,xyz3,dxyz1,dxyz2,dxyz3,
    vlpx,vlpy,vldx, vldy, idx):

     x1,y1,z1 = xyz1[0],xyz1[1],xyz1[2]
```

```
x2,y2,z2 = xyz2[0],xyz2[1],xyz2[2]
x3,y3,z3 = xyz3[0],xyz3[1],xyz3[2]

dx1,dy1,dz1 = dxyz1[0],dxyz1[1],dxyz1[2]
dx2,dy2,dz2 = dxyz2[0],dxyz2[1],dxyz2[2]
dx3,dy3,dz3 = dxyz3[0],dxyz3[1],dxyz3[2]

x0x = vlpx
x0y = vlpy
dx0x = vldx
dx0y = vldy


#dx1, dx2, dy1, dy2, dx3, dy3 = dx1*T, dx2*T, dy1*T, dy2*
   T, dx3*T, dy3*T
# Definitions of the graph
xVect = np.array([x1, x2, x3])  # vector of agents x
   positions
yVect = np.array([y1, y2, y3])  # y positions
dxVect = np.array([dx1, dx2, dx3])  # x velocities
dyVect = np.array([dy1, dy2, dy3])  # y velocities

# Define errors
e1x = -(L+B) @ (xVect - delta_x - x0x*eyebar)  # x pos
   error
e2x = -(L+B) @ (dxVect - eyebar*dx0x)  # x vel error
e1y = -(L+B) @ (yVect - delta_y - eyebar*x0y) # y pos
   error
e2y = -(L+B) @ (dyVect - eyebar*dx0y)  # y vel

e1x_plus = -(L+B) @ ((xVect + T*dxVect) - (delta_x) - (
```

```
    eyebar*x0x + T*eyebar*dx0x))
e1y_plus = -(L+B) @ ((yVect + T*dyVect) - (delta_y) - (
    eyebar*x0y + T*eyebar*dx0y))

ssx = ax@(e1x) + e2x
ssy = ay@(e1y) + e2y

sgnssx = np.array([np.sign(ssx[0]),
                   np.sign(ssx[1]),
                   np.sign(ssx[2])])

sgnssy = np.array([np.sign(ssy[0]),
                   np.sign(ssy[1]),
                   np.sign(ssy[2])])


# dxplus = np.linalg.inv(-(L+B))@(T*(-mux*ssx) + (L+B)@(
    ax@((xVect+T*dxVect)-delta_x-(eyebar*x0x+T*eyebar*dx0x
    ))-eyebar*dx0x-ax@(xVect-delta_x-eyebar*x0x)-(dxVect-
    eyebar*dx0x)))
# dyplus = np.linalg.inv(-(L+B))@(T*(-muy*ssy) + T*
    etay@sgnssy + (L+B)@(ay@((yVect+T*dyVect)-delta_y-(
    eyebar*x0y+T*eyebar*dx0y))-eyebar*dx0y-ay@(yVect-
    delta_y-eyebar*x0y)-(dyVect-eyebar*dx0y)))


# uxmat = np.linalg.inv(u1m)@(dxplus - dxVect)
# uymat = np.linalg.inv(u1m)@(dyplus - dyVect)

uxmat = np.linalg.inv(u1m) @ (T*(dxVect - eyebar*dx0x) -
    np.linalg.inv(L+B) @ ((T*mux*ssx) + T*etax@sgnssx + ax
```

```
        *e1x_plus - ssx))
    uymat = np.linalg.inv(u1m) @ (T*(dyVect - eyebar*dx0y) -
        np.linalg.inv(L+B) @ ((T*muy*ssy) + T*etay@sgnssy + ay
        *e1y_plus - ssy))


    ux1, ux2, ux3 = uxmat
    uy1, uy2, uy3 = uymat

    e_all = (e1x,e1y,e2x,e2y)
    ss_all = (ssx, ssy)
    controlInputsAll = (-ux1[0], uy1[0],-ux2[1], uy2[1], -ux3
        [2], uy3[2])

    if uri == URI1:
        controlInputs_list.append(controlInputsAll)
        ss_list.append(ss_all)
        errors_list.append(e_all)


    if uri == URI1:
        return -ux1[0], uy1[0]
    elif uri == URI2:
        return -ux2[1], uy2[1]
    elif uri == URI3:
        return -ux3[2], uy3[2]



def formation_control(scf,posdrones,veldrones,idx,droneNum):
```

```
global vlpx_old
global vlpy_old
cf = scf.cf
uri = scf.cf.link_uri



xyz1 = posdrones['radio://0/80/2M/E7E7E7E703']
xyz2 = posdrones['radio://0/80/2M/E7E7E7E702']
xyz3 = posdrones['radio://0/80/2M/E7E7E7E701']

dxyz1 = veldrones['radio://0/80/2M/E7E7E7E703']
dxyz2 = veldrones['radio://0/80/2M/E7E7E7E702']
dxyz3 = veldrones['radio://0/80/2M/E7E7E7E701']

#xyz1 = (deltax1,deltax2,0.5)
#xyz2 = (deltax2, deltax2, 0.5) #when drones are off,
   assume they are at their desired positions
#xyz3 = (deltax3, deltay3, 0.5) #when drones are off,
   assume they are at their desired postision
#dxyz1 = (0,0,0)
#dxyz2 = (0,0,0) #when drones are off, assume they are at
    their desired veloctities
#dxyz3 = (0,0,0)



if 300 <= idx < 3300 :
    vlpx = (0.4)*math.sin(4*pi*((idx-300))/2000) + 0.25
    vlpy = (0.4)*math.cos(4*pi*((idx-300))/2000)
    vlpz = 0.5
```

```
        vldx = (0.4)*math.sin(2*pi*((idx-300))/2000) - (0.4)*
           math.sin(2*pi*((idx-1-300))/2000)
        vldy = (0.4)*math.cos(2*pi*((idx-300))/2000) - (0.4)*
           math.cos(2*pi*((idx-1-300))/2000)



    else:
        vlpx = 0.3 + 0.25
        vlpy = 0.3
        vlpz = 0.5
        vldx = 0
        vldy = 0
        vldz = 0

    leaderpos = (vlpx, vlpy, vlpz,vldx, vldy, 0)


    #print(vldx, vldy)
    if uri == URI1:
        leaderxyz_list.append(leaderpos)


    if 0<= idx <200:

        a = 1
        #can run some initialisation here if needed


    #Takeoff Sequence, each drone goes to its desired
```

```
    position
elif 200 <= idx < 300 :
    if uri == 'radio://0/80/2M/E7E7E7E703':
        a = 1
        cf.commander.send_position_setpoint(xyz1[0],xyz1
            [1],0.5,0)   #set position to 0,0,1
        #if drone is drone 1
    elif uri == 'radio://0/80/2M/E7E7E7E702':
        cf.commander.send_position_setpoint(xyz2[0],xyz2
            [1],0.5,0)   #set position to 0,0,1
        a = 1
    elif uri == 'radio://0/80/2M/E7E7E7E701':
        a = 1
        cf.commander.send_position_setpoint(xyz3[0],xyz3
            [1],0.5,0)   #set position to 0,0,1


#Main loop where drones are controlled with formation
    controller
elif 300 <= idx < 3300 :
    if uri == URI1:
        ux,uy= get_control_inputs(uri,xyz1,xyz2,xyz3,
            dxyz1,dxyz2,dxyz3,vlpx,vlpy,vldx,vldy,idx)
        phi_d1, theta_d1 = uxuy2phidthetad(ux, uy)
        cf.commander.send_zdistance_setpoint(1*phi_d1,1*
            theta_d1, 0,0.5)

    elif uri == URI2:
        ux,uy= get_control_inputs(uri,xyz1,xyz2,xyz3,
            dxyz1,dxyz2,dxyz3,vlpx,vlpy,vldx,vldy,idx)
        phi_d2, theta_d2 = uxuy2phidthetad(ux, uy)
```

```
            cf.commander.send_zdistance_setpoint(1*phi_d2,1*
                theta_d2, 0,0.5)

        elif uri == URI3:
            ux,uy= get_control_inputs(uri,xyz1,xyz2,xyz3,
                dxyz1,dxyz2,dxyz3,vlpx,vlpy,vldx,vldy,idx)
            phi_d3, theta_d3 = uxuy2phidthetad(ux, uy)
            cf.commander.send_zdistance_setpoint(1*phi_d3,1*
                theta_d3, 0,0.5)


    #landing sequence, each drone goes to final position
    elif 3300 <= idx < 3400 :
        #print("landing")
        if uri == 'radio://0/80/2M/E7E7E7E703':
            cf.commander.send_position_setpoint(deltax1,
                deltay1,0.1,0)   #set position to 0,0,1
            a = 1
                #if drone is drone 1
        elif uri == 'radio://0/80/2M/E7E7E7E702':
            cf.commander.send_position_setpoint(deltax2,
                deltay2,0.1,0)   #set position to 0,0,1
            a = 1

        elif uri == 'radio://0/80/2M/E7E7E7E701':
            cf.commander.send_position_setpoint(deltax3,
                deltay3,0.1,0)   #set position to 0,0,1
            a = 1

    #run final landing sequence and shurdown motors
    elif 3400 == idx:
```

```
        print('powering down and saving logs')
        land(cf,-0.15)
        if uri == URI1:
            lp=pd.DataFrame(leaderxyz_list)
            lp.to_csv('leaderxyz_list.csv')
            print("adding errors and sliding surface to CSV")
            el=pd.DataFrame(errors_list)
            el.to_csv('errors_list.csv')
            ss=pd.DataFrame(ss_list)
            ss.to_csv('ss_list.csv')
            il = pd.DataFrame(controlInputs_list)
            il.to_csv('controlInputs_list.csv')


    else:
        quit()

def uxuy2phidthetad(ux, uy):

    psi = 0
    sinofphi = ux * math.sin(psi) - uy * math.cos(psi)
    #print(sinofphi)
    sinofphi = max(min(sinofphi, 1), -1)  # clamp value
       between -1 and 1
    phi_d = math.asin(sinofphi)

    sinoftheta = (ux * math.cos(psi) + uy * math.sin(psi)) /
       abs(math.cos(0))
    sinoftheta = max(min(sinoftheta, 1), -1)  # clamp value
       between -1 and 1
    theta_d = math.asin(sinoftheta)
```

```
    return phi_d, theta_d

if __name__ == '__main__':
    posdrones = dict()
    veldrones = dict()
    attdrones = dict()
    attrates = dict()
    tempdrones = dict()
    initlog = 0

    cflib.crtp.init_drivers(enable_debug_driver=False) #
       initialize drivers
    factory = CachedCfFactory(rw_cache='./cache')

    with Swarm(uris, factory=factory) as swarm:
        swarm.parallel_safe(simple_log_async, args_dict=
           seq_args)

        # while True:
        #     #swarm.parallel_safe(simple_log_async,
           args_dict=seq_args)
        #     a = 1
```

Appendix C

Gaussian-Process Regression Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt
from scipy.interpolate import griddata
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF,
   ConstantKernel as C

# Load the datasets
folder = <folder to test results>
file1 = folder + 'dronexyz1_list.csv'
file2 = folder + 'dronexyz2_list.csv'
file3 = folder + 'dronexyz3_list.csv'

data1 = pd.read_csv(file1)
data2 = pd.read_csv(file2)
data3 = pd.read_csv(file3)
```

```
# Concatenate the datasets
gridsize = 10

# Define a lowpass filter
def lowpass_filter(data, cutoff=0.01, fs=1.0, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=
       False)
    y = filtfilt(b, a, data)
    return y

# Apply the lowpass filter to the temperature data
filtered_temp1 = lowpass_filter(data1['temp'])
filtered_temp2 = lowpass_filter(data2['temp'])
filtered_temp3 = lowpass_filter(data3['temp'])

data1['temp'] = filtered_temp1
data2['temp'] = filtered_temp2
data3['temp'] = filtered_temp3

data = pd.concat([data1, data2, data3], ignore_index=True)

# Define bin size (2x2 cm squares)
bin_size = 0.05  # 2 cm in meters

# Create bins for x and y
x_bins = np.arange(data['x'].min(), data['x'].max() +
   bin_size, bin_size)
y_bins = np.arange(data['y'].min(), data['y'].max() +
```

```
    bin_size, bin_size)

# Digitize the x and y coordinates into bins
data['x_bin'] = np.digitize(data['x'], x_bins)
data['y_bin'] = np.digitize(data['y'], y_bins)

# Group by the bins and calculate the mean temperature for
    each bin
binned_data = data.groupby(['x_bin', 'y_bin']).agg({
    'x': 'mean',
    'y': 'mean',
    'temp': 'mean'
}).reset_index()

# Extract the binned x, y, and temperature data
X_binned = binned_data[['x', 'y']].values
y_binned = binned_data['temp'].values

# Define the kernel for GPR
kernel = C(1.0, (1e-4, 1e4)) * RBF(length_scale=1,
    length_scale_bounds=(1e-4, 1e3))

# Create and fit the Gaussian Process Regressor
gp = GaussianProcessRegressor(kernel=kernel,
    n_restarts_optimizer=10, alpha=2.0)
gp.fit(X_binned, y_binned)

# Create a grid for prediction
grid_x, grid_y = np.mgrid[data['x'].min():data['x'].max():10j
    , data['y'].min():data['y'].max():10j]
grid_points = np.vstack((grid_x.ravel(), grid_y.ravel())).T
```

```
# Predict the temperature at each grid point
grid_temp_pred, sigma = gp.predict(grid_points, return_std=
    True)
grid_temp_pred = grid_temp_pred.reshape(grid_x.shape)

# Plot the temperature distribution predicted by GPR
plt.figure(figsize=(10, 8))
plt.contourf(grid_x, grid_y, grid_temp_pred, levels=100, cmap
    ='viridis')
plt.colorbar(label='Temperature ( C )')
plt.xlabel('x Coordinate (m)')
plt.ylabel('y Coordinate (m)')

# Plot the trajectories of each drone
plt.plot(data1['x'], data1['y'], 'k-', label='Agent 1')
plt.plot(data2['x'], data2['y'], 'r-', label='Agent 2')
plt.plot(data3['x'], data3['y'], 'b-', label='Agent 3')

# Add labels to the trajectories
plt.legend()
# Plot the fan at (0.75, -0.75) with a longer arrow pointing
    left and tilted up by 5 degrees
dx = -0.2 * np.cos(np.radians(5))
dy = 0.2 * np.sin(np.radians(5))
plt.scatter(0.8, -0.6, color='black', s=100, marker='o')
# plt.annotate('', xy=(0.75 + dx, -0.75 + dy), xytext=(0.75,
    -0.75),
#              arrowprops=dict(facecolor='black', shrink
    =0.05, width=2, headwidth=8))
```

```
# Annotate the fan
plt.text(0.8, -0.65, 'Heat source', horizontalalignment='
    right', verticalalignment='top')

plt.arrow(x=0.8, y=-0.6, dx=-0.2, dy=0.05, width=0.01, color=
    'black')


plt.show()
```