Methods for Control and State Estimation of Unmanned Aerial Vehicles

Author: Fethi Candan

Supervisors: Prof. L. Mihaylova & Prof. M. Mahfouf

Department of Automatic Control and Systems Engineering

The University of Sheffield

Thesis

September 2023

This thesis is dedicated to My Wife, Sena. She deserves the most thanks for patiently waiting for me to finish my PhD and thesis during our honeymoon.

Declaration

I affirm that this dissertation is entirely my own original work unless otherwise referenced. It has not been submitted previously for any other degree or qualification at this or any other university, in whole or in part. I acknowledge that this dissertation does not contain any content that has been produced in collaboration with others, except as explicitly stated in the text and Acknowledgements.

Fethi Candan

September 2023

Acknowledgements

Over the past four years, composing this dissertation has been a daunting task. Nevertheless, upon completing it, I feel compelled to express gratitude towards those who aided me on this journey.

I would like to extend my appreciation to my supervisor, Prof. L. Mihaylova, for offering me guidance and support. Additionally, I would like to give special recognition to my second supervisor, Prof. M. Mahfouf, and my co-authors, Dr. Aykut Beke and Omer Faruk Dik, for sharing their valuable knowledge, feedback, and expertise.

I wish to express my profound gratitude to my family. My parents, Prof. Ercan Candan and Prof. Sennur Candan, and my never-grown-up sister in my eyes, Bengisu Candan, for their love and unwavering belief in me. Also, I am deeply grateful to my wife, Dt. Sena Candan, for being an enormous source of support throughout this process and for making significant sacrifices every day.

Finally, I want to express my utmost appreciation to the Turkish government for awarding me the doctoral scholarship.

Abstract

Unmanned Air Vehicles (UAVs) have many advantages for military and civil usage, such as aerial photography, surveillance, agricultural applications, to name a few. Even though they have many benefits, there are still problems that need to be solved in terms of control, observation and path planning of UAVs. This thesis investigates each problem separately and proposes novel methods: Maximum Correntropy Kalman Filter (MCStF) and Fuzzy Interacting Multiple Velocity Obstacle Avoidance Method (FIMVO). For comparisons and tests, DJI Tello and Crazyflie 2.0 UAVs have been used as real-time applications.

In the control part, three different controller methods, proportional integral derivative (PID), type-1 fuzzy PID (T1-FPID) and interval type-2 fuzzy PID (IT2-FPID), have been compared with each other. For comparison, two different scenarios have been used under the unknown payload connected with a flexible cable. Robustness, stability criteria, and task performance have been investigated, and simulation and real-time results have been shown.

The state estimation methods part has been the other main focus topic. Conventional Kalman filter (KF) and maximum correntropy Kalman filter (MCKF) have been compared to each other. Then, a novel method, MCStF, has been proposed and compared with KF and MCKF. For comparison, interacting multiple method has been applied for each filter. The proposed and compared methods have been tested on the real-time system; in addition, the computation time and root-mean-square error have been defined as performance criteria.

The last part is about path planning. Fuzzy controllers and improved state estimation algorithms have been applied for multi-UAV usage. Instead of global mapping, local mapping algorithms, geometric-based Velocity Obstacle (VO), Reciprocal Velocity Obstacle (RVO), and Hybrid Reciprocal Velocity Obstacle (HRVO) avoidance methods have been focused on. These methods are compared with histogram-based collision avoidance methods in the simulation

environment. Then, geometric approached velocity obstacle avoidance methods and the state-of-art velocity obstacle method, FIMVO, have been compared and tested on simulation and real-time systems. For comparison, the collision count and computation time are shown as performance criteria. Moreover, the added performance graph shows reliability, trajectory smoothness, task performance and algorithm simplicity.

Contents

List of Figures

List of Tables

Nomenclatures

BB-BC	:	Big Bang- Big Crunch Optimisation Algorithm
CKF	:	Cubature Kalman Filter
CoM	:	Center of Mass
CPP	:	Coverage Path Planning
EKF	:	Extended Kalman Filter
FIMVO	:	Fuzzy Interacting Multiple-Model Velocity Obstacle
HRVO	:	Hybrid Reciprocal Velocity Obstacle
IMM	:	Interacting Multiple Model
ISE	:	Integral Square Error
IT2-FPID	:	Interval type-2 Fuzzy PID Controller
KF	:	Kalman Filter
LQR	:	Linear Quadratic Regulator
LQG	:	Linear Quadratic Gaussian
MAS	:	Multi-Agent System
MCKF	:	Maximum Correntropy Kalman Filter
MCStF	:	Maximum Correntropy Student's T Filter
MF	:	Membership Function
MPC	:	Model Predictive Control
PF	:	Potential Field
PID	:	Proportional Integral Derivative
RMSE	:	Root Mean Square Error
RVO	:	Reciprocal Velocity Obstacle
T1-FPID	:	Type-1 Fuzzy PID

UAV	:	Unmanned Air Vehicle
UGV	:	Unmanned Ground Vehicle
UKF	:	Unscented Kalman Filter
VO	:	Velocity Obstacle

ϕ	:	Roll
θ	:	Pitch
ψ	:	Yaw (Heading Angle)
Ω	:	Angular Velocity

Publications

Main Publications

The following are the author's works that are pertinent to this thesis:

Published Journal Papers

J1 **F. Candan**, Y. Peng, and L. Mihaylova, "A Comparison of Obstacle Dependent Gaussian and Hybrid Potential Field Methods for Collision Avoidance in Multi-Agent Systems", *MJAIAS*, vol. 2, no. 1, Apr. 2021.

J2 **F. Candan**, O. F. Dik, M. Mahfouf, and L. Mihaylova, "Real-time Interval Type-2 Fuzzy Control of an Unmanned Aerial Vehicle with Flexible Cable-Connected Payload", *Algorithms*, MDPI, 2023.

Published Conference Papers

C1 **F. Candan**, A. Beke, C. Shen, and L. Mihaylova, "An interacting Multiple Model Correntropy Kalman Filter Approach for Unmanned Aerial Vehicle Localisation, in *Proceedings of the International Conference on Innovations in Intelligent Systems and Applications (INISTA)*, IEEE, 2022, pp. 1–6.

C2 **F. Candan**, A. Beke, and L. Mihaylova, "An Interacting Multiple Model Approach based on Maximum Correntropy Student's T Filter", In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Detroit, USA, 2023.

Journal Papers Under Review

U1 **F. Candan**, A. Beke, M. Mahfouf, and L. Mihaylova, “A Real-time Fuzzy Interacting Multiple Velocity Obstacle Approach for Unmanned Aerial Vehicles”, *Journal of Intelligent & Robotic Systems*, 2023.

Additional Publications

Published Conference Papers

A1 Qin, C., **F. Candan**, L. Mihaylova, and E. Pournaras, “3, 2, 1, Drones go! A testbed to take off UAV Swarm Intelligence for Distributed Sensing”, In *Proceedings of the 2022 UK Workshop on Computational Intelligence (UKCI)*, Springer, 2022.

Journal Papers Under Review

B1 J. Chen, R. Ma, **F. Candan**, L. Mihaylova and J. Oyekan, “A Cooperative Payload Trajectory Tracking Framework for Safe Adaptive Transportation”, *IEEE Transactions on Intelligent Transportation Systems*, 2023, under review.

Chapter 1

Introduction

The usage of Unmanned Aerial Vehicles (UAVs), commonly known as drones, has brought numerous benefits to various fields, including search and rescue operations, aerial photography, and industrial applications ([1], [2]). Aerial vehicles are autonomous or remotely operated aircraft systems that have revolutionised research, development, and transportation systems, enabling rapid reconfiguration, monitoring, inspection, and the delivery of sophisticated autonomous systems ([1]–[4]). These aircraft can be either electric or fuel-based and can fly autonomously or under remote control. UAVs can be broadly categorised into two types based on their aerodynamic principles: fixed-wing and rotary-wing UAVs. Fixed-wing UAVs feature a rigid wing that enables vertical take-off and landing (VTOL) and uses aerofoils, pitch control, and an elevator to achieve three axes of rotation ([5]). On the other hand, rotary-wing UAVs employ a rotating motor-based wing structure, with changes in rotor thrust allowing for three-axis rotation ([6]).

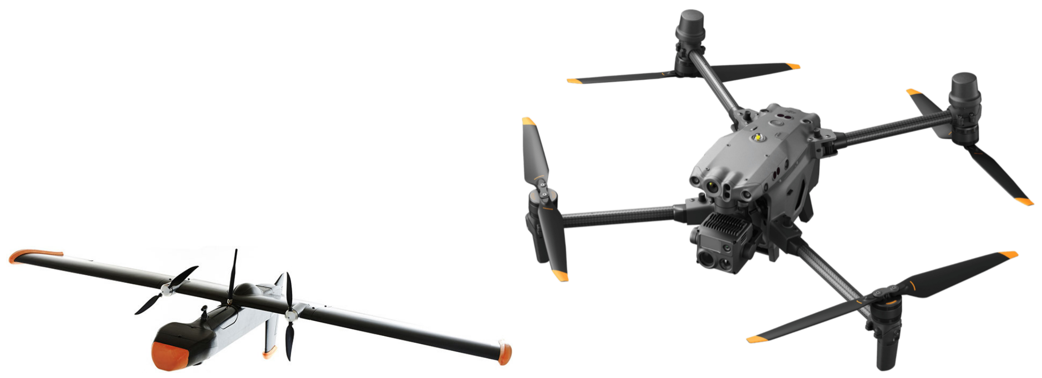
Figure 1.1: UAVER Avian-P3 commercial fixed-wing UAV ([7]) (Left) and DJI Matrice 30 commercial rotary-wing UAV ([8]) (Right).

In Fig. 1.1, the UAVER Avian-P3 fixed-wing UAV is depicted, which is commonly used for long-distance mapping systems. It is equipped with a long-lasting battery and has high-accuracy positioning capabilities. Additionally, the DJI Matrice 30 rotor-wing UAV is shown, often used for surveillance, thermal analysis, and 3D environment mapping [8]. However, both of these systems face challenges related to flight time and mission complexity. For instance, when dealing with farming areas larger than 100 acres, a rotor-wing UAV may require multiple flights, while using a fixed-wing UAV may not be a reliable solution. In such cases, employing multiple UAVs or a swarm of UAVs can be a potential solution.

Multiple UAVs, also known as aerial robots or autonomous UAVs, along with multiple Unmanned Ground Vehicles (UGVs), have gained significant attention in various fields such as tracking, inspection, and transportation systems, enabling the deployment of intelligent autonomous systems ([9]–[11]). Compared to a single large UAV, multiple UAVs controlled autonomously or through teleoperation prove to be more effective in many critical application domains. Therefore, various types of controllers have been developed for multiple UAV systems, including formation controllers ([11]–[13]) and flocking controllers ([14]–[16]). However, implementing real-time applications for multiple UAVs can be challenging due to factors such as high-cost equipment and inadequate environments. To address these challenges, multi-agent systems (MAS) have been utilised in research to simplify problems and find effective solutions ([17]). The considered problems are listed below:

1. Dynamical model complexity of UAVs: There are many different types of UAV models, such as linear or nonlinear. However, there are several inaccessible parameters in the UAV dynamics; therefore, the model of the UAV is able to be complex.

2. Control methodologies of UAVs: Many different controlling methods, PID, LQR, MPC or FPID, have been developed to control the UAV. Moreover, these methods have guaranteed the stability of the UAVs. Considering the multiple UAVs, the coupling parameters, such as cooperative or non-cooperative working, can change the stability performances of each UAV.

3. Internal and external noise effects on UAVs: In real-time systems, the sensors, IMU, GPS, INS or Optical Flow, have included some different types of noises. It has be to eliminated to get high accuracy. The state estimation methods, KF, EKF or UKF, are

able to be used to predict and estimate sensor actual data. However, considering the external noises, wind or radiation, it has been not easy to estimate.

This thesis focuses on addressing the main problems encountered in the context of multiple UAVs, which include the dynamic complexity of UAVs and multiple UAVs, control methodologies for UAVs and multiple UAVs, and the effects of internal and external noise on UAVs and multiple UAVs. These problems are of significant research interest. The complexity of dynamical models affects controller design and computation time, where a complex model can increase the computation time and lead to undesirable outcomes. Additionally, even with a complex model, there may exist uncertain or inaccessible parameters. Moreover, while internal noises can be mitigated using filter methods, predicting, and handling external noises present additional challenges.

These problems have practical implications in industrial, agricultural, and military fields, and several survey papers have extensively discussed and expanded upon these challenges ([18]–[20]). For example, agricultural UAVs deviating from their intended tracking line due to wind, military UAVs exhibiting incorrect responses under challenging conditions, and the coordination and timing requirements of multiple UAVs used in aerial shows are examples of the problems encountered. However, using expensive, large, and high-risk UAVs to solve these problems may not be practical. Instead, this research proposes the utilisation of small-scale and cost-effective UAV systems.

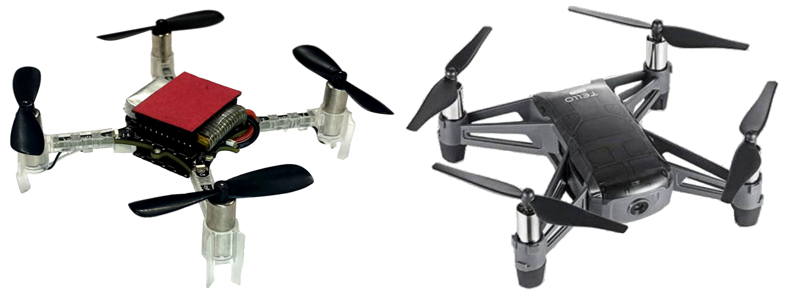

Figure 1.2: Bitcraze Crazyflie 2.1 (Left), DJI Tello EDU (Right).

In Fig. 1.2, the Crazyflie 2.1, developed by Bitcraze, and the Tello EDU UAVs, developed by DJI Company, are showcased. The Crazyflie 2.1, known as a "nano drone," is particularly suitable for indoor applications due to its small size, high manoeuvrability, affordability, and open-source nature ([21], [22]). Similarly, DJI Tello EDU UAVs possess similar advantages in terms of size, lightweight design, and manoeuvrability. However, unlike the Crazyflie 2.1, DJI Tello UAVs have a semi-open-source structure with a closed-loop inner system. These UAV platforms serve as suitable options for real-time application solutions, enabling the experimental testing of proposed methods.

To address the problems related to controllers, estimations, and local path planning, various comparison methods have been investigated and classified for each problem. The comparisons, along with the proposed approaches, are presented in Fig. 1.3.

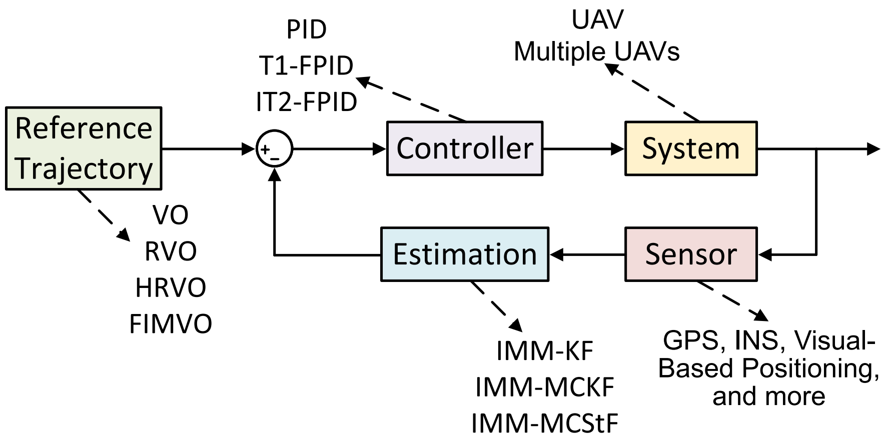

Figure 1.3: The proposed approaches of the UAVs

This research primarily focuses on modelling and control methods for UAVs. In the controller section, the proportional-integral-derivative (PID), interval type-1 fuzzy PID (IT1-FPID), and type-2 fuzzy PID (IT2-FPID) controller methods are compared in terms of stability, task time completion, disturbance rejection, and uncertainty handling. In the estimation part, the classical Kalman filter (KF), maximum correntropy Kalman filter (MCKF), and a novel method called maximum correntropy Student's T filter (MCStF) are compared under Gaussian and non-Gaussian noise conditions. Lastly, collision avoidance methods, crucial for local path planning, are explored using geometric approaches known for their computational efficiency and ease of implementation in real-time systems. Traditional velocity obstacle (VO), reciprocal velocity obstacle (RVO), hybrid reciprocal velocity obstacle (HRVO), and a proposed fuzzy interacting multiple velocity obstacle (FIMVO) approach are tested through simulations and real-time experiments.

1.1 Thesis Outline

The thesis has been divided into six parts, namely, the introduction, publications, related works, methods for control of UAVs, methods for estimation of UAVs with maximum correntropy filters, and methods for collision avoidance of UAVs, respectively. The last chapter consists of conclusions and future works.

The two parts present the introduction to UAVs and literature reviews. An introduction to UAV is given in Chapter 1. Then, related works about control, estimation and collision avoidance of UAVs are explained in Chapter 2. In addition, Chapters 3, 4 and 5 introduce the proposed and compared methods for the UAVs: control, estimation and collision avoidance, respectively. Methods for control of the UAVs represent PID, T1-FPID and IT2-FPID under uncertainties; also, simulation and real-time results are shown. In the methods for the estimation of the UAVs with maximum correntropy filters, a novel method, MCStF, is explained and tested on simulation and real-time systems. The proposed method is compared with KF and MCKF. The last chapter of the methods, collision avoidance methods, is started in Chapter 5. Potential field-based and geometric-based collision avoidance methods are compared in the simulation environment using Matlab. Then, VO, RVO and HRVO are combined with fuzzy and a new velocity obstacle avoidance approach, FIMVO, is proposed. The simulation and the real-time test results are also shown in these chapters. Lastly, the thesis is concluded, and future work is recommended in Chapter 6.

1.2 Contributions

In this section, it is presented the contributions of the thesis.

1. Multiple UAV problems, including collision avoidance, computation time, stability, and task performance, have been addressed both theoretically and practically. These problems have been extensively studied through simulations, and solutions such as Gaussian-based and non-Gaussian-based filters for state estimation and local path planning have been proposed to resolve them.

2. Real-time applications involving uncertain parameters and diverse types of noise have been undertaken. These uncertain parameters encompass both unknown payloads and disturbances affecting the payload. Specifically, when formulating the dynamic equations for UAVs, the resulting system does not account for the connected payload and the pendulum effects on the UAV's centre of mass. Additionally, different types of noise, such as Gaussian and heavy-tailed non-Gaussian noises, have been considered.

3. One of the significant contributions of this study is the successful implementation of the Interval Type-2 Fuzzy PID (IT2-FPID) controller under the nonlinear dynamic model with unknown/uncertain parameters of the UAV carrying a payload. This implementation involves tuning the controller parameters based on the model and aims to mitigate the oscillations caused by a flexible cable-connected payload on the DJI Tello UAV. The controllers, including PID, T1-FPID, and IT2-FPID, have been tuned using the Big Bang-Big Crunch (BB-BC) optimization method on the UAV's dynamical model without the payload. Subsequently, the controller coefficients were fine-tuned on a real-time UAV without the payload to account for unmodeled dynamics and uncertainties. The effectiveness of the IT2-FPID controller has been demonstrated by eliminating the disturbing payload effects and achieving efficient tracking of the desired trajectory. Furthermore, a comparative analysis of the performance between IT2-FPID, T1-FPID, and PID controllers has been conducted, revealing the superiority of the designed IT2-FPID control system over its counterparts.

4. This study introduces two novel methods: "An Interacting Multiple Model Approach based on Maximum Correntropy Student's T Filter" (MCStF) and "A Real-time Fuzzy

Interacting Multiple-Model Velocity Obstacle (FIMVO) Approach for Unmanned Aerial Vehicles". These methods constitute valuable contributions to the existing body of literature.

5. The MCStF method presents a novel approach to eliminate heavy-tailed non-Gaussian noises affecting the position sensor. A comparative analysis is conducted, comparing the proposed method with IMM-based MCKF and KF. While MCKF exhibits good performance under non-Gaussian noises, the results obtained with MCStF demonstrate better accuracy and computation time. Additionally, the proposed method is tested not only in a simulation environment but also in real-time systems using UAVs and cameras for position detection. The results obtained from both simulations and real-time experiments reinforce each other, establishing the superior accuracy and computation time efficiency of MCStF.

6. FIMVO is directly related to velocity obstacle methods. To enhance accuracy, task performance, and computation time efficiency, FIMVO incorporates VO, RVO, and HRVO simultaneously, utilizing a decision-making mechanism. The choice among these methods is determined by a fuzzy interaction, thereby impacting the computation time. Consequently, FIMVO outperforms HRVO in terms of speed and provides greater reliability compared to VO and RVO. The performance of these methods has been evaluated and compared in both simulation and real-time environments, with the simulation results validated through real-time experiments.

Chapter 2

Related Works

Literature on the methods for modelling and control of UAVs has been investigated in 3 parts: controller, observation, and local path planning.

2.1 Methods for Control of Unmanned Aerial Vehicles

Numerous types of linear and nonlinear control methods have been designed and implemented for the control of UAVs [21], [23]–[25]. While UAVs are nonlinear structures with highly coupled states, linear controllers such as traditional PID or LQR were used for the UAV's autonomous flight [25]. In general, modern controls have been developed and implemented to overcome the motion stability and coordination problems of UAVs, such as conventional proportional–integral–derivative (PID) ([26], [27]), fuzzy PID Control (FPID or T1-FPID) ([28], [29]), Linear Quadratic Regulator (LQR) ([30]), Linear Quadratic Gaussian and Loop Transfer Recovery (LQG/LTR) ([31]), Model Predictive Control (MPC) ([32], [33]) and H∞ ([34]) control methods. However, the controllers built previously are used for specific sizes of a single UAV. That means control parameters and control methods may be changed for different sizes and shapes of UAVs. In addition, the physical dynamics of the UAV represent a challenge for many of the control algorithms implemented on these UAVs, especially when the control coefficients are determined by using one type of dynamic model.

The controller algorithms and the calculation of their parameters are directly related to the physical dynamics of the system. However, the physical dynamics of the UAV represent a challenge for many of the control algorithms implemented on these UAVs, mainly when the control parameters are determined by using limited known dynamic model parameters. Moreover, knowing all parameters of the physical dynamics is almost impossible due to uncertainties or inaccessible parameters. Therefore, the stable controller parameter might cross to the unstable region in the presence of uncertainty or inaccessible parameters. To explain and illustrate the problem, it has been scaled and tested using a mini UAV. For indoor experimental research, nano or mini UAVs have proven their efficiency concerning easy access, small sizes and weights ([35]). As a result, researchers have used several commercial UAVs such as Bitcraze Crazyflie 2.0 ([22], [28], [36]), Parrot Mambo Minidrone ([29]) and DJI Tello ([37]).

In many studies, the connection with the payload represents a further research challenge ([37]–[40]), and the control problem under uncertain payloads has been solved using reinforcement learning. The UAV has a flexible cable, payloads, and magnets. Using the magnets, the payloads can connect with each other. Hence, the weights cannot be known, adding additional uncertain parameters. Thanks to model-based meta-reinforcement learning, the system's inputs and outputs have been trained and created model-based data using a human-based remote controller. After designing a model-based meta-reinforcement learning dynamic model, an MPC algorithm has been implemented to cope with such effects ([37]).

UAVs face other challenges - not only due to the payload but also due to wind disturbances. In ([39]), a control algorithm for trajectory tracking with a UAV with a cable-connected payload has been designed under wind disturbances. In addition, near hovering and cruising states have been included in the control and an employed attitude tracking control algorithm for the inner structure. In ([39]), a whole system model has been developed with payload and disturbances, which means that they are not unknown parameters. Considering unknown parameters such as the payload mass, an excellent example of associated research work can be attributed to ([38]) on adaptive UAV control for a cable-suspended unknown payload. The system has been tested with a nonlinear and adaptive PD controller. The coefficients/parameters of the UAV controllers have been searched and defined via many different optimisation methods ([41]), and each of them has its own advantages and disadvantages.

2.2 Methods for State Estimation of the Unmanned Aerial Vehicles

Positioning or localising of the UAV has revealed another problem, like controlling. In literature, there are several methods to accurately converge to exact positions like Kalman filter (KF), maximum correntropy Kalman filter (MCKF) [42], [43]. In the position of UAV, sensor fusion methods have been applied to the Inertial Measurement Unit (IMU) and Global Positioning System (GPS) sensor data. An essential part of this application is identifying the correct dynamical and kinematic model. However, the UAV states change drastically whether it follows a manoeuvring or non-manoeuvring trajectory. In this situation, interacting multiple model (IMM) approaches help to solve this problem.

There are several IMM methods have been deeply explained in [44]–[60]. In [48], a survey paper has been written about manoeuvre and non-manoeuvre models for a system. The conventional IMM affects the measurement to set information updates with active models. In [42], [43], [50], there are several advantages of IMM approaches for different applications, but the problem is model uncertainties. When the conventional IMM approach is used in fault-prognostic applications, the IMM approach is not able to estimate the actual states. In [45], to overcome this problem, Fuzzy and IMM structures have been merged to detect fault-tolerant. It is seen that defining dynamical models has played an important role in the application. For this reason, in [48], researchers have investigated different dynamical models for manoeuvring target tracking. Moreover, the Cubature Kalman filter implemented IMM approach has been used for nonlinear systems in [53], and this approach has been tested on manoeuvring target tracking to represent the switching probability; the approach employs a Markov process. Another research has shown different filtering for Markovian switching: interacting multiple model filtering in [58]. Considering the state estimation filters, the accuracy and error results of the conventional filters do not converge to actual data under different conditions, such as non-Gaussian or mixed distribution. However, the particle filter affects the mixture of Gaussian probability densities. That means that it is able to manage non-linearities in the system and non-Gaussian noise [51], [52].

The performance of these filters depends especially on the model parameters and noise variance or covariance values. However, knowing the exact parameter values in most real-time system applications is impossible, which could affect the estimation accuracy. The noise distributions of the sensor data can characterise these unknown parameters or uncertainties. The Gaussian distribution usually represents these noises. Then, the prediction and estimation are achieved by Kalman filters [42], [61]. However, these distributions are not always Gaussian. As a result of this, the predictions and the estimations of the filters cannot give high accuracy of the system state estimation under non-Gaussian, such as positioning problems with high error or vision-based localisation errors.

2.3 Methods for Collision Avoidance of Unmanned Aerial Vehicles

Modelling multiple agents or UAV systems (MAS or multiple UAVs) is another important task, especially linked with collision avoidance. In multiple UAVs, the peripheral equipment, such as sensors and internal controllers, can be seen at an aggregated level rather than at the individual level of each agent/UAV [62]. Considering the MAS or multiple UAVs, state estimation and path planning are other challenging problems for researchers [63]. Moreover, when designing formation control on swarms or MAS, the obstacle is another problem [64]–[66]. Therefore, the local path planning algorithm has to guarantee non-collision situations. Path planning algorithms can be changed for different tasks under static or dynamic obstacles, and each agent/UAV can be defined as an obstacle to each other. At this point, velocity or local path points may vary to avoid the collision. Therefore, it focuses on this subject intensely: Collision avoidance in real-time systems with a novel proposed method.

There are also challenges related to coordination, communication in real time and achieving these tasks for every single agent [67]. Multi-agent systems (MAS) have gained significant interest, and recent results are described in the survey paper [17].

Each agent/ UAV is considered an agent. Desired reference trajectory has been one of the important UAV tasks aiming to get a global trajectory that is optimal according to a certain criterion. In order to accomplish the created task, the robot/UAV must reach the desired target points after starting with an initial point. The agent trajectory generation needs to be performed to avoid both static and dynamic obstacles [67].

There are different methods able to solve efficiently path planning problems [68]–[75]. The potential field (PF) [68] and velocity obstacle (VO) methods [76] are two of the most efficient solutions.

The PF methods have been widely used to solve path planning problems since they have several advantages: they are easy to implement and provide smooth trajectories [68], [74], [75].

In [68], PF methods have been studied in depth for path planning. Then, the artificial potential field, based on the traditional PF method, has been validated with performance criteria in terms of solving local minima problems and obstacle avoidance. Collision avoidance can also be solved with geometric types of methods [73] and have been compared with an A* graph search method [77], especially for UAVs [69]. In that work, the A* method was employed for real-time pathfinding in the presence of dynamic obstacles. Nevertheless, in scenarios where both dynamic and static obstacles are encountered simultaneously, there is an observed increase in the frequency of collisions.

When comparing with the VO methods, the reciprocal velocity obstacle avoidance method (RVO) [78] outperforms the conventional VO method, and they have been tested in the real-time MAS. In [79] and [72], an interval VO approach, such as collision cone zone boundaries and velocity limits, with collision avoidance cone has been proposed and compared with traditional VO, reciprocal VO, hybrid reciprocal VO and the optimal reciprocal collision avoidance (ORCA) methods. Also, each algorithm has been assessed in terms of its advantages and disadvantages.

Traditional PF, VO, obstacle-dependant Gaussian Potential field (ODG-PF) and hybrid potential field methods (HPFM) have been applied in several fields [62], [72]. As the PF and VO were designed to deal separately with static and dynamic obstacles, they require improvements in complex situations where both static and dynamic obstacles exist.

The HPFM combines the strategy of PF and VO by using the form of Vector Field Histogram (VFH) [80], [81]. In the HPFM, features of two types of obstacles are considered, and the weights of each type of obstacle could be easily adjusted regarding different environments.

In [72], researchers have explained different geometric approached velocity obstacle avoidance methods for MAS. MAS and velocity obstacle (VO) have been tested on UAV swarms [82], [83]. In [83] and [82], VO methods have been implemented: selective VO and adaptive VO for UAV, respectively. These methods can be the geometric approach [66], the algebraic approach [84], or intelligent methods like machine learning or deep learning [85].

Potential Field and Velocity obstacle avoidance methods have been compared to each other [72], [86]–[88]. There are several advantages and disadvantages between them. There are two major types of collision avoidance when considering the geometric-based velocity obstacle approaches: cooperative and non-cooperative. Both methods often presuppose knowledge of each object's information geometry. Suitable collision avoidance methods perform on the assumption that there is an implicit, autonomous communication layer that allows any actor to express their purpose to another freely. In non-cooperative techniques, the agent's objective is only partially understood, relying solely on the kinematic model parameters that can be deduced from the agent's onboard tracking system. In [72], [79], [88], conventional methods such as VO, RVO, HRVO and ORCA have been simulated and compared to each other on MAS. In [89], a three-dimensional velocity obstacle has been designed for fixed-wing-based UAVs. By manipulating the vehicle's velocity vector in response to the geometry of the encounter, the velocity obstacle technique, a three-dimensional (3-D) version of the approach, has the ability to design an avoidance manoeuvre proactively. The results of a validation using Monte Carlo simulations in challenging super conflict scenarios show that none of the 25,000 samples had collisions. In another paper about obstacle avoidance method [90], the Pythagorean Hodograph (PH) curve trajectory re-planning serves as an illustration for the investigation of the 3-D direct obstacle avoidance technique in dynamic space. The simulation results demonstrate that the suggested method's ability to implement obstacle avoidance trajectory re-planning substantially and simultaneously also improves the flexibility of obstacle avoidance manoeuvre.

Chapter 3

Methods for Control of Unmanned Aerial Vehicles

This chapter has explained the controller problem pertaining to unmanned aerial vehicles (UAVs). To address this problem, three controller methods have been chosen: PID, Fuzzy PID (FPID), and Interval Type-2 Fuzzy PID (IT2-FPID). These methods have not only been implemented but also subjected to testing under various sources of uncertainty, including external disturbances and unknown connected payloads. Subsequently, these controller methods have been compared with respect to task completion time and performance criteria.

3.1 UAV Model and the Problem Definition

This section presents the mathematical model for the dynamics of DJI Tello UAV, and the problem statement and challenges. The first part of this section describes the dynamic model of the UAV. This dynamic model will be shown without any payload. After that, the problem of the UAV with a flexible-connected payload will be explained.

3.1.1 Model Dynamics of DJI Tello UAV

Before outlining the UAV model, the main coordinate frames used in this study are briefly described: $(.)^I$ represents the global Earth fixed Earth-centred inertial frame, and $(.)^B$ is the body fixed frame, fixed at the UAV Center of Mass (CoM) as given in Figure 3.1. There are 2 types of configurations for the UAV, which are $\times$ and $+$ configurations ([22], [28], [91]). In this study, $\times$ type UAV has been modelled and used.

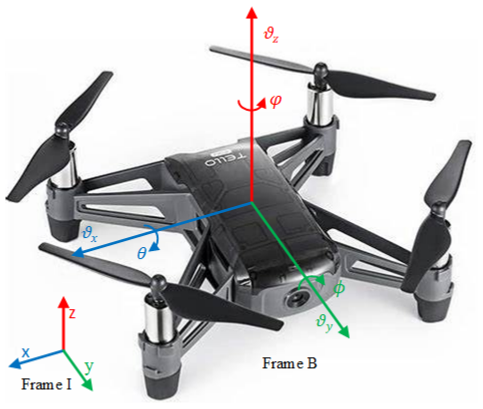

Figure 3.1: The DJI Tello UAV used in the implementation ([92]).

The position (p^I) and attitude (η^I) of the UAV with respect to the frame I are defined as follows:

$$\begin{aligned} \mathbf{p}^I = [x\ y\ z]^T, \eta^I = [\phi\ \theta\ \psi]^T, \\ \upsilon^I = \dot{p}^I = R\upsilon^B, \omega^I = \dot{\eta}^I = T\omega^B, \end{aligned} \tag{3.1}$$

where the vector p^I contains the x, y, and z Cartesian position coordinates of the UAV's inertial frame, υ^I and ω^I are the derivative of position and attitude values, respectively. The superscript $(.)^I$ and $(.)^B$ denote the inertial and body frames. The η^I vector contains the angles of roll ϕ, pitch θ, yaw ψ of the UAV's body frame, R is a rotational matrix, and T is a translation matrix used in frames conversions. These rotation and transformation matrices are used to transform the inertial frame to the body frame or vice versa. The rotation and transformation matrices have been written in Equation (3.2). The transformation between the body-fixed frame and the inertial frame is achieved with a rotation matrix R for inverse mapping and a transitional matrix T ([28], [29], [91], [93]) of the form:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & c(\theta)s(\phi) \\ 0 & -sin(\phi) & cos(\phi)c(\theta) \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix}. \tag{3.2}$$

The υ value is used to represent the linear velocity, and ω is used to represent the angular velocity. In the implementation, the heading angle of the UAV, yaw (ψ), is fixed and is equal to $\psi = 0$ in order to overlap the inertial and the body coordinate frames of the DJI Tello.

After the definition of UAV's position and orientation, a nonlinear dynamic equation for DJI Tello UAV can be written with the Newton-Euler formulation as given below ([22], [28]):

$$\begin{gathered} \mathbf{m}_{\text{uav}}\dot{v}^I = \mathbf{F}^I = \mathbf{F}^I_{gravity} + \mathbf{F}^I_{thrust}, \\ I^B\dot{\mathbf{w}}^B = -\mathbf{w}^B \times \left(I^B\mathbf{w}^B\right) + \tau^B_{rotor}. \end{gathered} \tag{3.3}$$

In Equation (3.3), m_{uav} is the mass of the UAV to the CoM of the UAV, $F^I_{gravity}$ is the gravity force, F^I_{thrust} is the total thrust force which comes from propellers, I^B represents a moment of inertia, and I^B is referred the inertia matrix.

In Equation (3.3), the cross product at the right-hand side of the angular velocity dynamics equation comes from the Coriolis force effect. Considering the DJI Tello UAV, this effect can be ignored because of the size of the UAV. [28] have tested a fixed heading angle and ignored the Coriolis effect on the DJI Tello UAV, which has an almost similar size to the DJI Tello UAV. Therefore, the formula can be written as $I^B\dot{w}^B = \tau^B_{rotor}$, and the UAV torques ($\tau^B_{rotor}$) are expressed by τ_ϕ,τ_θ, τ_ψ, respectively

$$\begin{gathered} \mathbf{F}^I_{gravity} = [0 \quad 0 \quad -m_{\text{uav}}g]^T, \\ \mathbf{F}^I_{thrust} = \mathbf{R}\mathbf{F}^B_{thrust}. \end{gathered} \tag{3.4}$$

To overcome the gravity effect, the force is applied to the CoM of the UAV as $F^I_{gravity}$. The acceleration due to Earth's gravity is denoted as g. The F^I_{thrust} forces are generated by propellers, and the formula is written as in Equation (3.5). The angular velocity of each rotor's propeller is denoted Ω_i and the thrust coefficient is referred to as b

$$\mathbf{F}^B_{thrust} = \left[0 \quad 0 \quad b\textstyle\sum_{i=1}^4 \Omega_i^2\right]^T, \tag{3.5}$$

$$\dot{v}^I = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{b}{m_{\text{uav}}}\textstyle\sum_{i=1}^4 \Omega_i^2 = \begin{bmatrix} s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi) \\ -s(\phi)c(\theta) + c(\phi)s(\theta)s(\psi) \\ c(\phi)c(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}. \tag{3.6}$$

In Equation (3.6), $s(.)$ and $c(.)$ denote sine and cosine, respectively. In the equations above, the nonlinear dynamical model for the translational motion of the UAV has been represented, and a detailed linear dynamical model of the UAV has been written In Equations. (3.10-3.12) of Appendix. Angular velocity dynamics of the UAV also can be found In Equations. (3.8 and 3.9). Considering the UAV dynamics equations above, the mass of UAV (m_{uav}) only depends on the UAV weight except for the extra payload.

In this study, the heading angle of the UAV, yaw (ψ), is fixed and is equal to $\psi = 0$. For this reason, the defined rotation matrix form is $R = R_\phi R_\theta$.

Unlike the nonlinear model of the UAV, the definition of the linear model is represented with simpler equations. As a consequence, U shows the control input values of the system Equation (3.7)

$$\mathbf{U} = [U_1 \;\; U_2 \;\; U_3 \;\; U_4] = [\mathbf{F}^B_{thrust} \;\; \tau_\phi \;\; \tau_\theta \;\; \tau_\psi]. \tag{3.7}$$

Equation (3.8) is obtained from the equation corresponding to each U input value

$$\mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ -bl & -bl & bl & bl \\ -bl & bl & bl & -bl \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}. \tag{3.8}$$

$$\dot{\mathbf{w}}^B = I_B^{-1} \tau^B_{rotor} = \begin{bmatrix} \frac{1}{I_x} bl U_2/\sqrt{2} \\ \frac{1}{I_y} bl U_3/\sqrt{2} \\ \frac{1}{I_z} d U_4 \end{bmatrix}. \tag{3.9}$$

In the above equations, Ω_i denotes the angular velocity of the UAV propeller, d is the drag coefficient of rotors, and l is the distance between the rotor and CoM. When writing the inertia matrix (I_B) for each torque, they are defined as inertia tensors (I_x, I_y, I_z).

To linearize the Equation (3.8) above by using the Taylor approximation, the following equation is used [28], [91]:

$$\mathbf{U}_i = U_i^0 + \frac{dU_i}{d\Omega_i}|_{\Omega_i^0}(\Omega - \Omega_i^0). \tag{3.10}$$

In the equation above, the subscript of $(.)^0$ means the hovering mode condition of the UAV.

Therefore, the UAV's hovering operation point is given by the following equation

$$\begin{gathered}\mathbf{U}^0 = [U_1^0 \;\; U_2^0 \;\; U_3^0 \;\; U_4^0]^T = [mg \;\; 0 \;\; 0 \;\; 0]^T, \\ \Omega_i^0 = \sqrt{\frac{mg}{4b}}.\end{gathered} \tag{3.11}$$

$$\begin{bmatrix}\ddot{x} \\ \ddot{y} \\ \ddot{z}\end{bmatrix} = \begin{bmatrix} g(s(\psi_0)\phi + c(\psi_0)\theta) \\ g(s(\psi_0)\theta + c(\psi_0)\phi) \\ F^B_{thrust}\end{bmatrix}. \tag{3.12}$$

In the above equations, the mass of the DJI Tello UAV equals $m_{\text{uav}} = 0.083$ kg, and the distance between CoM and propeller is 0.06 m. The detailed parameters of the DJI Tello UAV have been explained in ref [94], [95], and some of the parameters of the DJI Tello have been written as in Table 3.1.

Table 3.1: The list of the DJI Tello UAV parameters

Parameters	Name	Value
m	Mass	$80gr$
d	Arm length	$60mm$
g	Acceleration of gravity	$9.81m/s^2$
J	The matrix of rotational inertia	$I_x = I_y = 0679e^{-2}$, $I_z = 1.313e^{-2}\ kg.m^2$
T_m	Motor response time constant	$0.0163s$
J_r	The moment of inertia of propeller	$4.95e^{-5}\ kg.m^2$

3.1.2 Problem Statement and Challenges

In Figure 3.2, the UAV with the flexible cable-connected payload and its pendulum effects are shown. In the figure, the different perspectives of the UAV with cable-connected payload and the variable payload are shown. A 3D-printed part has been created and mounted to provide a UAV connection with a payload cable. The cable length is 35 cm and has been chosen as flexible. Taking into account the pendulum effects, forces F_P, F_R and the angles α, ε, β, and λ vary with the acceleration of the UAV due to Newton's second law. The maximum carrying payload has been measured with empirical methods, and it is determined that the maximum value is 75 grams for the experiments. This weight (m_{p}) is a significant value when considering the mini UAVs. Then, it is able to rewrite the equations as $m_{\text{total}} = m_{\text{uav}} + m_{\text{p}}$. Instead of rewriting the dynamical equations and calculating the controller parameters using swinging

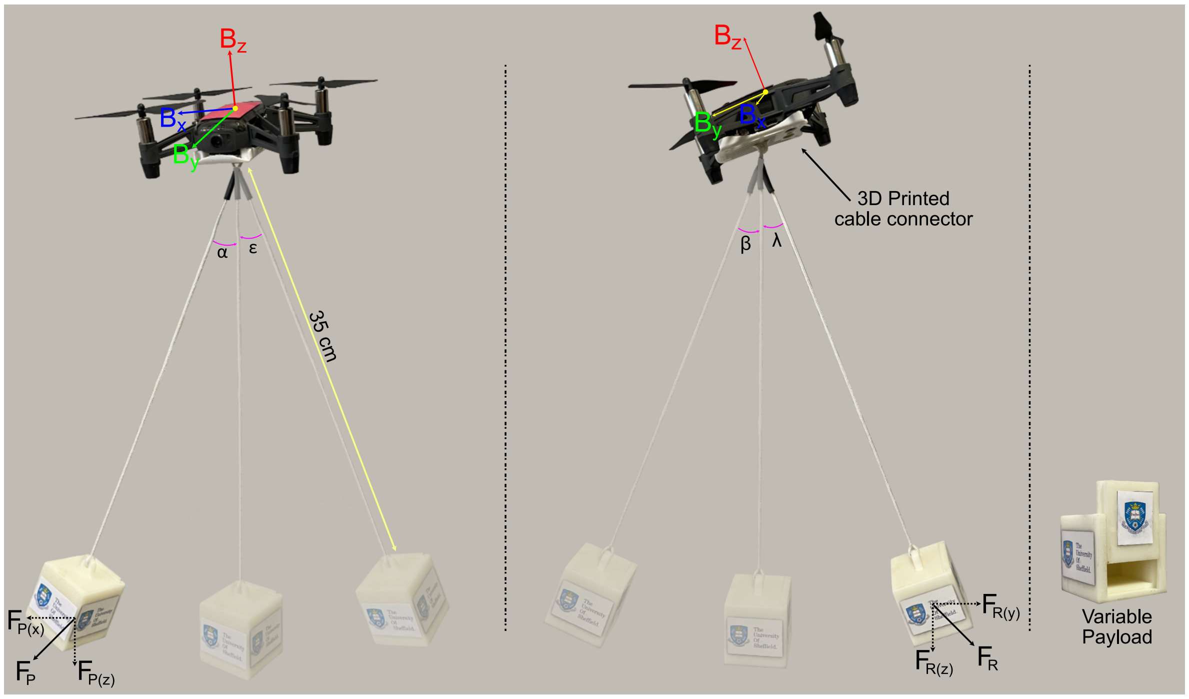

Figure 3.2: DJI Tello EDU and payload connection.

m_{total}, it is aimed that the compared controllers can eliminate the unknown payload effects and the variable CoM problems without redesigning or the calculation.

It is understandable that unknown payloads and disturbances directly relate to the system dynamics. Therefore, the main objective of this work is to overcome these adverse effects. At this point, instead of re-writing the equations of the UAV dynamics and redesigning the controllers by considering the whole mathematical model with additional dynamics of the swinging payload, the controllers are designed by considering the known dynamics of the UAV itself without taking the payload and disturbance effects in the account so that the robustness of the three controllers have been compared then the IT2-FPID controller has given the results in system stability and robustness.

The most important question is, "Why do we focus on uncertain parameters such as unknown payload and flexible cable effects?". This system is a simulation of a commercial agricultural UAV problem. Commercial agricultural UAVs may contain liquid or other payloads. However the problem is that these payloads, such as seeds or pesticides, are unknown loads because of their variable weights and variable centre of masses during the flight.

3.2 Implemented Controllers

This section presents the design and deployment of PID, T1-FPID and IT2-FPID controllers on the UAV as x-y position controllers that generate the x-y velocity reference signals υ_x, υ_y (m/s) to be tracked by the on-board velocity controller by DJI Tello UAV. In this context, first the position errors are defined that will be minimised by the controllers on the x and y axes e_x, e_y as follows:

$$\begin{aligned} e_x &= x_{ref} - x_{obs} \\ e_y &= y_{ref} - y_{obs} \end{aligned} \tag{3.13}$$

where (x_{ref}, y_{ref}) and (x_{obs}, y_{obs}) are the references and the feedbacks (actual position of the UAV), respectively. In this study, the reference and feedback units are in meters (m) in the position-based controller. To minimise e_x, e_y, two controllers are designed that generate the velocity signals $u_x = \upsilon_x$, $u_y = \upsilon_y$. In this study, the altitude z and yaw ψ control angles of the UAV are controlled by the onboard controller directly. In the particular implementation, the altitude reference angle is set up $z_{ref} = 90cm$ and heading angle reference of the UAV is also set up equal to zero,$\psi_{ref} = 0°$.

In this section, as the x-y position controllers have the same structure, the subscripts (x, y) in e_x, e_y, u_x, u_y will be dropped in the rest of the section for a better understanding.

3.2.1 PID Controller

In the literature chapter, Crazyflie 2.0 and DJI Tello UAVs-based systems have been investigated. These UAVs is able to be designed as decoupled dynamics. Therefore, the controller problem can be solved by SISO methods, such as PID, adaptive and more. For the comparison, conventional PID has been used.

The PID control law for the UAV is represented as follows

$$u = K_c \left(e + \frac{1}{T_i} \int edt + T_d \frac{de}{dt} \right), \tag{3.14}$$

where u is a reference signal for the inner controller. e is represented as the error, and also T_i, T_d, and K_c are the integrator, derivative, and proportional gain, respectively.

3.2.2 Type-1 Fuzzy PID Controller

In this section, the general structure of the T1-FPID will be presented. Before starting T1-FPID, the fuzzy PID control block for T1 and IT2 is shown in Figure 3.3 ([96]–[98]). The position error (e), and its rate of change (Δe) have been shown as controller inputs ([28], [29]).

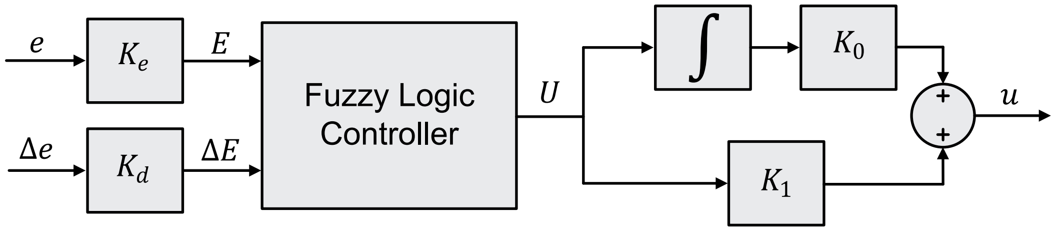

Figure 3.3: Illustration of the T1 and IT2 FPID controller.

As shown in Figure 3.3, the input Scaling Factors (SFs) are denoted with K_e and K_d, while the output scaling factor (SFs) are denoted K_0 and K_1. The equation is in the form:

$$E = K_e e, \quad \Delta E = K_d \Delta e, \tag{3.15}$$

where E and ΔE denote the FLC inputs. The output of the FLC (U) is transformed to the actual control signal of the T1 or IT2 fuzzy PID controllers (u) as follows

$$u = K_1 U + K_0 \int U dt. \tag{3.16}$$

Table 3.2: The rule table of T1 and IT2 FPID controller

$E/\Delta E$	**P**	**Z**	**N**
P	PB:1	PM:0.65	Z:0
Z	PM:0.65	Z:0	NM:-0.65
N	Z:0	NM:-0.65	NB:-1

The employed Fuzzy Logic Controller (FLC) has been designed with 3x3 rules as given in Table 3.2. The consequent part of the rules is described with crisp singletons. Moreover, they are represented with five linguistic fuzzy terms, which are defined with PB: Positive Big, PM: Positive Medium, Z: Zero, NM: Negative Medium, NB: Negative Big.

In Figure 3.4, the membership functions (MFs, μ) of the T1-FPID have been presented for the UAV [28], [29]. There are 3 membership functions in T1-FPID: "N, Z, P", abbreviated from "Negative", "Zero", and "Positive", which are the linguistic fuzzy variables. The antecedents

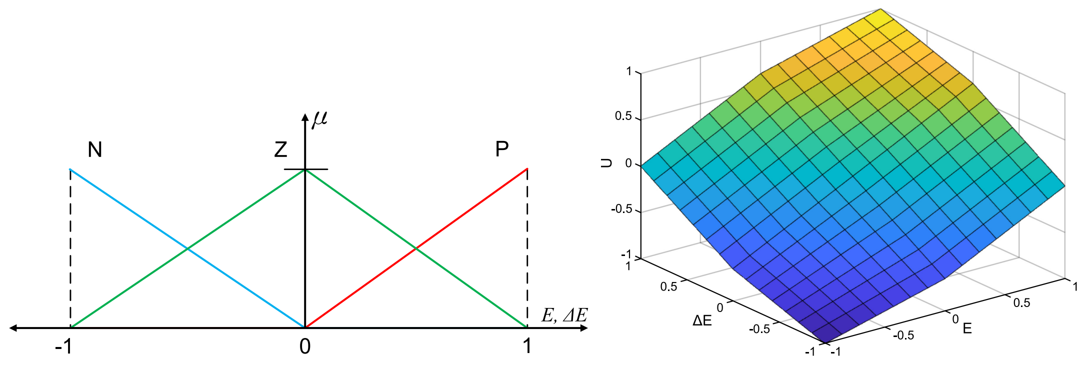

Figure 3.4: Illustrated T1 fuzzy MFs and control surface.

are defined with uniformly distributed symmetrical triangular membership functions. In this study, triangular membership functions have been selected instead of Gaussian or other types of membership functions. The aim of that is for fast computation time and calculation simplicity [99]. Moreover, the membership function values have been determined as empirical. The resulting control surface is shown in Figure 3.4. The implemented FLC uses the product implication and operates the centre of sets defuzzification method ([100]–[103]). In this study, the fuzzy weighted average defuzzification method has been used, and the equation is written in Equation (3.17).

$$U = \frac{\sum_{i=1}^{M} y^i f^i}{\sum_{i=1}^{M} f^i} \tag{3.17}$$

where U is the final output of the Takagi-Sugeno-Kang fuzzy inference system ([104]), and f is the firing function of T1-FPID, y is denoted as consequent membership functions, and M is the total number of rules and Table 3.2 has been used for T1-FPID.

3.2.3 Interval Type-2 Fuzzy PID Controller

Just like the T1-FPID controller, the IT2-FPID controller is composed of choosing E and ΔE as inputs and the control action of the IT2-FLC U as output, as displayed in Figure 3.3. The IT2-FLC has the same N=9 rules and is presented in Table 3.2. The membership functions of IT2-FPID are characterised by triangular membership functions, and the control surface of the IT2-FPID is given in Figure 3.5. The linguistic fuzzy variables are N, Z, and P, respectively. Here, m_n is the only parameter that creates the footprint of the uncertainty of the IT2-FPID ([105]). We will employ the following parameters: $m_2 = \alpha$ and $m_1 = m_3 = 1 - \alpha$ as suggested in ([100], [105]–[109]). Therefore, the selection of the parameter (α) is vital because it directly affects U.

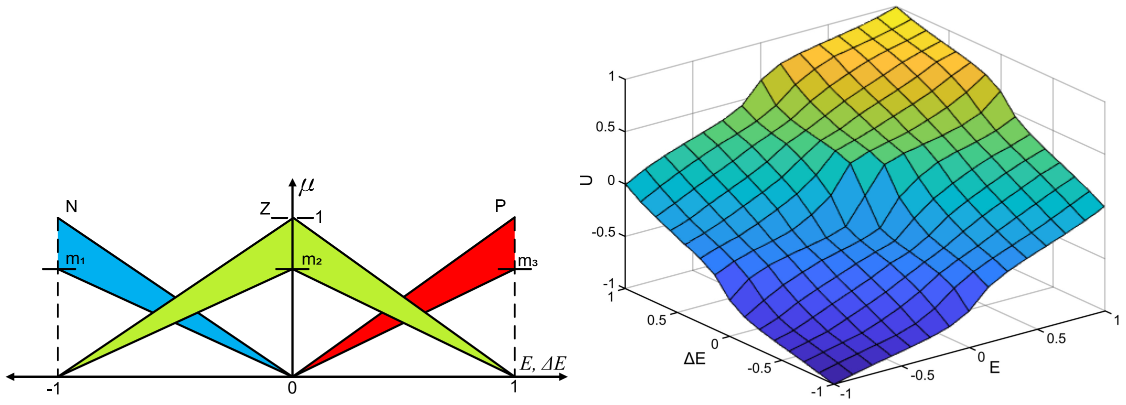

Figure 3.5: Interval type-2 fuzzy MFs and control surface.

Several methods have been used for the calculation of the controller output (U) ([105], [110]). In this study, Biglarbegian–Melek–Mendel method has been used ([105]). The equation relating to this method is given as follows:

$$\begin{aligned}\bar{y} &= \frac{\sum_{i=1}^{M} f_h^i y^i}{\sum_{i=1}^{M} f_h^i} \\ \underline{y} &= \frac{\sum_{i=1}^{M} f_l^i y^i}{\sum_{i=1}^{M} f_l^i} \\ \mathrm{U} &= \zeta \underline{y} + (1-\zeta)\bar{y}.\end{aligned} \tag{3.18}$$

where y^i shows consequent membership functions, M is the total number of rules and Table 3.2 has been used for IT2, ζ is a weighting parameter for type reduced set ($[\underline{y}, \bar{y}]$) and U is the output of FLC. Lastly, f_l and f_h are defined as lower and upper firing functions of IT2-FPID, respectively. In Figure 3.5, control surface which selected $\zeta = 0.25$ and $\alpha = 0.25$ for IT2-FPID has been illustrated.

Parameters of T1-FPID and IT2-FPID have been selected aggressively by using rule-based FLCs and membership functions. The ζ and α parameters have been selected by trial and error so that an aggressive control surface was created for IT2-FPID, then it is defined as $\zeta = 0.5$, $\alpha = 0.25$ in the study. After that, a comparison of the control surfaces (ΔU) of the T1-FPID (U_{T1}) and IT2-FPID (U_{IT2}) is presented in Figure 3.6, taking into account their aggressiveness and smoothness. The differences between them have been formulated as $\Delta U = U_{T1} - U_{IT2}$. A relatively higher control action output characterises an aggressive control action compared to the others. Specifically, if a line lies above another line in the positive region where $E, \Delta E \geq 0$ (or below in the negative region where $E, \Delta E < 0$), the resulting control action is deemed aggressive. Conversely, if this condition is not met, it is deemed a smooth control action.

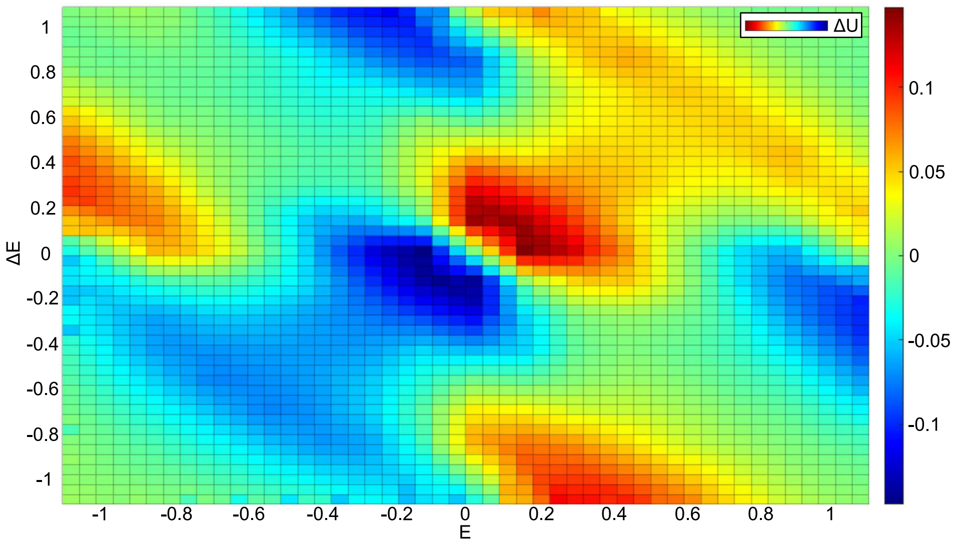

Figure 3.6: The control surface differences between T1-FPID and IT2-FPID.

3.2.4 Big Bang-Big Crunch Optimisation Method

To find the optimal parameters of the controllers, the Big Bang-Big Crunch (BB-BC) optimisation method has been used. The proposed optimisation method has shown a faster response than the global optimisation methods [111]. The BB-BC exists in 2 primary phases. The "Big Bang phase" involves randomly distributed candidate solutions across the search space, followed by the "Big Crunch phase," which involves calculating the population's centre of mass using a contraction procedure. Like other evolutionary search methods, the initial Big Bang population is produced randomly over the search space. Similarly, all following Big Bang phases are randomly distributed around the centre of mass or the best-fitted individual.

In ([97], [111], [112]), the working concept of this evolutionary approach consists of changing a converging result into a chaotic state, which is a new set of solutions. The algorithm starts by generating a random population of candidates for calculating fitness values. Using Equation 3.19, the centre of mass is calculated, and the selection will be based on either the best individual (fitness value) or the centre of mass. Similarly to evolutionary computing (EC) algorithms, a new set of candidates is generated based on the previously selected individual via adding or subtracting a random number whose value changes adaptively. The whole process is then repeated (without the initialisation) if the stopping criterion has not been met.

The BB-BC optimisation technique has been written in Algorithm 1.

Algorithm 1 The Pseudo-code of the Big Bang Big Crunch Optimisation Algorithm

```
• Step 1: (Big Bang Phase).
    An initial generation of N candidates is generated randomly in the search space.
• Step 2:
    The cost function values of all the candidate solutions are computed.
• Step 3: (Big Crunch Phase).
    The centre of mass is calculated. Either the best-fit individual or the centre of mass is
  chosen as the point of the Big Bang Phase.
• Step 4: (Path construction).
    New candidates are calculated around the new point.
• Step 5: (Path construction).
    Return to Step 2 until the stopping criteria have been met.
```

During the Big Crunch, a contraction method is used after the Big Bang. The contraction operator computes a centre of mass using the current locations of each candidate resulting in the population and its associated cost function value. The centre of mass x_c position may be calculated using the following formula:

$$x_c = \frac{\sum_{i=1}^{N} \frac{1}{f^i} x_i}{\sum_{i=1}^{N} \frac{1}{f^i}}. \tag{3.19}$$

and x_i is the candidate position, f^i is the cost function value for each candidate i and population size N. The new value x_i^{new} is

$$x_i^{new} = x_c + \sigma. \tag{3.20}$$

where x_i is the new candidate result and σ gives the standard deviation of a standard normal distribution.

For the next iteration Big Bang phase, the new generation is distributed around x_c as follows:

$$\sigma = \frac{r\alpha\left(x_{max} - x_{min}\right)}{k}, \tag{3.21}$$

In Equation (3.21), the standard deviation reduces as each repetition progresses. Moreover, r is a random number and α is a parameter bounding the size of the search space and

$$x_i^{new} = x_c + \frac{r\alpha(x_{max} - x_{min})}{k}. \tag{3.22}$$

Here, the maximum and minimum limits are given as x_{max} and x_{min}, respectively. Also, k is the number of iterations for the optimisation. While typically distributed numbers may exceed ± 1, the population must be limited to the search space boundaries. Therefore, the candidate results are constrained inside the search space boundaries due to this narrowing.

3.2.5 Optimisation Results and Simulation

All implemented controllers are optimised via the BB-BC algorithm described in the previous section. The cost function of BB-BC is defined as:

$$\text{ISE} = \int(e_x^2 + e_y^2)dt. \tag{3.23}$$

In the equation above, errors have been defined for two axes as e_x, e_y of the UAV. The error values have been explained in Equation (3.13), and integral square error (ISE) has been selected for performance criteria for the optimisation algorithm. In the BB-BC algorithm, the population size has been chosen as 50, and the maximum iteration number has been selected as 200 for the optimisation process of each controller. The BB-BC optimisation algorithm has been used to tune K_c, T_i, T_d, K_e, K_d, and K_0, K_1. It is noticed that α and ζ have been defined empirically. Control signals υ_x and υ_y have been constrained between $-12m/s$ and $12m/s$ during the BB-BC search.

In Table 3.3, parameters of the controllers and the best performance criteria (ISE) obtained by BB-BC have been listed. For tuning to the parameters of each controller, the position reference points are $x_{ref} = 1.1m$ and $y_{ref} = 0.52m$ for BB-BC optimisation.

The optimised controllers have been tested on the simulation model, and the results can be

Controller		Parameters		ISE
PID	:	$K_c = 0.0472$, $T_i = 0.0052$, $T_d = 0.0445$	:	82133.4
T1-FPID	:	$K_e = 0.019$, $K_d = 0.0019$, $K_0 = 0.0001$, $K_1 = 30$	:	55128.2
IT2-FPID	:	$K_e = 0.019$, $K_d = 0.0019$, $K_0 = 0.0001$, $K_1 = 30$, $(\alpha = 0.25, \zeta = 0.5)^\star$	:	36371.9

Table 3.3: Tuned parameters and their performance criteria results.
*: These parameters have been selected before BB-BC.

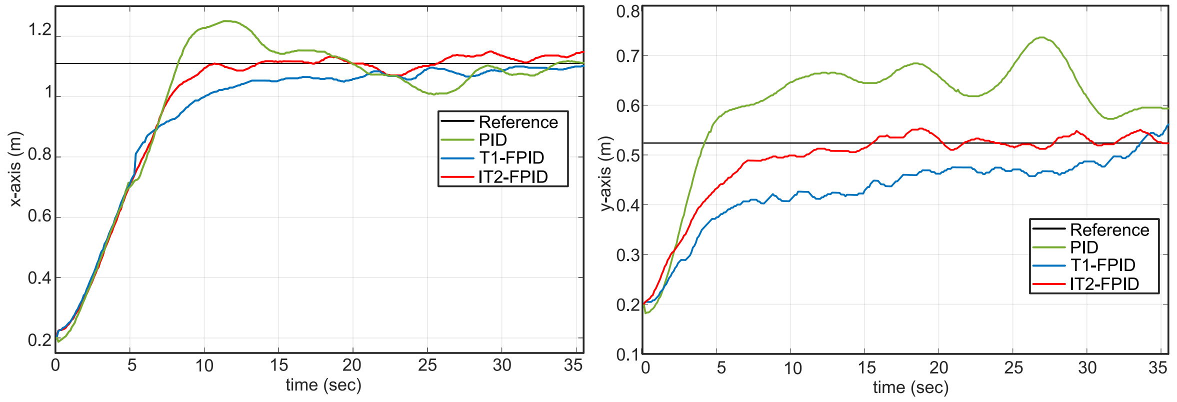

Figure 3.7: Testing results of the compared controllers in the simulation.

seen in Figure 3.7. In the simulation, the reference signals have been defined as $1.1m$ for the x-axis and $0.52m$ for the y-axis. They are the same as the references given in the controller parameter optimisation process using BB-BC.

These simulation results show that the IT2-FPID controller led to the best results by no overshoot, the slightest steady-state error, and rise time. In addition, T1-FPID gives a smoother response than the PID controller. Considering the compared controllers, they have not been the same system responses in both axes due to the fact that the feedback of the UAV in both axes has different levels of noise. The aim of choosing different levels of noise is that the payload will not be connected to the CoM of the UAV, and it will have a directly different effect on the axes. Oscillation and steady-state errors seen in Figure 3.7 are caused by the Gaussian disturbances added to the simulation model in the testing stage to show the robustness of the compared controllers against external disturbances. Gaussian distribution parameters for the x and y axes have been defined as $N(0, 5)$ and $N(0, 15)$, respectively. The selected distribution values have come from the general camera and GPS data. After testing the controllers in the simulation, the controllers were implemented in the UAV. In Appendix **A**, tuned parameters' results for all controllers have been shown.

3.3 Indoor Testing Area

In Figure 3.8, an overview of the system, including the proposed IT2-FPID-based position controller, is shown. In the overview system, references and controllers and position detection/estimation have been generated from the computer. The computer used in the real-time experiments has an Intel i7 CPU, 32GB RAM and NVidia GTX 1650Ti GPU. The generated outputs of the computer are the references of the UAV. Moreover, Python 3.10 environment has been used in the system [113]. The computer and the UAV have connected via a Wi-Fi communication protocol. Considering the main system, the process will start with the reference path generation. There are 2 different reference points in the proposed approach. The first one is about disturbing the payload. It means that; when the UAV tracks the reference points using a stick, it will be disturbed the flex cable-connected payload will a disturbed. Then, the second test will be initiated as Coverage Path Planning (CPP). Next, the offline CPP will be explained, and the resulting reference points are given [114]–[116]. The last part is about how to detect the UAV position with a camera and vision-based localisation algorithm. In the following 2 sections, these will be explained intensively.

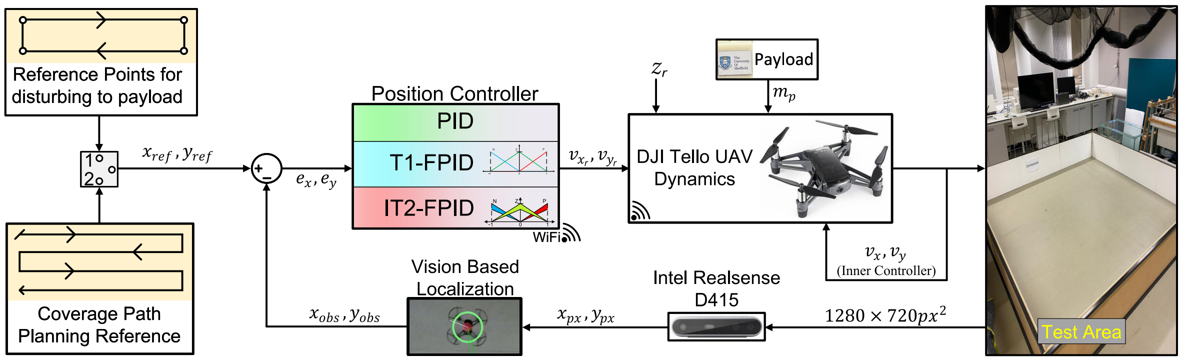

Figure 3.8: Fuzzy logic-based position control system.

3.3.1 Intel Realsense Depth Camera and Vision-Based Localisation Method

In this study, an Intel realsense depth camera has been used for the experiments ([117]) for it is a low-cost localisation system ([28], [118]). There are several advantages of using this particular depth camera in terms of its size, camera specifications and 3D Cartesian coordinate detection. Moreover, it has $69° \times 42° (\alpha \times \beta)$ field of view (FoV) for RGB sensor, $1280 \times 720px^2$ resolution, and $30fps$ RGB frame rate.

Table 3.4: Camera HSV parameters

HSV		**Hue**	**Saturation**	**Value**
Maximum Values	:	160	60	0
Minimum Values	:	200	255	255

To find the localisation of the UAV, RGB colour detection has been used to track it. A red-coloured circle piece has been put on the region of interest of the area, and the colour parameters are measured as shown in Table 3.4. These colour parameters are in the hue, saturation, and value colour space (HSV). Moreover, the HSV values have maximum and minimum limits, which are shown in Table 3.4.

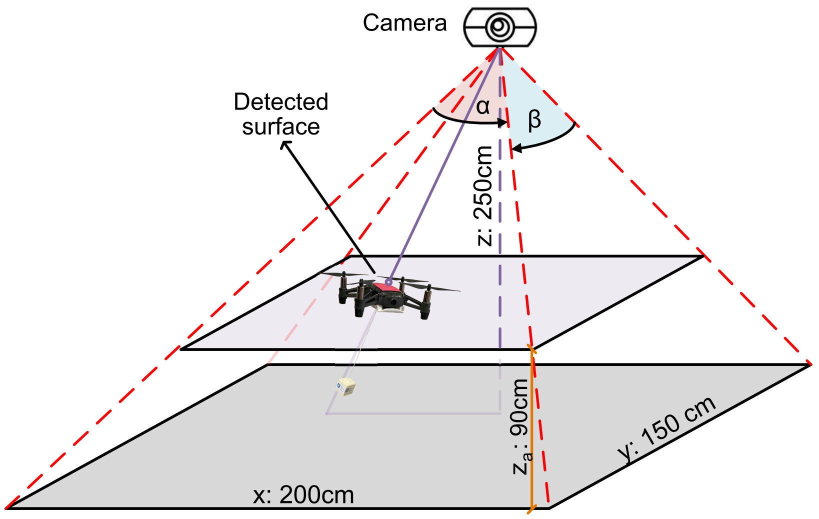

Figure 3.9: Camera trigonometric formation scheme and the UAV position.

After detecting the HSV parameters, some trigonometric formulas have been used to find the transformed centre points of the UAV. Figure 3.9 shows the three-axis sizes of the testing area $(x, y, z : 200, 150, 250$, respectively), camera position, UAV altitude level $(z_a : 90cm)$, and detected surface on the UAV.

In Equation (3.24), defined for the raw centre points of the UAV ($\hat{x}_{px}$ and $\hat{y}_{px}$), which have converged to the transformed centre points. In the matrix, s_x, s_y, sh_x, sh_y and t_x, t_y stand for scale factors, shear factors, and translation factors, respectively. Moreover, these parameters have been determined by using "Computer Vision Toolbox for Matlab" ([119]) and using the following model:

$$\begin{bmatrix} \hat{x}_{px} \\ \hat{y}_{px} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & sh_y & 0 \\ sh_x & s_y & 0 \\ t_x & t_y & 1 \end{bmatrix} \begin{bmatrix} x_{px} \\ y_{px} \\ 1 \end{bmatrix}. \tag{3.24}$$

Normally, this calculation has been set for the 2 axes because the $z-axis$ has been fixed. For the calculation for the 3-axes, the trigonometric formula has been used via Equation (3.25). In this equation, M_x and M_y show the total number of pixels for the x-y axes, and also σ_x and σ_y have the been defined as relative mounting angles of FoV. Lastly, h (or $z-axis$) has been used for the distance between the camera and the ground or floor. In Figure 3.9, $x-y-z-axes$ dimensions are given as 200cm, 150cm, and 250cm, respectively and by accounting for the relations:

$$\begin{aligned} x_{obs} &= -z\tan\left(\sigma_x + \beta\left(\hat{y}_{px}/M_x - 1\right)\right), \\ y_{obs} &= -z\tan\left(\sigma_y + \alpha\left(\hat{x}_{px}/M_y - 1\right)\right). \end{aligned} \tag{3.25}$$

3.3.2 Coverage Path Planning

This section presents the implemented CPP algorithm. CPP is another challenging problem for UAVs ([114]–[116]). The UAV aims to track the specific points defined by using the CPP method.

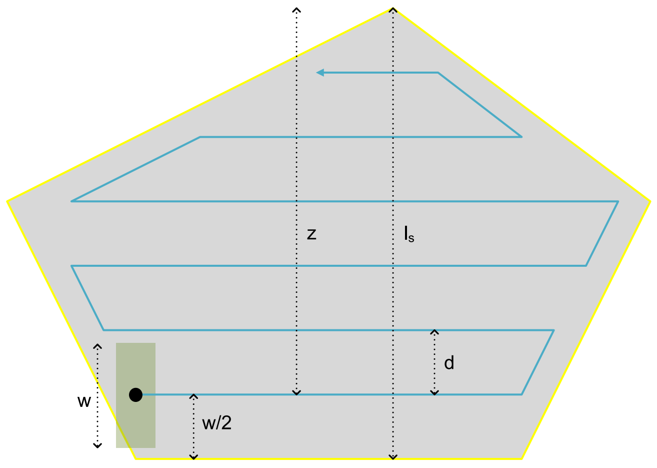

Figure 3.10: Example of coverage path planning.

In Figure 3.10, the boundaries, starting point, and the specific reference of CPP lines have been shown and coloured yellow, black, and turquoise, respectively. This is because the UAV has a task: reaching the specific points, which are the endpoints of the path segments. The following relation is used:

$$d = w.(1 - \upsilon). \tag{3.26}$$

The shape, length, and direction of the path constructed by the CPP algorithm depend on d, w, and l_s values, and these correspond to the distance among rows, the half of the camera footprint, and the width of the original polygon which is swept respectively and the percentage of vertical overlap between the width of the camera footprints on consecutive rows are denoted by υ and with a total number n of turns equal to:

$$n = \begin{cases} 2.\lceil z/d \rceil & \text{if } \ (z \bmod d) \ \leq w/2, \\ 2.(\lceil z/d \rceil + 1) & \text{if } \ (z \bmod d) \ > w/2. \end{cases} \tag{3.27}$$

The total number of turns n resulting from the CPP algorithm is given in Equation (3.27), and this algorithm aims to minimise n to some degree as it can be seen that this number depends on the z value, which is a function of the polygon width that is $z = l_s - w/2$. Therefore, the direction in which the polygon width is defined determines the value of n.

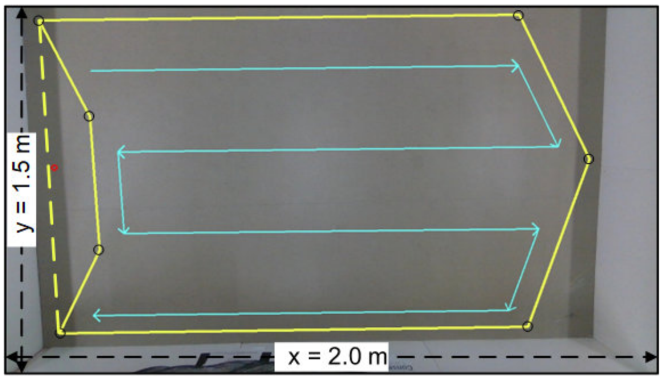

Figure 3.11: Coverage path planning for the UAV.

If the gap between the last row of the created path and the vertex or the edge of the original polygon is greater than half of the camera footprint w, then an extra row is added to the end of the path to reduce the gap as mentioned earlier; therefore the distance between the last row and the original polygon might be less than $w/2$. In Figure 3.11, the difference between the distances from the border of the polygon to the initial row and the last row of the CPP path is an example of this case. The optimal line sweep direction and the path are determined by the following algorithm given in 2 ([114], [115]).

Algorithm 2 Basic CPP algorithm.

- **Step 1:** (Farthest vertex v from the edge e).

  For each edge, find the farthest vertex with respect to Euclidean distance and keep this (e, v) pair and the calculated distance value.
- **Step 2:** (Minimum distance pair (e,v)).

  Compare the distance values and find the (e, v) pair, which gives the minimum distance value.
- **Step 3:** (Orthogonal direction vector).

  The optimal line-swept direction is the vector that is orthogonal to the edge from the (e, v) pair found in Step 2, and the rows constructed by the CPP algorithm will be parallel to this edge e.
- **Step 4:** (Path construction).

  The path is defined by the waypoints that are connected by line segments. These waypoints are placed in such a way that the line segments connecting them are parallel to the edge on which the polygon width is defined, and the distance between a waypoint and the border of the polygon satisfies the camera footprint requirements.

In the following section, the training and the testing results of the compared controllers will be given, and it will be shown that the IT2-FPID controller has obtained the best performance.

3.4 Real-Time Experimental Results

In this section, the performance of the IT2-FPID controller has been compared with the PID and the T1-FPID controllers by the data gathered from real-time experiments. All experiments are in 2 parts: without/with disturbance and payload on the set-point tracking. The tuned parameters without payload using the BB-BC algorithm have been employed in the UAV. After that, the set-point tracking shows all the results of the compared controllers under the flexible cable-connected payload and disturbance effects. Comparison results for these controllers are shown as Root Mean Square Error (RMSE) values. The sampling time (T_s) has been chosen as 0.055 s. For selecting T_s, the UAV IP address, communicated via WiFi, has been pinged, and latency has been calculated. With this latency has, been shown minimum T_s of the system. The performance of the proposed approach can be observed via the **video file** provided as **Supplementary Material**.

The implemented controllers have been tested for 3 cases. First, the UAV tracked the set point without any disturbance and the payload. Secondly, the controllers have tried to stabilise the system under disturbance and the payload. Also, the square reference points have been given to the UAV. The last case gives the reference path for the UAV by the CPP algorithm. In the CPP scenario, any external disturbance will not be applied because when the UAV follows the reference trajectory, the flexible cable-connected payload may generate disturbances on both axes due to the fact that the payload has not been connected to the CoM of the UAV.

Disturbing the payload aims to investigate the controller responses under disturbance. In the last test, CPP aims to compare the controller performances for minimum time and maximum accuracy. Disturbing the payload and the CPP results show the robustness of the controller in real-time.

3.4.1 Set-point Tracking without Disturbance and the Payload

Before testing with the unknown payload, all control methods and their control parameters have been tested without a connected payload. The reason for this is that the defined controller parameters may need further fine-tuning.

In Figs. 3.12 and 3.13, PID, T1-FPID and IT2-FPID position results can be seen. The reference, PID, T1-FPID, and IT2-FPID, have been coloured black, green, blue, and red, respectively. The starting point of the UAV has been selected as $x_0 = 0.2m, y_0 = 0.2m$ and the reference has been chosen as $x_{ref} = 1.1m, y_{ref} = 0.52m$.

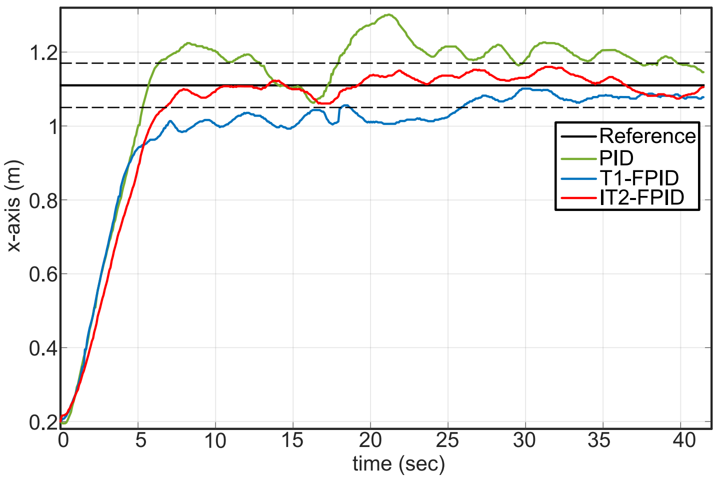

Figure 3.12: Set-point tracking results on the x-axis.

In Figure 3.12, the IT2-FPID results have shown the best result compared to other controllers regarding settling time, oscillation, steady-state error, and overshoot. PID gives more overshoot and then arrives at coverage of the reference area after 31 sec. T1-FPID has a slower response than compared with IT2-FPID, but it has also reached the reference faster than PID.

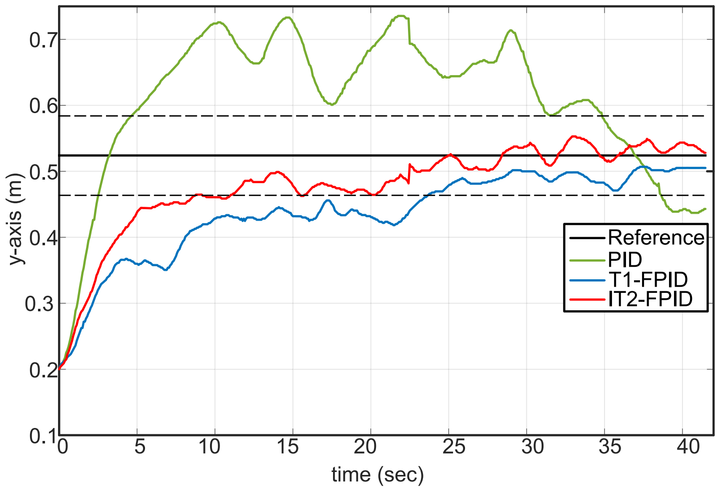

Figure 3.13: Set-point tracking results on the y-axis.

In Figure 3.13, unlike the results on the x-axis, all the controllers have not achieved good results on the y-axis. Significantly, the PID controller has a much overshoot problem and does not reach the reference or converge to the circle. Moreover, IT2-FPID looks faster than T1-FPID, and the system response is the best.

3.4.2 Set-point Tracking with Disturbance and the Payload

Set-point tracking with disturbance and the payload results consist of two different parts. The first relates to payload disturbing, and the second applies to CPP. In the second part, all controller structures were tested under disturbance. The payload was disturbed using a stick to implement the disturbances as external disturbances, directly impacting the system stability. After that, in the CPP scenario, the disturbances may be generated by themselves, as explained in the aforementioned above section.

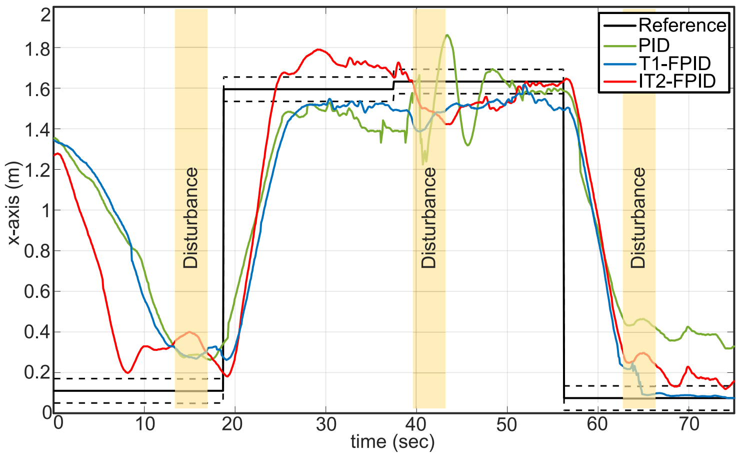

Figure 3.14: Disturbance results on the x-axis.

In Figure 3.14, disturbance effects have been seen on the x-axis. PID and T1-FPID have been given slow responses, and generally, they have not reached the reference points. IT2-FPID has not been as affected as the others.

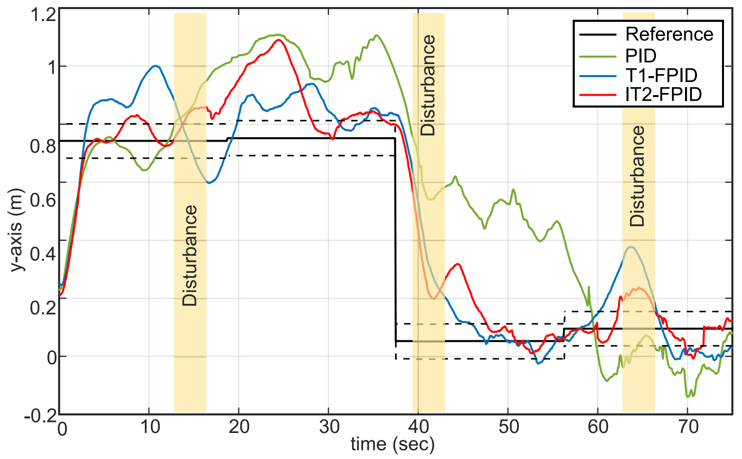

Figure 3.15: Disturbance results on the y-axis.

In Figure 3.15, the biggest problem for all controller types is on the y-axis. The PID cannot guarantee stability under disturbance, and T1-FPID and IT2-FPID showed a slight overshoot.

After the disturbance-based results, some specific points have been given for each controller. The second experiment of the IT2-FPID controller is coverage path planning. Each vertex of the coverage path planning has been fixed and determined before the test, and these are given as reference points. The reference point changes to the next vertex when the UAV converges or reaches the current vertex. Considering the task compilation time, IT2-FPID, T1-FPID, and PID have completed the task in 74.55 sec, 112.35 sec, and 120.15 sec, respectively. In addition to a faster completion time, IT2-FPID also shows better results in other ways.

The first experiment of the CPP has been the PID controller structure for the real-time UAV with a flexible cable-connected payload, as shown in Figure 3.16. PID shows a small steady-state error and slower system response on the x-axis. However, PID has a problem on the y-axis. It has not reached reference points properly.

In Figure 3.17, T1-FPID has not shown promising results compared with the PID results on the x-axis, but it completed the task time faster than the PID controller; these results have been plausible. However, T1-FPID oscillated on the y-axis. This issue can be explained by the fact that the payload is not at the CoM of the UAV but nearer the front side. Therefore, the problem on the y-axis has been expected for all the controller's structures.

The last and the best results achieved with the IT2-FPID are shown in Figure 3.18. It has completed the CPP task in 74.55 sec. In addition, it has a better settling time, oscillation, and steady-state error in terms of minor errors and short time. Although the payload position problem on CoM has existed in the other controllers, IT2-FPID has eliminated the problem steadfastly.

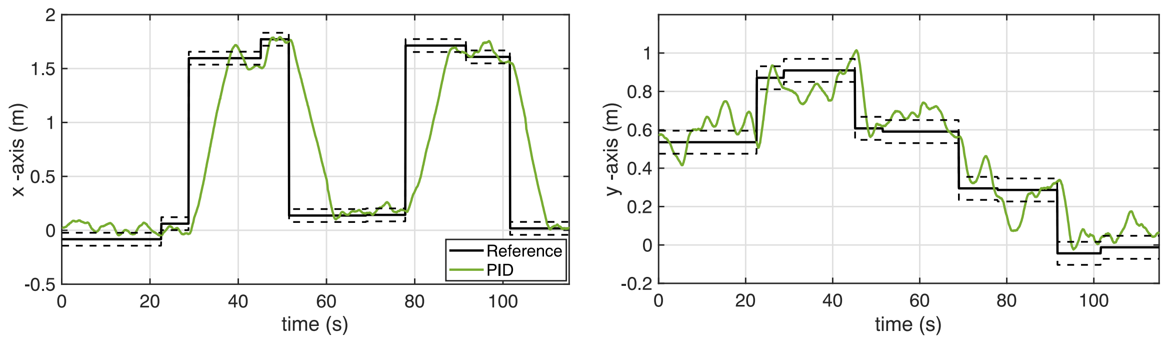

Figure 3.16: Coverage path planning with PID.

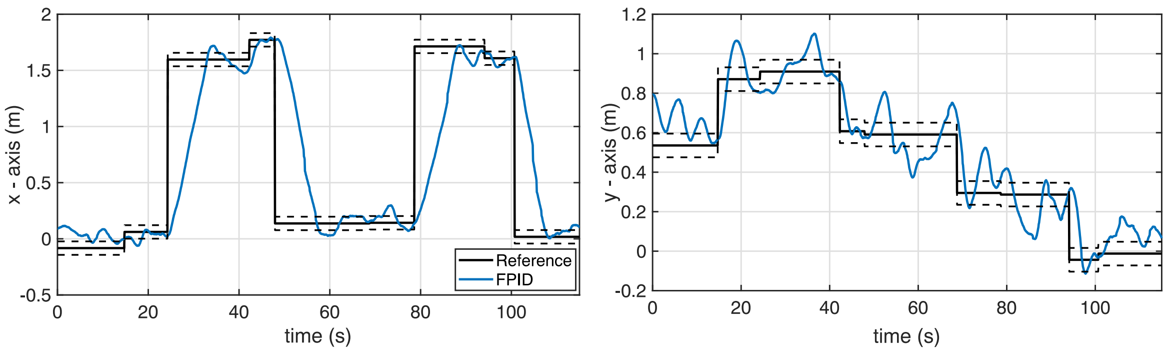

Figure 3.17: Coverage path planning with T1-FPID.

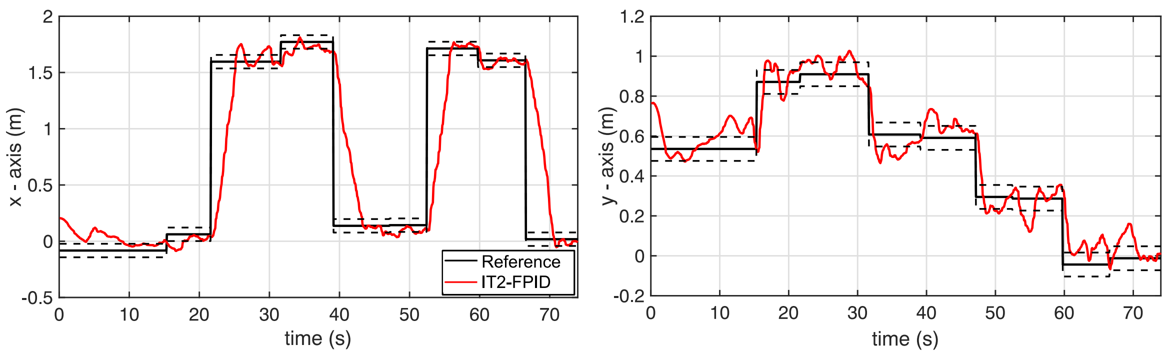

Figure 3.18: Coverage path planning with IT2-FPID.

The experimental results on each axis have been analysed separately. RMSE has been chosen as a performance index for the comparison of the controllers. In Table 3.5, the results of RMSE on the x-axis have been shown. The best results of each experiment have been IT2-FPID on the x-axis.

Table 3.5: RMSE Values on x - axis

RMSE Values (m)		PID	T1-FPID	IT2-FPID
Set-Point Tracking	:	0.2275	0.2255	**0.2218**
Under Disturbance	:	0.5574	0.5490	**0.4971**
CPP Results	:	0.5719	0.5597	**0.4540**

As far as RMSE values obtained in the set-point tracking scenario and in the testing against the disturbance that is disturbing the payload with a stick are considered, IT2-FPID have shown almost two times better results when compared to the PID and T1-FPID as given in Table 3.6.

Table 3.6: RMSE Values on y-axis

RMSE Values (m)		PID	T1-FPID	IT2-FPID
Set-Point Tracking	:	0.137	0.0964	**0.0805**
Under Disturbance	:	0.3103	0.1877	**0.1810**
CPP Results	:	0.1338	0.1298	**0.1068**

In summary, IT2-FPID has the best performance results in all two robustness tests. Considering the aim of the testing for IT2-FPID, the minimum task time, and RMSE values, maximum accuracy has been obtained. However, although T1-FPID has not been as good as IT2-FPID, it has better results than the compared PID controller. In the simulation, 20 independent tests have been done, and then 5 independent tests have been run in the real-time system. In Appendix **A**, more experiments which are disturbing-based and CPP, have been represented.

3.5 Conclusion

This study describes the challenge of controlling a UAV with an unknown payload connected by a flexible cable. The challenging problems are related to the swinging unknown payloads and point-based reference trajectories. For this challenging engineering problem, a proportional-integral-derivative (PID) controller, type-1 Fuzzy PID controller (T1-FPID), and interval type-2 Fuzzy PID controller (IT2-FPID) have been tuned and implemented on a real UAV. The control algorithms are generic and can be applied to different types and sizes of UAVs, although the experiments were performed using a DJI Tello UAV. The performance of the IT2-FPID has also been compared against a PID and a T1-FPID controller. After set-point tracking and robustness testing via disturbing the payload with a stick and changing the reference points in coverage path planning (CPP), causing the payload to swing, the experimental results show that the IT2-FPID is better than the controllers. There have been different aims for each test, such as minimum task time and RMSE. Disturbing effects of the flexible cable-connected payload on the UAV are shown throughout the experiments. These adverse effects can be eliminated by the three compared controllers to some point, but results show the superiority of the IT2-FPID. As an example from the experiments, although the PID controller may solve the trajectory tracking problem, the uncertainties, such as the unknown payload connection via flexible cable and disturbing the payload, led to highly detrimental results for the PID controller. Considering the RMSE values, the reference trajectory and the task time, these problems have been eliminated considerably when implementing the IT2-FPID controller. Consequently, the system stability has been guaranteed without considering the changes in the system dynamics equations and redesigning the controller.

After the controller design, the sensor part of the UAV will be improved to achieve better accuracy. In order to improve the sensor accuracy, the noise distribution will also be investigated, and then the system will be tested both in the simulation environment and in real-time. According to the type of distribution, a filter design will be included to eliminate such noise and general disturbances.

Chapter 4

Methods for State Estimation of the Unmanned Aerial Vehicles with Maximum Correntropy Filters

In this chapter, we have described state estimation methods. To enhance estimation accuracy and reduce computation time, we employed the multiple-model (MM) approach, leveraging its use of non-complex models. Consequently, we integrated the compared and proposed state estimation methods with the MM method.

4.1 Interacting Multiple Model

The multiple-model (MM) method is a significant approach employed for target tracking when dealing with motion uncertainty. It is often considered the most intuitive approach to hybrid estimation. This method utilises a bank of filters, each associated with a specific model representing various possible system behaviour patterns, such as manoeuvres, that are relevant to the problem at hand. These system behaviour patterns are commonly referred to as system modes [120].

By employing multiple models, the MM method aims to cover a range of potential system behaviours that the target being tracked may exhibit. This approach acknowledges the inherent uncertainty in the target's motion and accounts for various plausible scenarios. Each model within the bank of filters provides an estimation of the target's state based on a specific system behaviour pattern, allowing for a comprehensive analysis of the target's dynamics.

Overall, the MM method serves as an effective strategy for target tracking in the presence of motion uncertainty. It leverages a diverse set of models to capture different system modes, enabling a more robust and accurate estimation of the target's behaviour.

The early results obtained from noninteracting (static) MM estimation are primarily applicable to systems with time-invariant unknown or uncertain system modes. However, they are inadequate when it comes to addressing challenges such as tracking manoeuvring targets, where the system mode undergoes frequent transitions. The development of the interacting multiple-model (IMM) estimator was a significant breakthrough that made the MM approach practical for manoeuvring target-tracking scenarios [59].

Using the IMM-based filter, it is important to note that increasing the number of models and filters is not always the optimal solution Firstly, doing so significantly raises computational complexity, which can become impractical in many real-world situations. Additionally, even when utilising more models and filters optimally, theoretical evidence suggests that it does not guarantee performance improvement. As a result, a dilemma arises regarding the fixed structure of the MM approach: deciding whether to incorporate more or fewer models.

Finding the right balance between the number of models and filters is crucial. Increasing the number of models beyond a certain point may not yield significant benefits but would incur higher computational costs. On the other hand, reducing the number of models may lead to oversimplification, potentially limiting the ability to accurately capture and estimate the various system behaviours [59], [121].

Therefore, in practice, it is essential to strike a balance between the number of models and the associated computational complexity to achieve an effective and efficient MM estimation approach for maneuvering target tracking.

The IMM is a type of hybrid state estimation algorithm for Markovian linear systems ([59], [122]). It has many advantages in terms of computation time, target tracking, fault detection, and classification of the models [59]. The model structure has been illustrated in Figure 4.1. Each model has a different probability weight. Then, IMM calculates an estimate and covariance matrices to send to the Kalman filter as inputs [59]. In the figure, $\hat{x}$ and $\bar{x}$ are represented as prediction state and estimation state, respectively. Moreover, subscripts of the states $(._{i|i})$ are defined as the next step. In this section, general IMM formula parts have been explained.

In Appendix **B**, the detailed information for IMM and the algorithm have been written.

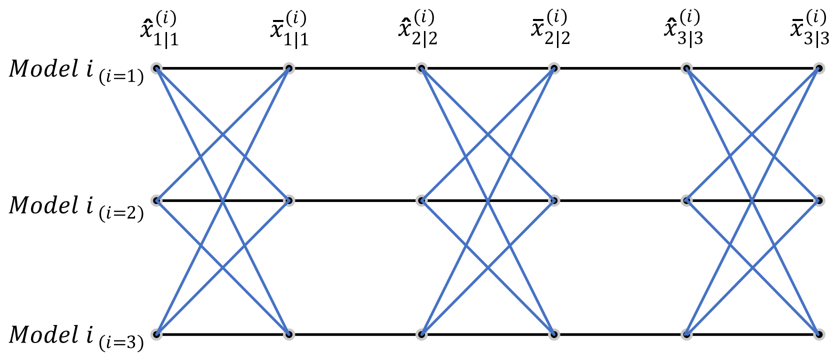

Figure 4.1: Interacting multiple model structure [59]

There are three kinematic models have been implemented in the IMM structure. They have been one constant velocity ($\mathbf{F}_{CVM}$) and two different constant turning models ($\mathbf{F}_{CTM}$). The respective matrices are of the form:

$$\mathbf{F}_{CVM} = \begin{bmatrix} 1 & \mathrm{T} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{T} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

$$\mathbf{F}_{CTM} = \begin{bmatrix} 1 & sin(\omega_i \mathrm{T})/\omega_i & 0 & -(1-cos(\omega_i \mathrm{T})/\omega_i \\ 0 & cos(\omega_i \mathrm{T}) & 0 & -sin(\omega_i \mathrm{T}) \\ 0 & (1-cos(\omega_i \mathrm{T}))/\omega_i & 1 & sin(\omega_i \mathrm{T})/\omega_i \\ 0 & sin(\omega_i \mathrm{T}) & 0 & cos(\omega_i \mathrm{T}) \end{bmatrix} \tag{4.2}$$

In Equation (4.1) and (4.2), sampling time is declared as T and manoeuvring of UAV ω have existed of 2 parts. For the left acceleration model, ω_i is $\omega = 2\pi/180$, on the other hand $\omega = -2\pi/180$ is chosen for right acceleration model. These two acceleration models generate a constant turning model($\mathbf{F}_{CTM}$).

The IMM consists of 4 parts; mixing, filtering, mode update, and output, respectively. Predicted model probability and mixing weight, estimate, and covariance are in the mixing part.

Firstly, predicted model probability ($\hat{\mu}_{k|k-1}^{(i)}$) formula in Equation (4.3), and mixing weight ($\mu_{k-1}^{j|i}$) can be seen in Equation (4.4). Here z^k shows current measurement data.

$$\hat{\mu}^{(i)}_{k|k-1} = \boldsymbol{P}\left\{m^{(i)}_k | z^{k-1}\right\} = \textstyle\sum_j \pi_{ji}\hat{\mu}^{(j)}_{k-1} \tag{4.3}$$

$$\mu^{j|i}_{k-1} = \boldsymbol{P}\left\{m^{(j)}_{k-1} | m^{(i)}_k, z^{k-1}\right\} = \pi_{ji}\mu^{(j)}_{k-1}/\hat{\mu}^{(i)}_{k|k-1} \tag{4.4}$$

After that, estimated state in Equation (4.5) and covariance in Equation (4.6) and can be calculated:

$$\bar{x}^{(i)}_{k-1|k-1} = E\left[x_{k-1}|m^{(i)}_k, z^{k-1}\right] = \textstyle\sum_j \hat{x}^{(j)}_{k-1|k-1}\mu^{j|i}_{k-1} \tag{4.5}$$

$$\begin{aligned}\bar{\boldsymbol{P}}^{(i)}_{k-1|k-1} = \textstyle\sum_j \Big[P^{(j)}_{k-1|k-1} + \left(\bar{x}^{(i)}_{k-1|k-1} - \hat{x}^{(j)}_{k-1|k-1}\right) \\ \left(\bar{x}^{(i)}_{k-1|k-1} - \hat{x}^{(j)}_{k-1|k-1}\right)'\Big]\mu^{j|i}_{k-1}\end{aligned} \tag{4.6}$$

These mixed-estimated state and covariance matrices have been used for the proposed Kalman filter. Next, the Maximum correntropy Kalman filter (MCKF) will be explained. In this section, it is assumed that the Kalman filter is applied. Also, Kalman filter outputs have been defined as updated state $x^{(i)}_{k|k}$ and updated covariance $\boldsymbol{P}^{(i)}_{k|k}$.

After skipping the filtering part, the model probability update part has been used. It means that model likelihood and model probability will be calculated.

The model likelihood and the model probability are given in Equation (4.7) and in Equation (4.8), respectively.

$$L^{(i)}_k = \boldsymbol{p}\left[\tilde{z}^{(i)}_k | m^{(i)}_k, z^{k-1}\right] \cong \frac{e^{-(1/2)(\tilde{z}^{(i)}_k)'(S^i_k)^{-1}(\tilde{z}^{(i)}_k)}}{\sqrt{|2\pi S^i_k|}} \tag{4.7}$$

$$\mu^i_k = \boldsymbol{P}\left[m^{(i)}_k | z^k\right] = \frac{\hat{\mu}^{(i)}_{k|k-1}L^{(i)}_k}{\sum_j \hat{\mu}^j_{k|k-1}L^{(j)}_k} \tag{4.8}$$

The last part of the IMM is described next. It is composed of the overall state estimate and the covariance matrix.

$$\hat{x}_{k|k} = E\left[x_k|z^k\right] = \textstyle\sum_i \hat{x}^{(i)}_{k|k}\mu^{(i)}_k \tag{4.9}$$

$$\boldsymbol{P}_{k|k} = \sum_i \left[\boldsymbol{P}_{k|k}^{(i)} + \left(\hat{x}_{k|k} - \hat{x}_{k|k}^{i}\right)\left(\hat{x}_{k|k} - \hat{x}_{k|k}^{i}\right)'\right] \mu_k^{(i)}. \tag{4.10}$$

The following Section 4.3 presents the developed IMM-MCKF algorithm.

4.2 Kalman Filter

In this section, the traditional Kalman filter will be described. In Figure 4.2, the Kalman filter connection with the controller and the system is represented. The Kalman filter uses the system output (y) as a measurement and the system state ($\hat{x}$).

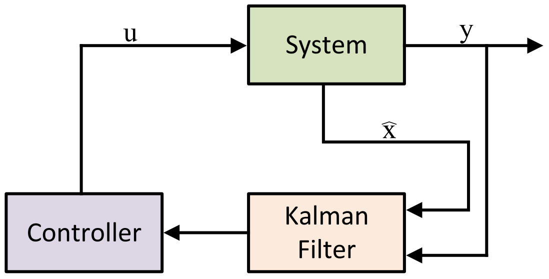

Figure 4.2: Kalman filter implemented system block diagram

There are many types of Kalman filters in terms of usage on the system, such as EKF, UKF or CKF. In the performance validation, linear models, such as the constant velocity and constant acceleration models, have been used instead of nonlinear state estimation filters, such as UKF or CKF.

In Figure 4.3, a general Kalman filter working system has been shown. Initial value with Kalman gain and measurement data create the updated estimate. Then, the output of the block is an input of the updated covariance matrices, and the system keeps the updated state estimates for the next step (k+1). After that, projected estimate values are compared with the next step.

The state and measurement equations of the Kalman filter can be represented as

$$\begin{aligned} x(k) &= \mathbf{F}\mathrm{x}(k-1) + \mathbf{q}(k-1), \\ y(k) &= \mathbf{H}(k)\mathrm{x}(k) + \mathbf{r}(k), \end{aligned} \tag{4.11}$$

where $x(k) \in \mathbb{R}^n$, $y(k) \in \mathbb{R}^m$, m represents the m-dimensional measurement vector instant k. **F** and **H** stand for the system transition matrix and observation matrix, respectively. Here

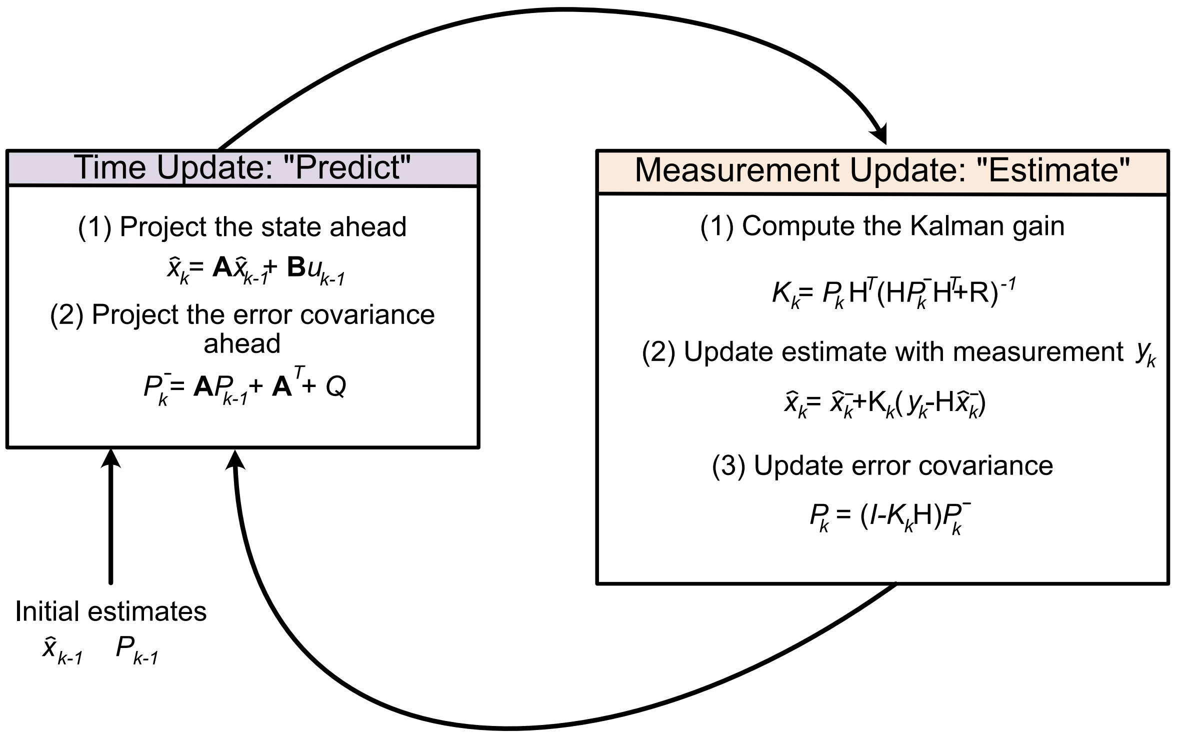
Time Update: "Predict"
(1) Project the state ahead
$\hat{x}_k^- = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}u_{k-1}$
(2) Project the error covariance ahead
$P_k^- = \mathbf{A}P_{k-1} + \mathbf{A}^T + Q$
Measurement Update: "Estimate"
(1) Compute the Kalman gain
$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$
(2) Update estimate with measurement y_k
$\hat{x}_k = \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-)$
(3) Update error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates
$\hat{x}_{k-1}$ P_{k-1}

Figure 4.3: The Kalman filter block diagram

$\mathbf{q}(k-1)$ and $\mathbf{r}(k)$ are mutually uncorrelated process noise and measurement noise with zero mean and covariance matrices and

$$\begin{aligned} \mathrm{E}[\mathbf{q}(k-1)\mathbf{q}^{\mathrm{T}}(k-1)] &= \mathbf{Q}(k-1), \\ \mathrm{E}[\mathbf{r}(k-1)\mathbf{r}^{\mathrm{T}}(k-1)] &= \mathbf{R}(k). \end{aligned} \tag{4.12}$$

In Equation (4.12), the processes covariance ($\mathbf{Q}(k-1)$) and measurement covariance ($\mathbf{R}(k)$) matrices have been derived from uncorrelated process noise and measurement noise, respectively.

In general, the Kalman filter includes the following two steps:

Prediction: The prior mean and covariance matrix are given by

$$\begin{aligned} \hat{\mathbf{x}}(k|k-1) &= \mathbf{F}\hat{\mathbf{x}}(k-1|k-1), \\ \mathbf{P}(k|k-1) &= \mathbf{F}\mathbf{P}(k-1|k-1)\mathbf{F}^T(k-1) + \mathbf{Q}(k-1). \end{aligned} \tag{4.13}$$

Update: The Kalman filter gain is computed as follows

$$\mathbf{K}(k) = \mathbf{P}(k|k-1)\mathbf{H}^{\mathrm{T}}(k)(\mathbf{H}(k)\mathbf{P}(k|k-1|)\mathbf{H}^{\mathrm{T}}(k) + \mathbf{R}(k))^{-1} \tag{4.14}$$

The posterior state is equal to the prior state plus the innovation weighted by the Kalman filter gain,

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)(\mathbf{y} - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)). \tag{4.15}$$

Additionally, the posterior covariance matrix is recursively updated as follows:

$$\mathbf{P}(k|k) = (\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k))(\mathbf{P}(k|k-1))(\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k))^{\mathrm{T}} + (\mathbf{H}(k)\mathbf{R}(k)\mathbf{H}(k)^{\mathrm{T}}). \tag{4.16}$$

4.3 Maximum Correntropy Kalman Filter

The Maximum Correntropy Kalman filter (MCKF) is based on maximum correntropy criteria and a fixed-point iterative algorithm [42], [43], [50]. It is well known that the conventional Kalman filter gives a good result under Gaussian noises. However, its performance worsens under non-Gaussian noises. This is an unavoidable outcome stemming from the utilization of the traditional Kalman filter, which relies on the Gaussian distribution [42], [43], [50].

Before starting the whole algorithm, correntropy criteria will be explained. When given two random variables $X, Y \in \mathbb{R}$ with joint distributed function $d\mathbf{F}_{XY}(xy)$, correntropy is described by

$$V(X,Y) = E\left[\kappa(X,Y)\right] = \int \kappa(x,y) d\mathbf{F}_{XY}(xy). \tag{4.17}$$

In Equation (4.17), the shift-invariant Mercer Kernel is denoted as $\kappa(.,.)$ and E represents the mathematical expectation operator. In this study, the Gaussian Kernel has been chosen, and it is given by

$$\kappa(x,y) = \mathrm{G}_{\sigma}(\epsilon) = \mathrm{e}^{\left(-\frac{\epsilon^2}{2\sigma^2}\right)}. \tag{4.18}$$

where $\mathrm{e} = x - y$ is calculated. Moreover, the σ value is chosen to be equal to 7 and the kernel bandwidth ε gets 10^{-3} values.

The predicted state vector and predicted covariance matrix can be calculated with the following equations:

$$\begin{aligned}\hat{\mathbf{x}}(k|k-1) &= \mathbf{F}\hat{\mathbf{x}}(k-1|k-1),\\ \mathbf{P}(k|k-1) &= \mathbf{F}\mathbf{P}(k-1|k-1)\mathbf{F}^T(k-1)+\mathbf{Q}(k-1).\end{aligned} \tag{4.19}$$

The kernel bandwidth σ and small positive threshold ε are set up accordingly from practical considerations. Also,

$$\begin{aligned}&\begin{bmatrix}\mathbf{P}(k|k-1|) & 0\\ 0 & \mathbf{R}(k)\end{bmatrix}\\ &=\begin{bmatrix}\mathbf{P}_p(k|k-1|)\mathbf{P}_p^T(k|k-1|) & 0\\ 0 & \mathbf{B}_r(k)\mathbf{B}_r^T(k)\end{bmatrix}\\ &=\mathbf{B}(k)\mathbf{B}^T(k).\end{aligned} \tag{4.20}$$

In Equation (4.20), $\mathbf{B}(k)$ represents the Cholesky factor. Then,

$$\begin{aligned}&\mathbf{D}(k)=\mathbf{W}(k)\mathbf{x}(k)+\mathbf{e}(k),\\ &\mathbf{D}(k)=\mathbf{B}^{-1}(k)\begin{bmatrix}\hat{\mathbf{x}}(k|k-1|)\\ \mathbf{y}(k)\end{bmatrix},\quad \mathbf{W}(k)=\mathbf{B}^{-1}(k)\begin{bmatrix}\mathbf{I}\\ \mathbf{H}(k)\end{bmatrix}.\end{aligned} \tag{4.21}$$

For the posterior estimation, it is updated by using fixed-point iteration for each $\hat{\mathbf{x}}^{(t)}(k|k)$

$$\hat{\mathbf{x}}^{(t)}(k|k)=\hat{\mathbf{x}}(k|k-1)+\widetilde{\mathbf{K}}^{(t-1)}(\mathbf{y}(k)-\mathbf{H}(k)\hat{\mathbf{x}}(k|k-1))\cdot \tag{4.22}$$

Considering the above equation, it includes some parameters which are detailed in Equation (4.23) and Equation (4.28)

$$\begin{aligned}\widetilde{\mathbf{K}}^{(t-1)}(k)&=\widetilde{\mathbf{P}}^{(t-1)}(k|k-1|)\mathbf{H}^{\mathrm{T}}(k)...\\ &\times\Big(\mathbf{H}(k)\widetilde{\mathbf{P}}(k|k-1|)\mathbf{H}^{\mathrm{T}}(k)+\widetilde{\mathbf{R}}^{(t-1)}(k)\Big)^{-1}\end{aligned} \tag{4.23}$$

$$\begin{aligned}\widetilde{\mathbf{P}}^{(t-1)}(k|k-1) = \mathbf{B}_p(k|k-1)\left(\widetilde{\mathbf{C}}_x^{(t-1)}(k)\right)^{-1}...\\ \times\mathbf{B}_p^{\mathrm{T}}(k|k-1)\end{aligned} \tag{4.24}$$

$$\widetilde{\mathbf{R}}^{(t-1)}(k) = \mathbf{B}_r(k)\left(\widetilde{\mathbf{C}}_y^{(t-1)}(k)\right)^{-1} \times \mathbf{B}_r^{\mathrm{T}}(k), \tag{4.25}$$

$$\widetilde{\mathbf{C}}_x^{(t-1)}(k) = \begin{bmatrix} \mathrm{G}_\sigma\left(\widetilde{e}_1^{(t-1)}(k)\right) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathrm{G}_\sigma\left(\widetilde{e}_n^{(t-1)}(k)\right) \end{bmatrix}, \tag{4.26}$$

$$\widetilde{\mathbf{C}}_y^{(t-1)}(k) = \begin{bmatrix} \ddots & 0 & 0 & 0 \\ 0 & \mathrm{G}_\sigma\left(\widetilde{e}_{n+1}^{(t-1)}(k)\right) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathrm{G}_\sigma\left(\widetilde{e}_{n+m}^{(t-1)}(k)\right) \end{bmatrix}, \tag{4.27}$$

$$\widetilde{e}_i^{(t-1)} = d_i(k) - \mathbf{w}_i(k)\hat{\mathbf{x}}^{(t-1)}(k|k)\cdot \tag{4.28}$$

In Equation (4.29), estimation states have been compared with the current and the last step by a small positive threshold (ε).

$$\frac{\left\|\hat{\mathbf{x}}^{(t)}(k|k)-\hat{\mathbf{x}}^{(t-1)}(k|k)\right\|}{\left\|\hat{\mathbf{x}}^{(t-1)}(k|k)\right\|} \leq \varepsilon. \tag{4.29}$$

When the required condition in the above equation is satisfied, the posterior covariance matrix can be determined as follows

$$\begin{aligned}\mathbf{P}(k|k) = \left(\mathbf{I}-\widetilde{\mathbf{K}}(k)\mathbf{H}(k)\right)\mathbf{P}(k|k-1)\left(\mathbf{I}-\widetilde{\mathbf{K}}(k)\mathbf{H}(k)\right)^{\mathrm{T}}\\ +\widetilde{\mathbf{K}}(k)\mathbf{R}(k)\widetilde{\mathbf{K}}(k)^{\mathrm{T}}.\end{aligned} \tag{4.30}$$

In Figure 4.8, the architecture of IMM-MCKF has been shown. In this study, unlike the conventional IMM structure, IMM and MCKF have combined with each other. Compared with the proposed architecture, the conventional one has used a traditional Kalman filter in the model-conditioned filtering part (IMM-KF or IMM).

In probability and statistics, the Student's t-distribution, often simply referred to as the t-distribution, is a continuous probability distribution that extends the characteristics of the standard normal distribution. It shares the traits of being symmetric around zero and exhibiting a bell-shaped curve.

However, what sets the Student's t-distribution apart is its heavier tails, and the extent of probability in these tails is determined by the degree of freedom parameter (ν). When $\nu = 1$, the Student's t-distribution becomes equivalent to the standard Cauchy distribution, whereas as ν approaches infinity, it converges to the standard normal distribution $N(0, 1)$.

Hence, the Student's t-distribution is a valuable choice for comparing state estimation methods due to its ability to model various tail behaviors, making it versatile for a range of statistical applications.

The proposed method has aimed to estimate localisation under non-uniform distribution. For this reason, the Students-T distribution, which is a type of non-uniform or non-Gaussian distribution, has been used for the experiments.

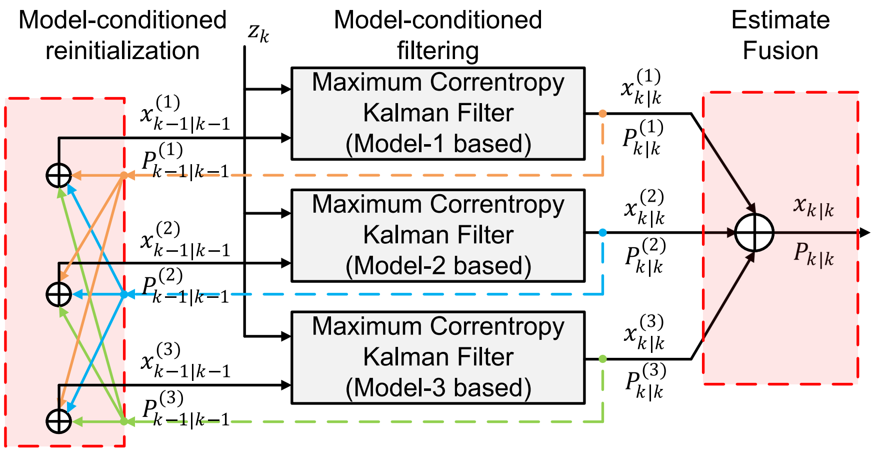

Figure 4.4: Architecture of IMM-MCKF

At the end of this section, IMM and MCKF have been explained in terms of equations and structures. Simulation results will be given by the following section.

4.4 Simulation Results

This study is about the implementation of IMM-MCKF for the UAV. Unlike conventional methods such as IMM-KF or single-model approaches, IMM-MCKF has been tested under non-uniform distribution. In this way, Student's-T distribution has been used to choose different degrees of freedom [49], [51], [52]. All experiments have been run 3000 independent Monte-Carlo times. The computer used for these experiments has an Intel i7 processor, 16GB RAM and a Matlab 2021 environment.

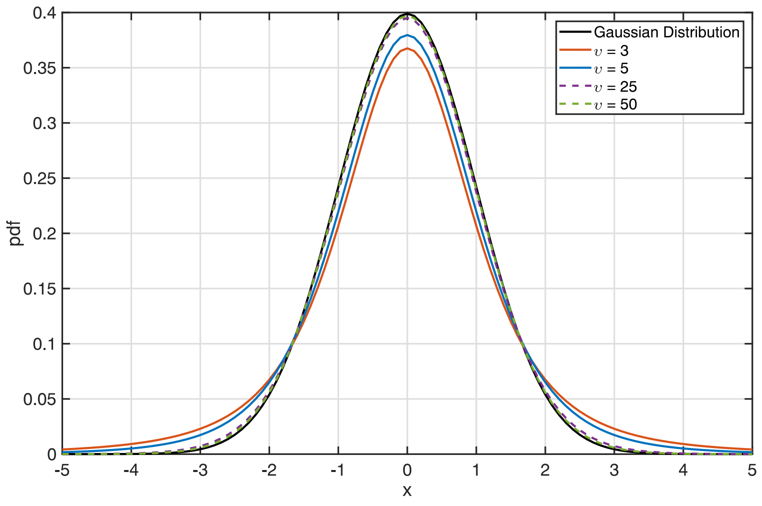

Figure 4.5: Gaussian and Student's-T distributions

For the experiments, the same error covariance matrices have been chosen. The covariance matrices have been defined as Gaussian covariances. Two different IMM methods have been compared under the Student's-T distribution. Also, variable degrees of freedom (DoF) have been selected for the Student's-T distribution in these tests. These distributions have been used as noises for the UAV measurement.

In Equation (4.31), **H** measurement matrix has been defined as below:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.31}$$

It has been remarked that initial covariance, process noise, and measurement noise matrices have been constructed with the same values for a fair comparison of IMM filters.

In Figure 4.5, Gaussian and Student's-T distributions have been shown. The important point of this figure has been about changing DoF. Noises have been implemented into measured data $(x, y, \upsilon_x, \upsilon_y)$ with using scaling matrix factor (β). The scaling matrix factor shows the power of noises.

In Figure 4.6 shows UAV reference trajectory under Student's-T noises with choosing $\mu_1 = 4$ and $\mu_2 = 8$. Moreover, β has been chosen 0.02. Trajectory reference, noise added reference, IMM-KF result, and IMM-MCKF result have been shown with different colours as green, black, blue, and red, respectively.

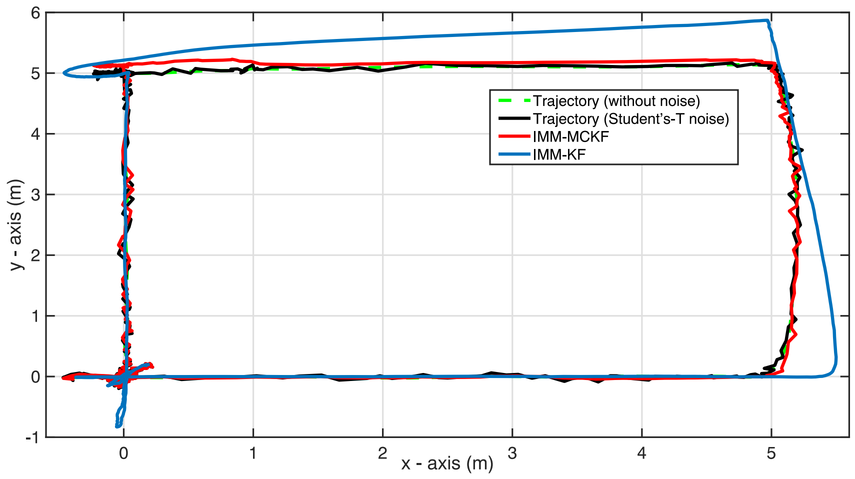

Figure 4.6: Reference trajectory tracking under Student's-T distribution ($\mu_1 = 4$ and $\mu_2 = 8$)

It is shown that IMM-MCKF and IMM-KF have given almost the same results for the first x-axis. Then, IMM-MCKF has shown better results than IMM-KF at the first y-axis and the second side of the x-axis.

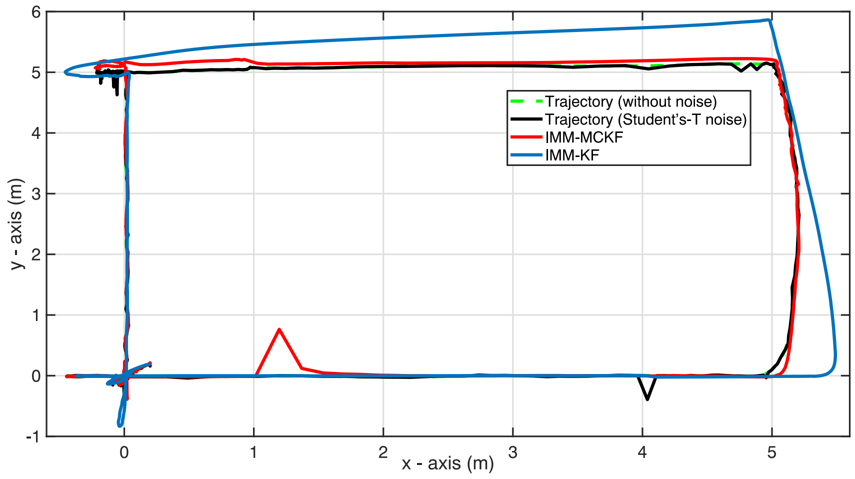

Figure 4.7: Reference trajectory tracking under Student's-T distribution ($\mu_1 = 1$ and $\mu_2 = 3$)

After the first experiment, $\mu_1 = 1$, $\mu_2 = 3$ and $\beta = 0.002$ have been selected for the second experiment. With these selections, DoF and scaling matrix factor effects have been seen in Figure4.7. Considering the IMM-KF reference trajectory estimation, the UAV has gone out of axes. Although, IMM-MCKF has continued to reference with a small oscillation.

The testing parameters have been given in Table 5.1. The average of Root Mean Square

Errors (RMSEs) values (in meters) over the 3000 Monte Carlo independent runs and their total times (in seconds) have been given in this table. Considering the total time differences for both experiments, the IMM-KF has given a 3.35 times faster response. In Table, Γ denotes $\mu = 4 - 8 \;\; \beta = 0.02$ parameters and Λ denotes $\mu = 1 - 3 \;\; \beta = 0.002$ parameters.

Table 4.1: Test Results of IMM-KF and IMM-MCKF

IMM-Based Filter	time_Γ	RMSE_Γ	time_Λ	RMSE_Λ
KF	2300 s	0.335 m	2024 s	0.343 m
MCKF	7710 s	0.081 m	6950 s	0.077 m

Table 5.1 shows the average RMSE and the total computational time of both algorithms. When choosing $\mu_1 = 4$, $\mu_2 = 8$ and $\beta = 0.02$, the IMM-MCKF has been given 4.1 times better RMSE results than IMM-KF. These consequences have been almost the same in the second experiment. The IMM-MCKF has been 4.45 times more accurate than the other algorithms in the presence of non-Gaussian noises.

4.5 Maximum Correntropy Student's T Filter

The proposed MCStF has the assumption of non-Gaussian distributed measurement noises. In this section, the maximum correntropy criterion (MCC) and Student's T filter to derive MCStF formulation have been explained deeply. Unlike under Gaussian noises, the traditional Kalman filter may perform dramatically worse when faced with non-Gaussian noises, and impulsive noises stimulate the particular underlying system.

Considering a linear model [42], [123]

$$\begin{bmatrix} \widehat{\mathbf{x}}(k|k-1) \\ \mathbf{y}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}(k) \end{bmatrix} \mathbf{x}(k) + \boldsymbol{\upsilon}(k), \tag{4.32}$$

where $\mathbf{x}(k) \in \mathbf{R}^n$ denotes an n-dimensional system state vector at discrete time k, $\mathbf{y}(k) \in \mathbf{R}^m$ is an m-dimensional measurement vector, $\mathbf{I}$ is the $n \times n$ identity matrix, $\mathbf{H}(k)$ is the measurement matrix and $(\widehat{\mathbf{x}}(k|k-1))$ is the predicted state estimate at time k. Following the notations of [42], an aggregated noise vector $\boldsymbol{\upsilon}$

$$\boldsymbol{\upsilon}(k) = \begin{bmatrix} -(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1)) \\ \mathbf{r}(k) \end{bmatrix} \tag{4.33}$$

comprises both the state error noise and measurement noise $\mathbf{r}(k)$. The state error covariance matrix $(\mathbf{P}(k|k-1))$ and measurement noise covariance matrix $(\mathbf{R}(k))$ can be written as

$$\mathbf{E}[v(k)v^T(k)] = \begin{bmatrix} \mathbf{P}(k|k-1) & 0 \\ 0 & \mathbf{R}(k) \end{bmatrix}, \tag{4.34}$$

where $\mathbf{E}[.]$ denotes the mathematical expectation operator. Equation (4.34) can be represented in the form

$$\begin{bmatrix} \mathbf{B}_p(k|k-1)\mathbf{B}_p^T(k|k-1) & 0 \\ 0 & \mathbf{B}_r(k)\mathbf{B}_r^T(k) \end{bmatrix} = \mathbf{B}(k)\mathbf{B}^T(k), \tag{4.35}$$

where $\mathbf{B}(k)$ is obtained by using the Cholesky factorisation of the expression on the left. When multiplying both sides of (4.32) by $\mathbf{B}^{-1}(k)$, we obtain

$$\mathbf{D}(k) = \mathbf{W}(k)\mathbf{x}(k) + \mathbf{e}(k) \tag{4.36}$$

where

$$\begin{aligned} \mathbf{D}(k) &= \mathbf{B}^{-1}(k)\begin{bmatrix} \hat{\mathbf{x}}(k|k-1) \\ \mathbf{y}(k) \end{bmatrix}, \\ \mathbf{W}(k) &= \mathbf{B}^{-1}(k)\begin{bmatrix} \mathbf{I} \\ \mathbf{H}(k) \end{bmatrix} \end{aligned}. \tag{4.37}$$

In Equation (4.36), the residual error $\mathbf{e}(k)$ is defined as $\mathbf{e}(k) = \mathbf{B}^{-1}(k)\upsilon(k)$. Moreover, it can be written as $\mathbf{E}[\mathbf{e}(k)\mathbf{e}^T(k)] = \mathbf{I}$ in the case of (4.36). Therefore, the residual error $\mathbf{e}(k)$ must be white.

Considering the MCC iterations, it can be written that the ith element of $\mathbf{e}(k)$ is:
$e_i(k) = d_i(k) - \mathbf{w}(k)\mathbf{x}(k)$. The MCC cost function J_L is in the form:

$$J_L(\widehat{\mathbf{x}}^*(k|k)) = \frac{1}{L}\sum_{i=1}^{L} G_\sigma(d_i(k) - \mathbf{w}_i(k)\widehat{\mathbf{x}}(k|k-1)), \tag{4.38}$$

with $L = m + n$ being the dimension of $\mathbf{B}(k)$ and $\mathbf{w}_i(k)$ is the row of $\mathbf{W}(k)$. The optimal state estimate $\widehat{\mathbf{x}}(k|k)$ is calculated from:

$$\widehat{\mathbf{x}}(k|k) = \underset{\mathbf{x}(k|k)}{\text{argmax}} J_L(\mathbf{x}^*(k|k)) = \underset{\mathbf{x}(k|k)}{\text{argmax}} \sum_{i=1}^{L} G_\sigma(e_i(k)), \tag{4.39}$$

also, G_σ denotes the kernel function which is the Gaussian kernel [42]. Unlike the MCKF, $\mathbf{M}_x$ and $\mathbf{M}_y$ need be calculated, as in Equation (4.40) and applied in Student's T filter to satisfy this condition. ,

$$\begin{aligned} \mathbf{M}_x(k) &= \text{diag}(\mathrm{G}_\sigma(\mathrm{e}_1(k)), \cdots, \mathrm{G}_\sigma(\mathrm{e}_n(k))) \\ \mathbf{M}_y(k) &= \text{diag}(\mathrm{G}_\sigma(\mathrm{e}_{n+1}(k)), \cdots, \mathrm{G}_\sigma(\mathrm{e}_{n+m}(k))) \end{aligned}, \tag{4.40}$$

these calculated diagonal matrices have been used to calculate error covariance $\mathbf{P}(k|k-1)$ and error measurement covariance $\mathbf{R}(k)$ matrices, in Eqns. (4.41) and (4.42)

$$\begin{gathered}\mathbf{P}(k|k-1) = \mathbf{B}_p(k|k-1)\left(\widetilde{\mathbf{M}}_x^{(t-1)}(k)\right)^{-1} \\ \mathbf{B}_p^{\mathrm{T}}(k|k-1)\end{gathered}, \tag{4.41}$$

$$\mathbf{R}(k) = \mathbf{B}_r(k)\left(\widetilde{\mathbf{M}}_y^{(t-1)}(k)\right)^{-1}\mathbf{B}_r^{\mathrm{T}}(k), \tag{4.42}$$

$$\begin{gathered}\widehat{\mathbf{x}}^*(k|k) = \widehat{\mathbf{x}}(k|k-1) + \mathbf{P}(k|k-1)\mathbf{H}(k)^{\mathrm{T}}(\mathbf{S}(k))^{-1} \\ (\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k|k-1))\end{gathered}, \tag{4.43}$$

$$\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}(k)^{\mathrm{T}} + \mathbf{R}(k)\cdot \tag{4.44}$$

Considering the MCC-based cost function, iteration σ and small positive threshold ε values have been defined. In ref. [42], [124], the effects of σ and ε have been investigated and compared with different values.

$$\frac{\left\|\widehat{\mathbf{x}}^{*,(t)}(k|k) - \widehat{\mathbf{x}}^{*,(t-1)}(k|k)\right\|}{\left\|\widehat{\mathbf{x}}^{*,(t-1)}(k|k)\right\|} \leq \varepsilon, \tag{4.45}$$

Equation (4.45) gives the cost function, which depends on the current state estimate $\widehat{\mathbf{x}}^{*,(t)}(k|k)$ and the previous state estimate $\widehat{\mathbf{x}}^{*,(t-1)}(k|k)$. The t index denotes the number of iterations, and $||.||$ denotes the Euclidean norm. Then, the estimated state value can be obtained as $\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}^*_{k|k}$ if the calculated value is ($\leq \varepsilon$) in Equation (4.45).

$$\begin{gathered}\mathbf{P}^*(k|k) = \frac{\upsilon(k-1) + \Delta^2(k)}{\upsilon(k-1) + d}(\mathbf{P}(k|k-1) - \mathbf{P}(k|k-1) \\ \mathbf{H}^{\mathrm{T}}(k)(\mathbf{S}(k))^{-1}\mathbf{H}(k)\mathbf{P}(k|k-1))\end{gathered}, \tag{4.46}$$

$$\begin{gathered}\Delta^2(k) = (\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k|k-1))^{\mathrm{T}} \\ (\mathbf{S}(k))^{-1}(\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k|k-1))\end{gathered}. \tag{4.47}$$

In Equation (4.46), the updated error covariance matrix has been calculated after the necessary condition is done. In this work, flexible Student's T distribution has been used under heavy-tailed noise. A method including noises with heavy-tailed distributions is proposed in [49], [125], in particular with a Student's T filter and the covariance matrix is represented as:

$$\mathbf{P}(k|k) = \frac{\upsilon^*(k)}{\upsilon^*(k)-2}\frac{\upsilon(k)-2}{\upsilon(k)}\mathbf{P}^*(k|k)^{\cdot} \tag{4.48}$$

In Equation (4.48), the updated error covariance matrix of the flexible Student's T filter has been defined. In the equation, υ is the degree of freedom, and d is the dimension of the state vector. In addition, it is obtained as $\upsilon^*(k) = \upsilon(k-1) + d$. After all the mathematical background, the MCStF's implementation pseudocode of the MCStF is shown in Algorithm (3).

Algorithm 3 The implementation pseudocode for one time-step of the MCStF

```
Inputs: $\hat{\mathbf{x}}(k-1|k-1)$, $\mathbf{P}(k-1|k-1)$, $\mathbf{Q}(k-1)$, $\mathbf{R}(k)$, $\upsilon$, $\varepsilon$, $\sigma$
Time update:
  1. $\hat{\mathbf{x}}(k|k-1) = \mathbf{F}(k)\hat{\mathbf{x}}(k-1|k-1)$.
  2. $\mathbf{P}(k|k-1) = \mathbf{F}(k)\mathbf{P}(k-1|k-1)\mathbf{F}^{\mathrm{T}}(k) + \mathbf{Q}(k-1)$.
Measurement update:
  1. $\mathbf{B}_p = \mathrm{Chol}(\mathbf{P}(k|k-1))$, $\mathbf{B}_r = \mathrm{Chol}(\mathbf{R}(k))$,
     $\hat{\mathbf{x}}^{(t=t_0)}(k) = \hat{\mathbf{x}}(k|k-1)$.
  2. $\mathbf{M}_x(k) = \mathrm{diag}(\mathrm{G}_\sigma(\mathrm{e}_1(k)), \cdots, \mathrm{G}_\sigma(\mathrm{e}_n(k)))$,
     $\mathbf{M}_y(k) = \mathrm{diag}(\mathrm{G}_\sigma(\mathrm{e}_{n+1}(k)), \cdots, \mathrm{G}_\sigma(\mathrm{e}_{n+m}(k)))$.
  3. $\mathbf{P}(k|k-1) = \mathbf{B}_p(k|k-1)(\widetilde{\mathbf{M}}_x^{(t-1)}(k))^{-1}\mathbf{B}_p^{\mathrm{T}}(k|k-1)$,
     $\mathbf{R}(k) = \mathbf{B}_r(k)(\widetilde{\mathbf{M}}_y^{(t-1)}(k))^{-1}\mathbf{B}_r^{\mathrm{T}}(k)$.
  4. $\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}(k)^{\mathrm{T}} + \mathbf{R}(k)$,
     $\widehat{\mathbf{x}}^{*,(t)}(k|k) = \widehat{\mathbf{x}}^{(t)}(k|k-1) + \mathbf{P}(k|k-1)\mathbf{H}(k)^{\mathrm{T}}$
     $(\mathbf{S}(k))^{-1}(\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k|k-1))$.
  5. $\dfrac{\left\|\widehat{\mathbf{x}}^{*,(t)}(k|k) - \widehat{\mathbf{x}}^{*,(t-1)}(k|k)\right\|}{\left\|\widehat{\mathbf{x}}^{*,(t-1)}(k|k)\right\|} \leq \varepsilon$,
     where $\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}^{*,(t)}(k|k)$ is performed and go to Step 6 if the termination condition is
     satisfied; else, Step 2 is returned, and the iteration is continued after setting $t = t+1$.
  6. $\Delta^2(k) = (\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k|k-1))^{\mathrm{T}}(\mathbf{S}(k))^{-1}$
     $\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k|k-1)$.
  7. $\mathbf{P}^*(k|k) = \dfrac{\upsilon(k-1) + \Delta^2(k)}{\upsilon(k-1) + d}(\mathbf{P}(k|k-1) - \mathbf{P}(k|k-1)$
     $\mathbf{H}^{\mathrm{T}}(k)(\mathbf{S}(k))^{-1}\mathbf{H}(k)\mathbf{P}(k|k-1))$.
  8. $\mathbf{P}(k|k) = \dfrac{\upsilon^*(k)}{\upsilon^*(k)-2}\dfrac{\upsilon(k)-2}{\upsilon(k)}\mathbf{P}^*(k|k)$.
Outputs: $\hat{\mathbf{x}}(k|k)$, $\mathbf{P}(k|k)$.
```

The presumed state-space model that accurately represents the real system significantly impacts how well state estimation techniques work. However, model uncertainties appear when the system is complicated and challenging to model precisely [126]. If measurement noise statistics are unknown, uncertainties may also exist from the method and their statistics [127]. The rapid shift in the system dynamics brought by, for example, target movement is another typical source of model uncertainty [121]. The presence of model uncertainty may severely hamper the state estimation performance.

In contrast, several candidate models are used in multiple-model techniques to address model uncertainty [128]. IMM produces the state estimate, combining the filtering results. In this work, IMM has been combined with multiple MCStFs by using different models. These models will be explained in the next section.

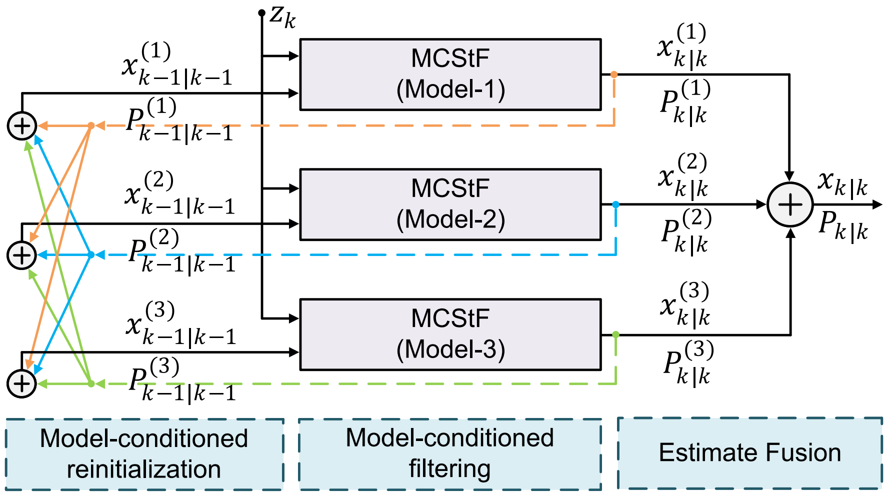

Figure 4.8: Architecture of IMM-MCStF.

The architecture diagram of IMM-MCStF is shown in Figure 4.8. The IMM-MCStF structure consists of three MCStFs that operate with three different motion models: the nearly constant velocity and coordinated turn models [59], [121], which are explained in the next section. Each motion model has been specified for each MCStF. The benefits of the IMM have been widely discussed in the literature [49], [121].

4.6 Performance Evaluation and Validations

This section explains how to perform numerical simulation and real-time experiments of IMM-based KF, MCKF and MCStF methods. All experiments are started on the simulation environment, and non-Gaussian noises are implemented on the system model. Then, the state estimation results are evaluated and compared using Root Mean Square Error (RMSE) as a performance measure. Afterwards, we present the real-time experimental results to show the performance of the proposed IMM-MCStF against the conventional IMM-KF and IMM-MCKF. Note that the height was set and fixed as 40 cm in all the conducted experimental studies.

4.6.1 Numerical Simulation

The simulations have been performed on the Crazyflie 2.0 UAV model. The detailed model has been explained in many research [22], [28], [124]. The model contains the Crazyflie 2.0's positions in the 2-D plane and is used as the reference benchmark data set to test the performances of the implemented filters; IMM-KF, IMM-MCKF, and IMM-MCStF. During the simulation, it is assumed that the measurements are corrupted with a Student's T distribution noise which can be defined as one of the non-Gaussian distribution types. Thus, the reference benchmark data set has been corrupted with a noise that has a Student's T distribution. In other words, the employed IMM methods are tested in an environment with noise with non-Gaussian distribution.

Table 4.2: Choosing DoF values

$\upsilon = 3$	extreme level heavy-tailed distribution
$\upsilon = 50$	intermediate level heavy-tailed distribution
$\upsilon = 100$	approximate Gaussian distribution

In [49], the effects of DoF have been explained and shown in Table 4.2. In this study, the distribution has been chosen between $[3-25]$, which means the noise is variable on intermediate and extreme levels heavy-tailed.

A constant velocity model and coordinated turn models with state transition matrices respectively ($\mathbf{F}_{CVM}$) and ($\mathbf{F}_{CTM}$) represent the different types of UAV motion. Two coordinated turn models have been implemented, representing two different directions (left turn and right turn motions). To determine the non-Gaussian process model, ($\mathbf{F}_{CTM}$). var(x) $= \alpha V$ has been merged with the defined covariance matrices.

The system state transition matrices of the constant velocity model and of the coordinated turn models have the form:

$$\mathbf{F}_{CVM} = \begin{bmatrix} 1 & \Delta\mathrm{T} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta\mathrm{T} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.49}$$

$$\mathbf{F}_{CTM} = \begin{bmatrix} 1 & sin(\omega_i \Delta\mathrm{T}) & 0 & -(1 - cos(\omega_i \Delta\mathrm{T})/\omega_i \\ 0 & cos(\omega_i \Delta\mathrm{T}) & 0 & -sin(\omega_i \Delta\mathrm{T}) \\ 0 & (1 - cos(\omega_i \Delta\mathrm{T}))/\omega_i & 1 & sin(\omega_i \Delta\mathrm{T})/\omega_i \\ 0 & sin(\omega_i \Delta\mathrm{T}) & 0 & cos(\omega_i \Delta\mathrm{T}) \end{bmatrix}, \tag{4.50}$$

and the respective system noise covariance matrix of the coordinated turn model is:

$$\mathbf{Q} = \alpha \begin{bmatrix} \frac{\Delta\mathrm{T}^4}{4} & \frac{\Delta\mathrm{T}^3}{2} & 0 & 0 \\ \frac{\Delta\mathrm{T}^3}{2} & \Delta\mathrm{T}^2 & 0 & 0 \\ 0 & 0 & \frac{\Delta\mathrm{T}^4}{4} & \frac{\Delta\mathrm{T}^3}{2} \\ 0 & 0 & \frac{\Delta\mathrm{T}^3}{2} & \Delta\mathrm{T}^2 \end{bmatrix}, \tag{4.51}$$

where the α value on the model side has been constant and chosen as “5”, but the υ value on the measurement side has been changeable between 3 and 25, and step size has been 2 for each Monte-Carlo step. Moreover, 3000 Monte-Carlo independent repetitions have been used for the experiment. After the 3000 times repetition, all processing time has been measured for each IMM structure. IMM-KF, IMM-MCKF and IMM-MCStF have given 766 *sec*, 4724 *sec* and 5121 *sec*, respectively. Although IMM-KF has been faster than the others, their RMSE results have shown real differences between them. In addition, the process time differences problem can be eliminated in a real-time system.

In Figure 4.9, noised reference, which means reference with Non-Gaussian distribution,

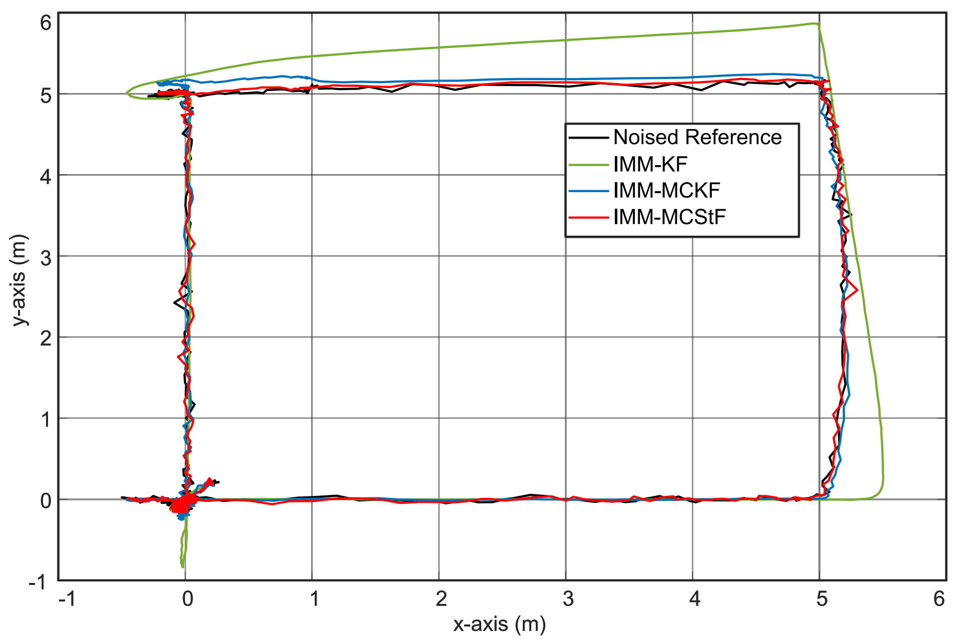

Figure 4.9: Reference trajectory tracking for each IMM structure.

IMM-KF, IMM-MCKF and IMM-MCStF, has been coloured as black, green, blue and red, respectively. The reference square reference has tracked the UAV in this figure. It shows that IMM-KF faces out of reference lines. It is expected due to discrepancies in the covariance matrices. Considering the IMM-KF and IMM-MCKF, process covariance matrices depend on Gaussian distribution. On the other hand, IMM-MCStF has worked under non-Gaussian noises.

The changes in the x and y coordinates (shown as a trajectory in Figure 4.9) are presented separately in Figure 4.10 and Figure 4.11. Unlike the compared IMM algorithms, IMM-MCStF has the best result considering the 3000 times Monte-Carlo running and reference trajectory.

From the trajectory given in Figure 4.10, it is evident that the IMM-KF algorithm shows an overshoot of the estimated trajectory in some parts, while the IMM-MCKF shows steadily converging state estimation error. These overshoots and steady-state errors affect the overall performance results.

Like the x-axis results, the same performance results can be observed on the y-axis; this has been shown in Figure 4.11. Moreover, the results for a specific area demonstrate that the IMM-MCStF tracks the reference line perfectly compared with other algorithms.

Both position results and axes velocities have been investigated and graphed. The system

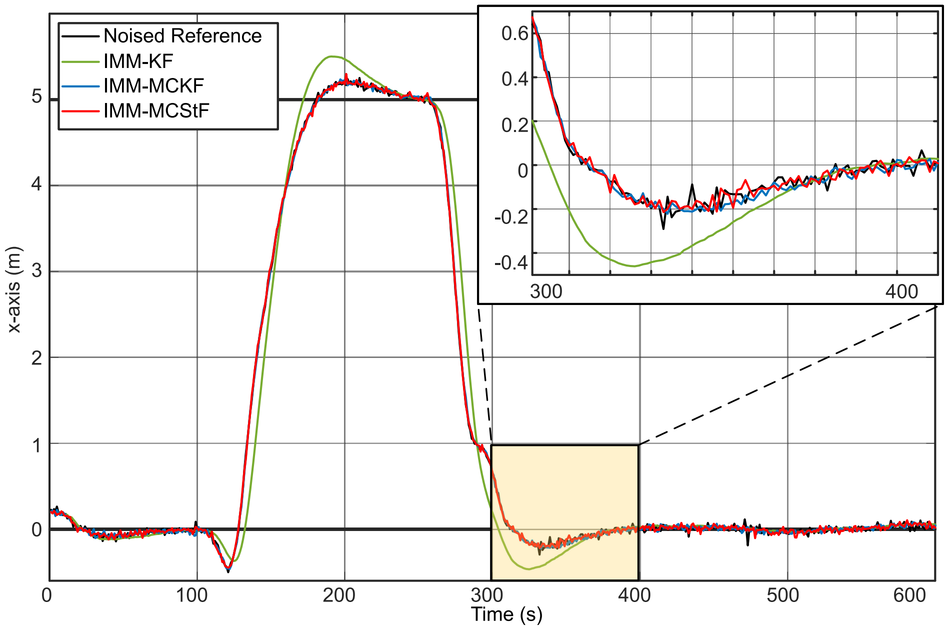

Figure 4.10: Results of compared IMM methods on the x-axis.

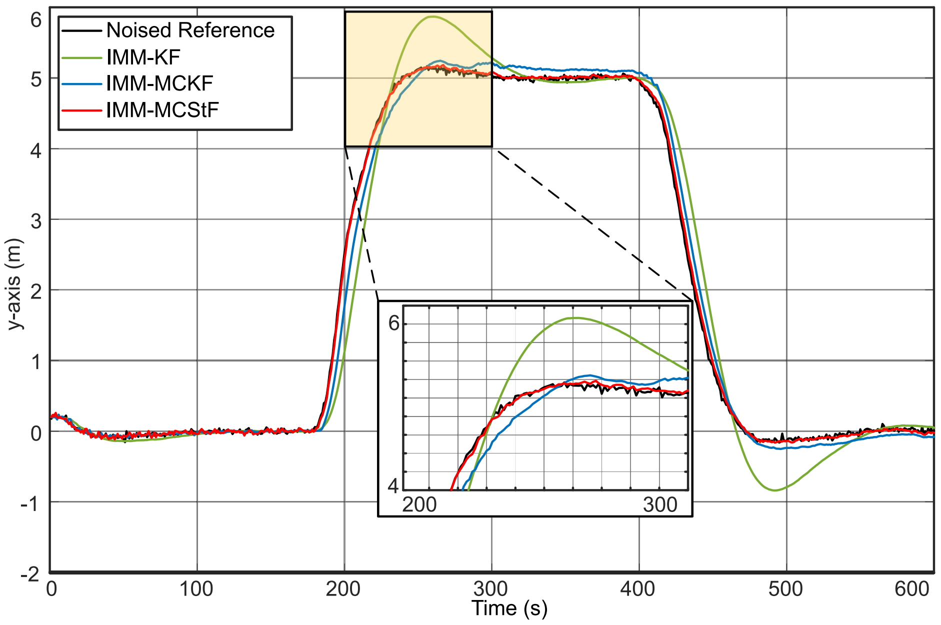

Figure 4.11: Results of compared IMM methods on the y-axis.

velocity responses can be seen in Figs. 4.12 and 4.13. The reference velocity has aggressive responses between $100-200$ and $250-350$ iterations. At these points, noise or distribution effects can be seen. Although IMM-KF has not had a good result, IMM-MCKF and IMM-MCStF have eliminated the distribution effect more than IMM-KF.

Although 3000 independent Monte Carlo repetition has been done, the number of samples

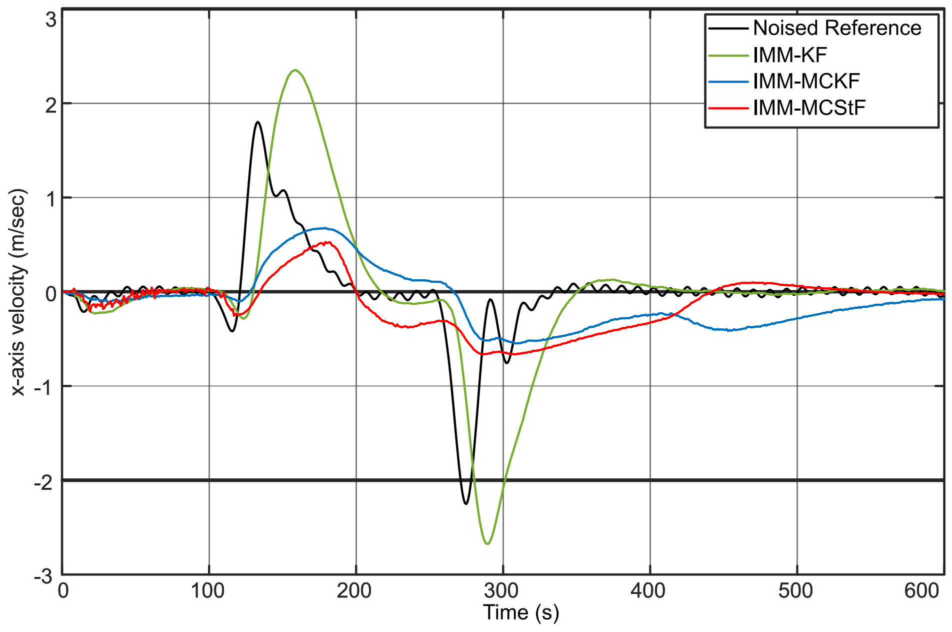

Figure 4.12: Results of compared IMM methods x-axis velocities.

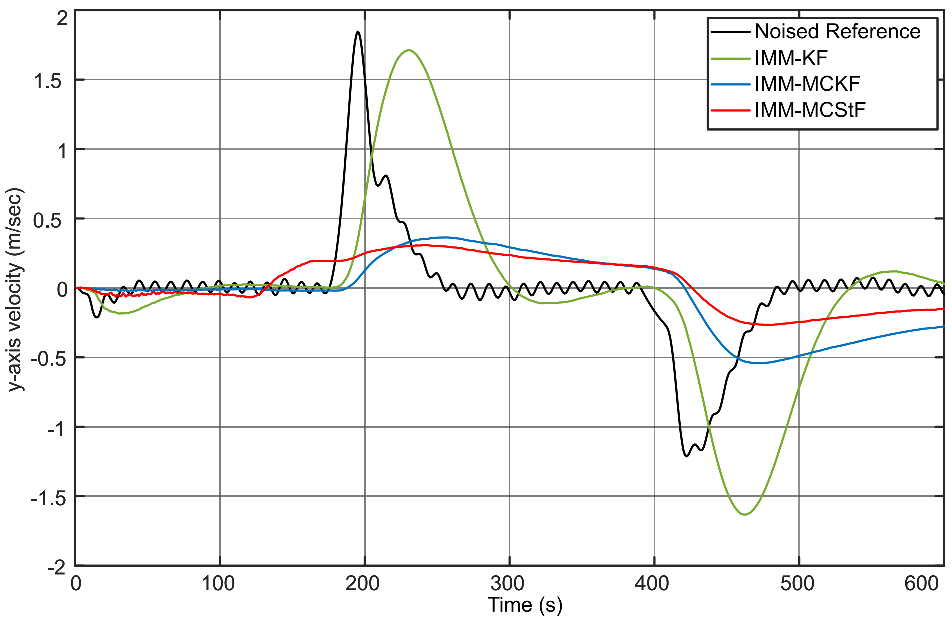

Figure 4.13: Results of compared IMM methods y-axis velocities.

$j = 1 - 100$ results have been shown in Figs. 4.14 and 4.15. Each sample shows the average RMSE value of the proposed and the compared methods. With these figures, it can be seen that the errors are repeated after 100 independent iterations.

With these 100 RMSE values, MCKF and MCStF have almost the same error values on the x-axis. On the contrary, these filters have 10 times better results than IMM-KF. Moreover,

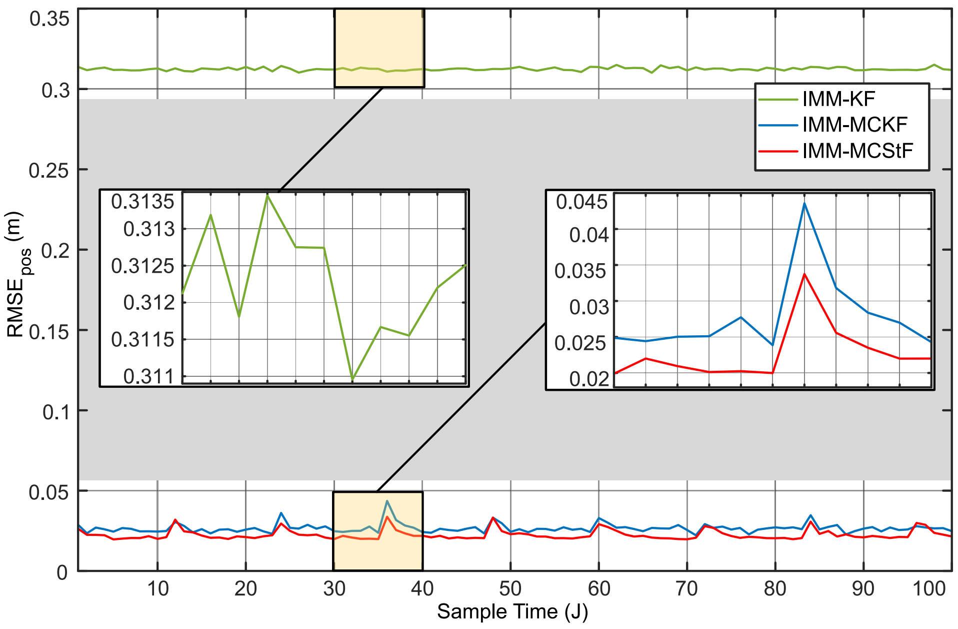

Figure 4.14: RMSE Results of compared IMM methods on the x-axis.

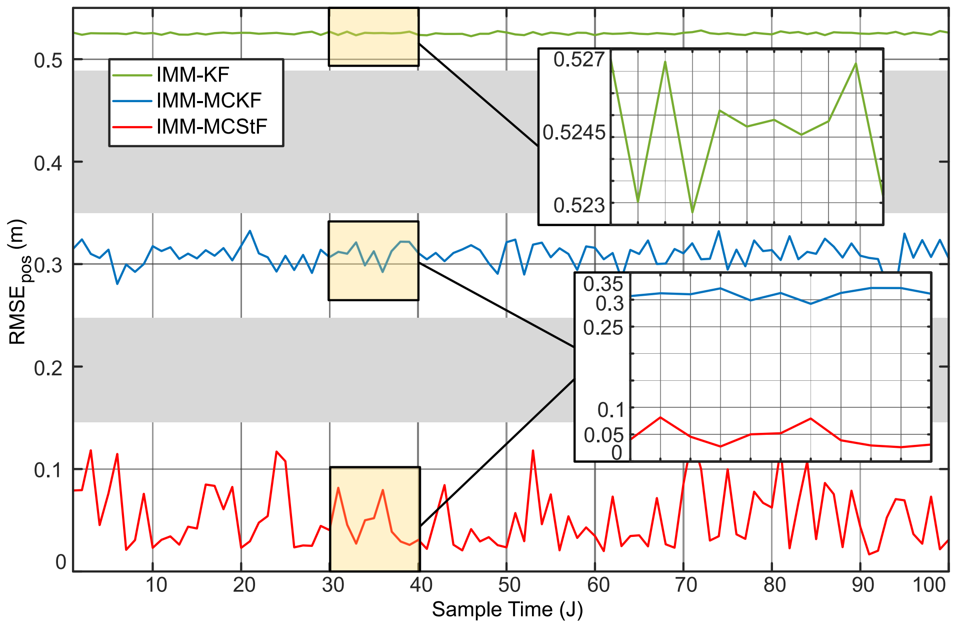

Figure 4.15: RMSE Results of compared IMM methods on the y-axis.

in Figure 4.15, it can be seen that IMM-MCStF has 6 and 10 times better results on the y-axis than IMM-MCKF and IMM-KF, respectively. The UAV movement dynamics directly affect the axes. The UAV has long distances on the axis, and the heading angle direction is on the y-axis. Therefore, part of the errors can be eliminated by the controller of the UAV. That is why the UAVs have almost the same RMSE values on the x-axis.

Table 4.3: RMSE Values of tested IMM methods.

RMSE Values	IMM-KF	IMM-MCKF	IMM-MCStF
x-axis results (m)	0.2413	**0.0221**	0.0230
υ_x-results (m/s)	0.6321	0.4566	**0.4408**
y-axis results (m)	0.4108	0.1841	**0.0388**
υ_y-results (m/s)	0.5687	0.4245	**0.3834**

Instead of 100 independent iterations, 3000 independent Monte-Carlo RMSE results have been shown in Table 4.3. Unlike RMSE figures, RMSE values on velocities have been shown in the table. The results show that the proposed method has significantly good results when compared to methods except for the results on the y-axis. On the x-axis, the proposed and the compared methods have almost the same response. It has shown the same performance values on both axes compared with RMSE figures and the RMSE table.

4.6.2 Real-time Application Results

In this subsection, the real-time experimental results of the implemented IMM filters are demonstrated on Crazyflie 2.0 commercial mini UAV [129] as seen in Figure 4.16. In the simulation studies, the performances are compared through an offline analysis with collected position data. On the other hand, real-time experiments are conducted online in the real-time. In this context, we have constructed an experiment environment as shown in Figure 5.20. On top of the experiment environment, an Intel real-sense camera system [117] is located that is used for detecting and tracking the positions of the Crazyflie 2.0 during the experiments in real-time. The frames are collected via the camera system and then processed by a vision-based localisation structure to obtain the positions in 2-D coordinate (x, y).

The experiment area is constrained by a protection net so that the mini UAV does not fly out, and also we limit the mini UAV's flying area. Intel real-sense camera, which provides accurate position information with low noise, has been used in the experiments. Therefore, we inject other noise sources into the real-world experiment area to decrease camera accuracy, assuming they have non-Gaussian distribution. Testing the performance of IMM-MCStF has been modelled for Non-Gaussian or Gaussian distribution, such as changing environment temperature, different initial levels of the battery, wind sources, and other unexpected reasons. In [28], [124], the dynamic model of the Crazyflie 2.0 is given afterwards; state estimation,

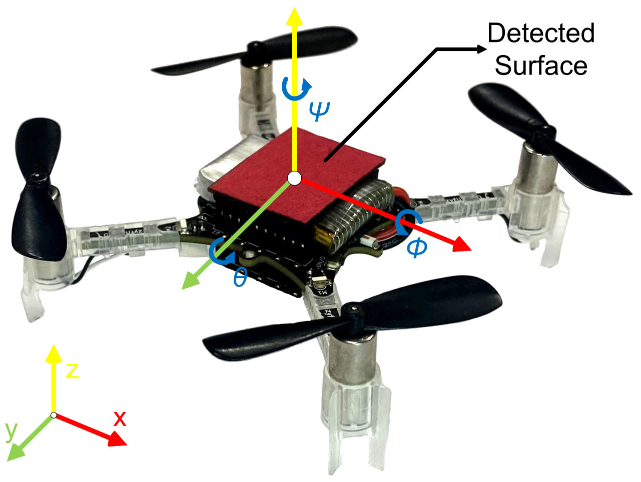

Figure 4.16: Crazyflie 2.0 coordinate frame.

attitude, altitude controllers and position controllers are explained in detail, which are the same control structures used in this chapter.

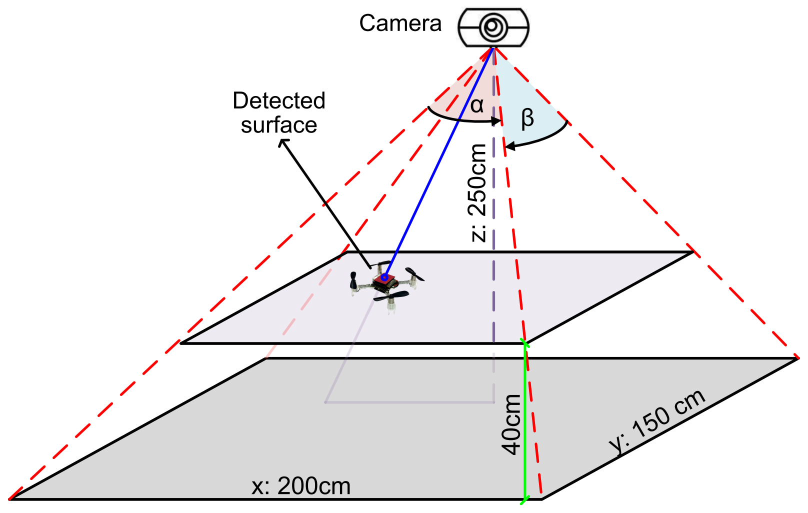

Figure 4.17: Global camera location.

Remark 1: Intel real sense has significant accuracy in getting the position of the desired object. For this reason, non-Gaussian noise with $[3-25]$ DoF has been injected as monotonic increasing and decreasing.

In Figure 4.18, Crazyflie 2.0's reference trajectory and the resulting performance trajectories of the implemented IMM filters under a noisy environment are presented. Here, point 0 represents the initial point of Crazyflie 2.0 just after take-off, while point 4 represents the final point of Crazyflie 2.0. One needs to remark that Crazyflie 2.0 started from the same point on the ground for all experiments conducted with all implemented IMM filters, and point 0 in Figure 4.18 shows the first positions in hovering just after taking

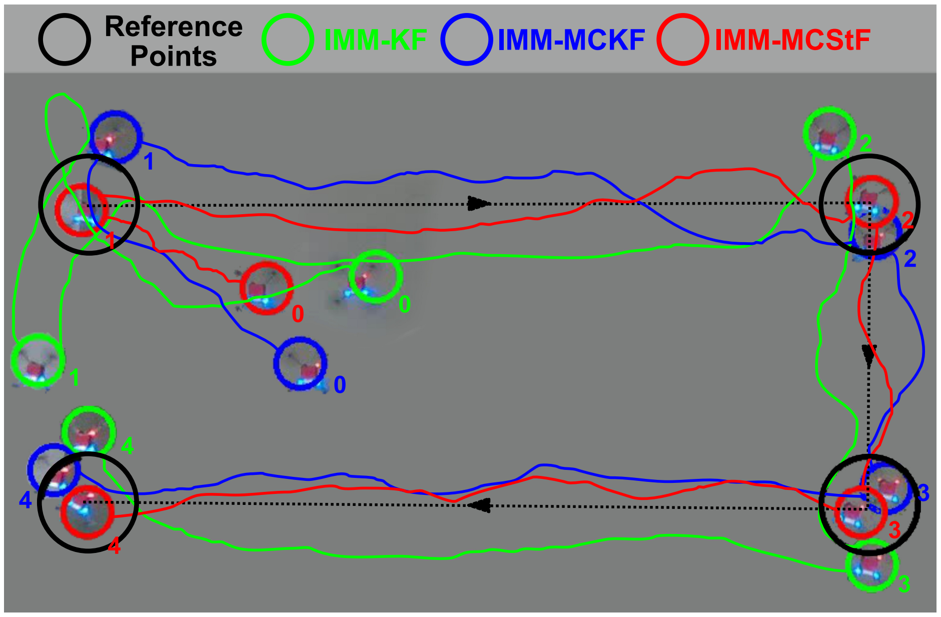

Figure 4.18: Crazyflie 2.0 reference trajectory results on the real-time experiments area.

Remark 2: It should be noted that initial values of covariance, measurement, and process noise matrices have been set as the same for all the implemented IMM filters, and they are selected as suggested in [124].

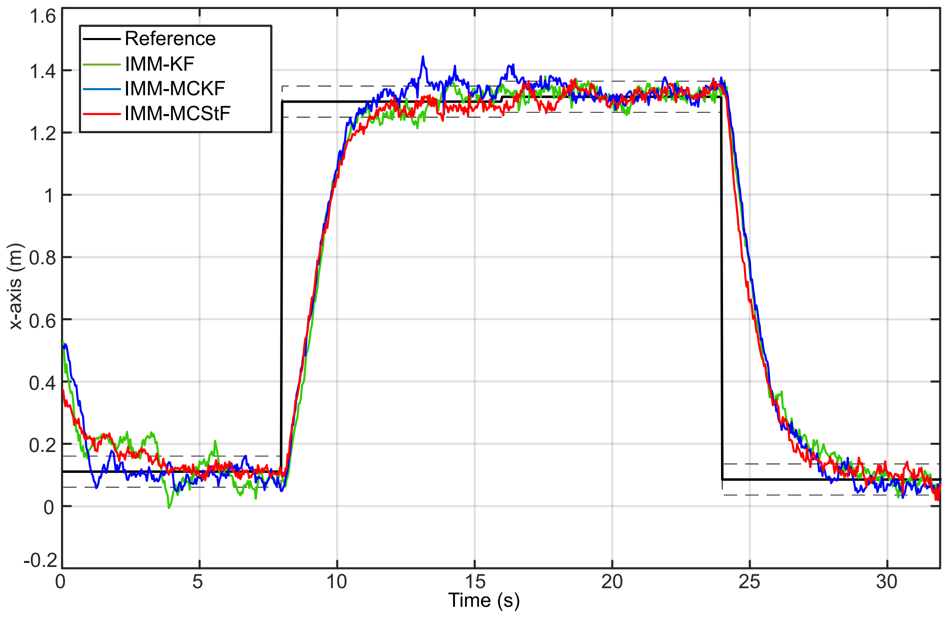

Figure 4.19: Crazyflie 2.0 reference trajectory results on the x-axis.

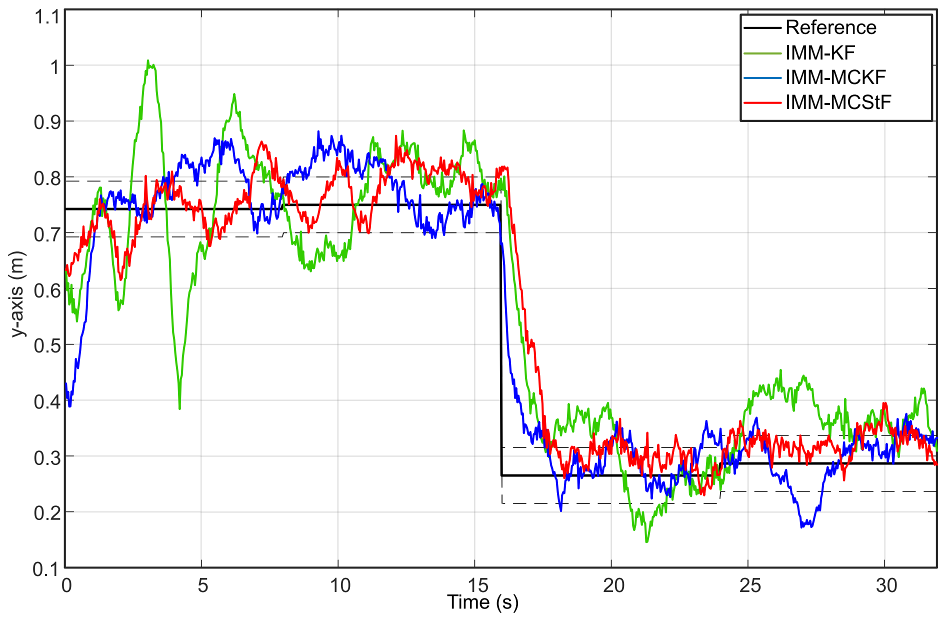

Figure 4.20: Crazyflie 2.0 reference trajectory results on the y-axis.

The real-time performances of the implemented IMM filters for the reference trajectories in $x-y$ are shown in Figure 4.19, 4.20. In order to handle the stochastic nature of the environment, these trajectories are the resulting mean values of the 10 independent experiments for each IMM filter. The resulting performance measures are tabulated in Table 4.4. As it can be clearly seen from Figure 4.19 and Figure 4.20, IMM-MCStF has a better system performance than its counterparts for both axes. It can be seen from Figure 4.19 that the resulting responses of IMM filters have close performance results, yet the IMM-MCStF has managed to increase the performance of its IMM-MCKF counterpart. On the other hand, it can be clearly seen from Figure 4.20 that the response of the IMM-MCKF oscillates around the reference between $[0-15]$ sec while IMM-MCStF resulted in an almost perfect response on the y-axis. For example, a point with a magnitude of $9cm$ while the FPID one resulted in an almost perfect response in the y-axis. Besides, the coupling effect can be observed from Figure 4.20 at 24^{th} sec. During this reference change on the x-axis at 24^{th} sec, it can be concluded that the IMM-MCStF has a more robust performance to handle this effect.

Table 4.4: RMSE Values of tested IMM methods on the real-time UAV system

RMSE Values	IMM-KF	IMM-MCKF	IMM-MCStF
x-axis results (m)	0.1368	0.1306	**0.1222**
y-axis results (m)	0.0929	0.0560	**0.0342**

From Table 4.4, it can be firstly commented that the IMM-MCStF outperformed the IMM-KF and IMM-MCKF significantly as it improved the RMSE value by an amount of 40% for the y-axis when compared to its IMM-MCKF counterpart. On the other hand, the RMSE value is by an amount of 10% for the x-axis compared to its IMM-KF counterpart.

4.7 Conclusion

This chapter, the first part, has shown how to estimate UAV states with an Interacting Multiple Model (IMM)-based Maximum Correntropy Kalman Filter (MCKF) approach. Unlike conventional methods such as IMM-Kalman filter (KF) or single-model approaches, IMM-MCKF has been tested under a non-uniform distribution. The Student's-T distribution has been used. Simulation results showed that IMM-MCKF results are better than the IMM-KF in terms of root mean square error (RMSE). Furthermore, although the computational time is a challenge for the IMM-MCKF, the IMM-KF is faster. In that experiment, the computational time is averaged over 3000 repeated Monte Carlo runs, and it is 3.3522 seconds for IMM-MCKF and 0.7667 seconds for IMM-KF.

After that, a novel estimation method called Interacting Multiple Model Maximum Correntropy Student's T Filter (MCStF) has been proposed in this study. The proposed method has been compared with the IMM-based KF and MCKF. For the comparison, non-Gaussian-based distributions have been applied to the reference trajectory. Moreover, the selected non-Gaussian distribution, Student's T-based, had variable degrees of freedom. It changed between 3 and 25. Therefore, the noises are between extreme and intermediate level heavy-tailed. In addition, the simulation and real-time experiment environments have been applied to analyse the MCStF response. Both environments gave the same error results. That means simulation and real-time results proved each other.

Future work will focus on implementing the MCStF on multiple UAVs whose system covariance matrices are unknown. Also, MCStF will be combined with local path planning methods, such as the velocity obstacle and reciprocal velocity methods. After these implementations, the MCStF will be investigated for collision avoidance.

Chapter 5

Methods for Collision Avoidance of Unmanned Aerial Vehicles

5.1 Obstacle Avoidance Methods

Considering points of goal, the goal is defined as a global reference, which can be seen in Figure 5.1. In ref [64], researchers design real-time motion planning; this plan includes manoeuvre and trajectory. With these 2 different plans, agents calculate their controller for steering angle and local path position. Considering formation and obstacle avoidance, we will focus on local motion planning in ref [64], [130], [131]. When the swarm system faces obstacles like different drones, the system may go to instability area. Therefore, we proposed to determine and estimate a local reference trajectory solution. In Figure 5.1, differences between global and local reference trajectories. The global system tries to find optimal roads, although the local system tries to avoid obstacles etc.

This section presents the velocity obstacle (VO) [76] avoidance and Potential Field (PF) [68] methods for obstacle avoidance in multi-agent systems (MAS). These methods are briefly described in the next section, and their performance is demonstrated over an example [132].

5.1.1 Potential Field Method

The potential field method is a well-known global path planning method able to determine easily distances to multiple agents. Assuming that the agent has a point mass and moves in

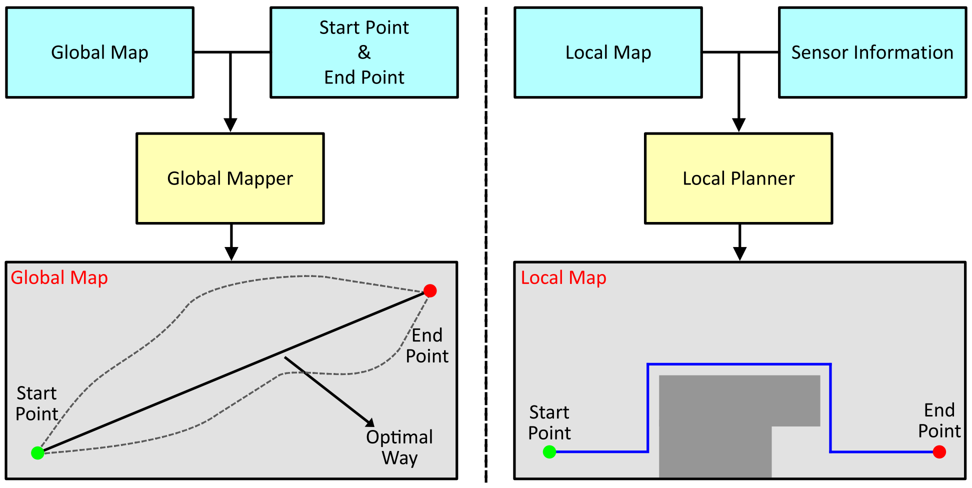

Figure 5.1: Difference between global and local map.

a two-dimensional (2-D) workspace, the following notation is adopted: $\mathbf{q} = [x\ , y]^T$, where x and y are the agent coordinates, and $(.)^T$ is the transpose operation.

Different cost functions have been suggested in the literature [133], [134]. Considering a 2-D Cartesian system, the agent and obstacle can be visualised as positioned at a certain distance from each other, as shown in Figure 5.2. The attractive potential function $\mathbf{U}_{att}(\mathbf{q})$ that is most widely used takes the form of:

$$\mathbf{U}_{att}(\mathbf{q}) = \frac{1}{2}\xi\rho^m(\mathbf{q}, \mathbf{q}_{goal}), \tag{5.1}$$

where a positive scaling factor ξ is defined and the distance $\rho(\mathbf{q}, \mathbf{q}_{goal})$ between the current agent's location $\rho(\mathbf{q}$ and its goal location $\mathbf{q}_{goal})$ is shown. Here the m coefficient can have values 1 or 2. When $m = 1$, the attractive potential function is conic in shape; otherwise, it is parabolic.

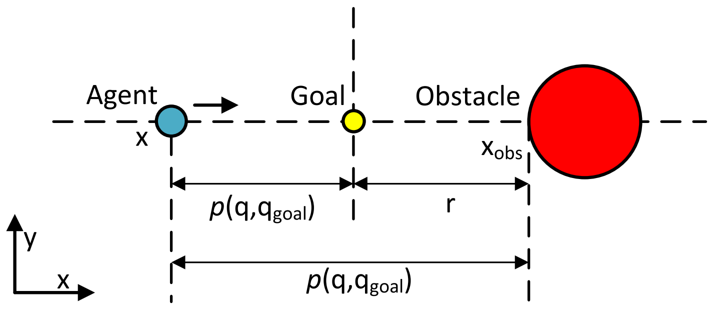

Figure 5.2: Positions of the agent, goal and obstacle in a 2-D space.

The associated attractive force $\mathbf{F}_{att}(\mathbf{q})$ is then given by the attractive potential's negative gradient $-\nabla U_{att}(\mathbf{q})$ as follows

$$\mathbf{F}_{att}(\mathbf{q}) = -\nabla U_{att}(\mathbf{q}) = \xi(\mathbf{q}_{goal} - \mathbf{q}). \tag{5.2}$$

One of the widely used potential driving functions takes the form of:

$$U_{rep}(\mathbf{q}) = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{\rho(\mathbf{q}, \mathbf{q}_{obs})} - \frac{1}{\rho_0} \right)^2 & , \text{if } \rho(\mathbf{q}, \mathbf{q}_{obs}) \leq \rho_0 \\ 0, & , \text{if } \rho(\mathbf{q}, \mathbf{q}_{obs}) > \rho_0 \end{cases} \tag{5.3}$$

where ρ_0 is the threshold distance.

When η is a positive scaling factor, $\rho(\mathbf{q}, \mathbf{q}_{obs})$ represents the minimum distance from agent $\mathbf{q}$ to the obstacle and $\mathbf{q}_{obs}$ represents the point on it.

$$\mathbf{F}_{rep}(\mathbf{q}) = -\nabla U_{rep}(\mathbf{q}). \tag{5.4}$$

The sum of the attractive and repulsive forces

$$\mathbf{F}_{total} = \mathbf{F}_{att} + \mathbf{F}_{rep}. \tag{5.5}$$

forms the combined force that is applied to the agent.

5.1.2 Obstacle Dependant Gaussian Potential Field Method

This subsection presents the Obstacle Dependant Gaussian method (ODG-PF), based on a Vector Field Histogram. In real-time path planning, this method relies on processing the sensor data because, generally, other methods use global environment data. Considered with respect to the Potential Field method, it is a notable global path planning algorithm in terms of easy calculation and accurate results.

For generating the potential field, the width ω_k, the distance d_k to the obstacle and the calculated angle Φ_k are given by equation (5.6) for this method.

$$\Phi_k = 2arctan\left(\frac{\omega_k}{2d_k}\right). \tag{5.6}$$

The distance d_k between obstacles and agents is linked with the function A_k as follows:

$$A_k = \frac{1}{2}e^{d_k}. \tag{5.7}$$

Equation (5.8) shows the repulsive function f_k and direction ϑ to avoid each obstacle. Generally, the variable ϑ is between 0 and 359 degrees, and it changes one degree for each step:

$$f_k(\vartheta) = A_k e^{-\frac{2(\theta_k - \vartheta)^2}{\Phi_k^2}}. \tag{5.8}$$

The repulsive function $f_{rep}(\vartheta)$ is determined when several obstacles are identified in a scan by basically summarising the repulsion region of each obstacle.

$$f_{rep}(\vartheta) = \sum_{k-1}^{n} f_k(\vartheta) = \sum_{k-1}^{n} A_k e^{-\frac{2(\theta_k - \vartheta)^2}{\Phi_k^2}}. \tag{5.9}$$

The attractive field $f_{att}(\vartheta)$ is based on the θ_{target} angle, referring to the orientation of the target and γ is a coefficient of the attractive function:

$$f_{att}(\vartheta) = \gamma \left|\theta_{target} - \vartheta\right|. \tag{5.10}$$

Lastly, the total potential field function is summed with the repulsive and attractive functions.

$$f_{total}(\vartheta) = f_{rep}(\vartheta) + f_{att}(\vartheta). \tag{5.11}$$

The ODG-PF in [75] shows an insensitive feature for the γ coefficient in an attractive field compared to the k_{att} coefficient in general PF, which provides greater flexibility of the method to the environment. Another benefit of ODG-PF is that it requires a reasonable amount of data, which makes it easy to deploy in real-time.

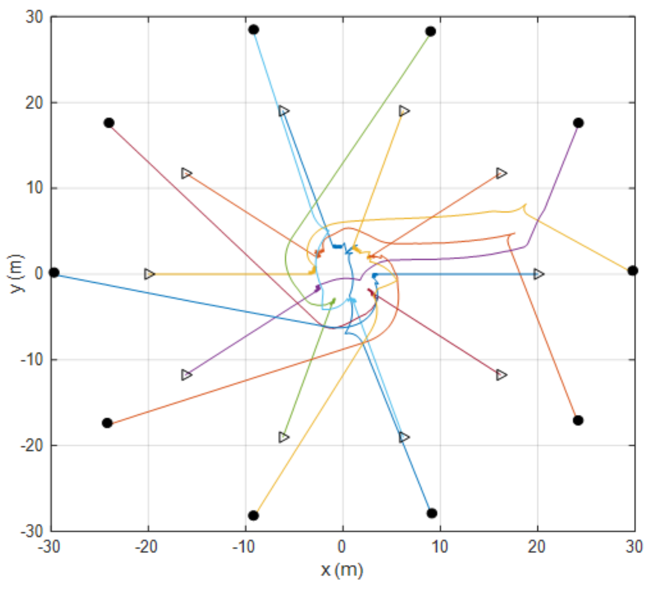

Figure 5.3: ODG-PF trajectory tracking result.

Figure 5.3 shows agents' trajectories with the optimal coefficient value of ODG-PF (γ). This coefficient value is determined by using empirical testing. Each colour shows the trajectory line of each agent.

5.1.3 Velocity Obstacle Method

The velocity obstacle (VO) avoidance method [76] considers a set of all velocities of an agent which, at some point in time, could result in a collision with another agent if the other agent retains its current velocity. If the agent chooses a velocity inside the velocity obstacle avoidance region, then the two agents will inevitably collide. If it chooses a velocity outside the velocity obstacle avoidance region, it is assumed that such a collision will not happen. Compared to other local collision-free trajectory planning methods, the VO method could help avoid moving obstacles since the possible future obstacle trajectory has been considered during the collision avoidance process.

The VO method [76] has a critical concept which is the Collision Cone (CC), a conical area in the velocity space. In Figure 5.4, it has been agents A, obstacles B and centred position of objects are p_A and p_B, respectively.

The variables r_A and r_B represent the radii of the two objects A and B, respectively. The cut line from the central point p_A to the circle based on p_B and $r_c = r_A + r_B$ radius can be

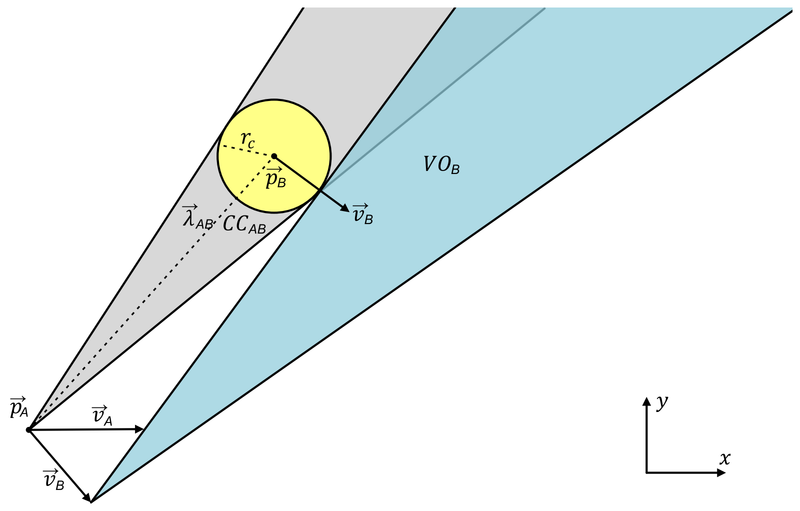

Figure 5.4: Velocity obstacle avoidance methods.

described as two boundaries λ_{AB}. The VO_{AB} area is a collision cone. So, it is necessary to apply only a small translation through Minkowski sum($\oplus$) to produce VO_{AB}

$$VO_{AB} = CC_{AB} \oplus v_B. \tag{5.12}$$

The cone CC_{AB}, the set of colliding relative velocities between A and B:

$$CC_{AB} = \left\{ \upsilon_{AB} | \lambda_{AB} \cap B \neq \emptyset \right\} \tag{5.13}$$

where υ_{AB} represents the relative speed of A with respect to B, $\upsilon_{AB} = \upsilon_A - \upsilon_B$, and λ_{AB} is the line of υ_{AB}. In Eq. 5.12, Minkowski Vector Sum operator has been explained in Algorithm 4:

The velocity group VO_{AB} is described with (5.12). It is possible to select the collision-free velocity in time k from the velocities that satisfy the condition: $v \notin VO_{AB}$

$$V_i^{optimal} = argmin_{v \notin VO_{AB}} |v - v_A^{pref}|. \tag{5.14}$$

For the VO technique, it is also important to find an optimal velocity beyond the VO area. Currently, only the derivation to the preferred velocity is included in the optimum velocity selection. It can be expressed as follows

$$VO_{AB} = CC_{AB} \oplus \frac{(v_A + v_B)}{2}. \tag{5.15}$$

Algorithm 4 Minkowski Vector Sum Algorithm

```
$i \leftarrow 1$; $j \leftarrow 1$
$\upsilon_A(n+1) \leftarrow \upsilon_A(1)$; $\upsilon_A(n+2) \leftarrow \upsilon_A(2)$
$\upsilon_B(n+1) \leftarrow \upsilon_B(1)$; $\upsilon_B(n+2) \leftarrow \upsilon_B(2)$

while $i \neq n+1$ and $j \neq m+1$ do
add $\upsilon_A(i) + \upsilon_B(j)$ as a vertex to $P \oplus R$
    if $\angle(\upsilon_A(i), \upsilon_A(i+1)) > \angle(\upsilon_B(j), \upsilon_B(j+1))$ then
        $i \leftarrow i+1$
    else if $\angle(\upsilon_A(i), \upsilon_A(i+1)) < \angle(\upsilon_B(j), \upsilon_B(j+1))$ then
        $j \leftarrow j+1$
    else
        $i \leftarrow i+1$ and $j \leftarrow j+1$
```

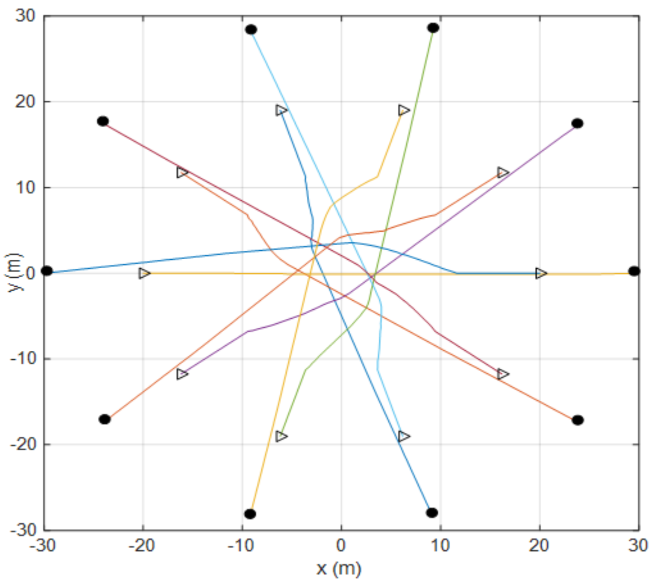

Figure 5.5: VO trajectory tracking result.

Next, the velocity obstacle avoidance coefficient values have been determined based on training, and these results are shown in Figure 5.5.

5.1.4 Hybrid Potential Field Method based on Velocity Obstacle avoidance and Potential Field

A Hybrid Potential Field method (HPFM), a novel collision avoidance method, is proposed. This method combines ideas from the "Potential Field and "Velocity Obstacle" avoidance methods.

The method considers a potential collision event that is based on a geometric space, while VO_{AB} is a term based on the velocity space. Equation (5.16) is used for the transformation of space velocity information into geometric space. For each agent, the speed is translated to relative displacements.

$$x_v = (v_{agt} - v_{obs}) \times t_s(v_{agt} \in VO_{AB}), \tag{5.16}$$

in VO_{AB} that are in the CC_{AB}. Then, the velocity danger level l_v is represented as:

$$l_v = \frac{d_v}{|x_v|}. \tag{5.17}$$

The repulsive function, which is related to dynamic obstacles, is described as a function. This function

$$f_{rep}^{VO}(\vartheta, v) = e^{-l_v}, \tag{5.18}$$

contains velocity(v) and angle (ϑ) values.

The total potential field function can be determined as below:

$$f_{total}(\vartheta, v) = f_{att}(\vartheta) + \sum_{s=1}^{S} f_{rep}(\vartheta) + \sum_{d=1}^{D} f_{rep}^{VO}(\vartheta, v). \tag{5.19}$$

Due to the effects of dynamics constraints, the total potential field function and velocity filtering vectors are determined with the equation:

$$f_{rech}(\vartheta, v) = \begin{cases} v < v_{max} \\ \sqrt{\left(\dfrac{dv}{dt} - v\dfrac{d\vartheta}{dt}\right)^2 + \left(2v\vartheta - v\dfrac{d\vartheta}{dt}\right)^2} < a_{max}. \end{cases} \tag{5.20}$$

When adding this inequality to the total function, five different coefficient values can be obtained. These coefficients are for attractive functions (γ, β), statics (ζ, δ) on repulsive functions and dynamics (α) on repulsive functions

$$f_{total}(\vartheta, v) = (\gamma \left|\theta_{target} - \vartheta\right| + \beta \left|v - v_{pref}\right|)$$

$$+ \left(\zeta \sum_{s=1}^{S} e^{-\dfrac{2(\theta_s - \vartheta)}{\delta \Phi_k^2}} \right) + \left(a \sum_{d=1}^{D} e^{-l_v^d} \right). \tag{5.21}$$

The five coefficient values of HPFM have been optimised to find the optimal trajectory and velocity for each agent. A genetic algorithm (GA) has been used in the evaluation structure for optimisation of HPFM [135]. The following equation

$$f_{fit}(\gamma, \zeta, \delta, \alpha, \beta) = L_{traj} + 10E_{coli} + \frac{\Delta\theta}{20} + L_{sep}, \tag{5.22}$$

defines the fitness function in a way to reduce the collision possibilities and provides an optimised trajectory.

In the HPFM, a two-dimensional potential field where the magnitude and direction of the velocity can be determined separately has been created. When the force HPFM is applied, the total cost is given as follows

$$T_{cost} = \frac{v_{max}}{v_{interval}} \times \frac{360^\circ}{\Theta_{interval}}. \tag{5.23}$$

and it includes a product of the considered velocity and angles.

When the speed or angle is small, the HPF optimisation method performs well. However, the computational cost increases when the resolution of the speed and angle increases. Since significant computational costs are not suitable for real-time applications, the smart optimisation algorithm can be a good solution to lower the computation cost T_{cost}.

Empirical methods have some difficulties in determining all coefficients. In order to use the fitness function via the GA, optimal coefficients have been determined. Figure 5.6 shows the obtained result after the training phase. The next section presents testing results from the VO, ODG-PF and HPFM using the same coefficients.

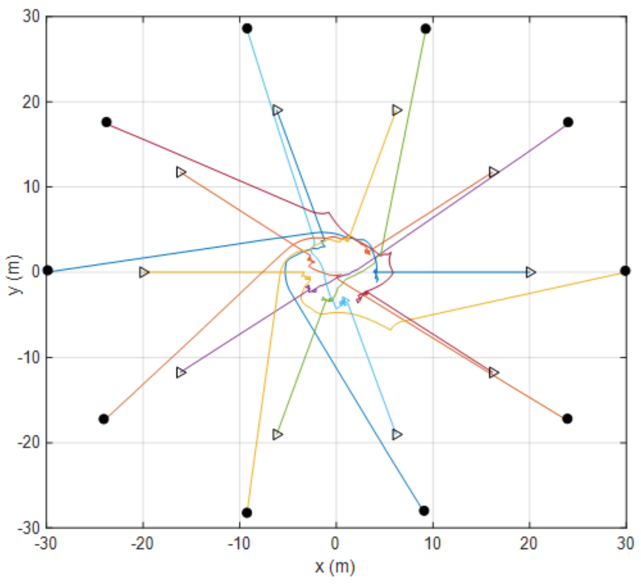

Figure 5.6: HPFM trajectory tracking result.

5.2 Simulation Results

In this study, the experiment has been carried out via MATLAB / Simulink. A point-based mathematical model of the agent has been used in the testing stage. Moreover, all determined obstacle avoidance methods have been tested with static and dynamic obstacles. The testing parameters are given in Table 5.1.

After 1000 independent Monte Carlo runs, several performance criteria have been evaluated to determine the accuracy of the algorithms.

Figure 5.7 shows the trajectory results of VO methods. Multiple collisions have been observed in the conventional VO methods. However, the VO method has some advantages in terms of choosing the optimal trajectory with the minimum task time. For testing, all coefficients have been fitted with the GA for HPFM and found with empirical methods for ODG-PF and VO.

Figure 5.8 shows results with the ODG-PF, which has the best results when compared to VO methods with respect to the minimum collision and computational time. Figure 5.8 shows that all agents achieve the goal although they have chosen long paths to move.

Figure 5.9 shows the optimal HPFM trajectory, found based on the optimal coefficients calculated using a GA. The HPFM is a combination of the VO and PF, with an improved cost function. Thanks to this combination, it has better collision avoidance results than the VO

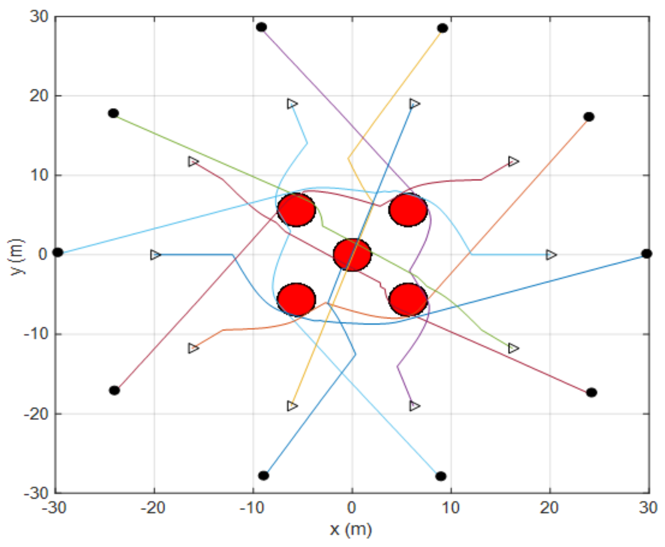

Figure 5.7: VO trajectory tracking result under static obstacles.

Table 5.1: Test Results of VO, ODG-PF and HPFM

Parameters	VO	ODG-PF	HPFM
Computation Time	0.017 sec	**0.008 s**	0.04 s
Collision	23.46 times	10.61 times	**5.75** times
Length of Trajectory	54.58 m	**47.83 m**	62.38 m
Task Time Cost	34.33 s	**26.40 s**	36.22 s

and better length trajectory than the ODG-PF.

Results for trajectories obtained with all methods are given in Table 5.1. Each value is determined from the mean value of the 1000 runs. Considering the computational time, the ODG-PF algorithm is faster than the others. Also, the trajectory length and task time cost of ODG-PF are better than those of the other algorithms. However, the collision avoidance results of HPFM are the best. After that, we can rank ODG-PF and VO, respectively, as good results. In this context, HPFM is more reliable than the others. This shows the potential of the HPFM algorithm to be part of MAS or swarm systems.

Figure 5.10 gives a performance map of the compared algorithms with respect to the easiness of use, task performance, trajectory smoothness and reliability, respectively. This performance map shows the importance of the considered criteria for MAS path planning. Different methods can be used for different tasks for MAS, depending on the mission requirements.

The ODG-PF method, which was designed for a static scenario, is obviously the best one with respect to simplicity and task performance aspects. However, in MAS, the level of

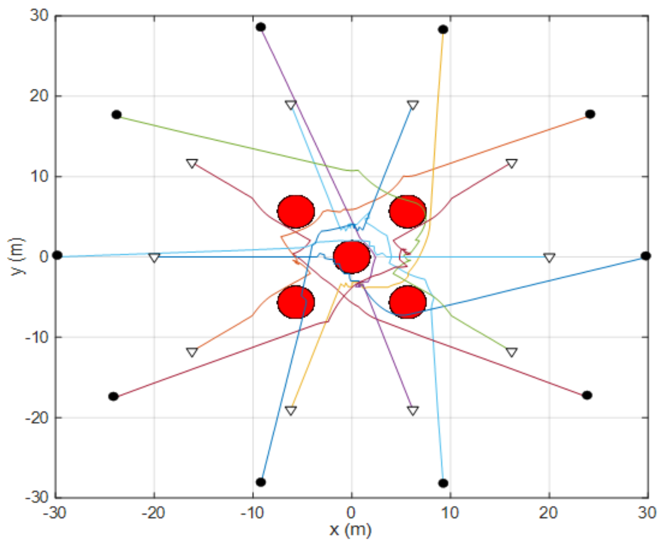

Figure 5.8: ODG-PF trajectory tracking result under static obstacles

reliability is one of the most critical aspects. With respect to reliability, the HPFM gives the best result in comparison to the other methods. Meanwhile, both HPFM and ODG-PF produce trajectories that have almost the same level of smoothness.

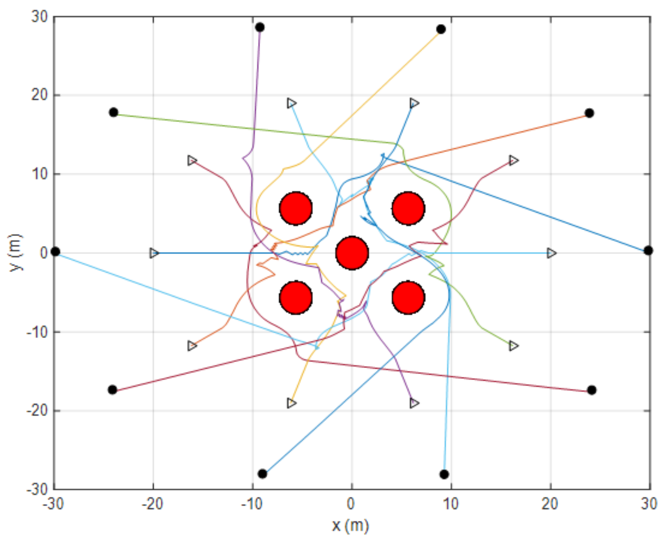

Figure 5.9: HPFM trajectory tracking result under static obstacles.

5.3 Geometric-based Velocity Obstacle Avoidance Methods

This section gives brief preliminary information about the three commonly used velocity obstacle avoidance methods, namely the velocity obstacle (VO), reciprocal velocity (RVO), and hybrid reciprocal velocity obstacle (HRVO) methods. In Figure 5.11, the behaviour of the commonly used velocity obstacle methods against the obstacle can be seen.

5.3.1 The Classical Velocity Obstacle Avoidance Methods

The conventional velocity obstacle avoidance methods have been based on geometric considerations, and they are directly related to Collision Cone (CC). In [136], it has been detailed and formulated deeply. In this project, we focused on the UAV's 2-D local plane (XY). In Figure 5.11, the geometrically relative position ($\lambda_{\alpha\beta}$) of the obstacle, defined radius r_c of obstacle, and the velocity vector $\mathbf{v}_\beta$ are used to generate the collision cone for obstacle β as $CC_{\alpha\beta}$. Moreover, $\mathbf{p}_\beta$ is represented as a planar cross-section centre.

The classical velocity obstacle avoidance formula

$$\mathrm{VO}_{\alpha\beta} = \mathrm{CC}_{\alpha\beta} \oplus \mathbf{v}_\beta, \tag{5.24}$$

is written by using the Minkowski sum ($\oplus$).

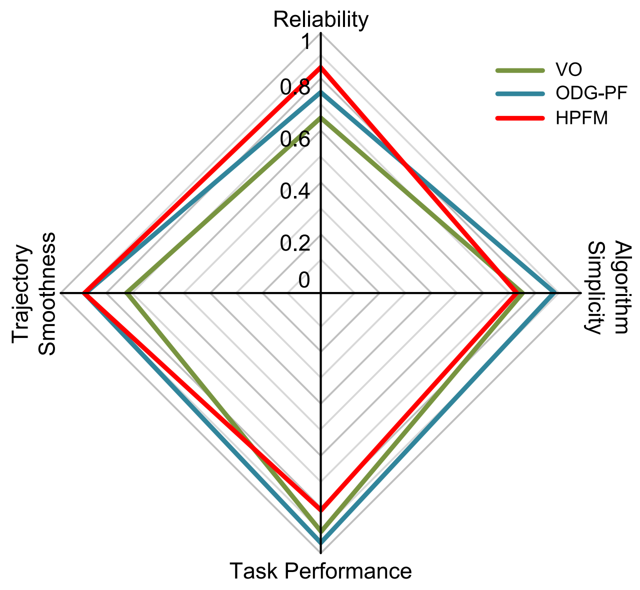

Figure 5.10: Performance Map.

The numerous number (n) of velocities $\mathrm{VO}_{1:n}$ is taken into account while considering multiple obstacles. Therefore, agent velocities are regarded as acceptable if $\mathbf{v}_{\alpha,k+1} \notin \mathrm{VO}_k = \cup_{\beta=1}^{n} \mathrm{VO}_{\beta,k}$ in [136]. In the existence of obstacles $\mathrm{VO}_{\beta=1:n}$ for current time $k = t_k$, velocities respecting this restriction define a collision-free track for each agent α.

5.3.2 The Reciprocal Velocity Obstacle Method

Considering the classical VO method, RVO creates smoother avoidance trajectories in each iteration. In [78], it makes an effort to take the reciprocal velocity of the subsequent decision-making agent β into account; also, in Figure 5.11, the RVO method is shown.

$$\mathbf{v}_{\alpha,k+1} \notin CC_{\alpha\beta} \oplus (\mathbf{v}_{\alpha,k} + \mathbf{v}_{\beta,k})/2. \tag{5.25}$$

Unlike VO, the created VO for each agent is defined as an enhanced apex by the mean of the agent and object velocities in RVO. In Equation (5.25), the RVO formula is written. With the help of this idea, the agent may successfully follow the safer trajectory $\mathbf{v}_{\alpha,k+1}$ in line with $\mathbf{v}_\beta$ when compared to the traditional VO.

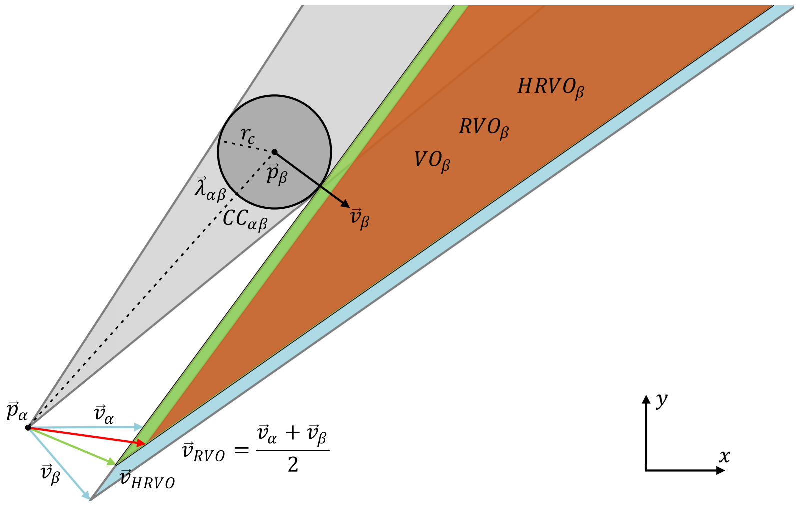

Figure 5.11: The geometric-based velocity obstacle avoidance methods.

5.3.3 The Hybrid Reciprocal Velocity Obstacle Method

By enhancing the VO and RVO areas, a solution to the VO problem has been presented to eliminate the reasons for reciprocal dance. In order to evaluate various behaviours based on the relative velocity of the obstacle $\mathbf{v}_\beta$, the HRVO, as illustrated in Figure 5.11, modifies the apex of the HRVO. Since the origin lines of VO$_\beta$ and RVO$_\beta$ are co-linear, the agent should resolve a trajectory $\mathbf{v}_{\alpha,k+1}$ to pass the obstacle on the left if it is going to the right, and vice versa. Inaction results in the phenomenon of reciprocal dancing.

Theoretically, the approach cannot ensure the development of smooth avoidance trajectories, despite evidence to the contrary in [137]. In the illustration shown in Figure 5.11, the directional bias is created by changing the RVO$_\beta$'s apex to be the point where the RVO$_\beta$'s leading edge and VO$_\beta$'s trailing edge connect, for example, HRVO$_\beta$ $= CC_{\alpha\beta} \oplus \mathbf{v}_{\text{HRVO}}$. The constraint set that is subsequently placed on agent α at current time $k = t_k$ is thus expressed as $\mathbf{v}_{\alpha,k+1} \notin \text{HRVO}_k = \bigcup_{\beta=1}^{n} \text{HRVO}_{\alpha,k}$ in [137], [138]. The VO and A_β for obstacles O_β:

$$\mathbf{v}_{\alpha,k} \notin \text{HRVO}_k = \bigcup_{A_\beta=1}^{n} \text{HRVO}_{A_\beta} \cup \bigcup_{O_\beta=1}^{n} \text{VO}_{O_\beta}. \tag{5.26}$$

The RVO and HRVO are often only required for computing inter-agent avoidance trajectories. Instead, the union of the reciprocal variants (RVO or HRVO) for the surrounding agents can be used to represent the global VO set for agent α.

5.4 Fuzzy Interacting Multiple Velocity obstacle avoidance method

In this study, we have designed the novel FIMVO for a bounded experimental environment as shown in Figs. 5.21, 5.22, 5.23, and 5.24. The FIMVO mechanism is constructed by choosing its inputs as the absolute relative distances of the UAVs and obstacles with respect to each other, as illustrated in Figure 5.12.

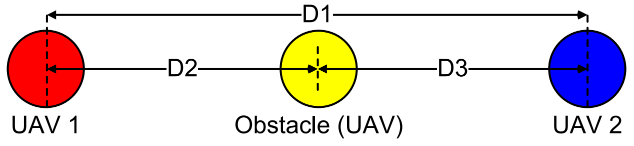

Figure 5.12: Distance-based decision making.

These absolute relative distances are the distances between UAVs and the obstacle are defined as in the following:

$$\begin{aligned}
\widetilde{D1} &= \sqrt{(x_1 - x_2)^2} + \sqrt{(y_1 - y_2)^2}, \\
\widetilde{D2} &= \sqrt{(x_1 - x_o)^2} + \sqrt{(y_1 - y_o)^2}, \\
\widetilde{D3} &= \sqrt{(x_2 - x_o)^2} + \sqrt{(y_2 - y_o)^2},
\end{aligned} \tag{5.27}$$

where, $(x_1, y_1), (x_2, y_2)$ and (x_o, y_o) represent the locations of the UAV1, UAV2 and obstacle in 2-D plane, respectively. The universe of discourse of the antecedents MFs of FIMVO are defined in between [0, 1], thus the inputs ($\widetilde{D1}, \widetilde{D2}$ and $\widetilde{D3}$) of the FIMVO are scaled into the universe of discourse as follows:

$$\begin{aligned}
D1 &= \widetilde{D1} * K_s^1, \\
D2 &= \widetilde{D2} * K_s^2, \\
D3 &= \widetilde{D3} * K_s^2,
\end{aligned} \tag{5.28}$$

where, K_s^1 and K_s^2 are the scaling factors for the relative distances D1, D2 and D3. In summary, the FIMVO uses the absolute relative distances of the UAVs and obstacles with respect to each other to decide the best obstacle avoidance method among the employed approaches in this work. The rule base of the FIMVO has been designed through expert knowledge and by taking into account the boundaries and limitations of the experimental environment. The rule structures of the mechanism are constructed as follows:

$$R_i : \quad \textbf{If } (D1 \; is \; A_i) \textbf{ and } (D2 \; is \; B_i) \textbf{ and } (D3 \; is \; C_i) \\ \textbf{Then } \quad \alpha_1 = o_i^1, \; \alpha_2 = o_i^2 \tag{5.29}$$

where $i : 1, ..., M$ is the rule number index, and M is the total rule number of the FIMVO. Here, A_i, B_i and C_i represent the antecedent MFs while o_i^1 and o_i^2 are the parameters of the consequent MFs.

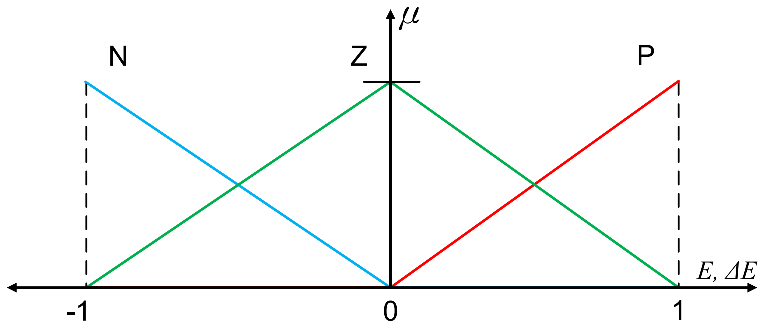

Figure 5.13: Fuzzy Membership Functions.

As seen in Figure 5.13, the antecedent MFs are specified using symmetrical triangular MFs that are uniformly distributed. Moreover, the linguistic fuzzy variables are defined as Small (S), Medium (M), and Big (B). The consequent parts of the rules are specified using crisp singletons that correspond to the three velocity obstacle approaches that are used (VO: 1, RVO: 2, HRVO: 3). The implemented FIMVO uses and employs the product implication and the centre of sets defuzzification method [109], [139], [140].

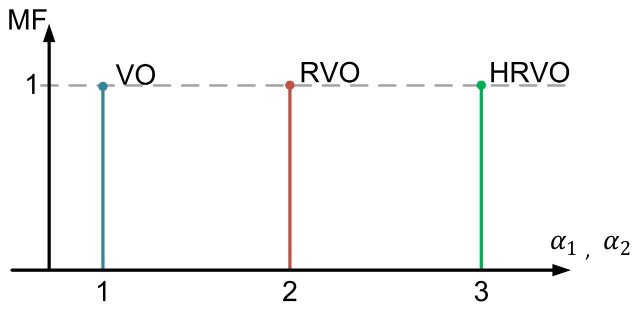

Figure 5.14: Consequents of the FIMVO.

Remark-1: It should be noted that all antecedent MFs for the D1, D2, and D3 selected same as shown in Figure 5.13. In a similar way, the consequents for the VO weighting coefficients of FIMVO (α_1,α_2) are chosen the same as illustrated in Figure 5.14.

The outputs of FIMVO (α_1, α_2) represent the fuzzy interacting value for each velocity obstacle avoidance method (VO, RVO, and HRVO). Moreover, $v_{(.)}$ denotes the output velocities of fuzzy-based VO methods. Thus, we proposed an algorithm, given in Algorithm 5, that combines implemented velocity obstacle avoidance methods which can be defined as multi-model velocity obstacle avoidance methods. In other words, the combining mechanism acts like a type of filter by weighing the implemented velocity obstacle avoidance methods in a factor of $\alpha_i, i \in 1, 2$.

Algorithm 5 Algorithm for combiner mechanism

```
if $\alpha > 2$ then
    $\overrightarrow{v}_{\text{FIMVO}} = (3-\alpha)\overrightarrow{v}_{\text{RVO}} + (\alpha-2)\overrightarrow{v}_{\text{HRVO}}$
else if $\alpha < 2$ then
    $\overrightarrow{v}_{\text{FIMVO}} = (2-\alpha)\overrightarrow{v}_{\text{VO}} + (\alpha-1)\overrightarrow{v}_{\text{RVO}}$
else
    $\overrightarrow{v}_{\text{FIMVO}} = \overrightarrow{v}_{\text{RVO}}$
end if
     =0
```

Remark-2: It should be noted that the same algorithm is used for both agents/UAVs.

5.5 Simulation and Real-Time Experimental Results

Simulation tests and real-time experiments have been done. Both simulation and real-time experiments are done on 2D Cartesian coordinates. For this reason, in real-time experiments, the altitude distance of the UAV has been set, and it is defined as $50cm$. Moreover, the heading angle (ψ) of the UAV has been also set $\psi = 0°$. The UAV mathematical model and specification of the UAV have been explained deeply. For simulation experiments, used agent/UAV model has been defined and tested. The parameters of the proposed and compared approaches have been selected with the same values and used in the simulation and real-time systems. For both experiments, the proposed and compared obstacle avoidance approaches have been performed in a Python 3.9 environment on a laptop computer running a 64-bit Windows 11 operating system with a 2.60 GHz Intel i7-10750H CPU processor. Moreover, The sampling times are the same for both real-time and simulation because of the latency in communication between the UAV and the computer.

5.5.1 Simulation Experiments

In this part, the robot model and one of the results for each velocity obstacle avoidance approach have been represented. After that, 1000 times independent repeated performance results, in terms of collision counts and task performance time, have been tabulated. A simple holonomic robot kinematic model has been used for each agent/UAV instead of using a complex UAV model as a transition model for the state estimation and the controller. The simple holonomic kinematic model for each agent is represented as:

$$\begin{bmatrix} \mathrm{x}_t \\ \mathrm{y}_t \end{bmatrix} = \begin{bmatrix} \mathrm{x}_{t-1} \\ \mathrm{y}_{t-1} \end{bmatrix} + \begin{bmatrix} \mathrm{v(x)}_t \Delta \mathrm{t} \\ \mathrm{v(y)}_t \Delta \mathrm{t} \end{bmatrix}, \tag{5.30}$$

in Equation (5.30), x_t and y_t are current positions of the agent; also x_{t-1} and y_{t-1} show the previous positions. The velocities of the agent are $\mathrm{v(x)}_t$ and $\mathrm{v(y)}_t$, respectively. The sampling time of the UAV (Δt) has been defined as 0.05s. Moreover, in the equation above, the heading angle has not been defined because it has been set to 0 and has not changed for the whole process. Therefore, it has not been added to the kinematic model.

Remark-3: Throughout the paper, agent and UAV have the same meaning. In the simulation tests, noise and disturbances have been neglected. The green star ($\star$) in the simulation is defined as the goal points of the agents. Moreover, all simulations are performed on Python environment [113].

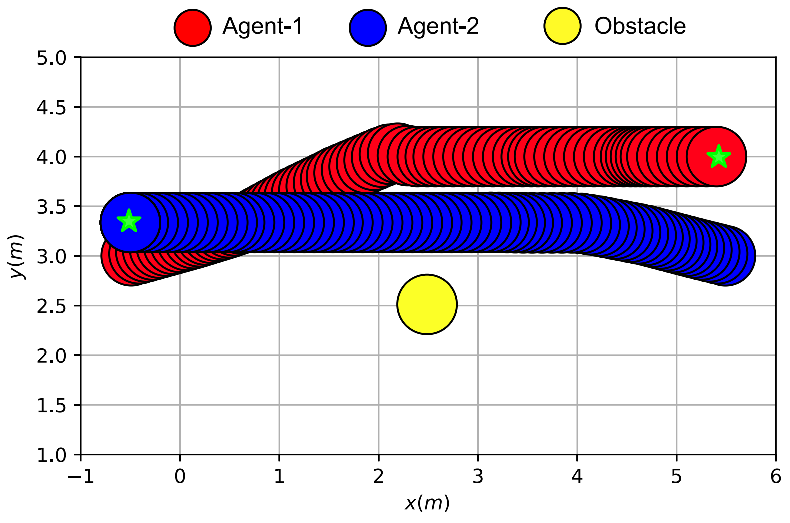

Figure 5.15: Simulation result of the VO avoidance method.

In Figure 5.15, the velocity obstacle avoidance simulation result can be seen. Total task time 7.5 sec, and agents are close to each other. It can increase the collision possibility.

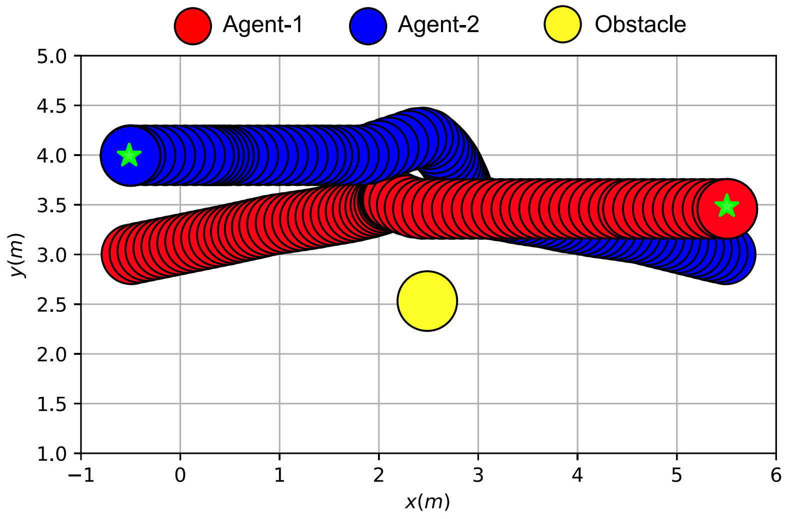

Figure 5.16: Simulation result of the RVO avoidance method.

In Figure 5.16, RVO avoidance approach results have been represented. The results of RVO show that it has done more manoeuvre than conventional one. The total task time of RVO has been measured as 9.2 sec.

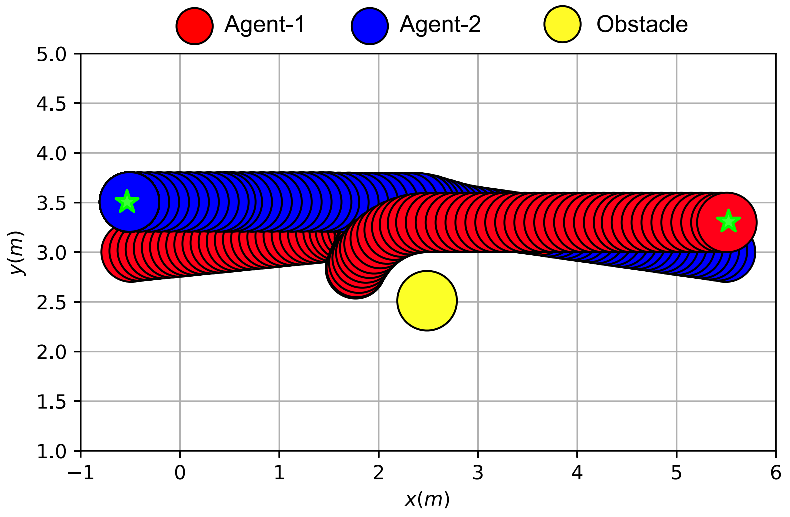

Figure 5.17: Simulation result of the HRVO avoidance method.

In Figure 5.17, each agent trajectory result has been seen. Total task time of HRVO has been found 13.6 sec. Red and Blue agents have shown manoeuvre motion when facing static and dynamic obstacles simultaneously. For this reason, computation time has increased to avoid collision cone area.

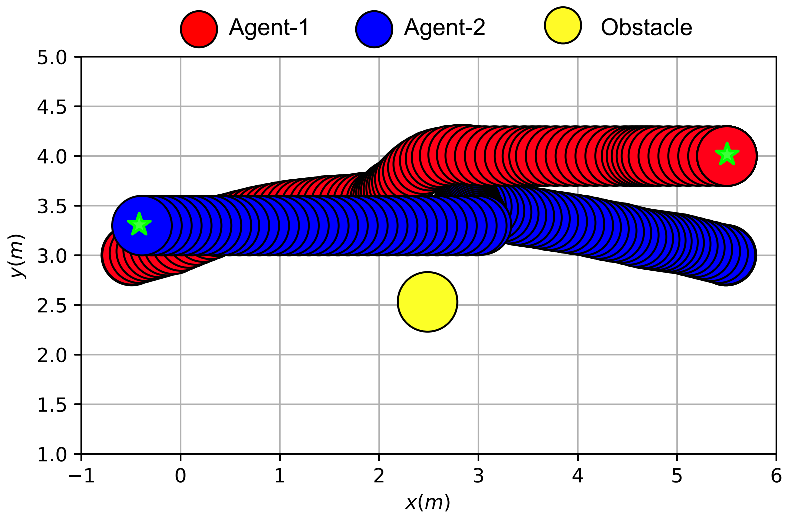

Figure 5.18: Simulation result of the FIMVO avoidance method.

Compared with VO, RVO, and HRVO, agents with FIMVO have smooth trajectory results, and the proposed novel method has 8.9 sec total time. The FIMVO method shows less manoeuvred motion than HRVO, and it is a better safe collision-free than RVO. However, the proposed method, FIMVO, is not faster than VO, but it has many advantages, which aforementioned before. The collision-free trajectory and manoeuvre reaction are shown in Figure 5.18.

Table 5.2: Total 1000 Monte-Carlo simulation test results

Method	Total collision counts	Mean of task performance time (s)
VO	692	8.58
RVO	347	13.67
HRVO	128	15.64
FIMVO	68	11.73

In Table 5.2, 1000 times independent simulation results of the proposed and compared velocity obstacle avoidance methods results are shown. It is seen that the highest number of collisions belongs to VO, but the fastest one is still VO. FIMVO, the proposed method, has significant results in terms of total collision and task performance time. The computation times gap between VO and FIMVO is negligible value, and also, FIMVO has the least number of collisions. In Appendix **C1**, the proposed and compared VO approaches have been tested on 14 agents. Moreover, the results of the trajectory for each agent have been represented as specific frames. The proposed method, FIMVO, has the least number of collisions. Moreover, the computation times gap between VO and FIMVO are negligible values; they are 0.001s and 0.0023s, respectively.

5.5.2 Real-time Experiments

For the real-time testing, three DJI Tello UAVs have been used; the UAV [141] and its coordinates can be seen in Figure 5.19. It is possible to calculate the conversion from the inertial frame to the body frame or vice versa using transformation and rotation matrices. Moreover, linear velocities and attitude angles are represented as $[\upsilon_x, \upsilon_y, \upsilon_z]$ and $[\phi, \theta, \psi]$, respectively. Each UAV is designed like a holonomic robot; therefore heading angle of each UAV (ψ) has been fixed, and it is $\psi = 0$. The detected surface is coloured for each UAV (in red, blue, and yellow, respectively).

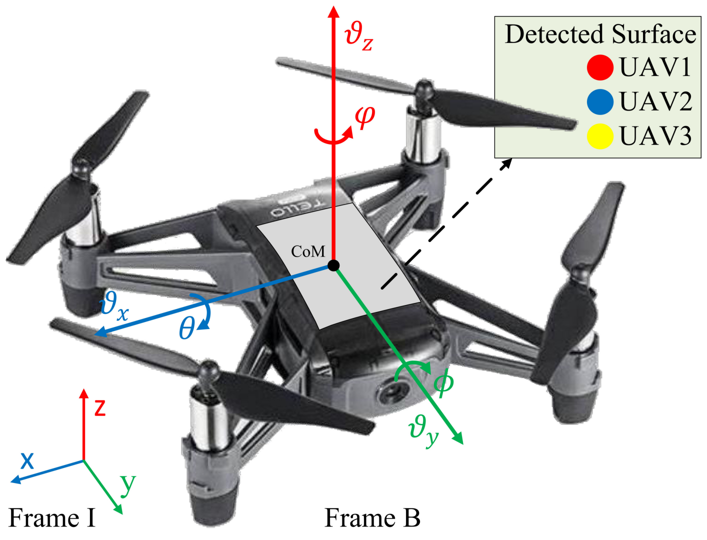

Figure 5.19: The DJI Tello UAV used in the implementation.

In Equation (5.31), a dynamical model of the UAV has been written. In this study, we directly focused on velocity obstacle avoidance methods; therefore, internal models and controllers have not been explained. Furthermore, controller inputs of the UAVs have been defined as linear velocities. In the equation, T, m, and g denote thrust from UAV rotors, the mass of the system and gravity acceleration, respectively.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{\upsilon}_x \\ \dot{\upsilon}_y \\ \dot{\upsilon}_z \end{bmatrix} = \frac{\mathrm{T}}{m} \begin{bmatrix} cos(\phi)cos(\psi)sin(\theta) + sin(\phi)sin(\psi) \\ cos(\phi)sin(\psi)sin(\theta) - cos(\psi)sin(\phi) \\ cos(\phi)cos(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}. \tag{5.31}$$

Remark-4: The dynamical model of the UAV has been designed as decoupled instead of creating highly-coupled dynamics. Due to the UAV size and weight, the coupled dynamics problems can be eliminated.

The detailed models of the DJI Tello UAV and designed controller are described in [28], [96]. Briefly, a Fuzzy PID (FPID) controller has been used, and controller parameters have been found using the "Big-Bang Big-Crunch" optimisation method. FPID and conventional controller methods have been tested and compared with each other [28], [142]. After that, it shows that FPID can guarantee stability because the selected optimisation methods have been used and tested in previous works (Chapter 4). That means when the system faces errors such as collision or out-of-reference trajectory problems, it has existed from the position estimation or velocity obstacle avoidance methods. Moreover, selected FPID controller parameters have been $K_e = 0.019$, $K_d = 0.0019$, $K_0 = 0.0001$, and $K_1 = 30$, respectively. The detailed structure of the FPID and the implementation have been explained in [28]. The sampling time for the real-time system has been chosen as 0.05 sec. The designed model and controller are the same for all UAVs. The properties of the used UAVs for experiments have been shown in Table. 5.3.

Table 5.3: DJI Tello UAV specifications

Parameters	Value
Number of UAVs	3
UAV dimension	98×92 mm
Time of flight	10 min
Maximum linear velocity	$[-8, 8]$ m/s
Boundry linear velocity	$[-12, 12]$ %
Boundry linear velocity	$[-12, 12]$ %

An Intel realsense depth camera [143] with a testing area provides the testing facilities (shown in Figure 5.20) for observing and localising the UAVs. Intel realsense depth camera has shown high precision results in terms of resolution, object detection and position detection. In this figure, the camera has been mounted on top of the floor, and some trigonometric formulation has been used. In [28], it has been explained and detailed intensely.

In the figure above, the positions of the UAVs have been defined as x_i, y_i and i is the number of the UAVs. Compared with the simulation experiments, the proposed and compared methods have faced different challenging problems, such as observation noise for positioning, affecting propeller torques between each other, and centralised-based velocity obstacle calculation. In

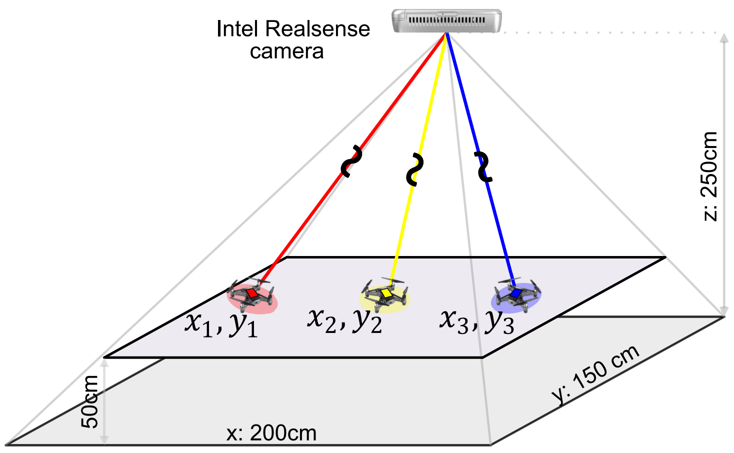

Figure 5.20: Global camera location and positions of the UAVs.

this study, position data come from the Intel realsense camera as the measurement distances of UAVs. Defined performance criteria which are trajectory smoothness, reliability, algorithm simplicity and task performance, have been directly related to velocity obstacle avoidance methods, controller and position detection. Considering performance criteria, controller and observation effects have been minimised by using previous work [28] and a high-accuracy depth camera system [143]. Before the test in real-time, position error tolerance from the depth camera has been measured, and it has been determined as ±50 mm. After the collected raw position data from UAVs, the maximum correntropy Kalman filter (MCKF) was applied to get high accuracy of the position estimation. Detailed mathematical explanations and application results have been found in [42], [124], [144].

Intel realsense depth camera has shown high precision results in terms of resolution, object detection and position detection. Non-Gaussian distribution-based noise has been merged with measurement data to increase the level of challenge. Added noise value has been represented in Equation. (5.32).

$$\omega \sim \mathcal{N}(0, 25^2). \tag{5.32}$$

The noise level has been between intermediate-level heavy-tailed and approximate Gaussian distributions. The heavy-tailed levels of distribution are variable between 50 and 100, monotonously. Detailed information about heavy-tailed and non-Gaussian distributions has been given in [49], [123], [124].

In the real-time tests, the proposed and compared velocity obstacle avoidance methods have to overcome static and dynamic obstacle problems and noised sensor measurement problems even if the measurement data is filtered. In order to make a significant performance comparison of the methods, all real-time experiments have been run ten times independently. In Figs. 5.21, 5.22, 5.23, and 5.24, red and blue UAVs have shown cooperative working, and yellow UAV has not interacted with blue and red UAVs. Other specific information is black circles. Black circles show the position of the UAVs with noises. That means the system does not know the exact position of the UAVs. Lastly, to employ FIMVO, K_s^1 and K_s^2 scaling factors have been defined as 1563 mm 750 mm, respectively.

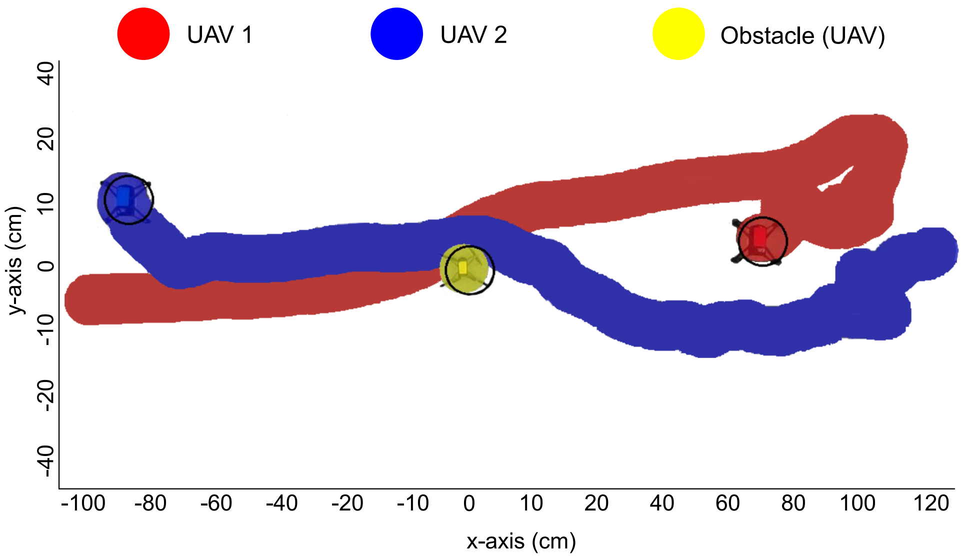

Figure 5.21: The VO results for UAVs in real-time.

In Figure 5.21, applied VO results have been seen as a real-time system. All methods have been tested ten times, and one of them has been chosen randomly. This figure shows the collision, which is expected because of noises and uncertain parameters in a real-time system. Considering the algorithm processing, this method is the fastest compared with others—total task performance time 9.15 sec.

The RVO is safer than the VO method, and the simulation results prove it. In real-time tests, this information shows the same result as the simulation. It shows safer and slower processing than VO in Figure 5.22. Total task performance time 12.35 sec.

In Figure 5.23, HRVO shows good cooperative and non-cooperative working in terms of reliability. It is the safest method when compared with traditional and proposed obstacle

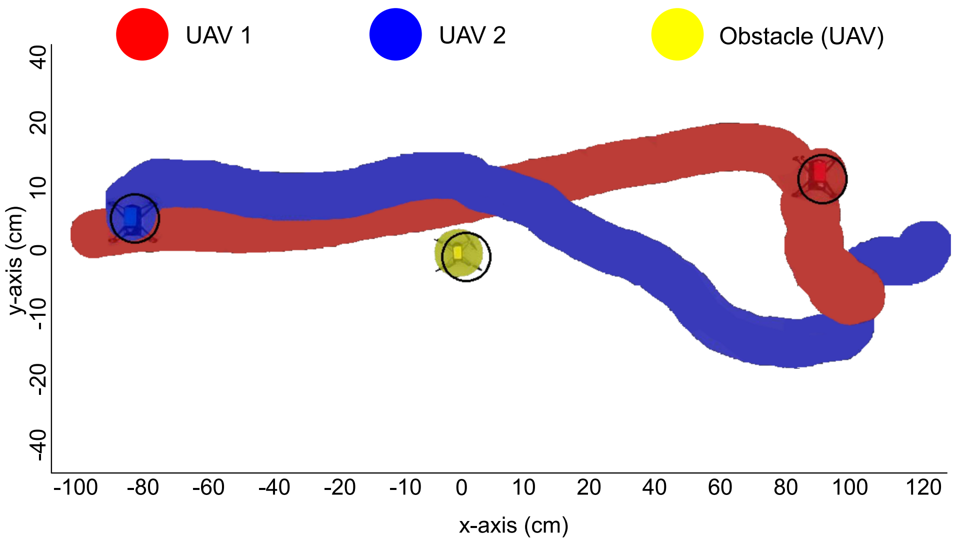

Figure 5.22: The RVO results for UAVs in real-time.

avoidance methods. However, computation time is a problem for real-time applications. Total task performance time has been measured at 15.25 sec.

In Figure 5.24, the proposed method results can be seen. It is faster than HRVO and safer than RVO. FIMVO scrutinize the capability of the system performance and reliability. At this point, it shows novelty. When compared with VO, RVO, and HRVO, FIMVO has 11.35 sec total time. In Appendix **C2**, the proposed and compared VO approaches have been tested on different scenarios. In the tested scenario, two UAVs were selected as static obstacles, and the other followed the trajectory, which was designed using the proposed and compared VO avoidance approaches.

In Table. 5.4, total test results of obstacle avoidance in multiple UAV real-time applications have been represented. In the real-time tests, the total collision counts and the mean of task performance times have been investigated. The results show that FIMVO has the best in terms of collision with obstacles; also, when the averages of their task performances are considered, it is seen that RVO and FIMVO have almost the same completion times. However, HRVO shows the slowest results even though it has considerable collision avoidance counts, such as 9 avoidance out of 10.

If we evaluate each result in a performance graph in terms of reliability, trajectory smoothness, task performance and algorithm simplicity, the results of the compared and proposed methods are seen in Figure 5.25. The performance results of the proposed and compared

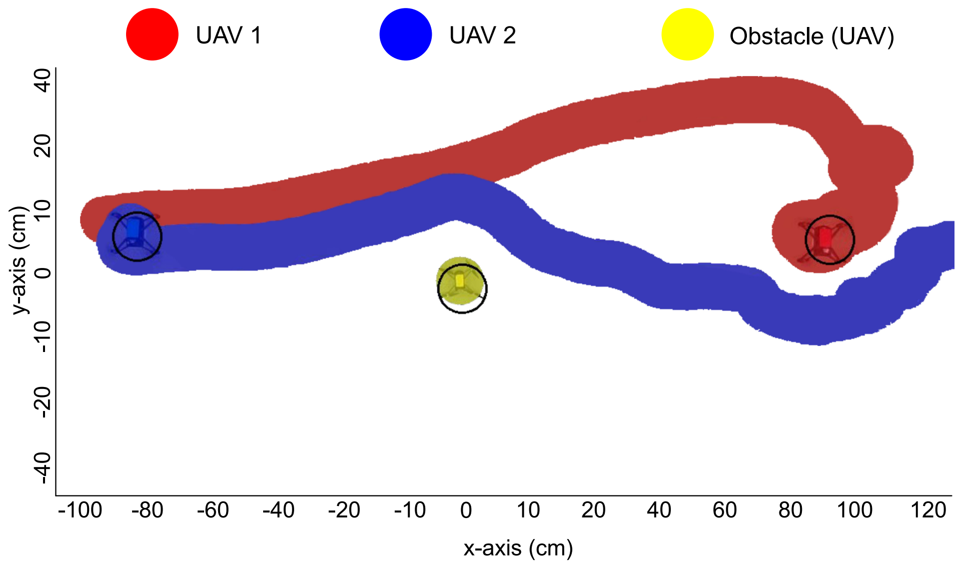

Figure 5.23: The HRVO results for UAVs in real-time.

Table 5.4: Total test results

Method	Total Collision counts (%)	Mean of task performance time (s)
VO	60	9.08
RVO	30	12.48
HRVO	10	16.01
FIMVO	0	12.33

avoidance methods come from ten times real-time testing independently. The results have been scaled between $[0, 1]$.

In the performance result figure, VO has been observed as the fastest algorithm when compared with others. However, considering the other comparison parameters, it has not acceptable for all applications. The RVO performance results show an almost average out of all of them. This does not mean that it has good performance because the HRVO results have the best reliability and task performance. The proposed method FIMVO uses multiple velocity obstacle avoidance methods with VO, RVO, and HRVO by using fuzzy logic. For this reason, it is more reliable and has a better trajectory smooth than RVO, and HRVO. Moreover, it has almost the same task performance with HRVO.

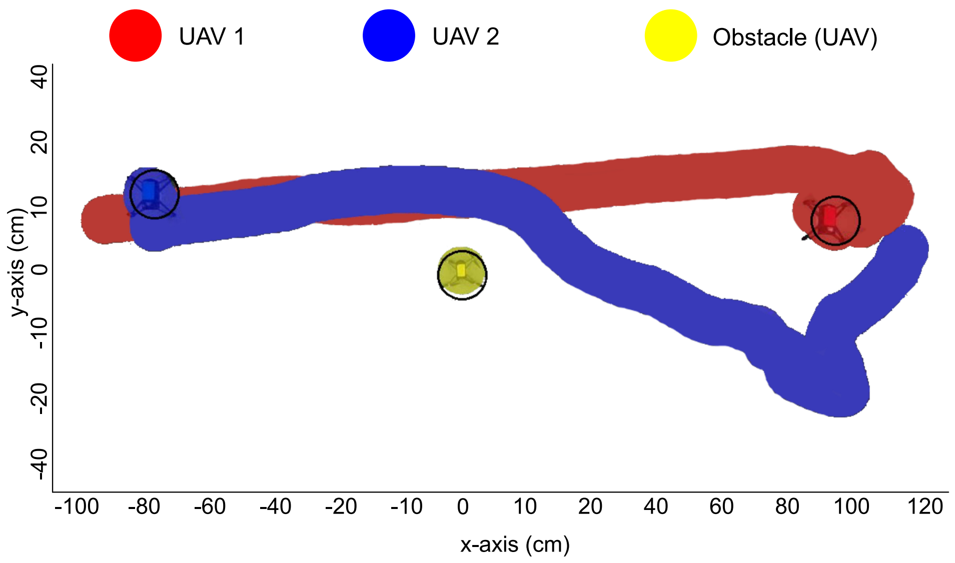

Figure 5.24: The FIMVO results for UAVs in real-time.

5.6 Conclusions

The first study is about the comparison between geometric-based velocity obstacle (VO), and histogram-based VO approaches. This comparison includes collision avoidance approaches that could be used in centralised and decentralised UAV collision avoidance systems. The trajectory smoothness and complexity of the velocity obstacle and potential field approaches were analysed. The evaluation of the approaches includes UAVs in unknown environments with static and dynamic obstacles. The Hybrid Potential Field approach (HPFM) is compared with the potential field and velocity obstacle approaches. The approach combines the PF and VO algorithms so that it can achieve the task of collision avoidance in complex environments. The HPFM used a form like the Vector Field Histogram to solve the collision avoidance task. The HPFM has been optimised using a Genetic Algorithm, and its performance improved by nearly fifty percentages compared with the non-optimised HPFM. The ODG-PF, VO and HPFM are compared over three different testing scenarios.

In a complex situation, such as a scenario with both dynamic and static obstacles and multiple agents, the HPFM behaviour is more reliable and stable. However, this comes with an increased computational cost. The average computational cost of HPFM is generally several times higher than VO and ODG-PF. The optimal Velocity for each agent must be evaluated via the fitness function for the HPFM.

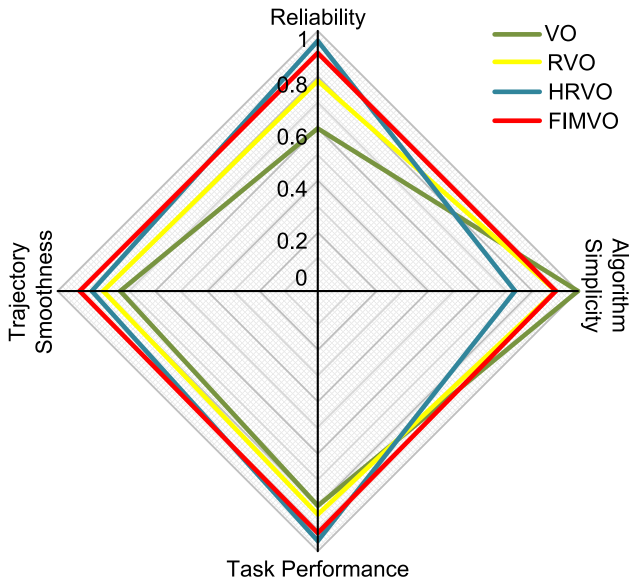

Figure 5.25: Performance result.

After the comparison of geometric and histogram-based VO approaches, fuzzy interacting multiple velocity obstacle FIMVO has been proposed as a new approach for swarm and multi-agent systems. The proposed approach has been compared with three different geometric approaches based on velocity obstacles. For comparison, simulation and real-time experiments have been tested ten times independently. For the real-time system, three UAVs were used as cooperative and non-cooperative. Moreover, these UAVs had the same controller structure and parameters. The test results were investigated in terms of trajectory smoothness, task performance, algorithm simplicity and reliability. Performance criteria showed that FIMVO had a better smooth trajectory and better algorithm simplicity than HRVO, and also, taking into account reliability, it was faster than RVO and VO.

The proposed approach will be tested with ten UAVs and different scenarios in future works. Moreover, different velocity obstacle approaches, such as optimal reciprocal velocity obstacle avoidance or generalised velocity obstacle avoidance approaches, will be combined with FIMVO. To increase the challenge in real-time, different noises will be added to the measurement part, and these noises can be extremely heavy-tailed non-Gaussian distributions. Lastly, the FIMVO approach will be investigated by different intelligent controllers, and then stability analysis will be chosen as another performance criterion for FIMVO and the compared approaches.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In brief, selected UAVs for the thesis, DJI Tello UAV and Crazyflie 2.0, have been used, and the controller, state estimation and local path planning methods for UAVs have been designed. Whole methods for the UAVs have been represented in Fig. 6.1.

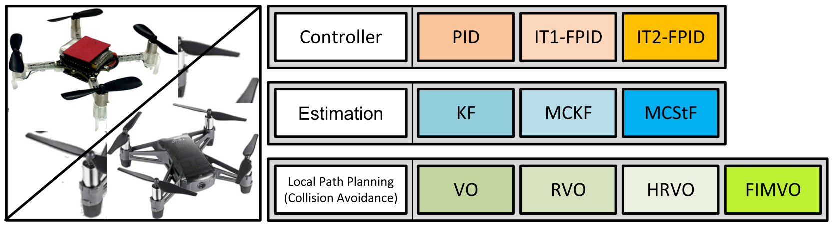

Figure 6.1: Methods for selected UAVs

The focus of the thesis revolved around addressing three key problems associated with UAVs: control, state estimation, and path planning. A comprehensive literature review was conducted for each problem domain, followed by the testing of existing methods in both simulation and real-time environments. Additionally, two novel methods, the Maximum Correntropy Student's T Filter (MCStF) and the Fuzzy Interacting Multiple Velocity Obstacle avoidance method (FIMVO), were proposed, contributing to the existing literature.

Prior to introducing the novel methods, a comparative analysis of different controllers for UAVs was conducted. The selected controllers, including Proportional-Integral-Derivative (PID), type-1 Fuzzy PID (T1-FPID), and interval type-2 Fuzzy PID (IT2-FPID), were thoroughly investigated, tested and analysed in terms of their ability to ensure stability criteria.

These controllers were also evaluated under uncertain conditions, such as the presence of a flexible cable-connected unknown payload and disturbances affecting the payload. Real-time application scenarios, such as agriculture UAV coverage path planning problems, were created to simulate these testing scenarios. The performance of each controller was compared based on total tasking time, Root-Mean-Square error (RMSE), and rising time. Despite T2-FPID demonstrating better performance and stability criteria, T1-FPID was chosen for the state estimation and geometric-approached velocity obstacle avoidance methods involving multiple UAVs. This decision was made because IT2-FPID can address sensor estimation issues and provide a smoother trajectory. Unexpected and disturbance rejection results were obtained when comparing the state estimation methods and velocity obstacle avoidance methods.

After implementing the T1-FPID controller in the UAVs, the conventional Kalman filter (KF), maximum correntropy Kalman filter (MCKF), and the state-of-the-art MCStF were utilized for state estimation. To simplify the estimation process, model reduction state transition matrices were employed in the filters instead of complex nonlinear and linear models of the UAVs. Constant velocity and constant acceleration models were selected to capture the manoeuvrability and non-manoeuvrability capabilities of the UAVs. Additionally, the interacting multiple model approach was integrated with each estimation method to enhance accuracy. Gaussian and non-Gaussian disturbances were introduced during the localization part of the experiments, with the non-Gaussian noise varying monotonously. After conducting simulations, real-time applications were performed using the proposed and compared sensor estimation methods in the UAVs. The performance of KF, MCKF, and MCStF was evaluated based on RMSE and computation time. MCStF exhibited the best RMSE results among the three methods, while KF was the fastest but had the worst RMSE performance.

The final phase of testing focused on geometric-approached velocity obstacle avoidance methods. Before delving into these methods, hybrid and potential field methods were compared with geometric-based approaches, and the simulation involved ten agents. Subsequently, geometric-based velocity obstacle avoidance methods took centre stage in the path planning aspect. The proposed method, FIMVO, was compared with conventional velocity obstacle (VO), reciprocal velocity obstacle (RVO), and hybrid reciprocal velocity obstacle (HRVO) methods in both simulation and real-time environments. Since FIMVO combines VO, RVO, and HRVO, it exhibited superior performance compared to the other methods. Performance criteria, such as reliability, smooth trajectory, task performance, and algorithm simplicity, were tabulated and graphically represented. Real-time experiments were conducted using three UAVs equipped with T1-FPID and MCKF, with Gaussian disturbances applied during the experiments.

6.2 Future Works

In future works, the authors plan to explore novel methods, namely FIMVO (which stands for Fuzzy Interacting Multiple-Model Velocity Obstacle) and MCStF (which stands for Maximum Correntropy Student's T Filter), in the context of multi-UAV systems. The scope of the research will expand to include more than three UAVs, and a nonlinear complex UAV model will be utilised for state estimation.

Specifically, the MCStF method will undergo modifications to accommodate the nonlinear dynamics of the UAVs or to be extended to nonlinear MCStF types. This adjustment is necessary because larger UAVs exhibit highly-coupled dynamics and parameters compared to the non-coupled dynamics assumed in the thesis.

Additionally, the controllers used in the UAV model will be upgraded. Model Predictive Control (MPC) and intelligent controllers, such as reinforcement learning or deep learning-based approaches, will be implemented to enhance the system's control performance.

Regarding state estimation, the thesis relied on camera-based localisation. However, the authors plan to shift towards global positioning systems (GPS) and real-time kinematic positioning systems (RTK) as the primary localisation sensors. It is worth noting that while the companies provide information on the internal noise characteristics of these sensors, modelling external noises (e.g., abnormal weather conditions, magnetic wave effects, light contrast problems) can be challenging. To address this, the MCStF method will be made adaptive and incorporate linear or nonlinear regression-based techniques to handle varying external noise conditions.

Furthermore, the adaptive MCStF approach will be tested using larger UAVs under the influence of highly-effective non-Gaussian or mix-Gaussian noise scenarios, simulating realistic and challenging operating environments.

The other important future work is about coloured noise. In the thesis, Gaussian and non-Gaussian distributions have been applied. This distribution is defined using Hidden Markov

Chain rules. That means the distribution noises do not depend on their previous values. However, noises, depending on the previous values, can be magnetic fields, wind, and extreme condition-based areas. Therefore, they will be modelled, and the proposed methods will be tested on these conditions. After that, all tests will be done in real-time systems.

These future developments aim to enhance the performance, robustness, and applicability of the UAV systems in various operating conditions and enable the deployment of advanced control and estimation techniques.

Appendix A

Control Experiments

The optimised controllers were subjected to thorough testing using the simulation model. The results obtained from the tests are visually represented in Figure A.1. Within the simulation, specific reference signals were defined for the x-axis and y-axis, namely $1.1m$ and $0.52m$, respectively. It is worth noting that these reference values remained consistent with the references employed during the process of optimising the controller parameters using BB-BC.

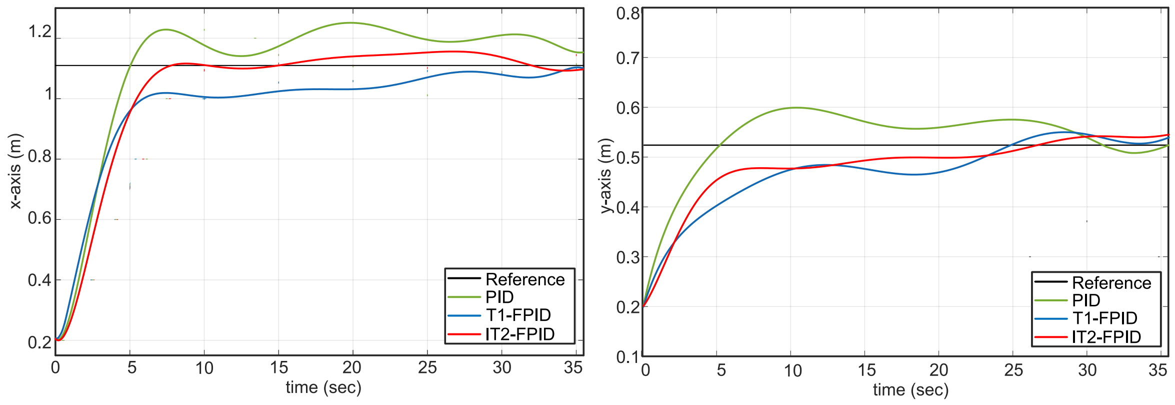

Figure A.1: Testing results of the compared controllers in the simulation without noise.

The simulation results conclusively demonstrate that the IT2-FPID controller outperformed the other controllers. It exhibited a remarkable absence of overshoot, minimal steady-state error, and swift rise time. Furthermore, in comparison to the PID controller, the T1-FPID controller showcased a smoother response. It is important to mention that the simulation results were obtained without the presence of Gaussian noise. The primary aim of this simulation was to achieve highly optimised values for the tuned parameters, utilising the BB-BC method.

Upon successfully testing the controllers in the simulation environment, the next step involved implementing them in the UAV system.

Fig. A.2 highlights the most significant challenge faced by all types of controllers, particularly on the y-axis. The PID controller struggles to maintain stability in the presence of disturbances, while both T1-FPID and IT2-FPID controllers exhibit slight overshoot. Examining Fig.A.2, the effects of disturbances are observed on the x-axis. The PID and T1-FPID controllers demonstrate slow responses and generally fail to reach the reference points. However, the IT2-FPID controller is less affected by these disturbances compared to the other controllers.

These findings emphasise that the y-axis poses a substantial challenge for all controller types, with the PID controller struggling to ensure stability. Additionally, the x-axis is affected by disturbances, causing slower responses and difficulties in achieving the desired reference points for the PID and T1-FPID controllers. In contrast, the IT2-FPID controller shows better resilience and performance in the presence of disturbances, both on the y-axis and the x-axis.

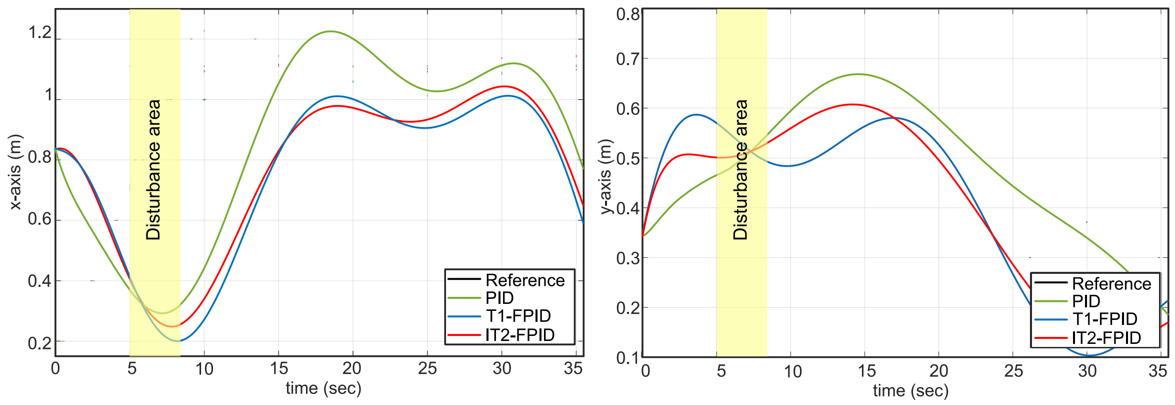

Figure A.2: Disturbing results of the compared controllers.

After analyzing the results based on disturbances, specific evaluations have been provided for each controller. In the case of the IT2-FPID controller, the second experiment involved coverage path planning. For this experiment, predetermined reference points were assigned to each vertex of the coverage path. The UAV would transition to the next vertex once it converged or reached the current vertex. Considering the task completion time, the IT2-FPID, T1-FPID, and PID controllers achieved completion times of 73.85 sec, 114.50 sec, and 122.05 sec, respectively. In addition to its faster completion time, the IT2-FPID controller also

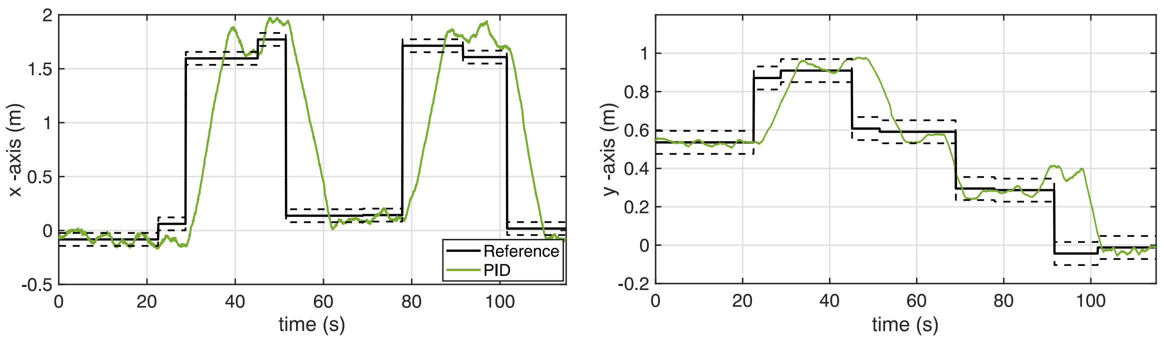

Figure A.3: Coverage path planning with PID.

demonstrated superior performance in various aspects.

The initial experiment in the coverage path planning (CPP) focused on the PID controller structure for a real-time UAV equipped with a flexible cable-connected payload, as illustrated in Fig. A.3. The PID controller exhibited a small steady-state error and a slower response on the x-axis. However, it encountered difficulties in accurately reaching the reference points on the y-axis.

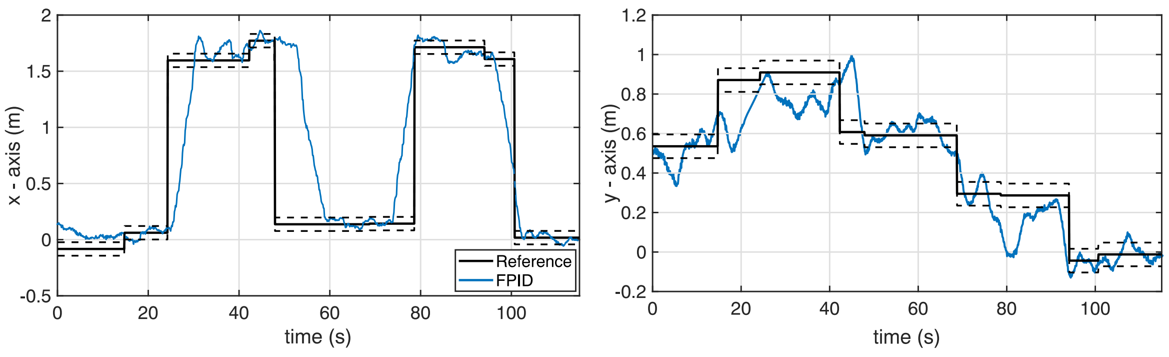

Figure A.4: Coverage path planning with IT2-FPID.

In Figure A.4, the results obtained with the T1-FPID controller in the CPP experiment were not as promising as those of the PID controller on the x-axis. Nonetheless, the T1-FPID controller completed the task in less time than the PID controller, which is considered a plausible outcome. However, oscillations were observed on the y-axis for the T1-FPID controller. This issue can be attributed to the fact that the payload is located closer to the front side of the UAV rather than at its centre of mass (CoM). Consequently, difficulties on the y-axis were anticipated for all controller structures.

The final and most impressive results were achieved with the IT2-FPID controller, as shown

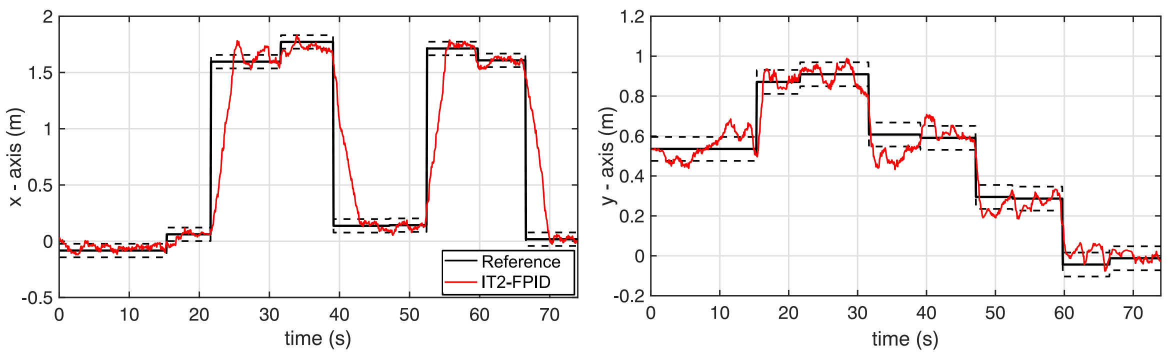

Figure A.5: Coverage path planning with IT2-FPID.

in Figure A.5. It completed the CPP task in a remarkable time of 73.85 sec. Moreover, it exhibited superior settling time, minimised oscillation, and reduced steady-state error, indicating minor errors and a shorter response time. Despite the payload position issue around the CoM experienced by the other controllers, the IT2-FPID controller effectively eliminated this problem.

Appendix B

Interacting Multiple Model

In the thesis, the IMM estimator method has been used, and the detailed algorithm has been represented as in Algorithm. 6. The IMM estimator is a probabilistic estimation algorithm commonly used in tracking and navigation systems to handle situations where the system dynamics can switch between different modes or behaviours. The general overview of the typical steps involved in one cycle of the IMM estimator:

1. Model Probability Prediction: In this step, the probabilities of each model (or mode) are predicted for the current time step based on the previous model probabilities and transition probabilities.

2. Mixing Weight Calculation: The mixing weights are computed to determine the contributions of each model in the filtering step. These weights consider the predicted model probabilities, the likelihood of the measurements given each model, and the transition probabilities.

3. Filtering Step: This step involves running individual filters (such as Kalman filters or other suitable filters) for each model to estimate the state variables and their covariance matrices. Each filter operates independently, taking into account the corresponding model and associated measurements.

4. Model Probability Update: The model probabilities are updated based on the filtering results and the measurements. This step adjusts the model probabilities according to the filtering performance and the agreement between the measurements and the predictions

made by each model.

5. Estimate Fusion: The state estimates and covariance matrices obtained from the individual filters are fused together to obtain an overall state estimate and covariance matrix that represents the combined information from all the models. This fusion step considers the model probabilities as weights in combining the estimates.

6. Output: The outputs of the IMM estimator typically include the overall state estimate and covariance matrix, which provide an estimate of the system's state and its associated uncertainty.

It's important to note that the specific details and equations involved in each step may vary depending on the implementation and the specific model used in the IMM estimator.

Algorithm 6 IMM estimator for each cycle

Model-conditioned reinitialisation (for ($i = 1, 2, \cdots, M$):

Predicted model probability:

$\hat{\mu}^{(i)}_{k|k-1} \triangleq P\left\{\mu^{(i)}_k | z_{k-1}\right\} = \sum_j \pi_{ji}\mu^{(j)}_{k-1}$.

Mixing weight:

$\mu^{j|i}_{k-1} \triangleq P\left\{m^{(j)}_{k-1} | m^{(i)}_k, z_{k-1}\right\} = \pi_{ji}\mu^{(j)}_k / \hat{\mu}^{(i)}_{k|k-1}$.

Mixing covariance:

$\bar{P}^{(i)}_{k-1|k-1} = \sum_j \left[P^{(i)}_{k-1|k-1} + \left(\bar{x}^{(i)}_{k-1|k-1} - \hat{x}^{(j)}_{k-1|k-1}\right)\left(\bar{x}^{(i)}_{k-1|k-1} - \hat{x}^{(j)}_{k-1|k-1}\right)'\right]\mu^{j|i}_{k-1}$.

Model-conditioned filtering (for ($i = 1, 2, \cdots, M$):

Selected Filters (KF, MCKF or MCStF)

Model probability update (for ($i = 1, 2, \cdots, M$):

Model likelihood:

$L^{(i)}_k \triangleq p\left[\tilde{z}^{(i)}_k | m^{(i)}_k, \tilde{z}^{(i)}_k\right] \approx \dfrac{e^{-\frac{(\tilde{z}^{(i)}_k)'(\tilde{z}^{(i)}_k)}{2S^{(i)}_k}}}{\sqrt{|2\pi S^{(i)}_k|}}$.

Model probability:

$\mu^{(i)}_k \triangleq P[m^{(i)}_k | z_k] = \dfrac{\hat{\mu}^{(i)}_{k|k-1} L^{(i)}_k}{\sum_j \hat{\mu}^{(j)}_{k|k-1} L^{(j)}_k}$.

Estimate fusion:

Overall state:

$\hat{x}_{k|k} \triangleq E\left[x_k | z_k\right] = \sum_i \hat{x}^{(i)}_{k|k}\mu^{(i)}_k$.

Overall covariance:

$P_{k|k} = \sum_i \left[P^{(i)}_{k|k} + \left(\hat{x}_{k|k} - \hat{x}^{(i)}_{k|k}\right)\left(\hat{x}_{k|k} - \hat{x}^{(i)}_{k|k}\right)'\right]\mu^{(i)}_k$.

Outputs: $\hat{\mathbf{x}}(k|k)$, $\mathbf{P}(k|k)$.

Appendix C

Geometric-based Obstacle Avoidance Results

C.1 Simulation Results

In this section, the proposed and compared approaches have been tested with 14 agents.

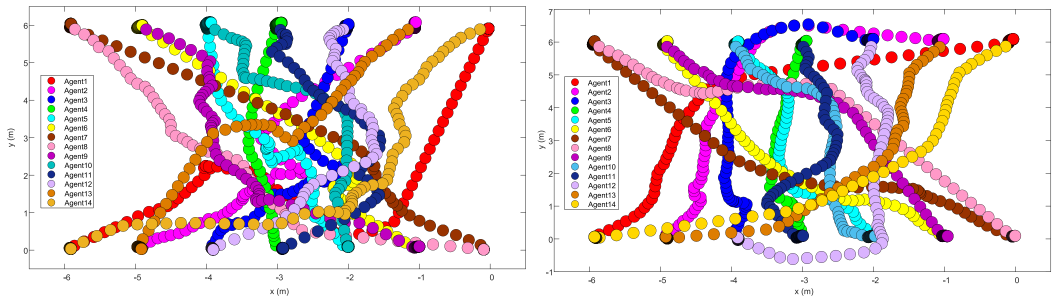

Figure C.1: VO (Left) and RVO (Right) avoidance approach.

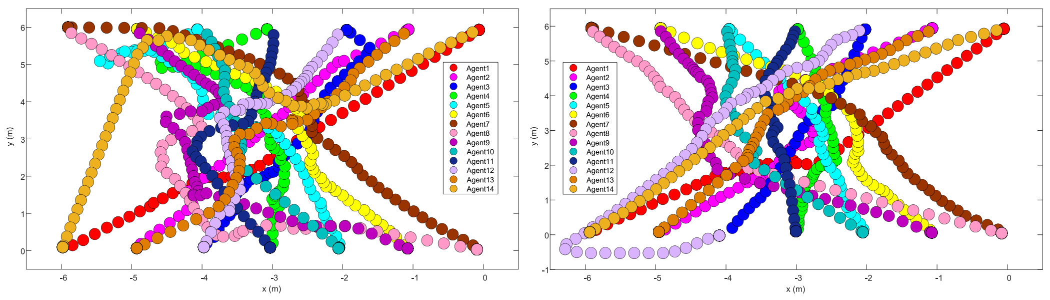

Figure C.2: HRVO (Left) and FIMVO (Right) avoidance approach.

In Fig. C.1 and Fig. C.2, trajectory results for all agents have been represented. The linear kinematic model has been used for each agent, and the sampling time has been selected as $10ms$.

In Figs. C.3- C.6, the important frames, such as starting points, endpoints, collision points, and non-collision points, have been shown. In these figures, specific points have been selected to show collision and non-collision points. Moreover, the behaviours of the proposed and compared velocity obstacle avoidance approaches under multi-agent systems have been detailed. The VO avoidance approach has collision points; the RVO trajectory shows a longer distance than VO, and HRVO increases the complexity and computation time.

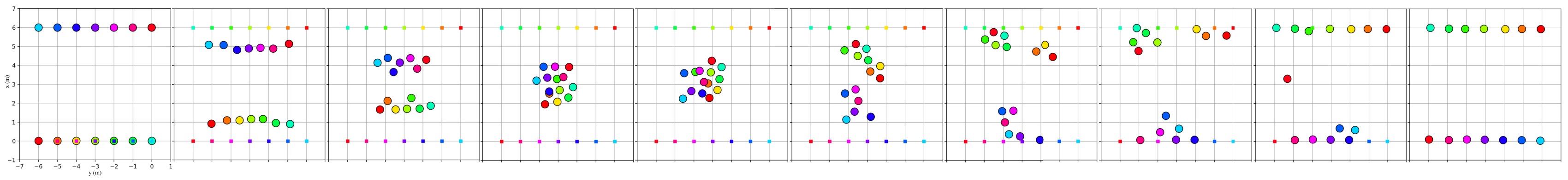

Figure C.3: VO avoidance approach.

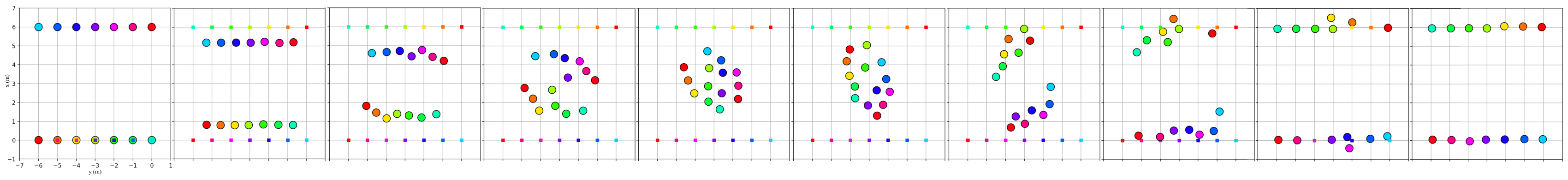

Figure C.4: RVO avoidance approach.

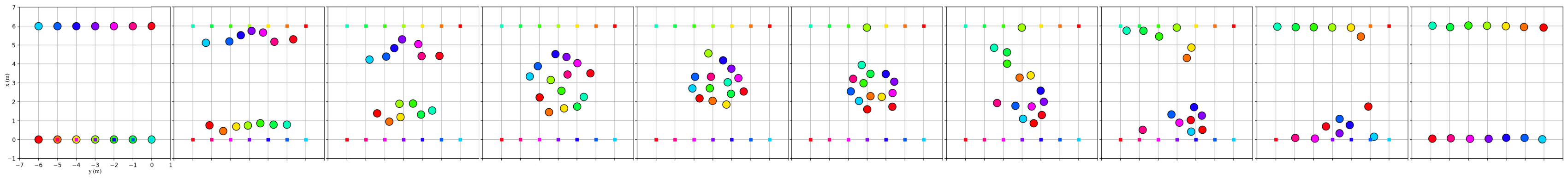

Figure C.5: HRVO avoidance approach.

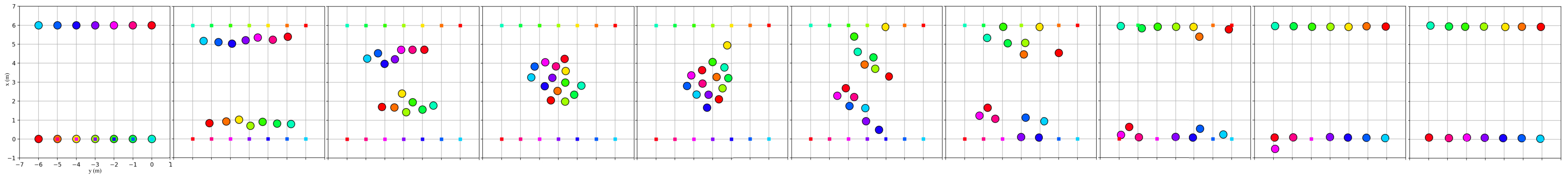

Figure C.6: FIMVO avoidance approach.

C.2 Real-time Results

This section shows the compared and proposed geometric-based obstacle avoidance results in a real-time environment. In the experiments, UAV1 (red) and UAV2 (blue) have been defined as static obstacles, and then the trajectory of UAV3 (yellow) has been drawn.

The tests show that VO is the fastest and collision rates are the highest. In Fig. C.7, the results of the VO avoidance approach are seen.

In Fig. C.8, the RVO avoidance approach has avoided the obstacle, and it has been near the obstacles.

The HRVO avoidance approach has given better results than VO and HRVO. the trajectory length is the longest when compared with others, the results have been represented In Fig. C.9

The proposed avoidance approach, FIMVO, has better results in terms of designed reference trajectory and simplicity. In Fig. C.10, The planned trajectory length is smaller than HRVO, and it has been also more reliable than VO and HRVO.

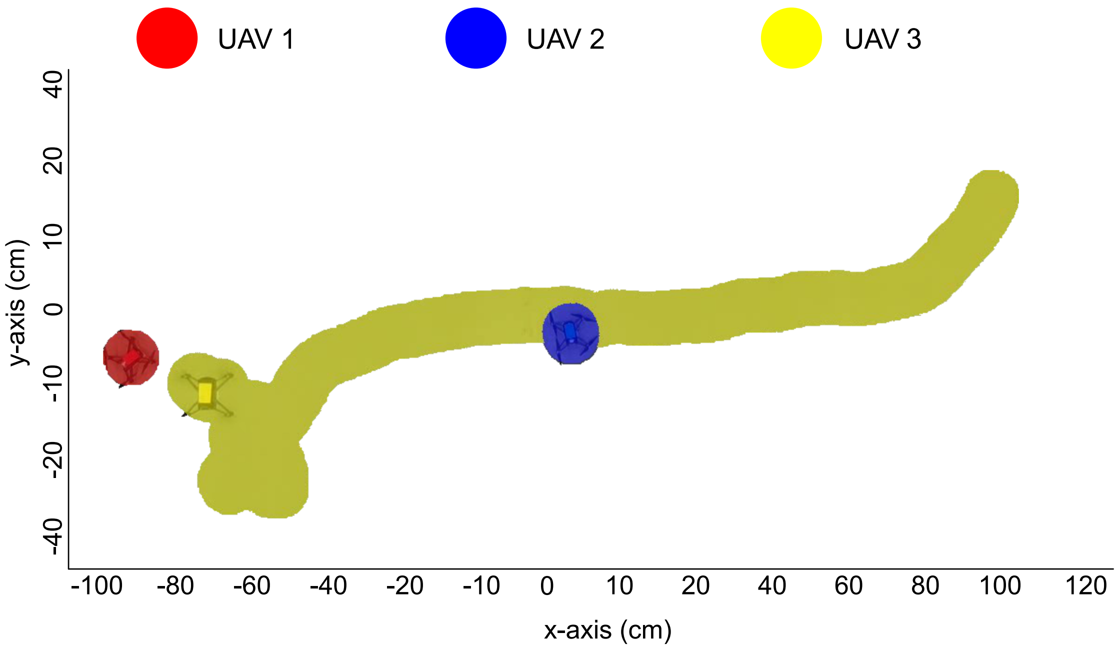

Figure C.7: Conventional VO avoidance approach.

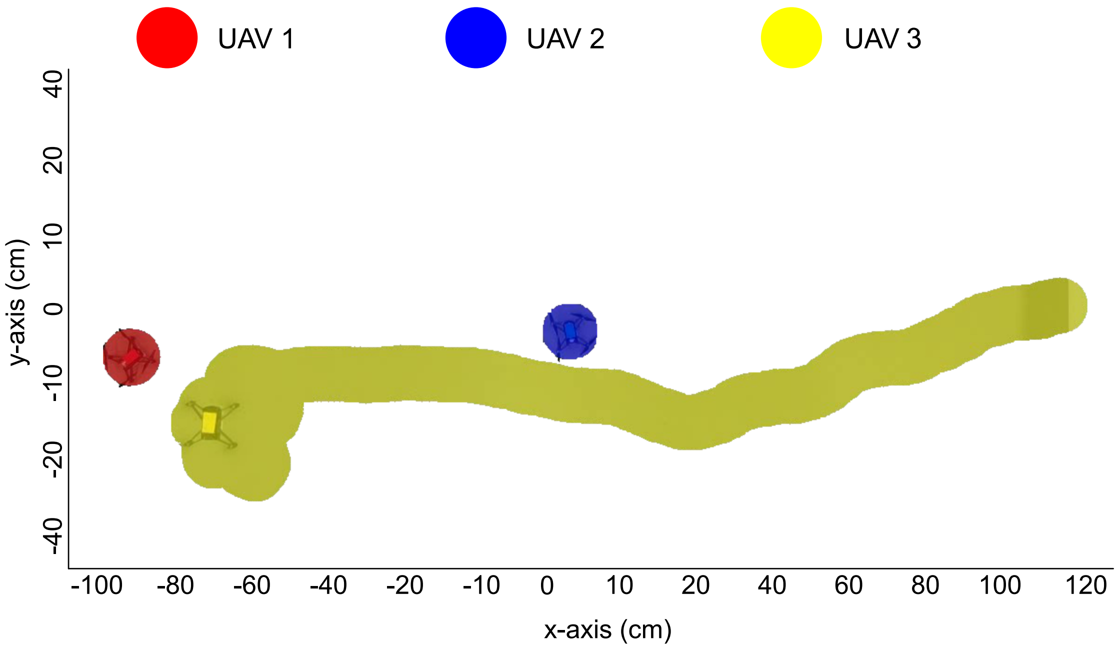

Figure C.8: RVO avoidance approach.

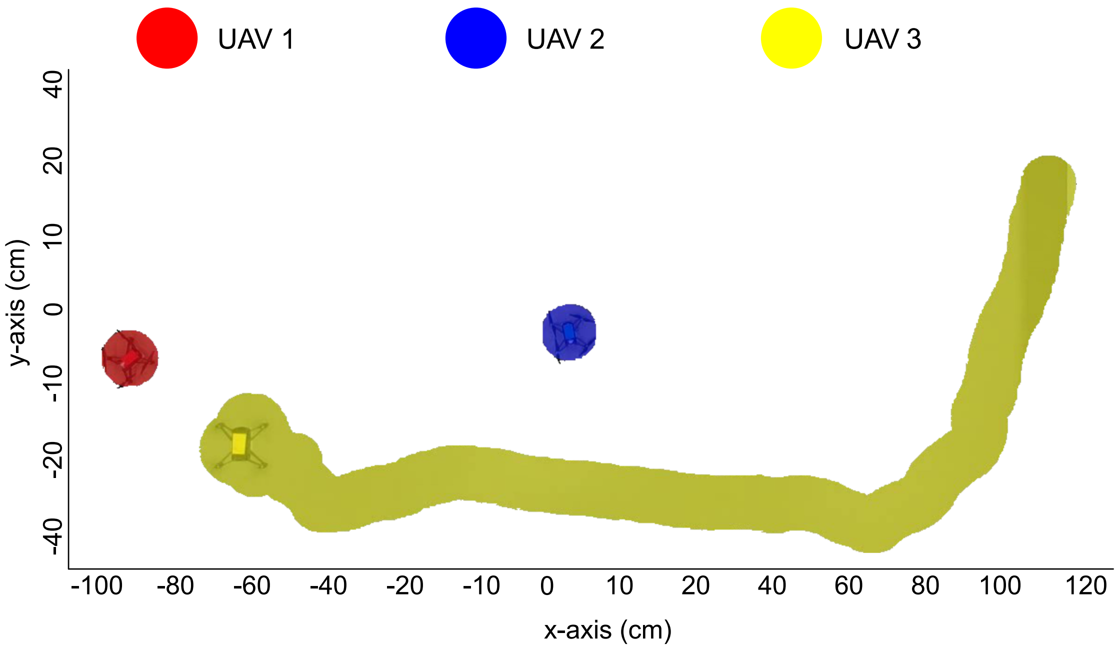

Figure C.9: HRVO avoidance approach.

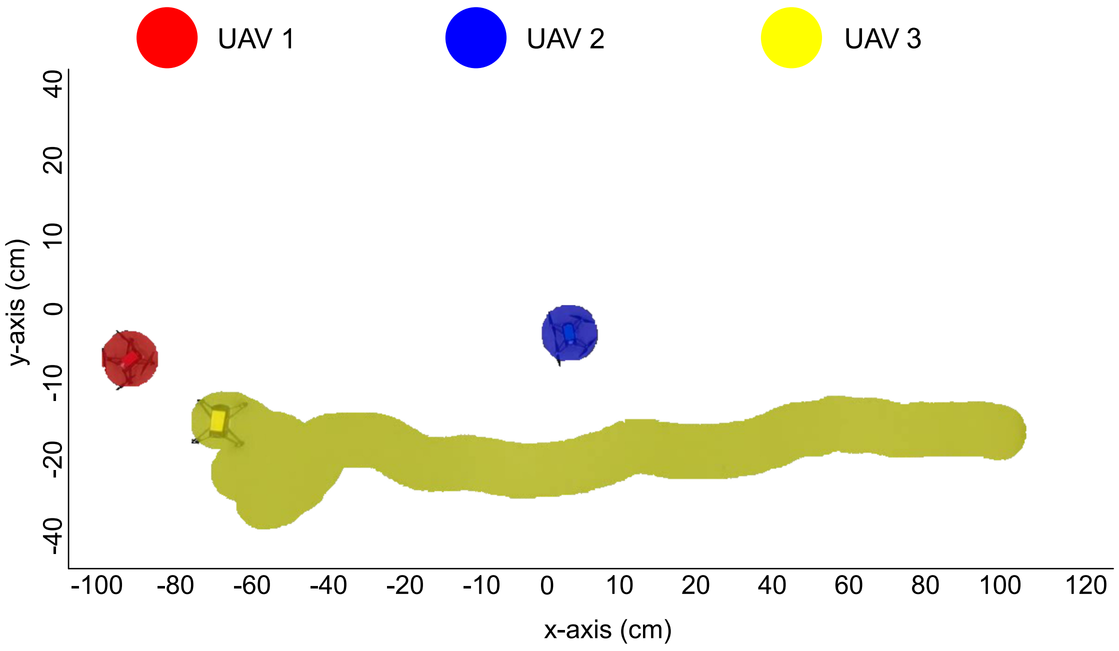

Figure C.10: FIMVO avoidance approach.

Bibliography

[1] M. Silvagni, A. Tonoli, E. Zenerino, and M. Chiaberge, “Multipurpose UAV for search and rescue operations in mountain avalanche events,” *Geomatics, Natural Hazards and Risk*, vol. 8, pp. 18–33, 2017.

[2] A. Sarabakha, C. Fu, E. Kayacan, and T. Kumbasar, “Type-2 Fuzzy Logic Controllers Made even Simpler: From Design to Deployment for UAVs,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 5069–5077, 2018.

[3] B. T. M. Leong, S. M. Low, and M. P. L. Ooi, “Low-cost microcontroller-based hover control design of a quadcopter,” in *Procedia Engineering*, vol. 41, 2012, pp. 458–464.

[4] G. A. Garcia, A. R. Kim, E. Jackson, S. S. Kashmiri, and D. Shukla, “Modeling and flight control of a commercial nano quadrotor,” in *Proceedings of the International Conference on Unmanned Aircraft Systems, ICUAS*, 2017.

[5] “Available at: Https://www.sensefly.com/drone/ebee-x-fixed-wing-drone.,” *[Accessed 1 August 2020]*, Sensefly Parrot Group.

[6] H. Ucgun, U. Yuzgec, and C. Bayilmis, “A review on applications of rotary-wing unmanned aerial vehicle charging stations,” *International Journal of Advanced Robotic Systems*, vol. 18, no. 3, p. 17 298 814 211 015 863, 2021.

[7] “Available at: Http://www.uaver.com/avian-p-aerial-mapping.html.,” *[Accessed 23 March 2023]*, Carbon-Based Technology Inc.

[8] “Triple Lense Camera Drone. DJI MAVIC 3 PRO , available at: Https://www.dji.com.,” *[Accessed 1 August 2020]*, DJI Platforms.

[9] S. J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, 2018, ISSN: 15523098. DOI: `10.1109/TRO.2018.2857475`.

[10] M. Furci, G. Casadei, R. Naldi, R. G. Sanfelice, and L. Marconi, "An open-source architecture for control and coordination of a swarm of micro-quadrotors," in *Proceedings of the International Conference on Unmanned Aircraft Systems, ICUAS*, 2015.

[11] X. Li, D. Zhu, and Y. Qian, "A survey on formation control algorithms for multi-auv system," *Unmanned Systems*, vol. 02, no. 04, pp. 351–359, 2014.

[12] K. A. Ghamry and Y. Zhang, "Formation control of multiple quadrotors based on leader-follower method," in *Proceedings of the International Conference on Unmanned Aircraft Systems, ICUAS*, Denver, CO, USA, 2015, pp. 1037–1042.

[13] Y. Kuriki and T. Namerikawa, "Experimental validation of cooperative formation control with collision avoidance for a multi-UAV system," in *Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications*, 2015.

[14] A. Barve and M. J. Nene, "Survey of flocking algorithms in multi-agent systems," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 6, p. 110, 2013.

[15] H. Manh La and W. Sheng, "Adaptive flocking control for dynamic target tracking in mobile sensor networks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 2009.

[16] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, 2006, ISSN: 00189286. DOI: `10.1109/TAC.2005.864190`.

[17] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, vol. 6, pp. 28 573–28 593, 2018, ISSN: 2169-3536. DOI: `10.1109/ACCESS.2018.2831228`. [Online]. Available: `https://ieeexplore.ieee.org/document/8352646/`.

[18] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles—a survey," *Journal of Industrial Information Integration*, vol. 16, p. 100 106, 2019.

[19] Y. Zhou, B. Rao, and W. Wang, "UAV swarm intelligence: Recent advances and future trends," *Ieee Access*, vol. 8, pp. 183 856–183 878, 2020.

[20] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[21] M. Bangura and R. Mahony, "Nonlinear dynamic modeling for high performance control of a quadrotor," in *Australasian Conference on Robotics and Automation, ACRA*, 2012, ISBN: 9780980740431.

[22] M. Greiff, "Modelling and control of the crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation," M.S. thesis, Lund University, Department of Automatic Control, Sweden, 2017.

[23] I. C. Dikmen, A. Arisoy, and H. Temeltaş, "Attitude control of a quadrotor," in *Proceedings of 4th International Conference on Recent Advances Space Technologies*, 2009.

[24] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "PID, LQR and LQR-PID on a quadcopter platform," in *Proceedings of the International Conference on Informatics, Electronics and Vision, ICIEV 2013*, 2013.

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

[26] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "PID, LQR and LQR-PID on a quadcopter platform," in *Proceedings of the International Conference on Informatics, Electronics and Vision (ICIEV)*, IEEE, 2013, pp. 1–6.

[27] F. Candan, Y. Peng, and L. Mihaylova, "A comparison of obstacle dependant gaussian and hybrid potential field methods for collision avoidance in multi-agent systems," in *Proceedings of the 1st International Conference on Computing and Machine Intelligence (ICMI 2021)*, Sheffield, 2021.

[28] F. Candan, A. Beke, and T. Kumbasar, "Design and deployment of fuzzy PID controllers to the nano quadcopter Crazyflie 2.0," in *Proceedings of the Innovations in Intelligent Systems and Applications (INISTA) Conference*, IEEE, 2018, pp. 1–6.

[29] M. R. Kaplan, A. Eraslan, A. Beke, and T. Kumbasar, "Altitude and position control of parrot mambo minidrone with PID and fuzzy PID controllers," in *Proceedings of the 11th International Conference on Electrical and Electronics Engineering (ELECO)*, IEEE, 2019, pp. 785–789.

[30] M. F. Everett, "LQR with integral feedback on a parrot minidrone," *Massachusetts Institute of Technology, Tech. Rep*, 2015.

[31] E. Barzanooni, K. Salahshoor, and A. Khaki-Sedigh, "Attitude flight control system design of UAV using LQG\LTR multivariable control with noise and disturbance," in *Proceedings of the 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, IEEE, 2015, pp. 188–193.

[32] W. Zhao and T. H. Go, "Quadcopter formation flight control combining mpc and robust feedback linearization," *Journal of the Franklin Institute*, vol. 351, no. 3, pp. 1335–1355, 2014.

[33] G. Ganga and M. M. Dharmana, "MPC controller for trajectory tracking control of quadcopter," in *Proceedings of the International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, IEEE, 2017, pp. 1–6.

[34] J. P. Ortiz, L. I. Minchala, and M. J. Reinoso, "Nonlinear robust h-infinity PID controller for the multivariable system quadrotor," *IEEE Latin America Transactions*, vol. 14, no. 3, pp. 1176–1183, 2016.

[35] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle," *Journal of intelligent & robotic systems*, vol. 61, no. 1, pp. 221–238, 2011.

[36] D. Palossi, J. Singh, M. Magno, and L. Benini, "Target following on nano-scale unmanned aerial vehicles," in *Proceedings of the 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, IEEE, 2017, pp. 170–175.

[37] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1471–1478, 2021.

[38] S. Islam, P. X. Liu, and A. El Saddik, “Robust control of four-rotor unmanned aerial vehicle with disturbance uncertainty,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1563–1571, 2014.

[39] L. Qian and H. H. Liu, “Path-following control of a quadrotor UAV with a cable-suspended payload under wind disturbances,” *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2021–2029, 2019.

[40] K. Guo, J. Jia, X. Yu, L. Guo, and L. Xie, “Multiple observers based anti-disturbance control for a quadrotor against payload and wind disturbances,” *Control Engineering Practice*, vol. 102, p. 104 560, 2020.

[41] S. A. Quintero and J. P. Hespanha, “Vision-based target tracking with a small UAV: Optimization-based control strategies,” *Control Engineering Practice*, vol. 32, pp. 28–42, 2014.

[42] B. Chen, X. Liu, H. Zhao, and J. C. Principe, “Maximum correntropy Kalman filter,” *Automatica*, vol. 76, pp. 70–77, 2017.

[43] X. Liu, B. Chen, H. Zhao, J. Qin, and J. Cao, “Maximum correntropy Kalman filter with state constraints,” *IEEE Access*, vol. 5, pp. 25 846–25 853, 2017.

[44] B. A. Gunawan, Y. Liu, and X. Li, “Adaptive localisation for unmanned surface vehicles using imu-interacting multiple model,” in *2020 International Conference on System Science and Engineering (ICSSE)*, IEEE, 2020, pp. 1–6.

[45] L. B. Cosme, W. M. Caminhas, M. F. S. V. D’Angelo, and R. M. Palhares, “A novel fault-prognostic approach based on interacting multiple model filters and fuzzy systems,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 519–528, 2018.

[46] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, “Interacting multiple model methods in target tracking: A survey,” *IEEE Transactions on aerospace and electronic systems*, vol. 34, no. 1, pp. 103–123, 1998.

[47] I. Hwang, C. E. Seah, and S. Lee, “A study on stability of the interacting multiple model algorithm,” *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 901–906, 2016.

[48] X. R. Li and V. P. Jilkov, “Survey of maneuvering target tracking: Dynamic models,” in *Signal and Data Processing of Small Targets 2000*, International Society for Optics and Photonics, vol. 4048, 2000, pp. 212–235.

[49] C. Shen and L. Mihaylova, “A flexible robust Student’s t-based multimodel approach with maximum versoria criterion,” *Signal Processing*, vol. 182, p. 107 941, 2021.

[50] X. Fan, G. Wang, J. Han, and Y. Wang, “Interacting multiple model based on maximum correntropy Kalman filter,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 3017–3021, 2021.

[51] Q. Li, Y. Ben, J. Tan, S. M. Naqvi, and J. Chambers, “Robust selection of the degrees of freedom in the student’st distribution through multiple model adaptive estimation,” *Signal Processing*, vol. 153, pp. 255–265, 2018.

[52] D. Li and J. Sun, “Robust interacting multiple model filter based on student’s t-distribution for heavy-tailed measurement noises,” *Sensors*, vol. 19, no. 22, p. 4830, 2019.

[53] H. Liu and W. Wu, “Interacting multiple model (imm) fifth-degree spherical simplex-radial cubature Kalman filter for maneuvering target tracking,” *Sensors*, vol. 17, no. 6, p. 1374, 2017.

[54] C. E. Seah and I. Hwang, “State estimation for stochastic linear hybrid systems with continuous-state-dependent transitions: An imm approach,” *IEEE Transactions on Aero space and Electronic Systems*, vol. 45, no. 1, pp. 376–392, 2009.

[55] H. A. Blom and Y. Bar-Shalom, “The interacting multiple model algorithm for systems with markovian switching coefficients,” *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[56] W. Blanding, P. Willett, and Y. Bar-Shalom, “Ml-pda: Advances and a new multitarget approach,” *EURASIP Journal on Advances in Signal Processing*, vol. 2008, pp. 1–13, 2007.

[57] X. R. Li and Y. Bar-Shalom, “A recursive multiple model approach to noise identification,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 3, pp. 671–684, 1994.

[58] Y. Boers and J. N. Driessen, "Interacting multiple model particle filter," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 150, no. 5, pp. 344–349, 2003.

[59] X.-R. Li, *Engineer's guide to variable-structure multiple-model estimation for tracking, Chapter 10, In Multitarget-multisensor tracking: Applications and advances*, Y. Bar-shalom and D. W. Blair, Eds. Boston, MA: Artech House, 2000, pp. 499–567.

[60] M. Jafarinasab, S. Sirouspour, and E. Dyer, "Model-based motion control of a robotic manipulator with a flying multirotor base," *Proceedings of the IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 5, pp. 2328–2340, 2019.

[61] L. Cao, D. Qiao, and X. Chen, "Laplace $L1$ Huber based cubature Kalman filter for attitude estimation of small satellite," *Acta Astronautica*, vol. 148, pp. 48–56, 2018.

[62] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[63] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based UAV path planning," *Knowledge-Based Systems*, vol. 158, pp. 54–64, 2018.

[64] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[65] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors," *IEEE Robotics and Automation Letters*, 2018.

[66] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[67] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[68] E. Sabudin, R. Omar, and C. Che Ku Melor, "Potential field methods and their inherent approaches for path planning," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 18, pp. 10 801–10 805, 2016.

[69] A. Alexopoulos, A. Kandil, P. Orzechowski, and E. Badreddin, "A comparative study of collision avoidance techniques for unmanned aerial vehicles," in *Proc. of the IEEE International C Conference on Systems, Man, and Cybernetics*, IEEE, 2013, pp. 1969–1974.

[70] F. Rossi, S. Bandyopadhyay, M. Wolf, and M. Pavone, "Review of multi-agent algorithms for collective behavior: A structural taxonomy," *IFAC-PapersOnLine*, vol. 51, no. 12, pp. 112–117, 2018.

[71] B. Albaker and N. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles," in *Proceedings of the International Conference for Technical Postgraduates (TECHPOS)*, IEEE, 2009, pp. 1–7.

[72] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity Obstacle Approaches for Multi-Agent Collision Avoidance," *Unmanned Systems*, vol. 07, no. 01, pp. 55–64, Jan. 2019.

[73] M. Radmanesh, M. Kumar, P. H. Guentert, and M. Sarim, "Overview of path-planning and obstacle avoidance algorithms for UAVs: A comparative study," *Unmanned Systems*, vol. 6, no. 02, pp. 95–118, 2018.

[74] Z.-z. Yu, J.-h. Yan, J. Zhao, Z.-F. Chen, and Y.-h. Zhu, "Mobile robot path planning based on improved artificial potential field method," *Harbin Gongye Daxue Xuebao(Journal of Harbin Institute of Technology)*, vol. 43, no. 1, pp. 50–55, 2011.

[75] J.-H. Cho, D.-S. Pae, M.-T. Lim, and T.-K. Kang, "A real-time obstacle avoidance method for autonomous vehicles using an obstacle-dependent gaussian potential field," *Journal of Advanced Transportation*, vol. 2018, Article ID 5041401, 2018.

[76] Y. I. Jenie, E. Kampen, C. D. Visser, J. Ellerbroek, and J. Hoekstra, "Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles," *Journal of Guidance Control and Dynamics*, vol. 38, no. 6, pp. 1140–1146, 2015.

[77] F. Duchoň, A. Babinec, M. Kajan, *et al.*, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[78] J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 1928–1935.

[79] J. A. Douthwaite, A. De Freitas, and L. S. Mihaylova, “An interval approach to multiple unmanned aerial vehicle collision avoidance,” in *Proceedings of the Sensor Data Fusion: Trends, Solutions, Applications (SDF) Workshop*, IEEE, 2017, pp. 1–8.

[80] J. Borenstein and Y. Koren, “The vector field histogram-fast obstacle avoidance for mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[81] S. Bolbhat, A. Bhosale, G. Sakthivel, D. Saravanakumar, R. Sivakumar, and J. Lakshmipathi, “Intelligent obstacle avoiding AGV using vector field histogram and supervisory control,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1716, 2020, p. 012 030.

[82] D. F. Campos, A. Matos, and A. M. Pinto, “An Adaptive Velocity Obstacle Avoidance Algorithm for Autonomous Surface Vehicles,” in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2019. DOI: 10.1109/IROS40897.2019.8968156.

[83] Y. I. Jenie, E. J. Van Kampen, C. C. De Visser, J. Ellerbroek, and J. M. Hoekstra, “Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles,” *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 6, pp. 1140–1146, 2015.

[84] D. Wilkie, J. Van Den Berg, and D. Manocha, “Generalized velocity obstacles,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2009.

[85] J. O. Gaya, L. T. Gonçalves, A. C. Duarte, B. Zanchetta, P. Drews, and S. S. Botelho, “Vision-Based Obstacle Avoidance Using Deep Learning,” in *Proceedings of the 13th Latin American Robotics Symposium and 4th Brazilian Symposium on Robotics, LARS/SBR*, 2016.

[86] M. Suzuki, K. Uchiyama, D. Bennet, and C. MacInnes, “Three-dimensional formation flying using bifurcating potential fields,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, 2009.

[87] F. Candan, Y. Peng, and L. Mihaylova, “A comparison of obstacle dependant gaussian and hybrid potential field methods for collision avoidance in multi-agent systems,” in *Proceedings of the 1st International Conference on Computing and Machine Intelligence (ICMI 2021)*, Sheffield, 2021.

[88] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, “A comparative study of velocity obstacle approaches for multi-agent systems,” in *Proceedings of the UKACC 12th International Conference on Control*, IEEE, 2018, pp. 289–294.

[89] Y. I. Jenie, E.-J. van Kampen, C. C. de Visser, J. Ellerbroek, and J. M. Hoekstra, “Three-dimensional velocity obstacle method for uncoordinated avoidance maneuvers of unmanned aerial vehicles,” *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 10, pp. 2312–2323, 2016.

[90] Y. Xiuxia, Z. Yi, and Z. Weiwei, “Obstacle avoidance method of three-dimensional obstacle spherical cap,” *Journal of Systems Engineering and Electronics*, vol. 29, no. 5, pp. 1058–1068, 2018.

[91] S. Abdelmoeti and R. Carloni, “Robust control of uavs using the parameter space approach,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5632–5637. DOI: `10.1109/IROS.2016.7759828`.

[92] C. DJI Tello, *Dji Tello EDU, Ryzerobotics*, `https://www.ryzerobotics.com/tello-edu`, Accessed: 2022-06-16.

[93] P. Castillo, R. Lozano, and A. Dzul, “Stabilization of a mini-rotorcraft having four rotors,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2693–2698.

[94] J. Rao, B. Li, Z. Zhang, D. Chen, and W. Giernacki, “Position control of quadrotor UAV based on cascade fuzzy neural network,” *Energies*, vol. 15, no. 5, p. 1763, 2022.

[95] W. Giernacki, J. Rao, S. Sladic, A. Bondyra, M. Retinger, and T. Espinoza-Fraire, "Dji tello quadrotor as a platform for research and education in mobile robotics and control engineering," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 735–744.

[96] A. Sarabakha, C. Fu, E. Kayacan, and T. Kumbasar, "Type-2 fuzzy logic controllers made even simpler: From design to deployment for UAVs," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 5069–5077, 2017.

[97] T. Kumbasar, E. Yesil, I. Eksin, and M. Guzelkaya, "Inverse fuzzy model control with online adaptation via big bang-big crunch optimization," in *Proceedings of the 3rd International Symposium on Communications, Control and Signal Processing*, IEEE, 2008, pp. 697–702.

[98] T. Kumbasar and H. Hagras, "Big bang–big crunch optimization based interval type-2 fuzzy pid cascade controller design strategy," *Information Sciences*, vol. 282, pp. 277–295, 2014.

[99] A. A. Khan and N. Rapal, "Fuzzy pid controller: Design, tuning and comparison with conventional pid controller," in *2006 IEEE International Conference on Engineering of Intelligent Systems*, IEEE, 2006, pp. 1–6.

[100] A. Sakalli, T. Kumbasar, and J. M. Mendel, "Towards systematic design of general type-2 fuzzy logic controllers: Analysis, interpretation, and tuning," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 2, pp. 226–239, 2020.

[101] D. H. Lee and D. Park, "An efficient algorithm for fuzzy weighted average," *Fuzzy Sets and Systems*, vol. 87, no. 1, pp. 39–45, 1997.

[102] P.-T. Chang, K.-C. Hung, K.-P. Lin, and C.-H. Chang, "A comparison of discrete algorithms for fuzzy weighted average," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 5, pp. 663–675, 2006.

[103] O. Karasakal, M. Guzelkaya, I. Eksin, E. Yesil, and T. Kumbasar, "Online tuning of fuzzy PID controllers via rule weighing based on normalized acceleration," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 184–197, 2013.

[104] L. B. Palma, R. A. Antunes, P. Gil, and V. Brito, "Takagi-sugeno-kang fuzzy pid control for dc electrical machines," in *Proceedings of the IEEE 14th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG)*, IEEE, vol. 1, 2020, pp. 309–316.

[105] M. Biglarbegian, W. W. Melek, and J. M. Mendel, "Design of novel interval type-2 fuzzy controllers for modular and reconfigurable robots: Theory and experiments," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 4, pp. 1371–1384, 2010.

[106] A. Sakalli, T. Kumbasar, E. Yesil, and H. Hagras, "Analysis of the performances of type-1, self-tuning type-1 and interval type-2 fuzzy pid controllers on the magnetic levitation system," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2014, pp. 1859–1866.

[107] A. Sakalli, A. Beke, and T. Kumbasar, "Analyzing the control surfaces of type-1 and interval type-2 flcs through an experimental study," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2018, pp. 1–6.

[108] A. Beke and T. Kumbasar, "Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 372–384, 2019.

[109] A. Beke and T. Kumbasar, "Type-2 fuzzy logic-based linguistic pursuing strategy design and its deployment to a real-world pursuit evasion game," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 211–221, 2018.

[110] K. Tai, A.-R. El-Sayed, M. Biglarbegian, C. I. Gonzalez, O. Castillo, and S. Mahmud, "Review of recent type-2 fuzzy controller applications," *Algorithms*, vol. 9, no. 2, p. 39, 2016.

[111] O. K. Erol and I. Eksin, "A new optimization method: Big bang–big crunch," *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.

[112] S. J. Upasane, H. Hagras, M. H. Anisi, S. Savill, I. Taylor, and K. Manousakis, "A big bang-big crunch type-2 fuzzy logic system for explainable predictive maintenance," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2021, pp. 1–8.

[113] Python.org, *Python3*, `https://www.python.org/`, [Accessed: 14-Jan-2023].

[114] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert Systems with Applications*, vol. 55, pp. 441–451, 2016.

[115] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, 2019.

[116] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[117] C. Intel Realsense, *Intel Realsense D415*, `https://www.intelrealsense.com/depth-camera-d415`, Accessed: 2022-06-16.

[118] M. C. Santos, L. V. Santana, A. S. Brandão, M. Sarcinelli-Filho, and R. Carelli, "Indoor low-cost localization system for controlling aerial robots," *Control Engineering Practice*, vol. 61, pp. 93–111, 2017.

[119] C. Mathworks, *Computer vision toolbox for Matlab, Matlab*, `https://uk.mathworks.com/help/vision/index.html`, Accessed: 2022-06-16.

[120] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 103–123, 1998. DOI: `10.1109/7.640267`.

[121] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[122] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988. DOI: `10.1109/9.1299`.

[123] S. Li, L. Li, D. Shi, W. Zou, P. Duan, and L. Shi, "Multi-kernel maximum correntropy Kalman filter for orientation estimation," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6693–6700, 2022. DOI: `10.1109/LRA.2022.3176798`.

[124] F. Candan, A. Beke, C. Shen, and L. Mihaylova, "An interacting multiple model correntropy Kalman filter approach for unmanned aerial vehicle localisation," in *Proceedings of the International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE, 2022, pp. 1–6. DOI: 10.1109/INISTA55318.2022.9894214.

[125] M. Roth, E. Özkan, and F. Gustafsson, "A Student's t filter for heavy tailed process and measurement noise," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 5770–5774.

[126] A. H. Sayed, "A framework for state-space estimation with uncertain models," *IEEE Transactions on Automatic Control*, vol. 46, no. 7, pp. 998–1013, 2001.

[127] Y. Huang, Y. Zhang, N. Li, Z. Wu, and J. A. Chambers, "A novel robust Student's t-based Kalman filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 3, pp. 1545–1554, 2017.

[128] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part V. Multiple-model methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1255–1321, 2005.

[129] C. Bitcraze Crazyflie, *Bitcraze Crazyflie 2.0*, https://www.bitcraze.io/products/old-products/crazyflie-2-0, Accessed: 2022-06-16.

[130] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, 1998, ISSN: 02783649. DOI: 10.1177/027836499801700706.

[131] C. Katrakazas, M. Quddus, W. H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, 2015, ISSN: 0968090X. DOI: 10.1016/j.trc.2015.09.011.

[132] Y. Peng, "The collision avoidance method for multi-agents system," M.S. thesis, University of Sheffield, United Kingdom, 2020.

[133] M. G. Park, J. H. Jeon, and M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, IEEE, vol. 3, 2001, pp. 1530–1535.

[134] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, “Evolutionary artificial potential fields and their application in real time robot path planning,” in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, IEEE, vol. 1, 2000, pp. 256–263.

[135] Y. Hu and S. X. Yang, “A knowledge based genetic algorithm for path planning of a mobile robot,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, IEEE, vol. 5, 2004, pp. 4350–4355.

[136] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[137] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, “The hybrid reciprocal velocity obstacle,” *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[138] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, “Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 5917–5922.

[139] İ. Şahin and C. Ulu, “Altitude control of a quadcopter using interval type-2 fuzzy controller with dynamic footprint of uncertainty,” *ISA Transactions*, 2022.

[140] D. Zhang and J. Wang, “Fuzzy pid speed control of bldc motor based on model design,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1303, 2019, p. 012 124.

[141] *Dji tello edu*, https://m.dji.com/uk/product/tello-edu/, [Accessed: 14-Sep-2022].

[142] C. Guzay and T. Kumbasar, “Aggressive maneuvering of a quadcopter via differential flatness-based fuzzy controllers: From tuning to experiments,” *Applied Soft Computing*, p. 109 223, 2022.

[143] *Intel realsense*, https://www.intelrealsense.com/stereo-depth/, [Accessed: 14-Sep-2022].

[144] X. Fan, G. Wang, J. Han, and Y. Wang, “Interacting multiple model based on maximum correntropy Kalman filter,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 3017–3021, 2021.