Explanations:

Institute of Aeronautics and Applied Mechanics

Automatic Control and Robotics

Robotics

UAV Decentralized Formation Flying For Surveying Missions, Using Behavioral Techniques

Charles Chano

317272 - index number

D.Sc., Ph.D., Elżbieta Jarzębowska

2023

Warsaw University of Technology

FACULTY OF
POWER AND AERONAUTICAL ENGINEERING

Institute of Aeronautics and Applied Mechanics

Master's diploma thesis

in the field of study of Robotics and Automatic Control
and the specialization of Robotics

UAV Decentralized Formation Flying For Surveying Missions, Using Behavioral Techniques

Charles Chano
Student record book number: 317272

Supervisor:
D.Sc., Ph.D., Elżbieta Jarzębowska

WARSAW October 1, 2023

Section 1. Summary

(ENG) UAV Decentralized Formation Flying For Surveying Missions, Using Behavioral Techniques

This thesis explores the application of multi-agent cooperation in the context of Unmanned Aerial Vehicles (UAV (Unmanned Aerial Vehicle)s), with a focus on emergency response scenarios. It discusses the use of UAV formations for various applications, emphasizing their potential in hazardous environments and large-scale surveys. This thesis discusses different cooperation architectures and decision-making structures, highlighting architectures for decentralized formation maintenance.

A specific mission scenario involving earthquake response is presented, detailing the use of the DJI Matrice 300RTK as the UAV and photogrammetry with DJI P1 cameras. The functional procedure for deploying a UAV swarm in emergency scenarios is outlined.

The thesis outlines the development of a control scheme based on a nonlinear dynamic model, feedback linearization, and PID controllers. Various behaviors are designed to enable single UAVs to integrate into assigned formations and execute survey missions. The use of ROS2 and Gazebo simulator for real-time simulation is discussed.

In conclusion, this work contributes to the development of flexible UAV swarms for emergency response, focusing on decentralized formation flying and survey missions. The study discusses limitations, capabilities, and potential future enhancements.

Keywords: *Decentralized Multi-Agent Cooperation, UAVs, Formation Flying, Decentralized Control, Emergency Response, Photogrammetry, Quadrotor Modeling, Feedback Linearization, Behavioral Control, ROS2, Gazebo Simulation.*

(PL) Sterowanie Formacją Obiektów Latających Bezzałogowych do Celów Misji Ratunkowych z Wykorzystaniem Technik Sterowania Typu "Behavior Based"

Ta praca badawcza analizuje możliwości zastosowania współpracujących w formacji obiektów autonomicznych, ze szczególnym uwzględnieniem scenariuszy reagowania na sytuacje awaryjne. Omawia wykorzystanie formacji UAV w różnych zastosowaniach, podkreślając ich potencjał w niebezpiecznych środowiskach i przy badaniach na dużą skalę. Praca ta omawia różne architektury współpracy oraz struktury podejmowania decyzji, ze szczególnym uwzględnieniem architektur do zdecentralizowanego utrzymania formacji.

Przedstawiony jest konkretny scenariusz misji związanej z reakcją na trzęsienie ziemi, w którym wykorzystywany jest dron DJI Matrice 300RTK jako UAV oraz fotogrametria z użyciem kamer DJI P1. Szczegółowo opisana jest procedura funkcjonalna wdrożenia grupy UAV w sytuacjach awaryjnych.

Praca przedstawia rozwinięcie schematu sterowania opartego na nieliniowym modelu dynamicznym, sprzężeniu zwrotnym i kontrolerach PID. Opracowane są różne zachowania, które umożliwiają pojedynczym UAV integrację w przypisane formacje i wykonywanie misji badawczych. Omówione jest wykorzystanie środowiska ROS2 i symulatora Gazebo do symulacji czasu rzeczywistego.

Podsumowując, ta praca przyczynia się do rozwoju elastycznych grup UAV w reakcji na sytuacje awaryjne, skupiając się na zdecentralizowanym lataniu w formacjach i misjach badawczych. W pracy omówiono ograniczenia, możliwości i potencjalne przyszłe udoskonalenia.

Słowa kluczowe: *Zdecentralizowana Wieloagentowa Współpraca, UAV, Lot w Formacjach, Zdecentralizowane Sterowanie, Reakcja na Sytuacje Awaryjne, Fotogrametria, Modelowanie Quadrotora, Sprzężenie Zwrotne, Sterowanie Behawioralne, ROS2, Symulacja Gazebo.*

Contents

1 Summary . . . 2
2 Introduction . . . 6
3 Motivation . . . 14
4 Mission Scenario Definition Adopted in the Thesis . . . 16
5 Approach and Methodology . . . 17
6 Dynamic Modelling and Controller Design for the UAVs . . . 19
6.1 Simulation Environment . . . 19
6.2 UAV Quadrotor Reference: Matrice 300 RTK . . . 20
6.2.1 Space Model of the UAV . . . 20
6.2.2 Powertrain Reference . . . 21
6.2.3 Moments of Inertia and Mass . . . 21
6.2.4 Propeller Coefficients and Forces . . . 23
6.3 UAV Quadrotor Modeling . . . 28
6.3.1 Kinematic Model . . . 28
6.3.2 Quadrotor Method of Flight . . . 29
6.3.3 Dynamic Model . . . 32
6.3.4 Control Input Relations for the UAV . . . 35
6.3.5 State Space Model of the UAV . . . 36
6.4 Single UAV Quadrotor Control . . . 38
6.4.1 Desired Linear Acceleration to Rotational Position . . . 38
6.4.2 Feedback Linearization . . . 39
6.4.3 Controller Output Motor Speed Command . . . 40
7 UAV Communication . . . 42
7.1 MANET Architecture . . . 42
7.2 Routing Protocol . . . 42
8 Behavioral Controller . . . 43
8.1 Behavioral Based Control . . . 43
8.1.1 Behavior Controller to Attitude Controller Integration . . . 44
8.1.2 Move-to-Goal Behavior . . . 46
8.1.3 Get to Altitude . . . 48
8.1.4 Collision Avoidance behavior . . . 50
8.1.5 Formation Maintenance Behavior . . . 56
8.2 Formation Formulation . . . 58
8.2.1 Centralized Formation Position Assignment . . . 58
8.2.2 Decentralized Reformation . . . 61
9 Simulation Studies . . . 62
9.1 Formation Reconfiguration . . . 62
9.1.1 Formation Reconfiguration: Simulation 1 . . . 62
9.1.2 Formation Reconfiguration: Simulation 2 . . . 73

9.1.3 Formation Reconfiguration: Simulation 3 77
9.2 UAV Failure within Formation 81
9.2.1 Simulation 1: Total Failure of UAV 1 82
9.2.2 Simulation 2: Partial Failure of UAV 1 84
9.3 Surveying Mission 85
9.3.1 Simulation 1 (Single UAV) 86
9.3.2 Simulation 2 (UAV Formation) 87
9.3.3 Simulation 3 (UAV Formation) 88
9.3.4 Simulation 4 (UAV Formation) 89
9.3.5 Simulation 5 (UAV Formation) 90
9.3.6 Simulation 6 (UAV Formation) 91
9.3.7 Simulation 7 (UAV Formation) 92
9.3.8 Survey Mission Results 93
10 Conclusion 95
10.1 Future Work 96
Acronyms 100

Section 2. Introduction

Multi-agent cooperation has been of increasing interest with increasing computing power and decreasing sensor size. The application of multi-agent cooperation to UAV holds great promise in applications such as military, commercial, and emergency response.

Single, operator-controlled UAVs have already proven to be useful in many applications, such as 3D mapping. In [1], they review UAVs for 3D mapping applications covering case studies in archeological site 3D modeling, geological and mining studies, and urban area surveying. They go on to note the advantages of having a low-cost system that is simple to deploy but also states that the lower resolution of cheaper cameras requires more images to achieve similar results to better equipped UAVs. Additionally, they state future expectations in improved geo-referencing through DGPS (Differential Global Positioning System) and INS (Inertial Navigation System).

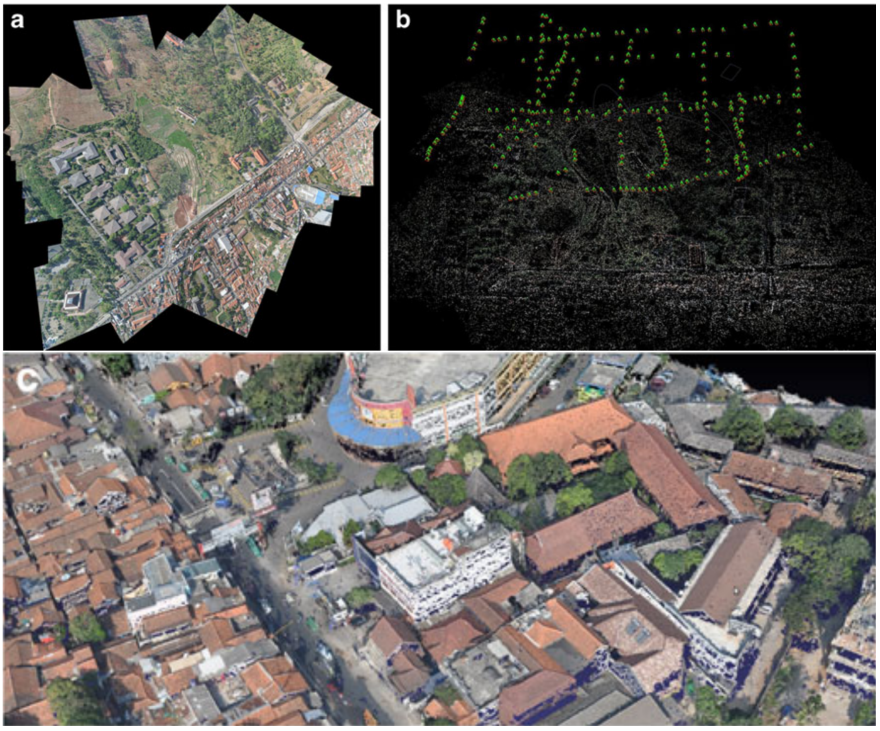


Figure 2.1. "A mosaic over an urban area in Bandung, Indonesia **(a)** Visualization of the bundle adjustment results of a large UAV block (ca. 270 images **(b)**) and a close view of the produced digital surface map over the urban area **(c)**." [1].

There are many situations where it's preferred to send a group of UAVs due to the risk of the environment. By sending a formation of UAVs to gather data in a dangerous situation, lives can be kept out of danger. With a swarm of UAVs in formation, coverage of large areas can be leveraged to get a macro level scale of a situation; such as in exploration, terrain model acquisitions, and aerial measurements in dangerous environments [2].

Cooperation between agents can be summarized to have the following architectures [3]:

- Physical coupling between agents.
- Formation of agents, maintaining organization while avoiding collisions.
- Swarms of homogeneous teams of agents collectively organized.
- Intentional cooperation where each agent achieves varying sub-goals to accomplish an overarching cooperative goal.

Additionally, architectures can have a variety of decision-making structures between agents summarized as [3]:

1. Centralized - a centralized agent has access to information from the rest of the group.
2. Decentralized - each agent solves a control problem for itself, but there are varying degrees of shared information between nodes:
   (a) Completely independent - algorithms only utilizing local information (sensors, pre-configured data).
   (b) Semi-independent - algorithms utilizing some global information available to all nodes.
3. Semi-Decentralized - utilizes the combined computational capacity of the group while depending on a central agent for sensor fusion.

With these classifications in mind, many works fall squarely in one or in between these architecture types to best solve their problem application.

For formation maintenance architectures, there are various types of methods but generally fall into the following categories:

- Leader-Follower
- Virtual Structure/Virtual Leader
- Behavioral Methods

In [4], fixed-wing UAV formation is maintained with a virtual structure where the formation points are set as minima in an artificial potential field, becoming followers to the virtual leader (located at the center of the virtual structure). Additionally, collision avoidance is introduced to the formation using a repulsive force around the object. The virtual structure flight trajectory is induced using artificial potential fields with a desired minimum as the waypoint/goal. The results of the work indicated the UAV displayed some oscillatory motion when collision avoidance gains were too weak or when the UAV moved at higher velocities.

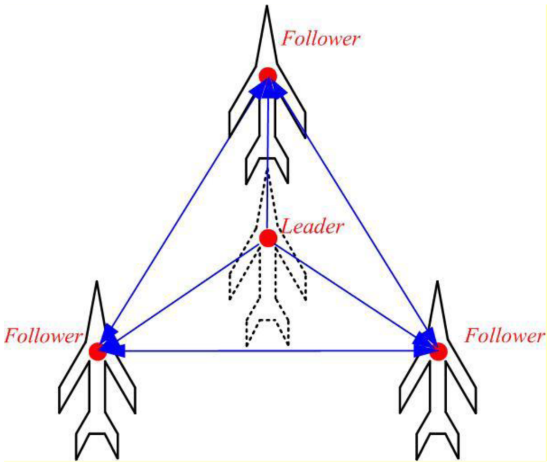


Figure 2.2. Example of virtual leader (virtual structure)/leader-follower formation approach presented by the authors in [4]. Each UAV has an estimate of the leader's position and bases their position in the formation on the leader's position.

In [2], the authors take a decentralized behavioral-based approach to maintain formation and maneuvering with differential drive UGV (Unmanned Ground Vehicle). The formation is maintained by defining a given formation relayed to each UGV and defining the dynamics of a single agent based on coupling between neighboring agents. In this work, three control strategies are explored, where formation between agents is maintained using:

1. Relative position and velocity between neighboring agents.
2. Relative position of neighboring agents with introduction of damping.
3. Relative position and velocity of neighboring agents with damping and complete actuator saturation of the UGV to achieve zero error at the goal position.

Without incorporated damping, the UGV had problems with oscillation.

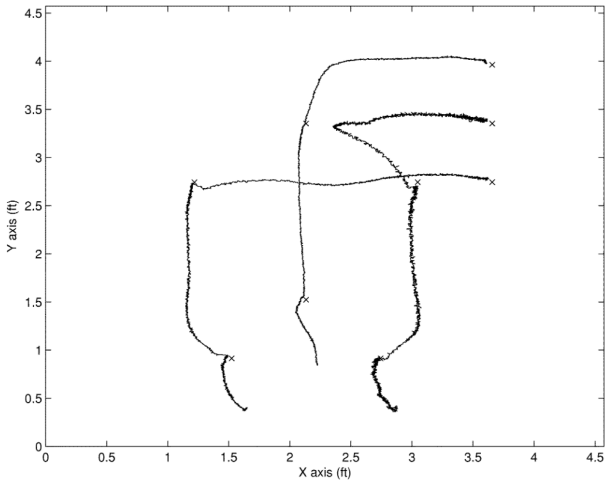


Figure 2.3. Differential drive robot positions for coupled dynamics with the input saturation control strategy studied by the authors in [2]. The X's indicate the desired waypoints for the formation to reach.

In [5], with a focus on complete decentralization with formation maintenance for differential drive robots, formation maintenance is achieved through behavioral techniques with a leader-follower approach to each neighbor. To ensure formation changes are possible despite the lack of knowledge passed from other agents, formation assignment is determined independently with given knowledge of the mission waypoints and formation.

In [6], behavior-based formation maintenance was achieved for military UGV applications as part of a DARPA (Defense Advanced Research Projects Agency) project. This work explores the various types of formation maintenance referencing:

- Unit-center-referenced
- Leader-referenced
- Neighbor-referenced

These formation reference structures are compared for formation maintenance performance with the other behaviors (avoid-static-obstacle, move-to-goal, and avoid-robot).

The work indicates that unit-centered-referenced formation performs best overall with certain situations providing an edge to leader-referenced formations, where it performed better in:

- $90\,\mathrm{deg}$ turns in a wedge or single-row formation.
- Crossing an obstacle field in single-file column formation.

However, it is indicated that certain cases rule out the use of unit-center-referenced formation.

- For a complete lack of communication, neighbor-referenced formations are less computationally intensive. Requiring less identification and computation of each UGV, needing to only track the leader.
- With communication-restricted applications, bandwidth requirements can be severely reduced (up to 70%) using a leader-referenced approach since only the leader needs to broadcast information.

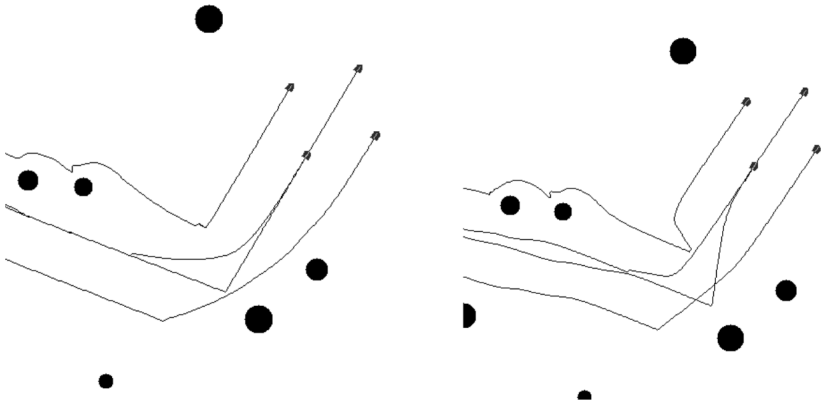


Figure 2.4. (left) UGV in diamond formation with leader reference making a $90\,\mathrm{deg}$ turn in obstacle filled environment. **(right)** UGV in diamond formation with unit-center reference making a $90\,\mathrm{deg}$ turn in obstacle filled environment. [6]

In [7], 16+ differential drive robots maintain formation and get around obstacles to reach desired goal points using behavioral methods. Formation maintenance is done using a neighbor-referenced approach. Inter-agent collision avoidance, obstacle avoidance, wall-following, and goal-reaching behaviors are weighted position-based behaviors. Where the magnitude of the behavior linearly adjusts based on the distance in between boundary conditions set. The output commanded velocity magnitude from each behavior is set with a weight. The directions of each behavior are defined based on the task:

- Goal-seeking behavior.
    - Move towards the goal.
- Obstacle avoidance behavior.
    - Move perpendicular to the straight line path between the robot and the stationary obstacle.
- Wall-following behavior.
    - Move perpendicular to the straight line path between the robot and the stationary obstacle.
- Robot collision avoidance behavior.
    - Move away from the robot colinear to the straight line distance between the robots.
- Formation maintenance behavior.
    - Move towards the formation position relative to the neighbor, oriented towards the goal position.

To ensure success in obstacle avoidance, the weights in formation maintenance are reduced to provide greater importance to obstacle avoidance. The result when simulating the formation traveling through various complex obstacle-filled environments, the swarm is successful in going around obstacles and returning to the rigid formation.

Another topic to consider when getting into formation is how to assign UAV to specified formation points without forcing collisions or taking significant time to settle into the formation positions.

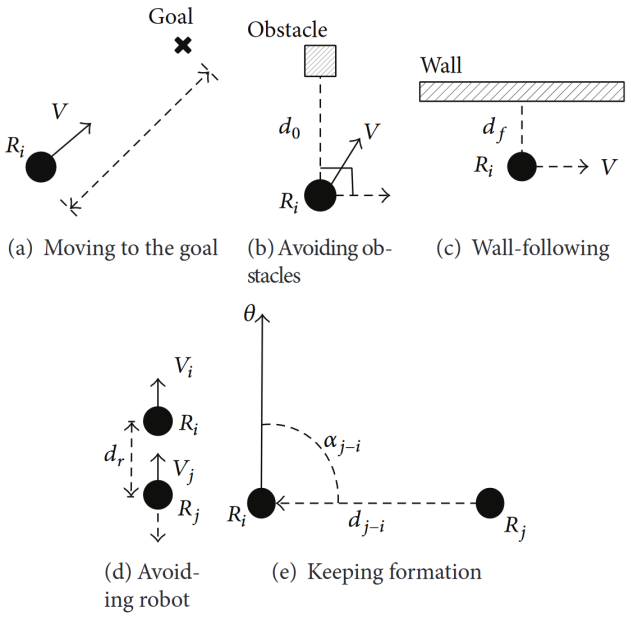


Figure 2.5. Visualization of each respective behavior presented by the authors in [7]. In each example "V" indicates the resultant velocity vector direction observed. **(a)** An example of the goal-seeking behavior, where the behavior vector is directed towards the goal position. **(b)** An example of the obstacle avoidance behavior, once the robot is at a minimum distance (d_o) from an obstacle, the behavior will direct the robot 90from the obstacle. **(c)** An example of the wall-following behavior, where at a defined distance d_f the robot is directed to move parallel to the wall. **(d)** An example of the robot collision avoidance behavior, where once the two robots are within a defined distance (d_f), the behavior directs each robot to move in the opposite direction of the opposing robot. **(e)** An example of the formation maintenance behavior, where the robot defines its position to maintain based on the formation's heading (α_{j-i}) and the relative distance to its neighbor (d_{j-i}).

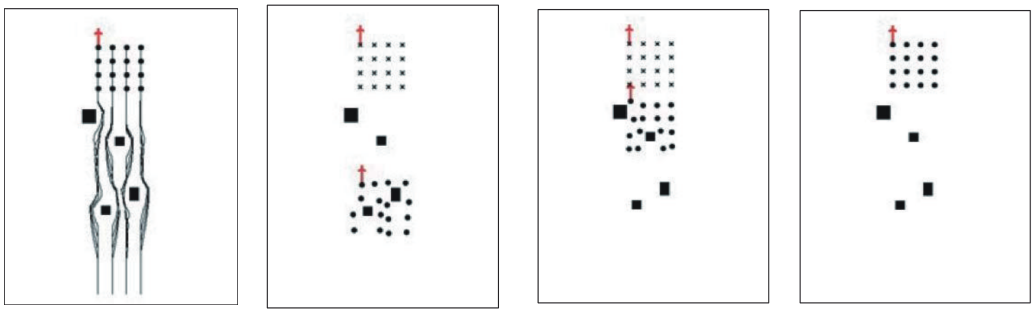


Figure 2.6. Visualization of obstacle avoidance performance while also maintaining formation using behavioral methods presented by the authors in [7].

In [5], formation is assigned through a cost table. Information on all other robot positions is known. The formation position is based on a selected leader, which is the robot with no

other robots in front of it towards the direction of the goal. Then all other robots perform their distance cost table calculation, giving themselves the position they should take and the robot it should follow in the single lead-neighbor-referenced formation.

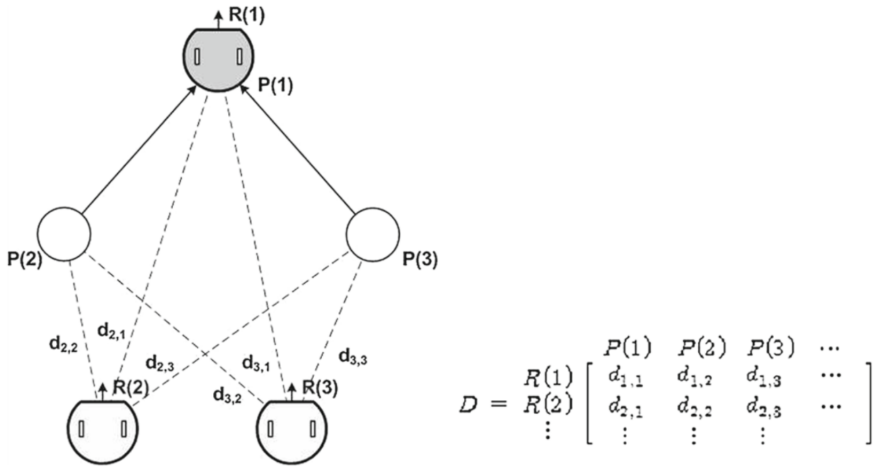
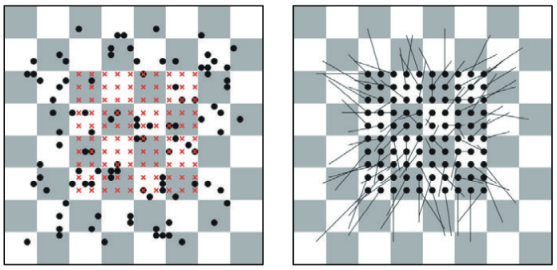


Figure 2.7. Example of development of distance cost table presented by the authors in [5]. $R(1)$ is considered the reference and leader for the formation. All other differential drive robots determine their relative distances to all available formation positions, filling in the cost table as presented **(right)**. Based on this cost table, the ideal assignment of positions is determined.

In [7], formation formulation is also done with a distance cost table and is expanded upon to improve formation formulation with large numbers of agents. This is done by classification, where the area covered by all agents and the formation is separated into cells, which are then used to classify the conditions they fall into.

The result of using this classification-based formation assignment algorithm versus the pure distance cost table is improved computational performance in all cases of formation sizing (20 - 200 agents) and formation shape (dense circle and dense square). The time taken to complete the formation position assignment calculation depends on the formation shape, formation density, and number of agents.



(a) The initial state (b) The final state after transformation

Figure 2.8. Classification-based formation assignment for a large number of differential drive robots proposed by the authors in [7].

There exists a variety of solutions for formation flight that each best suits their respective design application. Every task/application has differing priorities when maintaining formation that cannot be easily applied to every other task, underscoring the importance of tailoring these strategies to suit the specific requirements of each task/application. In applications requiring strict cooperative control, a large number of agents, and access to the ground station benefit from a centralized control structure. But in designing multi-role agents with formation tasks in mind, decentralization with cooperation is required.

By building upon the foundations laid out by previously discussed works, this thesis seeks to expand the capabilities of multi-agent UAV systems for critical scenarios. It is why for this thesis we look at the application of decentralized formation flying in surveying tasks during emergencies.

Section 3. Motivation

UAV swarms with flexible control architectures allow for the accomplishment of various tasks utilizing a single system. In an emergency situation, such flexibility can prove to be useful and even vital in some cases.

UAVs have continued to make an impact with regard to emergency response, such as surveillance of large areas to find people in danger, gathering situational awareness information during a disaster such as a flood or fire, easily deployable emergency cellular networks for improved coverage, etc.

Having a drone swarm at the disposal of first responders could be tremendously beneficial for developing situational awareness; infrastructure status, passable routes, stranded individuals, etc. But, with a centralized control system, each UAV is beholden to the centralized controller for all commands. This control structure provides less flexibility and possesses a single point of failure. Whereas introducing a decentralized control system architecture, each UAV possesses information to perform the commanded tasks from ground control, thus less affected by failures of a single UAV in the formation.

To have a swarm of UAVs of a single physical platform accomplishing various mission tasks, the UAVs would follow a hierarchical system, providing flexibility between missions such as surveying, cell-tower repeater ([8] [9]), or emergency kit delivery ([10]). Where the planning layer covers task planning and task execution monitoring. The management layer takes the assigned tasks and properly allocates the control behaviors. Finally, the control layer applies the control behaviors and feeds it through the UAV attitude/altitude controller.

Mission Scenario
Planning Layer
Flight Management Layer
Control Layer
Sensors and Actuators

Table 3.1. Layered architecture of UAV system [10].

This thesis aims to contribute the use of behavioral techniques with quadrotor UAVs for the use of surveying in a decentralized formation. With the use of behavioral techniques, behavior weights can be easily adjusted for the task at hand autonomously. Additionally, its implementation is simple, and easily integrates into a hierarchical system (section 3). While this thesis focuses on the implementation of a survey mission, it remains as a piece of an overall architecture for a multi-faceted mission in an emergency situation utilizing the same UAVs to accomplish various assigned missions.

As discussed in [5], to maintain a formation, information on the other agent's states is required. But, there can be times when communication is impossible; specifically with ground control. This thesis aims to consider situations where robust communication is used to

overcome difficult communication transmission areas. This thesis makes use of a flooding MANET (Mobile Ad-Hoc Network) topology within the UAV swarm to increase robustness of networking, with an expectation of possible communication loss to ground control once entering a hazardous area.

Using this thesis as a basis, future works can extend the capability with the addition of behaviors to perform more diverse tasks/operations and introduce more mission capabilities.

Section 4. Mission Scenario Definition Adopted in the Thesis

In the imagined mission scenario, an earthquake has occurred, first responders need to quickly gather information on the ground in its current state.

To get a 3D picture of this emergency response, post-earthquake scenario, the drone formation will take advantage of photogrammetry. The camera used in this scenario is the DJI P1 [11], with a 24mm lens. The flight path taken is a single pass forth and back route as shown in fig. 4.1.

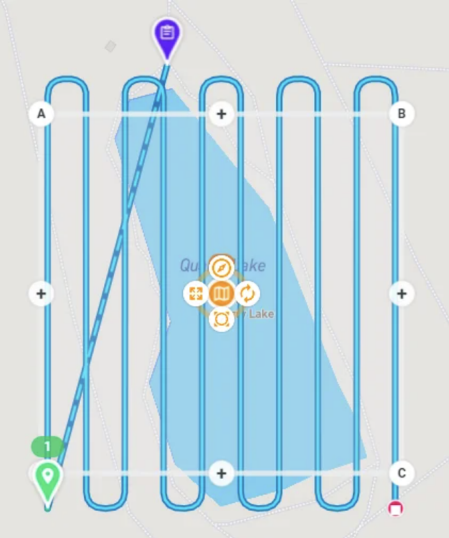


Figure 4.1. Lawnmower flight path for surveying area [12].

Additionally, there is an expectation that geolocation tags would be done using a PPK (Post Processing Kinematics) method in order to improve position data during a surveying mission.

The functional procedure for deploying the swarm is as follows:

1. The quadrotors are transported to a location for take-off.
2. The quadrotors are activated on the ground emergency base location.
3. The user inputs mission criteria to be passed to the quadrotors.
4. The information is passed to the quadrotor acting as the gate node, which is then passed to the whole group.
5. The quadrotors then get to altitude and get into the formation at their average location.
6. Then the quadrotors proceed to their goal in the assigned formation. At this point, the formation will continue with their mission as commanded.

Section 5. Approach and Methodology

To solve this problem, first, the non-linear dynamic model of a quadrotor UAV is developed, based on the DJI Matrice 300 RTK and E2000 Powertrain. The Matrice 300 RTK is used as the framework model due to its commercial availability and its design geared towards commercial industry use such as surveying and inspection [13]. To fill in missing data in powertrain information, the E2000 powertrain [14] is referenced as it is similar based on stator size, propeller pitch, and propeller diameter.

Then, a controller of a single UAV is designed with control theory tools to linearize control of the non-linear system using feedback linearization. With feedback linearization, PID controllers are used to control each axis of rotation and altitude control. This method of control is used due to its simplicity in implementation and simple integration with the behavioral planner.

Next, various behaviors are designed to integrate a single UAV into a formation of UAVs and provide the single UAV with its trajectory. The following behaviors are developed for quadrotor control and completion of the survey mission in formation. The formation maintenance behavior utilizes a virtual center approach to take advantage of improved localization of its position with respect to the formation [6].

Finally, to simulate the drones in real-time, ROS2 (Robot Operating System 2) with integration to Gazebo simulator on WSL (Windows Subsystem for Linux) was used for this thesis.

Gazebo is a physics engine with complete customizability, where simulation of sensors, motors, joints, physical interactions, dynamics, and the environment can be made. Gazebo allows for integration to ROS2, allowing for complex simulations of multiple agents in a simulation environment.

ROS2 is an open-sourced graph-structured system with various packages that can integrate with it. Fundamentally ROS2 works by having nodes that are executables running indefinitely by default, and these nodes interact with each other by sending messages between each node. These messages can be set to have a service/client interaction, where a node makes a request, and the corresponding node responds with the desired information. The messages can also be set to have a publisher/subscriber interaction, where a node constantly publishes messages at a predefined rate, and all other nodes are subscribed to this publisher. Additionally, the user may interact with the nodes through the Linux shell to pass data parameters for example.

The nature of ROS2 allows for the ability to represent control structures as intended for function in the physical world.

In conclusion, the integration of analytical dynamics and control design methods on powerful computational platforms is employed to execute control actions and generate visualizations.

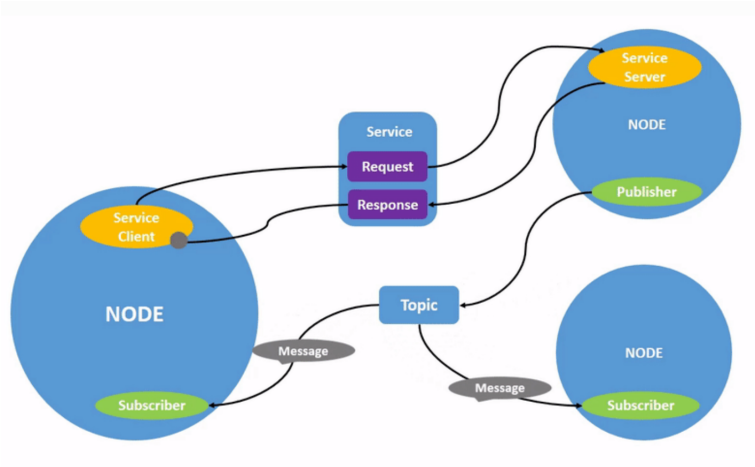


Figure 5.1. Example of ROS2 nodes communicating through service/client and publisher/subscriber topics.

Section 6. Dynamic Modelling and Controller Design for the UAVs

6.1 Simulation Environment

The Gazebo world is initialized in the default world (single floor plane) with gravity, air resistance, and the ECEF (Earth-Centered, Earth-Fixed) coordinates arbitrarily set to Los Angeles, California.

As in the real world, a local inertial frame is used with ENU (East-North-Up) coordinates (Note: ENU is utilized to keep consistency with references). This local inertial frame ECEF coordinate is provided to Gazebo in the world description.

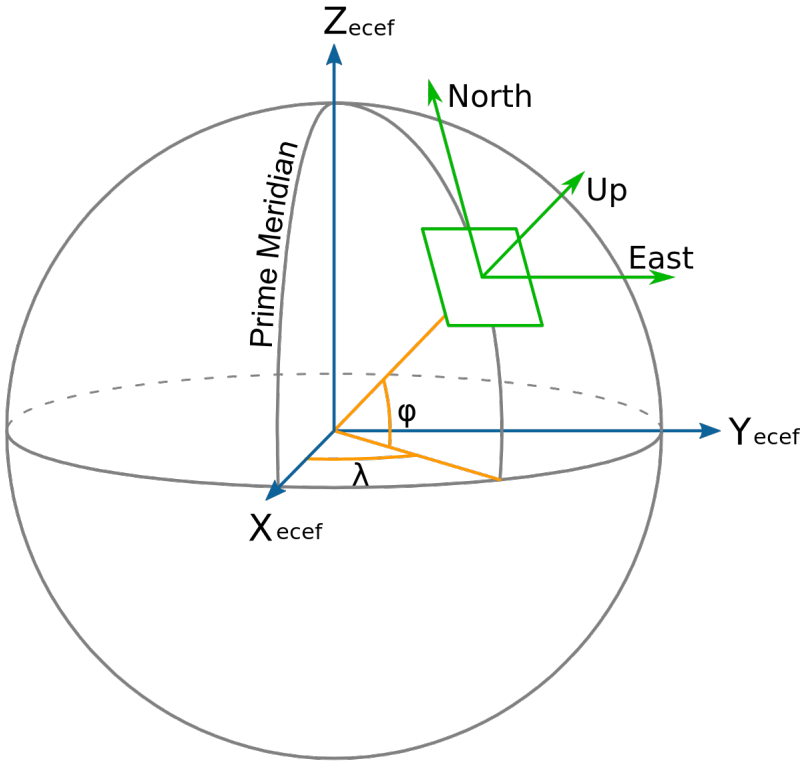


Figure 6.1. Relation between ECEF and ENU coordinates [15]

6.2 UAV Quadrotor Reference: Matrice 300 RTK

The model uses the “MATRICE 300 RTK” commercial UAV as a reference for dimensions, mass, etc. (see figs. 6.2 and 6.3).

6.2.1 Space Model of the UAV

The 3D model provided by DJI of the Matrice 300RTK is used along with the specification data to attain pertinent data (mass and inertia). The model only provides material volumetric data while all electronics are missing from the model (fig. 6.2).

To accommodate the model for the application of material properties, modifications were made to the model to properly estimate the mass distribution (fig. 6.3). The main quadrotor body was changed to a simplified body that had the same height as the batteries since the batteries make up a majority of the mass in the main body of the quadrotor. The overall shape of the simplified main body was maintained from the platform of the lower body connecting the 4 arms; assuming the mass distribution would be even across the body. Lastly, the rotor positions in the model were adjusted to ensure an equal distance to the UAV’s center of gravity and equidistant angular positioning along the circular perimeter of the UAV’s diagonal wheelbase.

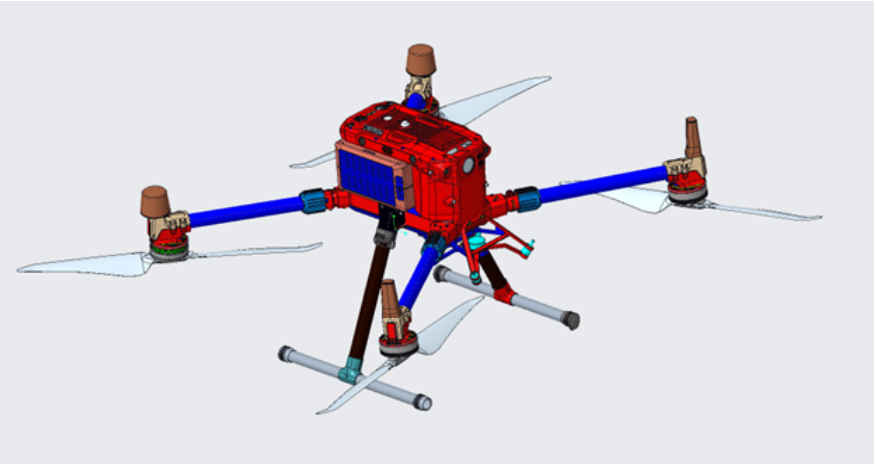


Figure 6.2. 3D model of the DJI Matrice 300RTK provided by DJI [16]. The model cannot be directly used for analysis due to errors with some incomplete surfaces, many missing internal components, and edits to mounting locations to the body, requiring simplifications/changes to the model.

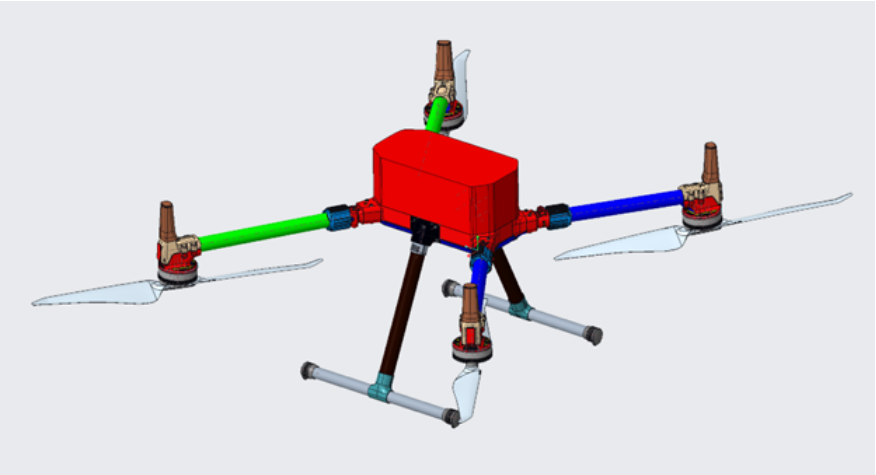


Figure 6.3. Simplified 3D model of the DJI Matrice 300RTK, derived from the downloaded model shown in fig. 6.2. With this model, material properties are applied to determine inertial data.

6.2.2 Powertrain Reference

To estimate information on the propeller powertrain, specifications were extrapolated from E2000, a stand-alone powertrain system from DJI [14], since an insufficient amount of powertrain data was provided for the Matrice 300RTK (refer to [13]). The reasoning behind selecting the E2000 specification was due to the fact that the stator was similar in size (60 mm diam. X 10 mm [Matrice 300RTK] versus 60mm diam. X 9mm [E2000]) and the propeller had similar diameters despite a difference in pitch (21 x 7.0 inch [533 x 178 mm] [Matrice 300RTK] versus 21 x 10 inch [533 x 254 mm] [E2000]). As discussed by the author in [17], the stator volume is a reliable indicator of the torque output of a motor.

6.2.3 Moments of Inertia and Mass

Calculation of the moment of inertia was done by taking the model of the M300RTK drone and specifications of the overall mass and applying material properties to the model. While also supplementing powertrain data with the E2000 powertrain mass data [14].

Material Name	**Material Density**	**Drone Components**
Cyanate Ester Carbon Composite	1.64×10^{-3} kg/m^3	Landing gear tubes and arms to powertrain
Stainless Steel (Austenitic)	7.9×10^{-3} kg/m^3	Screws on main body
Aluminum (6061 T6)	2.7102×10^{-3} kg/m^3	Powertrain screws, washers and motor enclosure
Silicone Rubber	1.12×10^{-3} kg/m^3	Seals in powertrain assembly
ABS	1.04×10^{-3} kg/m^3	All complex geometric parts
Neodymium Magnet (NdFeB) [18]	7.5×10^{3} kg/m^3	Powertrain stator magnets

Table 6.1. Material densities for the drone 3D model sourced from PTC Creo software's pre-installed library. Magnet material density referenced from [18].

E2000 Powertrain Mass Properties	
Total Powertrain Weight	430 g
Motor Weight	230 g
ESC Weight (with cables)	90 g
Propeller Weight	58 g

Table 6.2. Powertrain weight data referencing E2000 powertrain [14].

Inertia Tensor of Quadrotor [$kg \cdot m^2$]			
$I_{xx},\ I_{xy},\ I_{xz}$	1.9978×10^{-1}	-8.7175×10^{-3}	-4.3907×10^{-4}
$I_{yx},\ I_{yy},\ I_{yz}$	1.9978×10^{-1}	-8.7175×10^{-3}	-4.3735×10^{-4}
$I_{zx},\ I_{zy},\ I_{zz}$	-4.3907×10^{-4}	-4.3735×10^{-4}	3.5833×10^{-1}

Table 6.3. Inertia tensor determined from PTC Creo analysis.

Rotor Rotational Inertia [$kg \cdot m^2$]	
J_{zz}	1.98×10^{-1}

Table 6.4. Inertia determined from PTC Creo analysis.

6.2.4 Propeller Coefficients and Forces

With the data from the E2000 powertrain specifications [14], the experimentally derived data is used to determine the relation between the motor angular speed to the torque and thrust. The thrust is given explicitly by the data, but the torque is derived from the motor efficiency and input power data (fig. 6.4). The resultant relation between the motor angular velocity to the applied torque and thrust can be applied to the simulation model based on the commanded motor angular speed from the controller.

In addition to the forces and torques applied to the model, to get the commanded motor angular speed from the quadrotor's state space, the coefficients of thrust and torque are needed (see eq. (52)). Therefore, taking the experimentally derived relations of motor angular speed to thrust and power fig. 6.4, the coefficients of thrust and torque are derived.

Generally, as explained by the authors in [19], the rotor blade experiences aerodynamic lift and drag. For the purpose of this thesis, taking the experimental data, lift and drag are inherent to the data, allowing us to work without explicitly determining the coefficient of lift or drag.

For a single rotor, thrust is produced along its axis of rotation, and overall torque is in the direction of rotor motion. To relate the motor angular speed to the rotor thrust and torque, equations from [20] are used [eqs. (1) and (2)] The coefficient of thrust (C_T) and coefficient of torque (C_Q) are aerodynamic coefficients determined through experimental data.

$$F_i = C_T \, \rho \, A \, (\Omega_i R)^2 \tag{1}$$

$$T_i = C_Q \, \rho \, A \, (\Omega_i R)^2 \, R \tag{2}$$

where,

C_T is the coefficient of thrust
C_Q is the coefficient of torque
ρ is the air density
A is the circular area swept by the rotor blade
R is the radius of the of the circular area swept by the rotor blade
Ω is the angular velocity of the rotor in rad/sec

Looking at the relations for thrust, data provided on the specification of the E2000 powertrain fig. 6.4 gives the relation between rotor angular velocity and thrust per rotor shown in fig. 6.6 and eq. (4). To get the coefficient of thrust, the relation in eq. (1) is used.

The following data is used to determine the coefficient of thrust:

Propeller Radius [R] [13]	$0.266\,701 \;\; m$
Propeller Sweep Area [A]	$0.223\,459 \;\; m^2$
Air Density [ρ] [21]	$1.2250 \;\; kg/m^3$

Table 6.5. Input data constants in order to determine force and torque relations of the powertrain system in standard air conditions.

Taking eq. (1), and inputting data from fig. 6.4 (thrust (T) and angular velocity (Ω)) and table 6.5, the data in fig. 6.5 is produced.

Two linear curve fit lines (fig. 6.5) are used for determining the coefficient of thrust during control of the quadrotor (see eqs. (3) and (52)). Taking the most recent recorded motor angular velocity [Ω_{t-1}], the thrust coefficient can be determined for the current control loop iteration by inputting into eq. (3). Thus, giving a motor speed command to be passed to the simulation model.

Since we have the relation of motor angular speed to rotor thrust (fig. 6.6), the force using the experimental data (eq. (4)) can be applied to the simulation model.

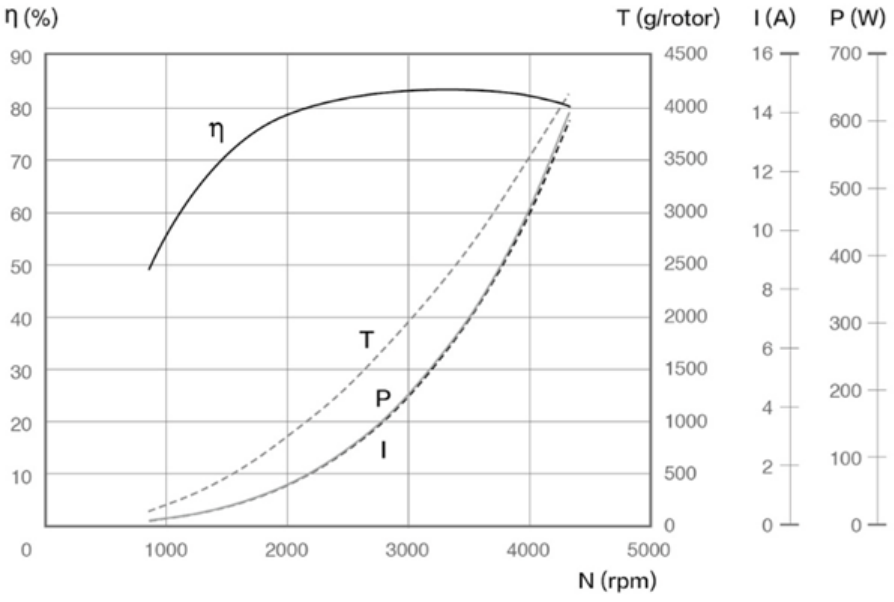


Figure 6.4. DJI E2000 powertrain specifications provided by the DJI product specification page [14]. Data points for motor efficiency, input power, and thrust were estimated from this figure.

$$C_T = \begin{cases} -2.43658 \times 10^{-5}\ (\Omega_i)\ +\ 0.014033693 & \text{for } \Omega_i\ \epsilon\ [104.7198, 169.263) \\ 7.60634 \times 10^{-7}\ (\Omega_i)\ +\ 0.009780691 & \text{for } \Omega_i\ \epsilon\ [169.263, 445.059] \end{cases} \tag{3}$$

$$F_i\ =\ 0.000210795\ (\Omega_i^2)\ -0.008276388\ (\Omega_i)\ +\ 1.006862 \tag{4}$$

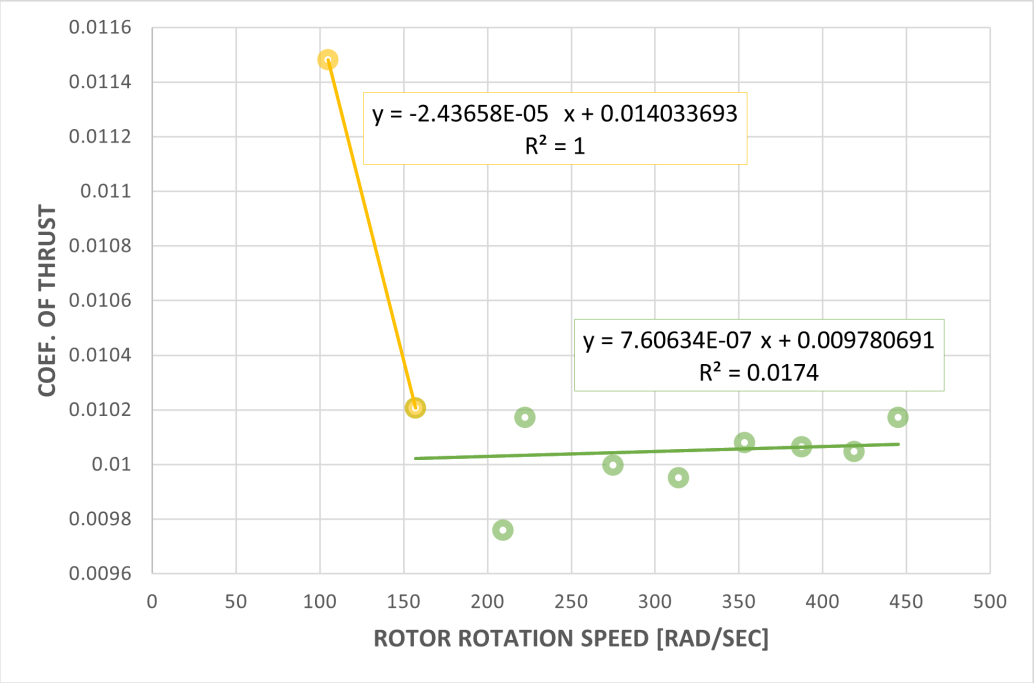


Figure 6.5. Piece-wise curve fit functions used to estimate the coefficient of thrust at a specific rotor angular speed (refer to eq. (3)).

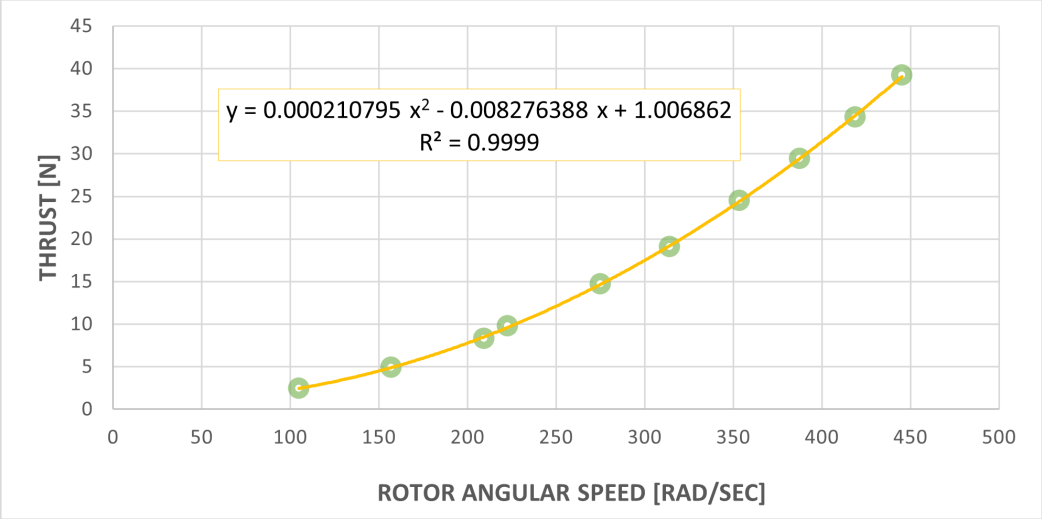


Figure 6.6. Curve fit function used to determine the thrust for a specific rotor angular speed (refer to eq. (4)).

As for the applied rotor torque, since torque data isn't given directly in relation to motor speed, it is derived using the relation from eq. (5) and eq. (6) [20] with the data provided from the specification sheet in [14]. Referencing fig. 6.4, output power is derived by eq. (7).

As it was for thrust, the torque coefficient is needed in eq. (52) to determine the commanded motor speed. Taking the calculated output power (eq. (7)) and solving for the coefficient of power (eq. (6)) gives the coefficient of torque due to the relation in eq. (5) described by the authors in [20]. The result is shown in fig. 6.7, where two linear curve fit lines are used for determining the coefficient of torque during control of the quadrotor (see eqs. (8) and (52)). Taking the most recent recorded motor angular velocity [Ω_{t-1}], the torque coefficient can be determined for the current control loop iteration by inputting into eq. (8). Thus, giving a motor speed command to be passed to the simulation model.

To then develop a relation between the motor angular speed to rotor torque, the previously calculated coefficient of torque is plugged into eq. (2) using constants from table 6.5. With the derived relation of motor angular speed to rotor torque (fig. 6.8), the torque (eq. (9)) can be applied to the simulation model.

$$C_P = C_Q \tag{5}$$

where,

C_P is the coefficient of output power

C_Q is the coefficient of the rotor shaft torque

$$\begin{aligned} C_Q &= \frac{T_i}{\rho\ A\ (\Omega_i R)^2 R} \\ C_P &= \frac{P_i}{\rho\ A\ (\Omega_i R)^3} \end{aligned} \tag{6}$$

where,

P_i is the output power for a single rotor

T_i is the shaft torque of a single rotor

$$P_{out} = P_{in}\ \eta_{motor} \tag{7}$$

where,

P_{in} is the input power given in fig. 6.4 for each motor

η_{motor} is the motor efficiency given in fig. 6.4

P_{out} is the output power of the motor

$$C_Q = \begin{cases} 5.2928 \times 10^{-6}\ (\Omega_i)\ +\ 0.000202 & \text{for } \Omega_i\ \epsilon\ [91.6298, 154.071) \\ -2.943\,18 \times 10^{-8}\ (\Omega_i)\ +\ 0.001015 & \text{for } \Omega_i\ \epsilon\ [154.071, 453.24] \end{cases} \tag{8}$$

$$T_i = 4.94068 \times 10^{-6} \left(\Omega_i^2\right) + 0.000182 \left(\Omega_i\right) - 0.02804 \tag{9}$$

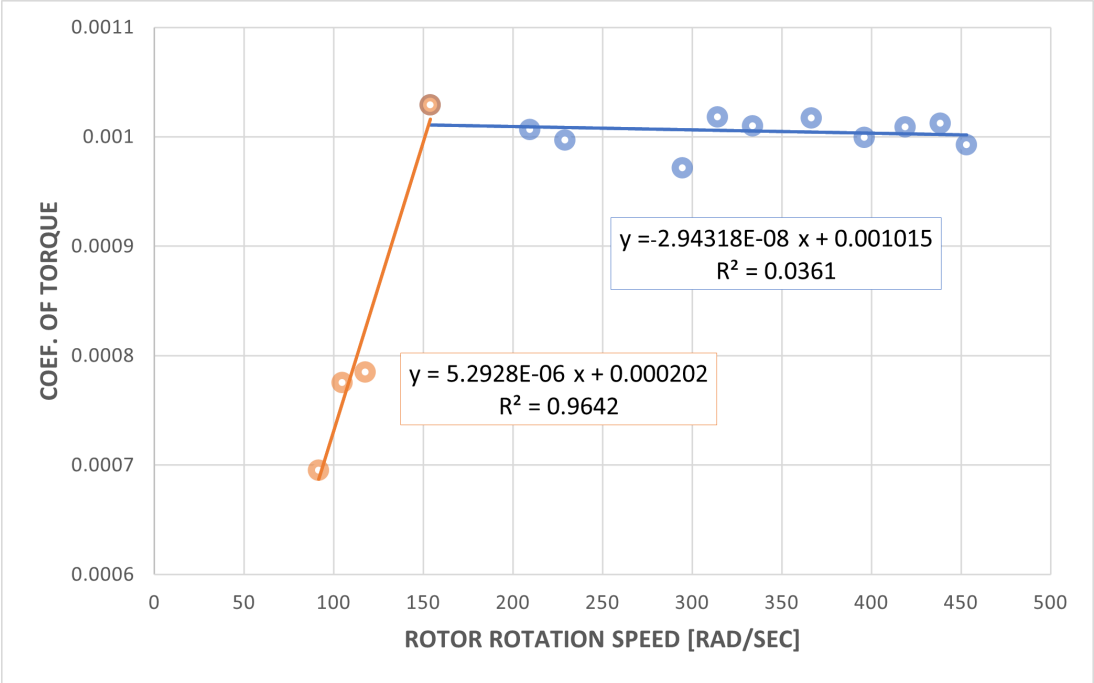


Figure 6.7. Piece-wise curve fit functions used to estimate the coefficient of torque at a specific rotor angular speed eq. (8)).

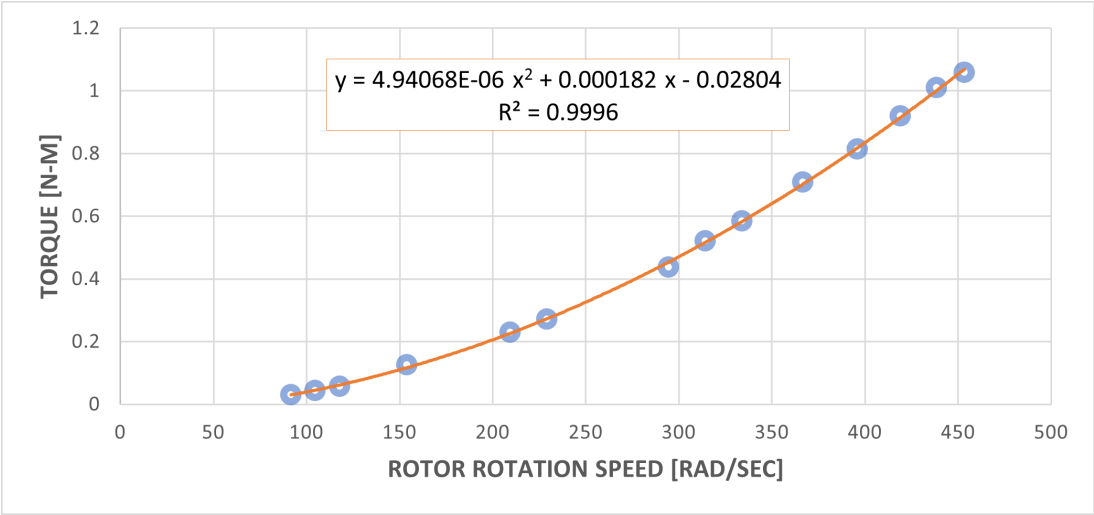


Figure 6.8. Curve fit function used to determine the torque for a specific rotor angular speed eq. (9)).

6.3 UAV Quadrotor Modeling

The quadrotor model is based on Newton-Euler methods, with the assumptions:

1. The quadrotor is simplified to be symmetric.
2. The quadrotor is rigid.
3. The quadrotor center of gravity is defined at the origin of the body-fixed frame.

6.3.1 Kinematic Model

To define the quadrotor's position in space, two coordinate frames are established:

1. The EF (Earth-Fixed) [aka inertial] frame.
2. The BF (Body-Fixed) frame.

The EF frame is defined at the location specified in the Gazebo world, which for this thesis is arbitrary.

The BF frame is defined along the arms of the quadrotor, with motor M_1 lying on the *x-axis* of the BF frame and motor M_2 lying on the *y-axis* of the BF frame. The BF frame *z-axis* points toward the sky.

The position of the quadrotor as defined by the EF frame to the BF frame is defined as $\zeta = [X \quad Y \quad Z]^T$.

The rotation matrix eq. (10) relates the orientation from the BF frame to the EF frame following the Tait-Bryan convention. Where orientation of the quadrotor is expressed in the EF frame as $\eta = [\phi \quad \theta \quad \psi]^T$ about the x, y, and z-axis respectively. This rotation matrix is used in the formulation of the dynamics model of the quadrotor.

$$R = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta c\psi & s\phi s\theta s\psi + c\theta c\psi & c\phi s\theta c\psi - s\theta c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{10}$$

where, $c = cos()$ and $s = sin()$

The angular velocities about the BF frame are expressed as Tait-Bryan angle rates with respect to the EF frame, which is shown in eq. (11) [19], [22].

Since the transformation matrix W_η is not orthogonal, the inverse is used to get the angular rates w.r.t. EF frame ($\dot{\eta}$) from the angular rates w.r.t. BF frame (ω), as shown in eq. (12) [22].

$$\omega = W_\eta \, \dot{\eta}; \quad \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{11}$$

where,

$\omega = [p \quad q \quad r]^T$ is the vector of the angular rates in the BF frame
W_η is the transformation matrix from the EF frame to the BF frame
$\dot{\eta} = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T$ is the vector of the Euler rates in the EF frame
$\phi, \quad \theta, \quad \psi$ are the Euler angles relating the orientation of the BF frame to the EF frame

$$\dot{\eta} = W_\eta^{-1} \, \omega; \quad \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{12}$$

The rotation matrix in eq. (11) can be simplified using small angle approximation when considering UAV flight to take place around hover position. So, $\cos\phi = \cos\theta \equiv 1$, $\sin\phi = \sin\theta = 0$, which reduces the rotation matrix in eq. (11) to I, as seen in eq. (13). This will allow for the conversion of angular body rates (ω), which is collected from the IMU (Inertial Measurement Unit), into the EF frame as inertial body rates ($\dot{\eta}$).

$$\omega = I \, \dot{\eta} \tag{13}$$

Because of this simplification, $\int \omega \, dt = \eta$ and the quadrotor orientation in the BF frame can be described as $\eta = [\phi \; \theta \; \psi]^T$.

6.3.2 Quadrotor Method of Flight

A quadrotor is an inherently non-linear system due to the fact that the linear motion of the quadrotor is coupled to its rotational motion. The four motors in a quadrotor must be controlled to maintain position and orientation in all six degrees of freedom.

Each rotor produces a thrust force F_i that can be manipulated to translate along the x,y, and z axes and to roll and pitch (see fig. 6.10). Additionally, the rotors exert a torque T_i that allows for yaw control (see fig. 6.10).

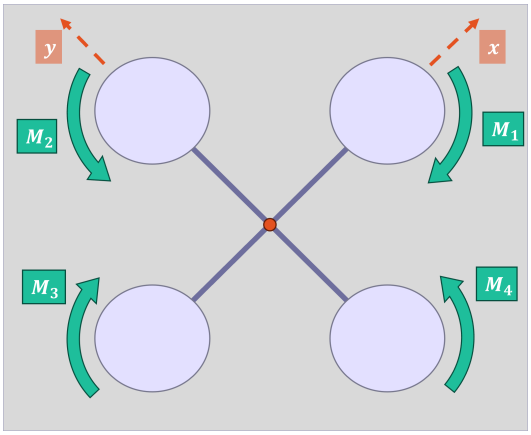


Figure 6.9. Motor labeling assigned from 1-4 counter-clockwise from the motor on the positive x-axis

To maintain altitude (Z), all motors (M_1 to M_4 [fig. 6.9]) must produce thrust in the EF frame's *z-axis*. When the quadrotor is level to the ground ($\phi = \theta = 0$), the thrust is equal from all motors with a resultant upward force (refer to *(a)* in fig. 6.10).

$$F_z = F_1 + F_2 + F_3 + F_4 \tag{14}$$

where,

F_i is the thrust force produced by each rotor

In this thesis, modeling of the quadrotor dynamics assumes that moments are taken about the BF frame's axes, which coincide with the rotors.

To control the roll angle (ϕ) of the quadrotor, motors M_1 and M_3 must inversely produce thrust to produce a moment about the BF frame's *x-axis* (refer to *(b)* in fig. 6.10).

When the quadrotor adjusts its roll angle, it will move along the BF frame's *y-axis*.

$$M_x = -F_2 l + F_4 l \tag{15}$$

where,

F_i is the thrust force produced by each rotor
l is the length from the origin of the BF frame to the axis of rotation of the rotor

To control the pitch angle (θ) of the quadrotor, motors M_2 and M_4 must inversely produce thrust to produce a moment about the BF frame's *y-axis* (refer to *(c)* in fig. 6.10).

When the quadrotor adjusts its pitch angle, it will move along the BF frame's *x-axis*.

$$M_y = F_1 l - F_3 l \tag{16}$$

where,

F_i is the thrust force produced by each rotor

l is the length from the origin of the BF frame to the axis of rotation of the rotor

To control the yaw angle (ψ) of the quadrotor, motors M_1 and M_3 together must produce a torque inverse to motors M_2 and M_4 together. Since each motor produces a torque (counterclockwise for M_1 and M_3, clockwise for M_2 and M_4 [fig. 6.9]), the overall effect is a torque about the BF frame's *z-axis* (refer to *(d)* in fig. 6.10).

$$M_z = -T_1 + T_2 - T_3 + T_4 \tag{17}$$

where,

T_i is the torque produced by each rotor

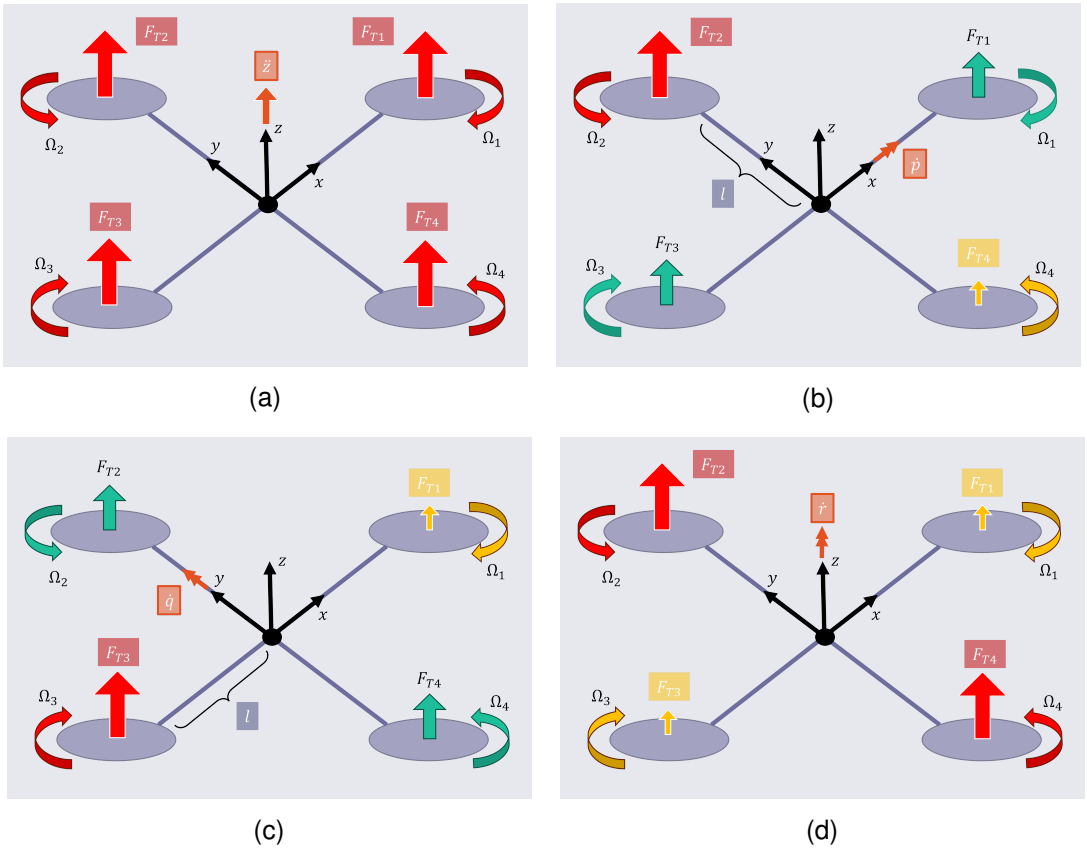


Figure 6.10. (a) All motors exert an upward thrust to control altitude. **(b)** *Motor 2* is exerting greater thrust than *Motor 4*, to produce a positive moment about the *x-axis*. **(c)** *Motor 3* is exerting greater thrust than *Motor 1*, to produce a positive moment about the *y-axis*. **(d)** *Motors 2 and 4* are producing greater torques than *Motors 1 and 3*, resulting in a positive moment about the *z-axis*.

6.3.3 Dynamic Model

The equations of motion of the quadrotor are derived for attitude and translation respectively.

Attitude Dynamic Model

The rotational equations of motion are derived in the body frame using the Newton-Euler method, as shown in eq. (18) [19]. Additionally, for the purpose of this thesis, moments due to aerodynamic effects in the model are neglected. It is assumed that aerodynamic conditions are nominal for the purpose of simulating the quadrotor.

$$J\dot{\omega} + \omega \times J\omega + M_G = M_B \tag{18}$$

where,

J is the quadrotor's inertia matrix.
ω is the vector of the angular rates of rotation in the BF frame
M_G is the vector of gyroscopic moments due to the inertia of the rotors
M_B is the vector of the overall moments acting on the quadrotor in the BF frame

The inertia matrix of the quadrotor model is a diagonal matrix from the assumption that the quadrotor is symmetric [19].

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{19}$$

where,

$I_{xx},\ I_{yy},\ I_{zz},$ are the moments of inertia about the principle axes of the BF frame

The gyroscopic moments due to the rotor inertia can be defined as:

$$M_G = \omega \times [0,\ 0,\ J_t\Omega_t]^T \tag{20}$$

where,

J_t is the sum of z-component inertias of all of the quadrotor's rotating masses (a single motor about the rotating axis [refer to eq. (21) and fig. 6.9])
Ω_t is the sum of the motor's angular velocity considering counter-clockwise rotation as positive (refer to eq. (22) and fig. 6.9)
ω is the vector of the angular rates of rotation in the BF frame

$$J_t = J_{M1} + J_{M2} + J_{M3} + J_{M4} \tag{21}$$

$$\Omega_t = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \tag{22}$$

As described in section 6.3.2, the propellers create positive thrust force F_i along the axis of rotation, and by adjusting the thrust of each rotor, moments about the BF frame induce rolling (ϕ) and pitching (θ) motion. Additionally, the applied torque from all rotors can induce a moment about the yaw axis (ψ).

Taking the moment components about the BF frame, referencing eqs. (15) to (17) we get eq. (23).

$$M_B = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \tag{23}$$

Then, a relation is derived from the powertrain experimental data presented in section 6.2.4 to the motor speeds Ω_i for our model. Expanding eq. (23) to get everything in terms of Ω_i, where eqs. (1) and (2) are simplified [19], [23]:

$$F_i = K_T \Omega_i^2 \tag{24}$$

$$T_i = K_Q \Omega_i^2 \tag{25}$$

where,

$K_T = C_T\ \rho\ A\ R^2$; containing all the constants found in eq. (1)
$K_Q = C_Q\ \rho\ A\ R^3$; containing all the constants found in eq. (2)

So eq. (23) expands to the following shown in eq. (26), which can be plugged back into eq. (18).

$$M_B = \begin{bmatrix} lK_T(-\Omega_2^2 + \Omega_4^2) \\ lK_T(\Omega_1^2 - \Omega_3^2) \\ K_Q(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 \end{bmatrix} \tag{26}$$

Translation Dynamic Model

The equation of linear motion (eq. (27)) is derived in the EF frame using Newton's second law as shown in eq. (27). Note that aerodynamic forces are neglected for this thesis [19].

The only acting forces on the quadrotor are the acting thrust from the rotors in the BF frame (reference eq. (28)) and gravity in the EF frame (eq. (27)).

To relate the model (eq. (27)) to the rotor speed (Ω_i), eq. (24) is expanded to eq. (29) using eq. (24).

$$m\,\ddot{\zeta} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_B \tag{27}$$

where,

m is the total mass of the quadrotor

g is the gravitational constant (earth's gravity)

$\ddot{\zeta}$ is the linear acceleration vector of the quadrotor in the EF frame

R is the rotation matrix shown in eq. (10)

F_B is the vector of the acting forces on the drone body in the EF frame

$$F_B = \begin{bmatrix} 0 \\ 0 \\ F_z \end{bmatrix} \tag{28}$$

$$F_B = \begin{bmatrix} 0 \\ 0 \\ K_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{29}$$

6.3.4 Control Input Relations for the UAV

For our state space model (section 6.3.5) and controller formulation (section 6.4), a relation between the control ouputs ($\bar{U}$) and the commanded motor speeds (Ω_i) is needed by taking the relation derived from the powertrain experimental data presented in section 6.2.4 as our model.

First, the control inputs are described in section 6.3.2, where altitude, roll, pitch, and yaw are controlled distinctly. The control vector ($\bar{U}$) is equated to eqs. (14) to (17) respectfully, as shown in eq. (30). Then a relation to Ω_i is made using eqs. (26) and (29) (reference eq. (30)).

$$\bar{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} K_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ lK_T(-\Omega_2^2 + \Omega_4^2) \\ lK_T(\Omega_1^2 - \Omega_3^2) \\ K_Q(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 \end{bmatrix} \tag{30}$$

6.3.5 State Space Model of the UAV

To control the system, a state space model is required and created, starting by presenting the state space vector of the model in eq. (31).

$$\bar{X} = [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad Z \quad \dot{Z} \quad X \quad \dot{X} \quad Y \quad \dot{Y} \quad]^T \tag{31}$$

Deriving the state equations based on rotational equations of motion in terms of the attitude control outputs ($U_2,\ U_3,\ U_4$), we start by extracting the relation eq. (32) from eq. (30) to have the same form as eq. (23).

$$M_B = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{32}$$

Next, using the relation established by eq. (13), the rotational motion eq. (18) can be re-written as:

$$J\ddot{\eta} + \dot{\eta} \times J\dot{\eta} + M_G = M_B \tag{33}$$

Now that we have eq. (32) in terms of attitude control outputs, we sub eq. (32) into eq. (33) and expand the equation, giving eq. (34).

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ J_t\Omega_t \end{bmatrix} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{34}$$

Expanding and re-writing eq. (34) in terms of angular acceleration in the EF frame gives our state equations for angular acceleration:

$$\begin{aligned} \ddot{\phi} &= \frac{U_2}{I_x x} - \frac{J_t}{I_x x}\dot{\theta}\Omega_t + \frac{I_y y - I_z z}{I_x x}\dot{\psi}\dot{\theta} \\ \ddot{\theta} &= \frac{U_3}{I_y y} - \frac{J_t}{I_y y}\dot{\phi}\Omega_t + \frac{I_z z - I_x x}{I_y y}\dot{\psi}\dot{\phi} \\ \ddot{\psi} &= \frac{U_3}{I_z z} + \frac{I_x x - I_y y}{I_z z}\dot{\theta}\dot{\phi} \end{aligned} \tag{35}$$

To get the state space representation based on the translation dynamics with the control output U_1, we start by extracting the relation eq. (36) from eq. (30) to have the same form as eq. (28).

$$F_B = \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} \tag{36}$$

Then, subbing eq. (36) into eq. (27) and expanding the equation gives eq. (37).

$$m \begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta c\psi & s\phi s\theta s\psi + c\theta c\psi & c\phi s\theta c\psi - s\theta c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} \tag{37}$$

Expanding and re-writing eq. (37) in terms of linear acceleration in the EF frame gives our state equations for linear acceleration:

$$\begin{aligned} \ddot{X} &= \frac{U_1}{m}\left(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\right) \\ \ddot{Y} &= \frac{U_1}{m}\left(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\right) \\ \ddot{Z} &= \frac{U_1}{m}\left(\cos\phi\cos\theta\right) - g \end{aligned} \tag{38}$$

Thus, the final state space representation of the UAV dynamics takes the form:

$$\begin{aligned} \dot{X}_1 &= \dot{\phi} = X_2 \\ \dot{X}_2 &= \ddot{\phi} = \frac{U_2}{I_{xx}} - \frac{J_t}{I_{xx}} X_4\Omega_t + \frac{I_{yy} - I_{zz}}{I_{xx}} X_6X_4 \\ \dot{X}_3 &= \dot{\theta} = X_4 \\ \dot{X}_4 &= \ddot{\theta} = \frac{U_3}{I_{yy}} - \frac{J_t}{I_{yy}} X_2\Omega_t + \frac{I_{zz} - I_{xx}}{I_{yy}} X_6X_2 \\ \dot{X}_5 &= \dot{\psi} = X_6 \\ \dot{X}_6 &= \ddot{\psi} = \frac{U_3}{I_{yy}} + \frac{I_{xx} - I_{yy}}{I_{yy}} X_4X_2 \\ \dot{X}_7 &= \dot{Z} = X_8 \\ \dot{X}_8 &= \ddot{Z} = \frac{U_1}{m}\left(\cos X_1\cos X_3\right) - g \\ \dot{X}_9 &= \dot{X} = X_{10} \\ \dot{X}_{10} &= \ddot{X} = \frac{U_1}{m}\left(\cos X_1\sin X_3\ cosX_5 + \sin X_1\sin X_5\right) \\ \dot{X}_{11} &= \dot{Y} = X_{12} \\ \dot{X}_{12} &= \ddot{Y} = \frac{U_1}{m}\left(\cos X_1\sin X_3\ sinX_5 - \sin X_1\cos X_5\right) \end{aligned} \tag{39}$$

6.4 Single UAV Quadrotor Control

Due to the coupling of linear motion along the x-y axes to the control of roll and pitch angles, a nested control structure is utilized for the control of the quadrotor (shown in fig. 6.11). The control structure for a single UAV consists of the position controller and the attitude controller. The behavioral controller functions as the trajectory planner (section 8.1), passing the reference velocity vector ($\dot{\zeta_R} = [\dot{X}_R\ \dot{Y}_R\ \dot{Z}_R]^T$) to the position controller. The position controller passes the reference pitch (θ_R) and roll (ϕ_R) positions (section 8.1.1) to the attitude controller while also receiving the reference yaw (ψ_R) position from the behavioral controller. Finally, all the motor speed commands ($\Omega_1,\ \Omega_2,\ \Omega_3,\ \Omega_4$) are passed to the UAV system.

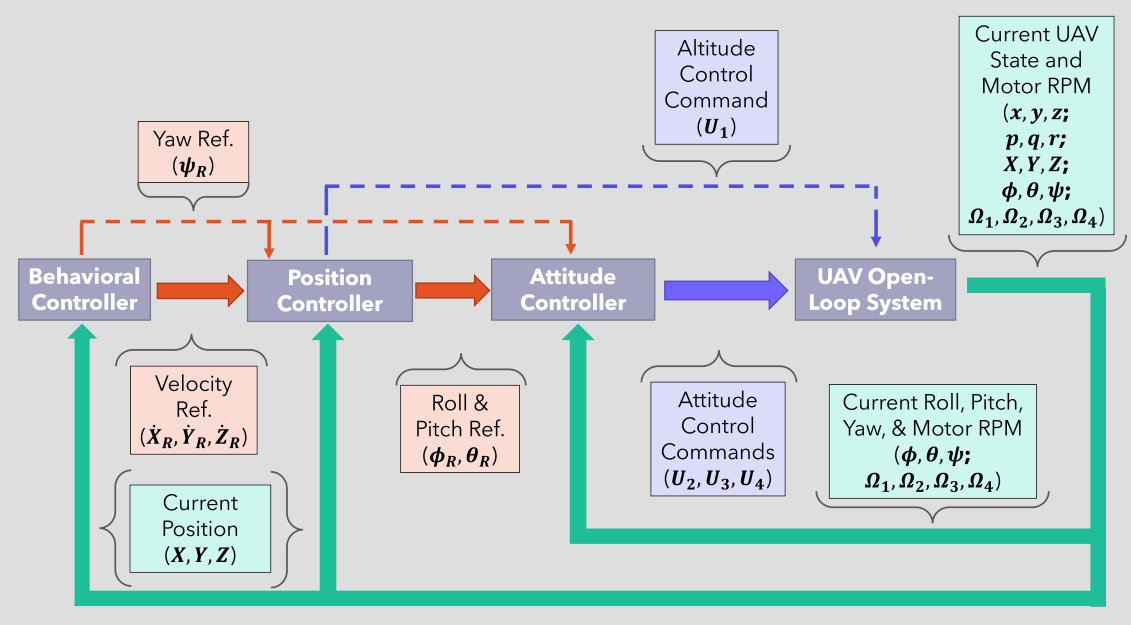


Figure 6.11. Single UAV controller flow diagram.

6.4.1 Desired Linear Acceleration to Rotational Position

To provide the attitude controller with the reference pitch (θ_R) and roll (ϕ_R) positions, a relation is established.

With state equations $\ddot{X}$ and $\ddot{Y}$ from eq. (40) a simplification can be made using small angle approximation ($\sin\phi_R \equiv \phi_R$, $\sin\theta_R \equiv \theta_R$, and $\cos\phi = \cos\theta = 1$) giving eq. (40).[19]

$$\begin{aligned} \ddot{X}_R &= \frac{U_1}{m}\left(\theta_R \cos\psi + \phi_R \sin\psi\right) \\ \ddot{Y}_R &= \frac{U_1}{m}\left(\theta_R \sin\psi - \phi_R \cos\psi\right) \end{aligned} \tag{40}$$

Writing eq. (40) into matrix form, inverting the equation, then expanding gives:

$$\begin{bmatrix} \phi_R \\ \theta_R \end{bmatrix} = \frac{m}{U_1} \begin{bmatrix} \ddot{X}_R \sin\psi - \ddot{Y}_R \cos\psi \\ \ddot{X}_R \cos\psi + \ddot{Y}_R \sin\psi \end{bmatrix} \tag{41}$$

where,

ϕ_R and θ_R are within the range of $\pm 20°$ [19]

6.4.2 Feedback Linearization

Feedback linearization for the position and attitude controller is utilized to linearize the nonlinearities of the quadrotor system. [24] [25]

$$\bar{X} = f(x) + G\,u \tag{42}$$

where,

$\bar{X}$ is the state vector
$f(x)$ is the vector of the non-linear model function
G is the control input matrix
u is the control input vector to the linearized model

Input vector u is determined by modifying the non-linearities of the model in the form shown in eq. (43).

$$u = G^{-1}(-f(x) + v) \tag{43}$$

where,

v is the linear control input vector

Then substituting eq. (43) into eq. (42) yields the following linear relation shown in eq. (44).

$$\bar{X} = v \tag{44}$$

So then extending it to the quadrotor state equations responsible for attitude and altitude control, $\ddot{Z}$, $\ddot{\phi}$, $\ddot{\theta}$, and $\ddot{\psi}$ are linearized by setting U_1 to U_4 (eq. (46)) to contain all the non-linearities as done in eqs. (42) to (44).

$$\begin{aligned}
\ddot{Z} &= \frac{U_1}{m}\left(\cos\phi\cos\theta\right) - g \\
\ddot{\phi} &= \frac{U_2}{I_{xx}} - \frac{J_t}{I_{xx}}\dot{\theta}\Omega_t + \frac{I_{yy}-I_{zz}}{I_{xx}}\dot{\psi}\dot{\theta} \\
\ddot{\theta} &= \frac{U_3}{I_{yy}} - \frac{J_t}{I_{yy}}\dot{\phi}\Omega_t + \frac{I_{zz}-I_{xx}}{I_{yy}}\dot{\psi}\dot{\phi} \\
\ddot{\psi} &= \frac{U_3}{I_{zz}} + \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi}
\end{aligned} \tag{45}$$

$$\begin{aligned}
U_1 &= \frac{\left(v_Z + g\right)m}{\cos\phi\cos\theta} \\
U_2 &= I_{xx}\left(-i_1\dot{\theta}\dot{\psi} - j_1\dot{\theta}\Omega + v_2\right) \\
U_3 &= I_{yy}\left(-i_2\dot{\phi}\dot{\psi} - j_2\dot{\phi}\Omega + v_3\right) \\
U_4 &= I_{zz}\left(-i_3\dot{\theta}\dot{\phi} + v_4\right)
\end{aligned} \tag{46}$$

where,

$$i_1 = \frac{I_{yy}-I_{zz}}{I_{xx}}\ ; \quad j_1 = \frac{J_t}{I_{xx}}\ ; \quad i_2 = \frac{I_{zz}-I_{xx}}{I_{yy}}\ ; \quad j_2 = \frac{J_t}{I_{yy}}\ ; \quad i_3 = \frac{I_{xx}-I_{yy}}{I_{zz}}$$

After establishing a linear relation as shown in eq. (49), a PID controller is set as the input to control the reformulated linear system (eq. (47))

$$\begin{aligned}
v_Z &= K_p e_Z + K_d \dot{e}_Z + K_i \int e_Z dt \\
v_\phi &= K_p e_\phi + K_d \dot{e}_\phi + K_i \int e_\phi dt \\
v_\theta &= K_p e_\theta + K_d \dot{e}_\theta + K_i \int e_\theta dt \\
v_\psi &= K_p e_\psi + K_d \dot{e}_\psi + K_i \int e_\psi dt
\end{aligned} \tag{47}$$

where,

$$e_Z = Z_R - Z\ ; \quad e_\phi = \phi_R - \phi\ ; \quad e_\theta = \theta_R - \theta\ ; \quad e_\psi = \psi_R - \psi \tag{48}$$

and

$$\ddot{Z} = v_Z\ ; \quad \ddot{\phi} = v_\phi\ ; \quad \ddot{\theta} = v_\theta\ ; \quad \ddot{\psi} = v_\psi \tag{49}$$

6.4.3 Controller Output Motor Speed Command

Now that we have our control output (refer to eq. (46)) in terms of $\bar{U}$, the output needs to be equated to motor speed commands (Ω_i).

Taking the relation established in eq. (30), the equation can then be rearranged to the following form in eq. (50). Recall that K_T and K_Q are the coefficients modeling the UAV motor

speed relation to the thrust and torque output respectively (reference eqs. (24) and (25)) and "l" is the length from the motor rotating axis to the UAV BF frame origin (reference fig. 6.10 and section 6.3.2).

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ 0 & -lK_T & 0 & lK_T \\ lK_T & 0 & -lK_T & 0 \\ -K_Q & K_Q & -K_Q & K_Q \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{50}$$

Determining the inverse of the constant matrix and moving the $\bar{\Omega}^2$ vector term to the left side of the equation gives us eq. (51).

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4K_T} & 0 & -\frac{1}{2lK_T} & -\frac{1}{4K_Q} \\ \frac{1}{4K_T} & \frac{1}{2lK_T} & 0 & \frac{1}{4K_Q} \\ \frac{1}{4K_T} & 0 & \frac{1}{2lK_T} & -\frac{1}{4K_Q} \\ \frac{1}{4K_T} & -\frac{1}{2lK_T} & 0 & \frac{1}{4K_Q} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{51}$$

Finally, taking the square roots of both sides of eq. (51) results in the following equations in eq. (52). Using the equations shown in eq. (52) we can get the output rotor speed ($\bar{\Omega}$) from the control output ($\bar{U}$).

$$\begin{aligned} \Omega_1 &= \left(\frac{U_1}{4K_T} - \frac{U_3}{2lK_T} - \frac{U_4}{4K_Q} \right)^{\frac{1}{2}} \\ \Omega_2 &= \left(\frac{U_1}{4K_T} + \frac{U_2}{2lK_T} + \frac{U_4}{4K_Q} \right)^{\frac{1}{2}} \\ \Omega_3 &= \left(\frac{U_1}{4K_T} + \frac{U_3}{2lK_T} - \frac{U_4}{4K_Q} \right)^{\frac{1}{2}} \\ \Omega_4 &= \left(\frac{U_1}{4K_T} - \frac{U_2}{2lK_T} + \frac{U_4}{4K_Q} \right)^{\frac{1}{2}} \end{aligned} \tag{52}$$

Section 7. UAV Communication

The communication network is based on a single-group swarm ad hoc network as described in [26].

7.1 MANET Architecture

Within the swarm of quadrotors, communication is based on a meshed structure (fig. 7.1). Considering the extra equipment for long-range communication to ground control, a single node within the swarm is considered the Gate Node to the swarm (fig. 7.1). Commands from ground control are received and passed by publishing the information to all adjacent nodes. The information then spreads using a simple flood routing protocol.

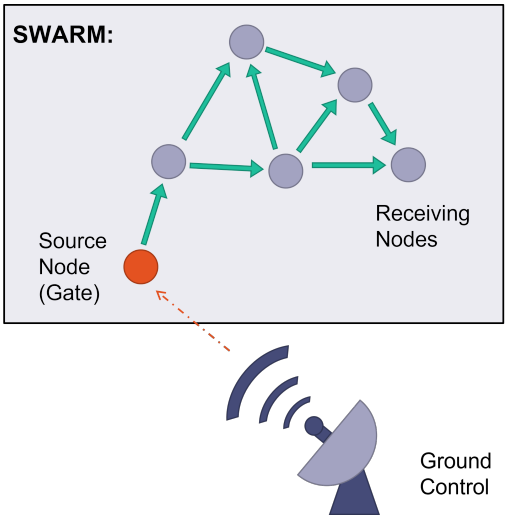


Figure 7.1. Ground control relays information to the swarm through a gateway node (red). Internal mesh communication is independent of the infrastructure (ground control) [26].

7.2 Routing Protocol

Each quadrotor requires the latest state of the other agents in the swarm for the behavioral schema. Every agent will possess a list of the current state of all agents in the swarm as they know it. This list is published to all neighboring agents (within the communication radius of the drones), and from all the duplicated state messages, the most recently updated state is the one taken and updated (fig. 7.2).

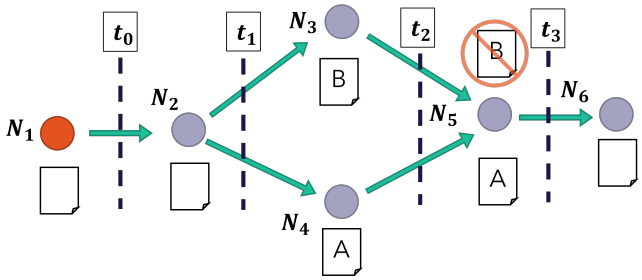


Figure 7.2. Publishing of data forwarded to the swarm.

Section 8. Behavioral Controller

8.1 Behavioral Based Control

Each quadrotor possesses a behavioral controller determining the commanded velocity vector passed to the quadrotor positional/attitude controllers (fig. 6.11).

The outcome trajectory of the quadrotor is determined by the summation of the behaviors as described in [5]–[7].

Each behavior adjusts its respective weights based on set criteria, thus providing a value for that behavior's effective magnitude ($f_{behavior}$). Each behavior additionally has a velocity unit vector to provide direction for the specific behavior ($V_{behavior}$). Therefore, for each behavior there is an output velocity vector ($V = V_{behavior} f_{behavior}$), and by summing all the behaviors (refer to eq. (53)) and rescaling the velocity vector (refer to eq. (54)) we get an emergent behavior (or output velocity command, $\dot{\zeta}_{beh}$), based on all environmental criteria the quadrotor experiences.

$$\dot{\zeta}_{beh} = [f_{goal} \quad f_{form} \quad f_{alt} \quad f_{c-avoid_i} \quad \cdots \quad f_{c-avoid_N}] \begin{bmatrix} V_{goal} \\ V_{form} \\ V_{alt} \\ V_{c-avoid_i} \\ \vdots \\ V_{c-avoid_N} \end{bmatrix} \tag{53}$$

where,

f_{goal} is the weight (magnitude) assigned to the goal-finding behavior.
f_{form} is the weight (magnitude) assigned to the formation maintenance behavior.
$f_{c-avoid}$ is the weight (magnitude) assigned to the collision avoidance behavior. For each UAV-to-UAV collision instance, a distinct magnitude is calculated.
f_{alt} is the weight (magnitude) assigned to the altitude maintenance behavior.
V_{goal} is the commanded velocity vector direction from the goal-finding behavior.
V_{form} is the commanded velocity vector direction from the formation maintenance behavior. For each UAV-to-UAV collision instance, a distinct unit vector is calculated.
$V_{c-avoid}$ is the commanded velocity vector direction from the collision avoidance behavior.
V_{alt} is the commanded velocity vector direction from the altitude maintenance behavior.

The output behavior velocity vector from eq. (53) is normalized and re-scaled to the maximum capable linear speed of the quadrotor (eq. (54)). The resultant velocity vector ($\dot{\zeta}_R$) is then used to derive the reference acceleration ($\ddot{\zeta}_R$) and reference altitude position (Z_R) for the quadrotor's attitude and altitude controller.

$$\dot{\zeta}_R = \left(\dot{X}_{beh}^2 + \dot{Y}_{beh}^2 + \dot{Z}_{beh}^2\right)^{-\frac{1}{2}} \begin{bmatrix} \dot{X}_{beh} \\ \dot{Y}_{beh} \\ \dot{Z}_{beh} \end{bmatrix} \begin{bmatrix} v_{max_{XY}} & v_{max_{XY}} & v_{max_Z} \end{bmatrix} \tag{54}$$

where,

$\dot{\zeta}_{beh} = \begin{bmatrix} \dot{X}_{beh} & \dot{Y}_{beh} & \dot{Z}_{beh} \end{bmatrix}^T$ refer to eq. (53).
$v_{max_{XY}}$ is the maximum speed of the quadrotor along the EF frame *X*-*Y* axes.
v_{max_Z} is the maximum speed of the quadrotor along the EF frame *Z* axis.
$\dot{\zeta}_R = \begin{bmatrix} \dot{X}_R & \dot{Y}_R & \dot{Z}_R \end{bmatrix}^T$ reference velocity vector

For inputs of the controller requiring linear acceleration (eq. (41)) from the behavioral controller, the acceleration is based on the reference velocity (eq. (54)) from the past iteration (at $t = i - 1$) and the current iteration (at $t = i$) as shown in eq. (55).

$$\ddot{\zeta}_R = \frac{\dot{\zeta}_R(t = i) - \dot{\zeta}_R(t = i - 1)}{t_i - t_{i-1}} \tag{55}$$

where,

$\ddot{\zeta}_R = \begin{bmatrix} \ddot{X}_R & \ddot{Y}_R & \ddot{Z}_R \end{bmatrix}^T$ is the output acceleration reference vector in the EF frame.
$\dot{\zeta}_R(t = i)$ is the output reference velocity (reference eq. (54)) from the current calculation iteration.
$\dot{\zeta}_R(t = i - 1)$ is the output reference velocity (reference eq. (54)) from the previous calculation iteration.
t_i, t_{i-1} are the timestamps of when the output reference velocities were calculated from the current (i) and last ($i - 1$) iteration.

8.1.1 Behavior Controller to Attitude Controller Integration

To integrate the behavioral controller output to the quadrotor's altitude controller ($e_Z = Z_R - Z$ in eqs. (47) and (48)), the desired position is determined using eq. (56). By setting the desired interval of time (dt) to extract the distance (taking the integral) with the given commanded velocity ($\dot{Z}_R$) from the behavioral controller (reference eq. (54)), the EF frame z-axis position (Z_R) is determined.

Utilizing the same minimum boundary set in eq. (61) for the formation maintenance behavior (see fig. 8.2), the goal altitude is passed as the EF frame z-axis reference position (Z_R) as shown in eq. (57) (result is passed to the following eqs. (47) and (48)).

The algorithm for determining the EF frame z-axis reference position (Z_R) is shown in table 8.1.

$$Z_R = \dot{Z}_{beh}\, dt + Z_i \tag{56}$$

where,

Z_R is the EF frame z-axis reference position.

$\dot{Z}_{beh}$ is the z-axis commanded behavior velocity in the EF frame towards the goal altitude from the UAV's current position.

dt is the defined time interval setting to determine the desired reference position to reach.

$$Z_R = Z_{goal} \tag{57}$$

where,

Z_R is the EF frame z-axis reference position.

Z_{goal} is the goal altitude (EF frame).

Reference Z Position Algorithm
IF $\left(Z_{goal} - Z_i\right) \leq b_{min}$
THEN Use eq. (56) to determine the EF frame z-axis reference position (Z_R)
ELSE
Use eq. (57) to determine the EF frame z-axis reference position (Z_R)

Table 8.1. Algorithm to determine the EF frame reference Z_R position to the altitude controller (reference eq. (48)).

8.1.2 Move-to-Goal Behavior

In the goal-seeking behavior, once the quadrotor is near the goal altitude, it is commanded to go a velocity based on the distance from the goal until it reaches the goal's boundary (b_{max}), as shown in eq. (59). The weight parameter is negatively linearly varied from the maximum boundary to the minimum boundary (eq. (59) and fig. 8.1).

Overall, there is a constant attractive force at distances greater than the maximum boundary. Then, from the maximum boundary to the minimum boundary, the attractive force drops off at a linear rate until it reaches zero once below the minimum boundary (reference fig. 8.1).

Additionally, this behavior remains inactive until within the maximum boundary of the goal altitude, as shown in eq. (59) and fig. 8.2.

$$V_{goal} = \left(\left(X_{goal} - X_{Ni} \right)^2 + \left(Y_{goal} - Y_{Ni} \right)^2 \right)^{-\frac{1}{2}} \begin{bmatrix} X_{goal} - X_{Ni} \\ Y_{goal} - Y_{Ni} \end{bmatrix} \tag{58}$$

where,

$[X_{goal}, Y_{goal}]^T$ is the vector of the goal position.
$[X_{Ni}, Y_{Ni}]^T$ is the vector of the quadrotor position.

$$f_{goal}(d_i) = \begin{cases} 0, & \text{for } d_{alt} \, \epsilon \, (b_{alt-max}, \, +\infty) \\ 0, & \text{for } d_i \, \epsilon \, [0, \, b_{min}] \\ w_{goal}, & \text{for } d_i \, \epsilon \, (b_{max}, \, +\infty) \\ w_{goal} \frac{d_i}{b_{max}}, & \text{for } d_i \, \epsilon \, [0, \, b_{max}] \end{cases} \tag{59}$$

where,

d_{alt} is the distance from the quadrotor to the goal altitude position.
$b_{alt-max}$ is the adjustable maximum boundary parameter shared from the altitude maintenance behavior (reference fig. 8.2).
d_i is the distance from the quadrotor to the goal position.
w_{goal} is the adjustable goal weight parameter.
b_{max} is the adjustable boundary parameter.

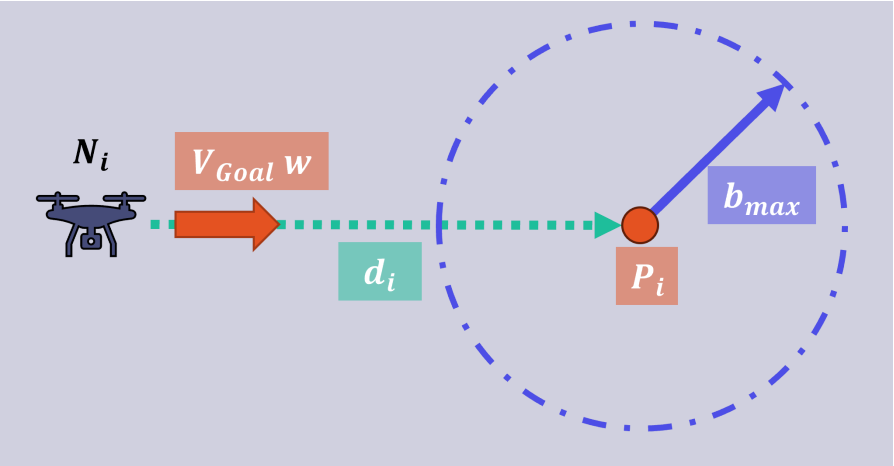


Figure 8.1. UAV moves towards the goal and adjusts velocity within the boundary condition.

8.1.3 Get to Altitude

In the altitude maintenance behavior, the quadrotor is commanded to go at maximum velocity based on the distance from the goal altitude set, until it reaches the goal altitude boundary (b_{max}), as shown in eq. (61). The weight parameter is negatively linearly varied from the maximum boundary to the minimum boundary (eq. (61)). Which will also negatively linearly vary the commanded velocity when multiplied together as seen in eq. (53). Once the quadrotor is within the minimum boundary (b_{min}) as shown in eq. (61)), the applied weight is set to zero.

Overall, there is a constant attractive force at distances greater than the maximum boundary (b_{max}), then from the maximum boundary to the minimum boundary (b_{min}), the attractive force drops off at a linear rate until reaches zero once below the minimum boundary (fig. 8.2).

$$V_{alt} = \frac{Z_{goal} - Z_{Ni}}{\left(\left(Z_{goal} - Z_{Ni}\right)^2\right)^{\frac{1}{2}}} \tag{60}$$

where,

Z_{goal} is the goal position along the EF frame *Z* axis (altitude).
Z_{Ni} is the quadrotor position along the EF frame *Z* axis.

$$f_{alt}(d_i) = \begin{cases} 0, & \text{for } d_i \epsilon [0,\ b_{min}] \\ w_{alt}\frac{d_i}{b_{max}}, & \text{for } d_i \epsilon [b_{min},\ b_{max}] \\ w_{alt}, & \text{for } d_i \epsilon (b_{max},\ +\infty) \end{cases} \tag{61}$$

where,

d_i is the distance from the quadrotor to the goal altitude.
w_{alt} is the adjustable altitude weight parameter.
b_{max} is the adjustable maximum boundary parameter.
b_{min} is the adjustable minimum boundary parameter.

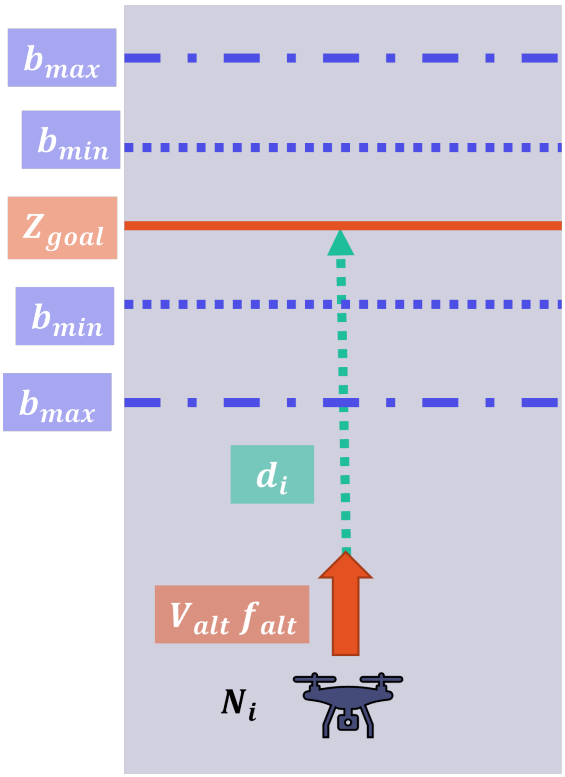


Figure 8.2. UAV moves towards the goal altitude and adjusts velocity within the maximum boundary condition but outputs a zero condition within the minimum boundary condition.

8.1.4 Collision Avoidance behavior

Unlike the other behaviors, collision avoidance is based on the velocities of each UAV to determine time until collision for calculating the magnitude of the behavior response, while using the UAV-to-UAV distance to determine the direction of avoidance.

The time to collision is based on the equations of motion (eq. (62)):

$$\begin{aligned} X(t) &= 0.5\ddot{X}\ t^2 + \dot{X}\ t + X_{initial} \\ Y(t) &= 0.5\ddot{Y}\ t^2 + \dot{Y}\ t + Y_{initial} \end{aligned} \tag{62}$$

The distance between the UAV, including the radius encompassed by the physical UAV, is represented by eq. (63) and eq. (64):

$$d_{ij} = \left[\left(X_j(t) - X_i(t) \right)^2 + \left(Y_j(t) - Y_i(t) \right)^2 \right] \tag{63}$$

$$d_{ij} = R_{UAV_i} + R_{UAV_j} = 2\ R_{UAV} \tag{64}$$

where,

R_{UAV_i} is the collision radius of the UAV being controlled ("self").
R_{UAV_j} is the collision radius of the UAV potential in collision with ("other").
R_{UAV} the collision radius of both ("self" and "other") if the same.

Taking eq. (62), considering acceleration as constant, and plugging it into eq. (63), and equating to eq. (64) results in the following equation in quadratic form as shown in eq. (65) and eq. (66):

$$a\ t^2 + b\ t + c = 0 \tag{65}$$

where,

$$\begin{aligned} a &= \left(\dot{X}_j - \dot{X}_i \right)^2 + \left(\dot{Y}_j - \dot{Y}_i \right)^2 \\ b &= 2\ \left[\left(X_j - X_i \right) \left(\dot{X}_j - \dot{X}_i \right) + \left(Y_j - Y_i \right) \left(\dot{Y}_j - \dot{Y}_i \right) \right] \\ c &= \left(X_j - X_i \right)^2 + \left(Y_j - Y_i \right)^2 - (2\ R_{UAV})^2 \end{aligned} \tag{66}$$

Taking advantage of the quadratic form, the roots can be determined using the quadratic formula (eq. (68) or eq. (70)), giving the time(s) collision(s) occur(s).

Before calculating the roots, the discriminant is calculated (eq. (67)). The discriminant indicates the following based on the output:

Discriminant Collision Check
IF $D > 0$, then there are two roots indicating an entry collision and an exit collision. **THEN** Use eq. (68) to find the two possible time to collisions. Then take the smaller value using eq. (69) which can be assumed to be the entry collision time.
IF $D = 0$, then there is one root indicating an entry collision and an exit collision. **THEN** Use eq. (70) to find the time to collision.
IF $D < 0$, then there is no collision.

Table 8.2. Algorithm check by taking the discriminant to determine if another UAV is on a path that collides with UAV "self".

$$D = b^2 - 4\ a\ c \tag{67}$$

$$t_1, t_2 = \frac{\left(-b \pm \sqrt{D}\right)}{2\ a} \tag{68}$$

In the case where the discriminant is greater than zero ($D > 0$), two roots are given, so the minimum value is taken as the entry collision (eq. (69)).

$$t_{collision} = min(t_1,\ t_2) \tag{69}$$

$$t_{collision} = \frac{-b}{2\ a} \tag{70}$$

To determine if a collision occurs, two criteria are investigated:

1. The discriminant (eq. (67))
2. The dot product between the relative position and relative velocity (eq. (71) and table 8.3)

As previously discussed, a discriminant value of less than zero indicates that there are no collisions (table 8.2). But to supplement this, the dot product of the relative position and velocity of the "other" UAV provides information on whether the "other" UAV is moving towards "Self", away, or maintaining its distance (table 8.3). This checks to ensure the UAVs are moving fast enough on a collision course before calculating the collision. Since the "b" term in eq. (66) is similar to taking the dot product, it is used in the collision avoidance algorithm (table 8.5).

$$v_{ij} = \dot{\zeta}_j - \dot{\zeta}_i$$
$$r_{ij} = \zeta_j - \zeta_i \tag{71}$$

Dot Product Collision Check
IF $r_{ij} \cdot v_{ij} > 0$, $\lvert r_{ij} \rvert$ increases, indicating UAV "other" (j) is not on a collision course with UAV "self" (i).
IF $r_{ij} \cdot v_{ij} < 0$, $\lvert r_{ij} \rvert$ decreases, indicating that UAV "other" (j) is on a collision course with UAV "self" (i).
IF $r_{ij} \cdot v_{ij} = 0$, $\lvert r_{ij} \rvert$ remains the same, indicating UAV "other" (j) is not on a collision course with UAV "self" (i).

Table 8.3. Algorithm check to determine if another UAV is coming closer, maintaining distance, or moving away from UAV "self".

To get the magnitude of the collision avoidance behavior, the type of collision is first determined, and the time to collision is calculated following the algorithm in table 8.5.

Time to Collision Algorithm
IF $D > 0$ and $b < -1 \left(10^{-4}\right)$
THEN, find $t_{collision}$ with eq. (68) and eq. (69)
ELSE IF $D == 0$ and $b < -1 \left(10^{-4}\right)$
THEN, find $t_{collision}$ with eq. (68)
ELSE
$t_{collision} = None,$ There is no collision.

Table 8.4. Algorithm to determine the time for two UAV on a collision course will collide.

Once the time to collision has been determined ($t_{collision}$), the following cases determine the magnitude (eq. (72)).

$$f_{c-avoid}(t_{collision}, d_{jk}) = \begin{cases} 0, & \text{for } d_{jk}\ \epsilon\ (b_{max},\ +\infty) \\ 0, & \text{for } collision_{bool}\ ==\ False\ and\ d_{jk} b_{max} \\ \frac{w_{c-avoid}}{t_{collision}}, & \text{for } collision_{bool}\ ==\ False \end{cases} \tag{72}$$

where,

d_{jk} is the distance from the quadrotor ("self") to the quadrotor to be avoided ("other") fig. 8.3.

$w_{c-avoid}$ is the adjustable collision avoidance behavior weight parameter. Represents the desired distance to cover within the time until a collision occurs.

b_{max} is the adjustable maximum boundary parameter around the UAV to be avoided fig. 8.3. This value is set to be $50\ -\ 75\%$ of the formation spacing.

To determine the direction of the collision avoidance behavior, two components are introduced to prevent collision and induce the UAV to pass around each other (eq. (75)). For eq. (74) the distance between the UAV "self" (j) and UAV "other" (k) is used to get a unit vector with the UAVs repelling each other directly away from each other. In eq. (73), a vector perpendicular to the straight line distance between the UAV "self" (j) and UAV "other" (k) is used, directed towards the direction of the formation position relative to UAV "self" (j) (fig. 8.3).

$$V_{c-avoid_{tan}} = d_{jk}^{-1} \begin{bmatrix} copysign\big(\big(Y_{Nk} - Y_{Nj}\big),\ V_{form}(X)\ \big) \\ copysign\big(\big(X_{Nk} - X_{Nj}\big),\ V_{form}(Y)\ \big) \end{bmatrix} \tag{73}$$

where,

$d_{jk}^{-1} = \Big(\big(X_{Nk} - X_{Nj}\big)^2 + \big(Y_{Nk} - Y_{Nj}\big)^2\Big)^{-\frac{1}{2}}$ is the distance from UAV N_j to UAV N_k.

$copysign(a, b)$ function to take the absolute value of a and assign the sign ($\pm$) of b.

V_{form} unit vector direction for formation maintenance behavior (eq. (79)).

$[X_{Nk}, Y_{Nk}]^T$ is the position vector of the quadrotor to avoid fig. 8.3.

$[X_{Nj}, Y_{Nj}]^T$ is the current quadrotor position vector fig. 8.3.

$$V_{c-avoid_{norm}} = d_{jk}^{-1} \begin{bmatrix} -\big(X_{Nk} - X_{Nj}\big) \\ -\big(Y_{Nk} - Y_{Nj}\big) \end{bmatrix} \tag{74}$$

$$V_{c-avoid} = V_{c-avoid_{norm}} + V_{c-avoid_{tan}} \tag{75}$$

To temporarily reduce the effect of the goal and formation behaviors, they are linearly scaled down the closer the UAVs get to each other while within each UAV's respective collision avoidance boundary (b_{max}). However, a minimum value is specified to ensure the other behaviors remain active to induce movement toward the formation points and goal. It is important to note that this calculation is done after calculating the goal and formation behaviors.

$$r_{redux} = max\left(\frac{(b_{max} - d_{jk})}{(b_{max} - b_{min})}, w_{min}\right) \tag{76}$$

where,

$max(a, b)$ is a function where the larger value between a and b is returned.

b_{max} is the adjustable maximum boundary parameter around the UAV to be avoided fig. 8.3. This value is set to be $50 - 75\%$ of the formation spacing.

b_{min} is the adjustable minimum boundary parameter fig. 8.3.

d_{jk} is the distance from the quadrotor ("self") to the quadrotor to be avoided ("other") fig. 8.3.

w_{min} is the minimum weight value assigned to the goal and formation behavior.

Then we define the new formation and goal behavior magnitude as:

$$\begin{aligned} f'_{form} &= f_{form}\ r_{redux} \\ f'_{goal} &= f_{goal}\ r_{redux} \end{aligned} \tag{77}$$

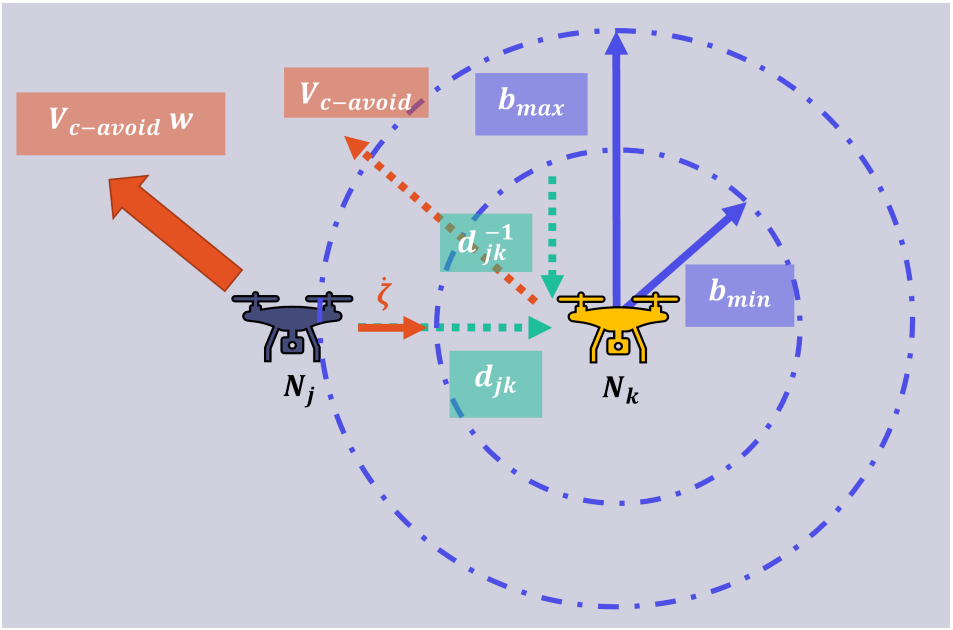


Figure 8.3. When within the maximum boundary condition of other UAV, UAV "self" (j) constantly checks for a possible collision to all UAV "other" (k). UAV "self" (j) moves away from UAV "other" (k) when on a collision course to each other. In addition to this, the goal-seeking behavior and formation maintenance behavior are linearly reduced from the maximum boundary to the minimum boundary, to prioritize collision avoidance but still ensure the UAV move with the rest of the formation.

The entire collision avoidance behavior is summarized in the following algorithm:

Collision Avoidance Algorithm
FOR UAV in "swarm size" find d_{ij} from eq. (63) **IF** d_{ij} <= $b_{max_{c-avoid}}$ **and** $drone\_id_j$!= $drone\_id_i$ **THEN**, $\zeta_i, \zeta_j, \dot{\zeta}_i, \dot{\zeta}_j$ # Get current position and velocity of UAV_i and UAV_j Calculate coefficients in eq. (66) Calculate discriminant (D) with eq. (67) **IF** $D > 0$ and $b < -1 \left(10^{-4}\right)$ **THEN**, find $t_{collision}$ with eq. (68) and eq. (69); $Collision_{bool} = True$ **ELSE IF** $D == 0$ and $b < -1 \left(10^{-4}\right)$ **THEN**, find $t_{collision}$ with eq. (68); $Collision_{bool} = True$ **ELSE** $t_{collision} = None,$ $Collision_{bool} = False$ There is no collision. Move on to next "for loop" iteration. Calculate behavior unit vector ($V_{c-avoid}$) with eqs. (73), (74) and (75) **IF** $Collision_{bool} == True$ **and** $t_{collision}$ <= $t_{collision_{MAX}}$ **IF** $t_{collision}$ <= $t_{collision_{MIN}}$ **THEN**, $t_{collision} = t_{collision_{MIN}}$ **THEN**, Determine behavior magnitude with eq. (72) **IF** d_{ij} <= $b_{min_{c-avoid}}$ **THEN**, Set other behaviors to minimum magnitude: $f_{goal} = f_{goal_{MIN}}$ $f_{form} = f_{form_{MIN}}$ **ELSE** Determine behavior weight reduction shown in eqs. (76) and (77)

Table 8.5. Algorithm for collision avoidance behavior.

8.1.5 Formation Maintenance Behavior

Each UAV possesses knowledge of the current formation, orientation of the formation, their assigned position in the formation, and the current position/velocity of the other UAV. Using a formation-center-based approach, the average location of all the UAVs is determined (eq. (78)). This locates the formation, while the next waypoint orients the formation as shown in fig. 8.4. With knowledge of the UAV's assigned formation position in the calculated formation position at the average formation center, the respective UAV's formation position can be maintained. The magnitude of the behavior is provided by eq. (79), functioning similarly to the goal-seeking behavior. The unit vector direction provided in eq. (78) directs the UAV towards the known formation position.

$$\alpha_{avg} = \left(\sum_{i=1}^{n} \alpha_i\right)(n)^{-1} = \begin{bmatrix} X_1 + X_2 \ + \ ... \ + \ X_n \\ Y_1 \ + \ Y_2 \ + \ ... \ + \ Y_n \end{bmatrix}(n)^{-1} \tag{78}$$

where,

α_i is the *X*-*Y* position of each quadrotor in the EF frame.

$$V_{form} = \left(\left(X_i - X_{Nj}\right)^2 + \left(Y_i - Y_{Nj}\right)^2\right)^{-\frac{1}{2}} \begin{bmatrix} X_i - X_{Nj} \\ Y_i - Y_{Nj} \end{bmatrix} \tag{79}$$

where,

$[X_i, Y_i]^T$ is the assigned formation node position.
$\left[X_{Nj}, Y_{Nj}\right]^T$ is the current quadrotor position.

$$f_{form}(e_i) = \begin{cases} 0, & \text{for } d_{alt} \ \epsilon \ (b_{alt-max}, \ +\infty) \\ 0, & \text{for } d_i \ \epsilon \ [0, \ b_{min}] \\ w_{form}, & \text{for } e_i \ \epsilon \ (b_{max}, \ +\infty) \\ w_{form}\frac{e_i}{b_{max}}, & \text{for } e_i \ \epsilon \ [0, \ b_{max}] \end{cases} \tag{80}$$

where,

d_{alt} is the distance from the quadrotor to the goal altitude position.
$b_{alt-max}$ is the adjustable boundary parameter from the altitude maintenance behavior.
e_i is the distance from the quadrotor to the designated formation position (fig. 8.4).
w_{form} is the adjustable formation maintenance behavior weight parameter.
b_{max} is the adjustable maximum boundary parameter around the designated formation position (P_i in fig. 8.4) as similarly shown in fig. 8.1.

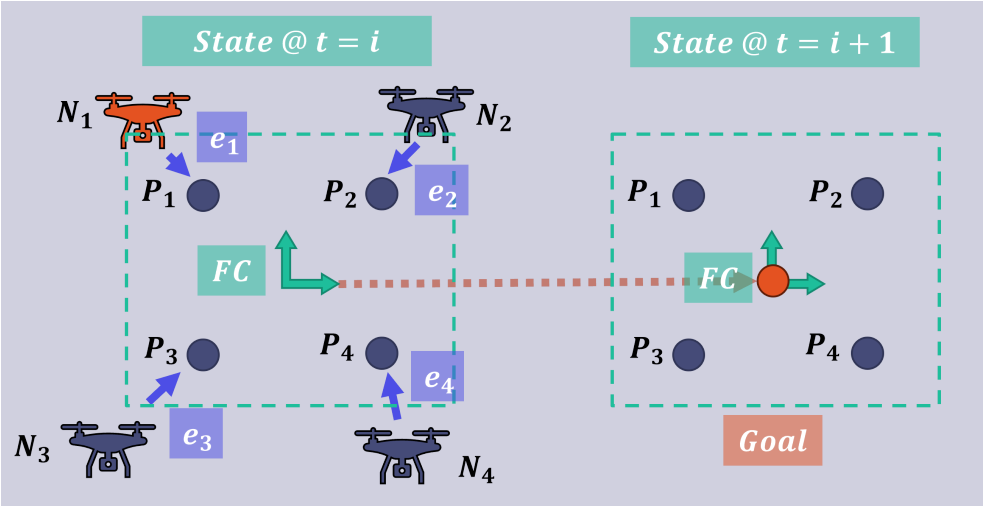


Figure 8.4. Example of formation maintenance behavior, where each UAV possesses knowledge of the formation and their assigned position in it, positions of the other UAV, and the goal the formation must reach. At every time instance, the formation positions are calculated based on the current formation centers, thus giving the UAV's position error from its assigned formation position and the direction to the assigned formation position. Which is then maintained with the formation maintenance behavior shown in eqs. (79) and (80).

8.2 Formation Formulation

Two methodologies for formation assignment were implemented and compared. A centralized formation assignment structure and a decentralized formation assignment structure.

8.2.1 Centralized Formation Position Assignment

Initial formation formulation occurs after all quadrotors reach altitude from the position where they have been activated (or spawned in simulation). The quadrotor designated as the gate node then performs the calculation for the formation position/orientation and the quadrotors assigned to each formation positional node.

Taking the commanded formation, formation spacing, and number of nodes in the formation, the *gate node* calculates the average *X*-*Y* position of all quadrotors to determine the formation center and orients the formation towards the next desired waypoint/goal (as shown in fig. 8.5).

Next, the *gate node* (N_1 in fig. 8.5) assigns each UAV to a position in the formation based on it's distance. The quadrotor nearest to the formation position is the quadrotor assigned (N_3 in fig. 8.5).

In fig. 8.4, it is shown how the formation is located in the center of the swarm of quadrotors. Then, when the *gate node* is assigning formation positions, a table (section 8.2.1 is produced relating the distances (d_i, j) of each formation point (P_i) to each quadrotor (N_i).

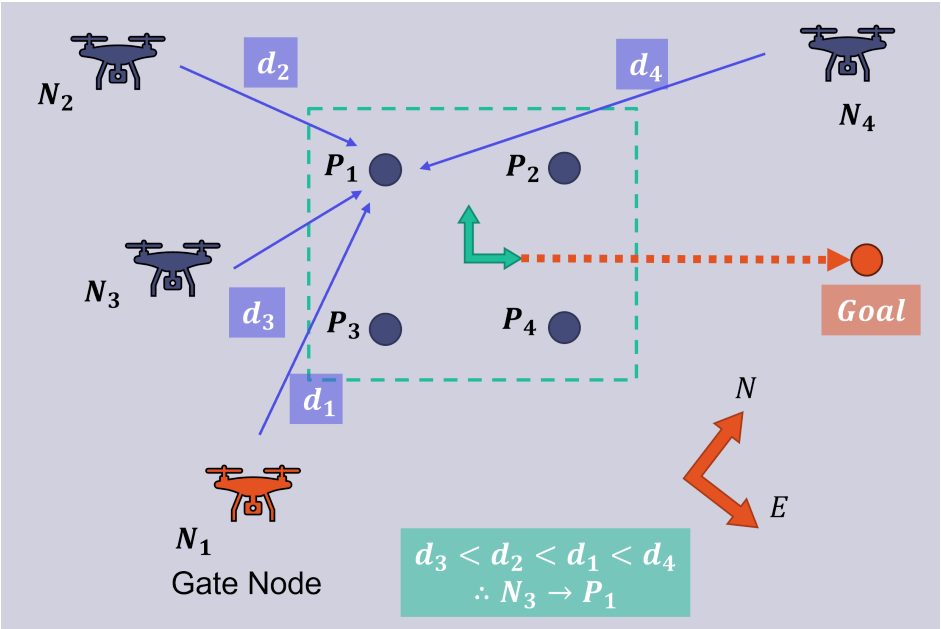


Figure 8.5. Gate node assigns UAV positions in the commanded formation based on distance.

	P_1	P_2	P_3
N_1	$d_{1,1}$	$d_{2,1}$	$d_{3,1}$
N_2	$d_{1,2}$	$d_{2,2}$	$d_{3,2}$
N_3	$d_{1,3}$	$d_{2,3}$	$d_{3,3}$

Table 8.6. Matrix of UAV distances to each position in formation.

A separate numerical example (table 8.8) shows the allocation of three quadrotors (N_1 to N_3) to three positions (P_1 to P_3) in a single-file line formation.

	P_1	P_2	P_3
N_1	4	5.656	8.944
N_2	4	0	4
N_3	8.944	5.656	4

(a)

	P_1	P_2	P_3
N_1	4	inf	8.944
N_2	inf	inf	inf
N_3	8.944	inf	4

(b)

	P_1	P_2	P_3
N_1	inf	inf	inf
N_2	inf	inf	inf
N_3	inf	inf	4

(c)

	P_1	P_2	P_3
N_1	inf	inf	inf
N_2	inf	inf	inf
N_3	inf	inf	inf

(d)

Table 8.8. Example of allocating the quadrotors to their nearest formation position. **table 8.7a** Matrix of quadrotor distances to each position in the formation. **table 8.7b** Selection of smallest distance between a UAV and formation position by determining the minimum value in the matrix, then setting the respective row and column to infinite values. Since there are two positions in the matrix that are the lowest value, the selected value is the first encountered minima in the matrix (N_2 to P_2). **table 8.7c** Selection of smallest distance between UAV and formation position (N_1 to P_1) by determining the minimum value in the matrix, then setting the respective row and column to infinite values. **table 8.7d** Selection of smallest distance between UAV and formation position (N_3 to P_3)

Once the quadrotors are assigned positions, the assigned positions of each UAV and the commanded formation data is published to the swarm (section 8.2.1). From this point on, each quadrotor determines the formation goal positions independently, with the knowledge passed regarding the formation and assigned positions in the commanded formation.

Formation Command Data	
Formation	single-file line; single-file row; wedge; circle
Formation Spacing	> 3 m
Formation Altitude	> 10 m
Swarm Size	> 1 UAV
Formation Goal Position	(X,Y) m
Formation Assignments	"drone id": "formation position id"

Table 8.9. Data passed from the *gate node* which is forwarded from *ground control* for formation commands (excluding formation assignments which are determined by the *gate node*).

After the initial formation assignment, the UAV swarm will head toward the goal position in the previously assigned formation. At any point, the ground control can assign new formation criteria as shown in section 8.2.1. As discussed in section 8.2, the *gate node* provides the assigned formation positions and each respective UAV determines their trajectory to maintain formation.

Formation Command Data	
Formation	single-file line; single-file row; wedge; circle
Formation Spacing	> 3 m
Formation Altitude	> 10 m
Formation Goal Position	(X,Y) m

Table 8.10. Data passed from *ground control* for re-formation commands (excluding formation assignments which are determined by the *gate node*).

8.2.2 Decentralized Reformation

Following the same procedures as the centralized formation formulation, initial formation occurs once all UAV reach altitude. All data regarding the mission plan is still routed through the *gate node*. Instead of relying on the *gate node* for reformation, which ensures all nodes are never assigned to the same formation positions, minimum weight matching in bipartite graphs are used. With the Scipy optimization library, the implementation of finding the formation assignments is a modified Jonker-Volgenant algorithm with no initialization [27]. With this algorithm, each UAV can independently determine their formation position at the same time instance, while avoiding assignments of similar formation positions.

In addition to independent formation formulation, failures of UAVs during flight missions have minimal effects on formation assignments, since the algorithm can be solved for rectangular (non-square) cost tables. By identifying UAV to ignore, the cost table is solved for un-ignored UAV.

The formation position assignment algorithm is as follows:

Decentralized Formation Assignment

```
from scipy.optimize import linear_sum_assignment
import numpy as np
drone2form_dist = np.zeros([swarm_size, form_size]) # distance between each UAV and
each available formation node position.
for i in range(swarm_size): # cycle through all UAV.
        for j in range(form_node_size): # cycle through each formation node position.
                drone2form_dist[i, j] = dist_calc_2d(form_pos[i], drone_pos[j])
modified_matrix = drone2form_dist.copy()
for k in ignore_drone_id_array: # ignore_drone_id_array possesses id's starting from 1
        modified_matrix[(k - 1), :] = np.max(drone2form_dist) + 1 # Set distance cost to +1
greater than the maximum value in the distance cost table.
drone_ind, form_pos_ind = linear_sum_assignment(modified_matrix) # output indices for
the assigned formation positions to each un-ignored UAV
```

Table 8.11. Algorithm for formation assignment based on UAV's distance to formation node positions. Consideration of UAV(s) to ignore is done by comparing to an array providing id's (ignore_drone_id_array).

Section 9. Simulation Studies

In the simulations, three overarching studies were conducted:

- Formation adherence, reformation, and collision avoidance.
- Effect of a single UAV failure to the formation.
- Comparison between a single UAV photogrammetry survey mission and a multi-UAV formation photogrammetry survey mission.

9.1 Formation Reconfiguration

The swarm was tested on its ability to :

- Get out and into tight formations without inducing instability of each UAV
- Avoid collisions with other UAV during reformation
- Formation maintenance
- Time to reach the new formation

9.1.1 Formation Reconfiguration: Simulation 1

A square path was conducted to introduce 90 degree turns since the swarm is expected to perform well consistently making 90 degree turns in formation when conducting surveys of large areas. Reformation is introduced at each corner and mid-point of the square path's edge to examine near consecutive reformation ability (every 20m). Various consecutive formation combinations are examined going in and out of corners.

As can be seen in fig. 9.1 the test has UAV spawn near the local origin, get to altitude (15m), then go 20 meters to the square edge waypoint, continuing the square path with waypoints every 20 meters, then return to the local origin.

During the test, all the UAV initially got into formation without issue after reaching altitude. Getting into and out of formations was conducted smoothly. It was observed that the UAV responded quickly upon new formation assignment, but there were no collision avoidance responses except for at *waypoint 8* (fig. 9.1) going from *circle* formation to *single-row* (fig. 9.3).

Focusing on Waypoints 7 - 9 :

In fig. 9.4 and fig. 9.3, the UAV positions are shown in increments of 2 seconds to show the speed at which the UAV travel to get into formation. In fig. 9.3, UAV 1 accelerates to get into the lead position transitioning from row formation, then the circle formation slowly gets to the waypoint at (-20,-20). Next, the formation transitions to single row formation, and for a moment UAV 1 and 2 are on collision paths to each other (shown in fig. 9.5 from the perspective of UAV 1). Considering that the test is conducted at the minimum formation spacing recommended of 8 meters, this is very positive. But without larger number of agents in the formations, it is unknown how much reliance would be placed on collision avoidance to get into dense formations from linear formations. During development tests

for formation assignment, the timing of when formation assignment was determined could cause problems where one UAV may be obstructing another UAV in its direct line path to their formation position. Collision avoidance mitigated full-speed collisions provided their formation spacing was above the minimum of 8 meters and the boundary at which collision avoidance is detected is large enough to slow the colliding UAV.

For cases where the UAV need to weave through other UAV to get into formation, the goal and formation behaviors are minimized to ensure that the UAV avoid each other but still slowly get into formation position (which can be seen in fig. 9.5 the effect of other behaviors minimized with increased collision avoidance behavior). Minimization of the other behaviors is done since increasing the collision avoidance behavior introduces unstable reactions, resulting in a downed UAV.

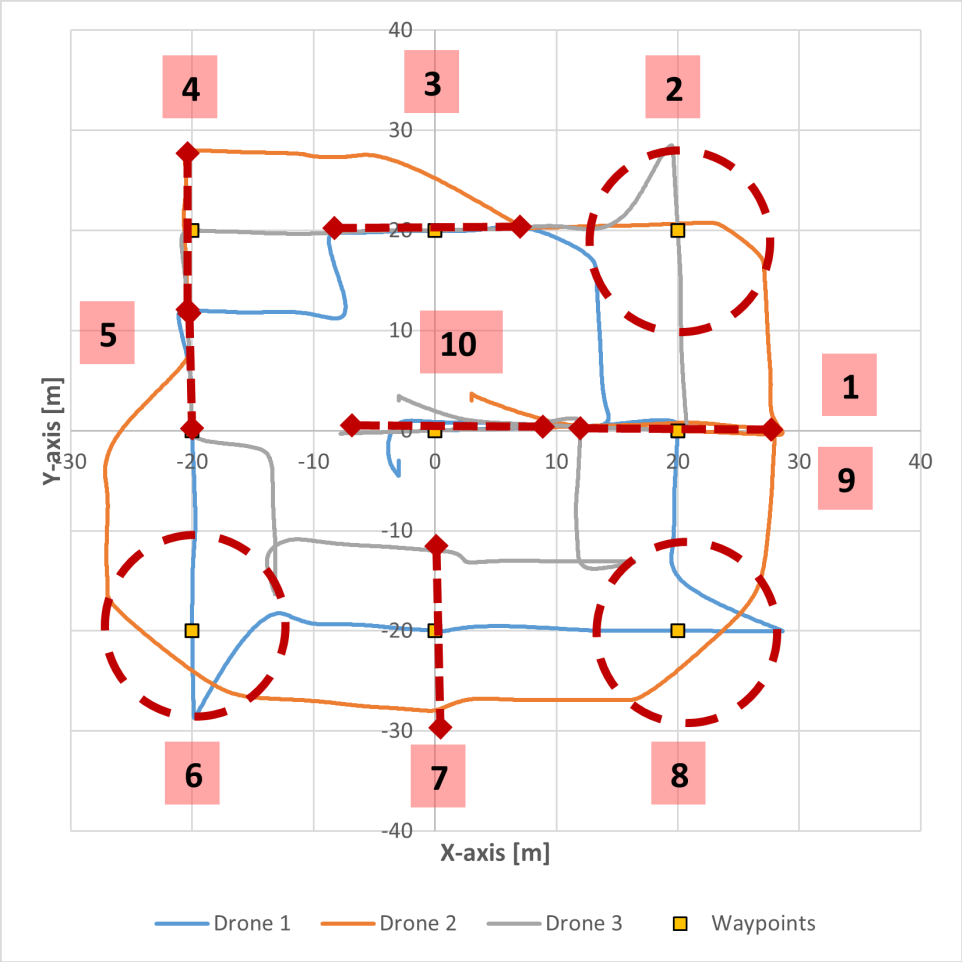


Figure 9.1. Reformation test, where UAV swarm moves along a square path changing formations. Formations indicated at waypoints are the finished formation up until the reached waypoint. *single-row formation* (4,7)*; single-file formation* (1,3,5,10)*; circle formation* (2,6,8)

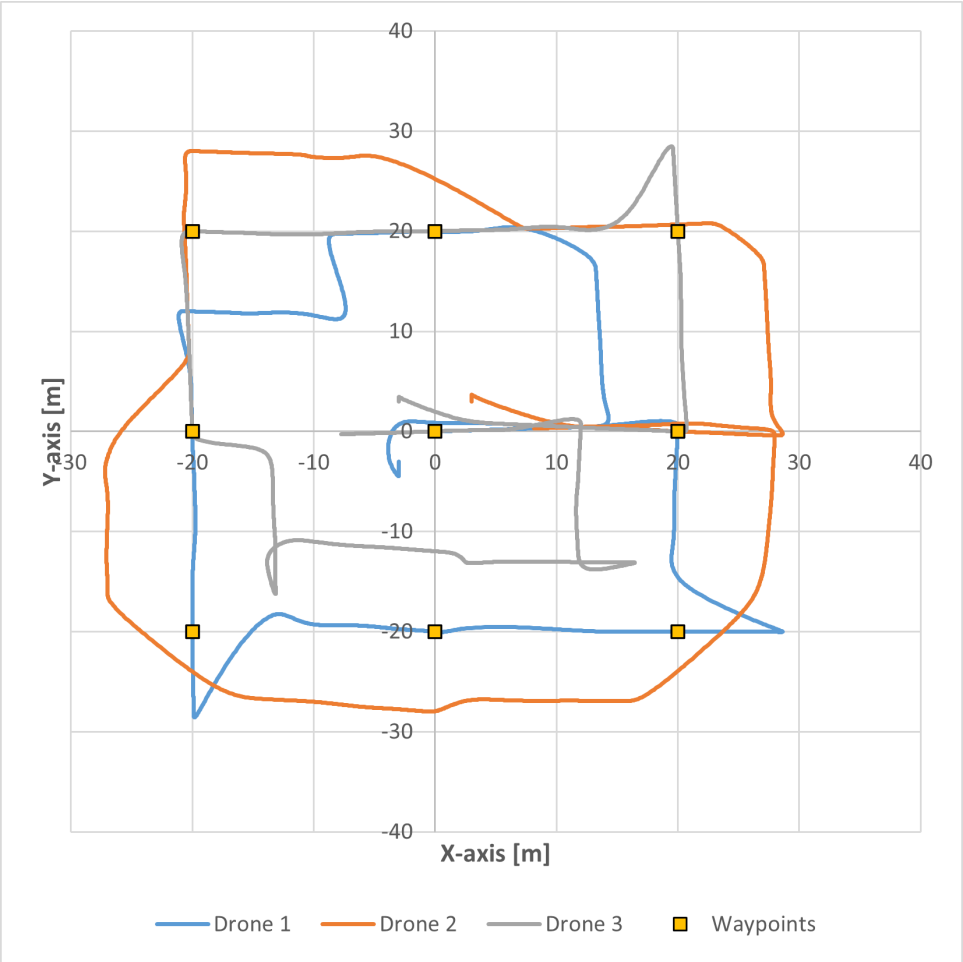


Figure 9.2. Reformation test, where UAV swarm moves along a square path changing formations.

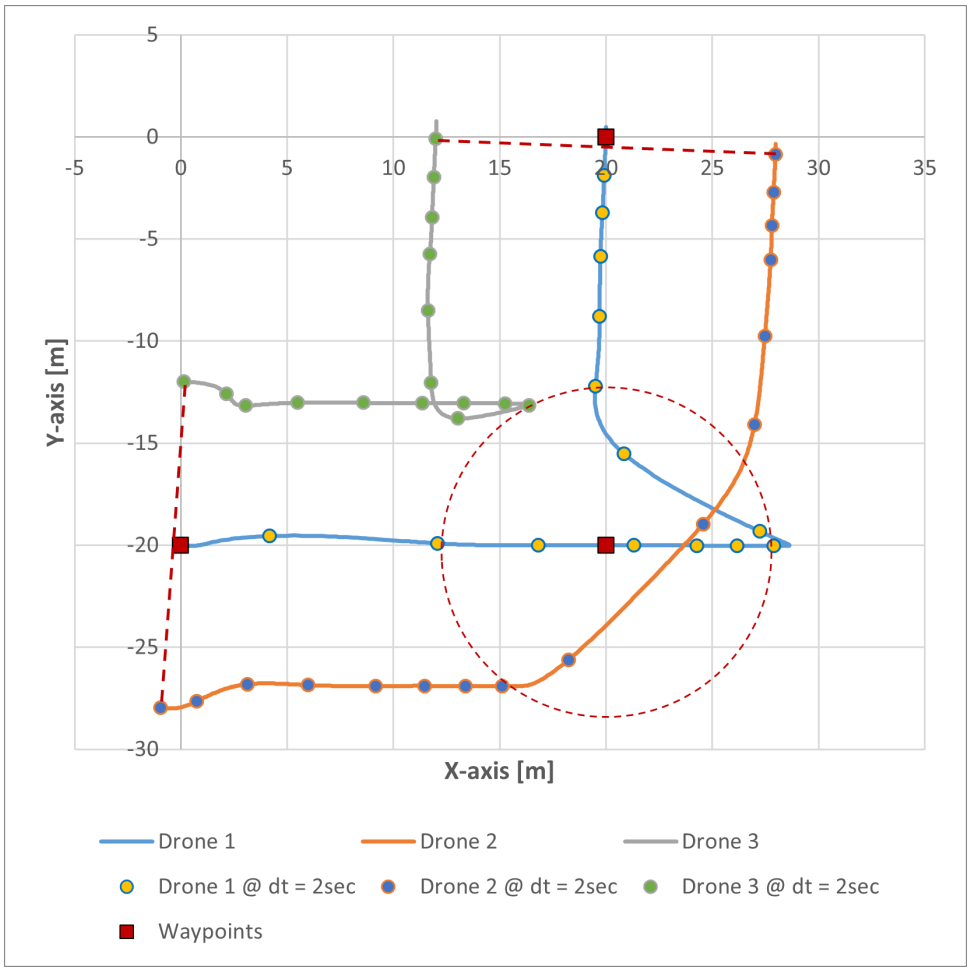


Figure 9.3. Zoomed-in view of reformation test through waypoints 7, 8, and 9 referencing fig. 9.1. The path of each UAV can be visualized, as well as their respective position in 2-second increments.

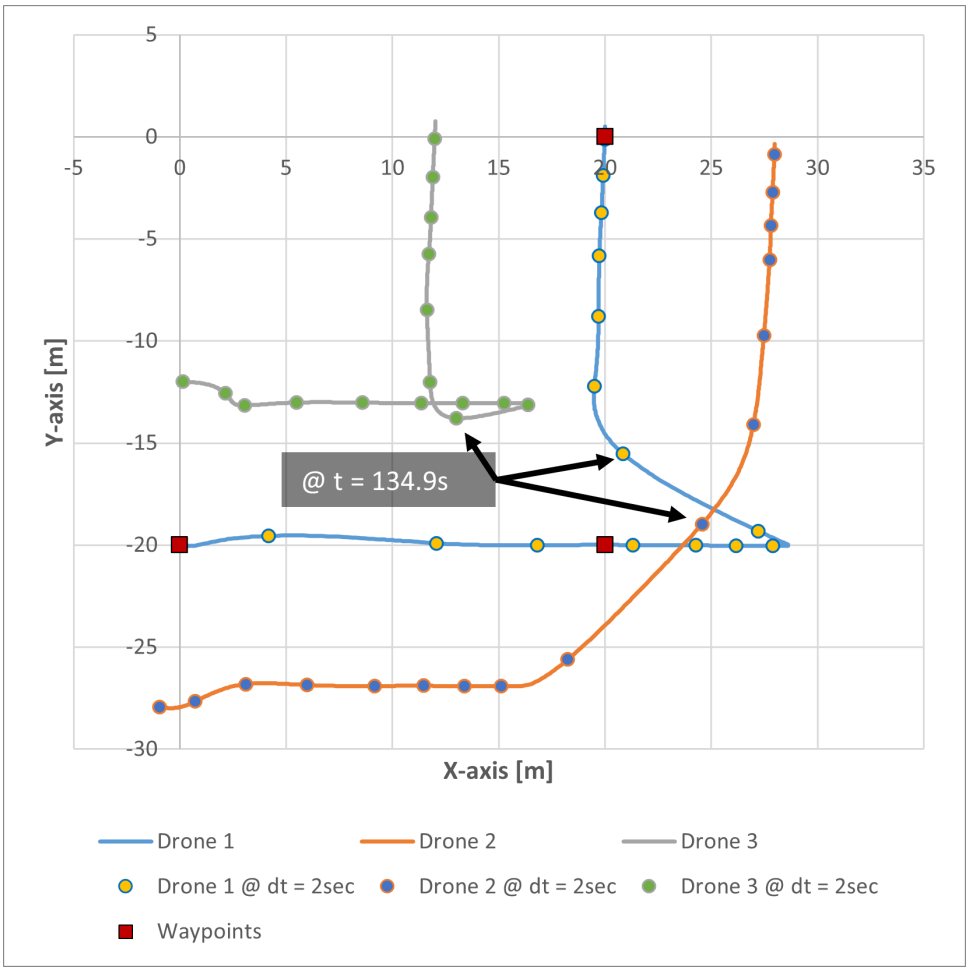


Figure 9.4. Zoomed-in view of reformation test through waypoints 7, 8, and 9 referencing fig. 9.1. The path of each UAV can be visualized, as well as their respective position in 2-second increments. Position of UAVs at $t = 134.9\ sec$ is called out to show reformation and collision avoidance at work (fig. 9.5).

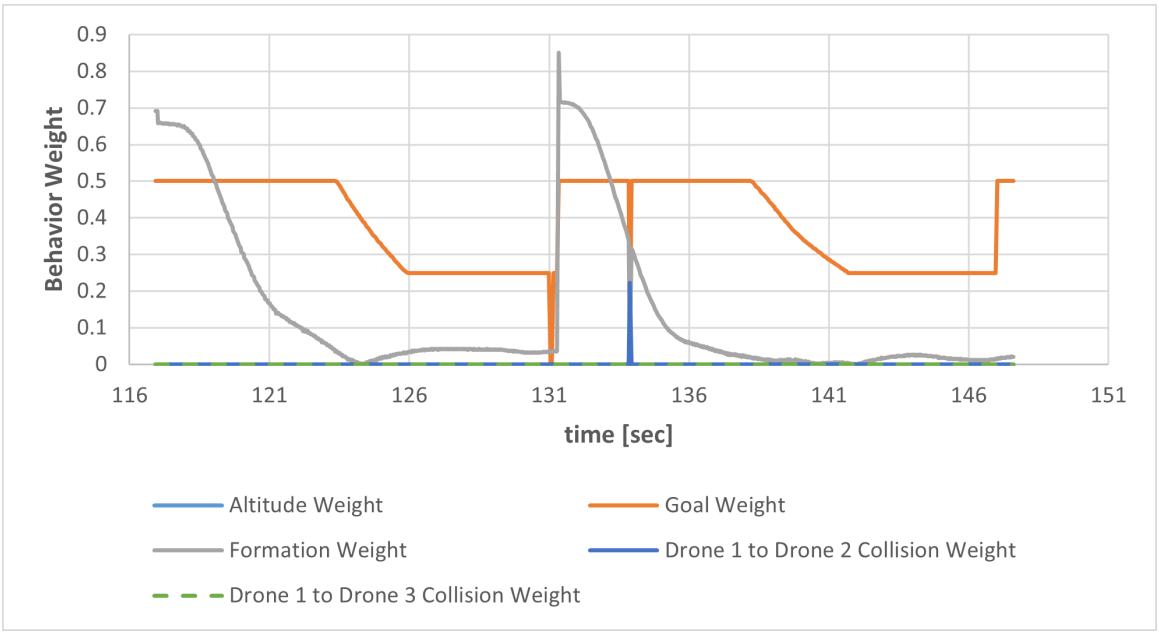


Figure 9.5. Behavior magnitude outputs of UAV 1 during reformation test (fig. 9.4)

Formation Maintenance:

Formation maintenance behavior is based on the UAV's position error from the assigned formation position. So, it's expected to see this behavior increase during formation changes/-formation re-assignment (fig. 9.5). Then the formation behavior hovers near zero to maintain formation. Looking at the time to transition from the previous formation to the new formation, the time for all UAV to settle reaches a maximum of 8.795 seconds (table 9.2). Thus, the effect of the settling time into formation positions on survey missions can easily be accounted for when setting up the mission plan.

Looking at fig. 9.6 at $t = 131.279\ sec$, there is a sharp increase in the distance to the assigned formation position. But in fig. 9.6, the distance quickly reduces (see table 9.1). The distance to the formation position changed but the self-assigned formation position has not been updated to reduce the distance traveled to the assigned formation position. But, on the next iteration of the behavioral controller the new formation position assignment is passed, reducing the needed distance to travel into formation (see table 9.1). Since the length of time this unoptimized formation assignment is used for 0.030 seconds, the effect on the UAV's velocity is minimized (table 9.1).

Recalling that each UAV determines the swarm formation average position from provided data, the true position of all the drones differs from the positions known by every UAV at any time instance. Due to timing and any possible communication failures, errors can affect the ability of the swarm to maintain formation accurately. Simulating a healthy communication network scenario, a maximum error of 0.25 meters was found across all drones during formation reconfiguration. Once settled into their respective formation positions, the maximum error in the swarm average position along the axis of travel is about 0.1 meters (fig. 9.7, fig. 9.8, fig. 9.12, fig. 9.13, fig. 9.15, fig. 9.16,).

When discussing the effects to the application of surveying, these formation deviation errors are minimal since what is of greater importance is ensuring that every UAV is accurately geolocated. With accurate positioning data in post-processing, such errors are negligible. With photogrammetry, a percentage of overlap is necessary, but the expected error can be easily compensated for [28] [12]. For the survey mission, the set distance between UAVs was $22.85\ meters$ to ensure 75% overlap, therefore the overlap can be increased by 0.5 meters, changing from 22.85 to a 22.35 meter spacing.

$Timestamp$ $[sec]$	$UAV\ 1\ to$ $form.\ pos.$ $distance$ $[m]$	$Behavior$ $Accel\ Cmd$ $Out,$ $X-axis$ $\left[\frac{m}{s^2}\right]$	$Behavior$ $Accel\ Cmd$ $Out,$ $Y-axis$ $\left[\frac{m}{s^2}\right]$	$UAV\ 1$ $Velocity$ $X-axis$ $\left[\frac{m}{s}\right]$	$UAV\ 1$ $Velocity$ $Y-axis$ $\left[\frac{m}{s}\right]$
131.279	0.4308	−0.2082	0.0945	0.6395	−0.0041
131.349	16.4022	−47.6332	21.6558	0.8663	0.0061
131.379	8.4280	32.5827	8.9305	0.8551	−0.0066

Table 9.1. Data points of UAV 1 displaying the effect of the spike seen in fig. 9.5 and fig. 9.6. It exhibits the minimal effect on the UAV's velocity (also shown in fig. 9.9 and fig. 9.10) given the behavior output acceleration command.

$UAV\ ID$	$Reformation$ $Start$ $Timestamp\ [sec]$	$Reformation$ $End\ Timestamp$ $[sec]$	$\delta t\ [sec]$
1	131.279	137.889	6.610
2	131.383	139.734	8.351
3	131.361	140.156	8.795

Table 9.2. Referencing fig. 9.6, fig. 9.11, and fig. 9.14; This table shows the time taken to get to a distance of 0.25 meters from goal formation position for each respective UAV.

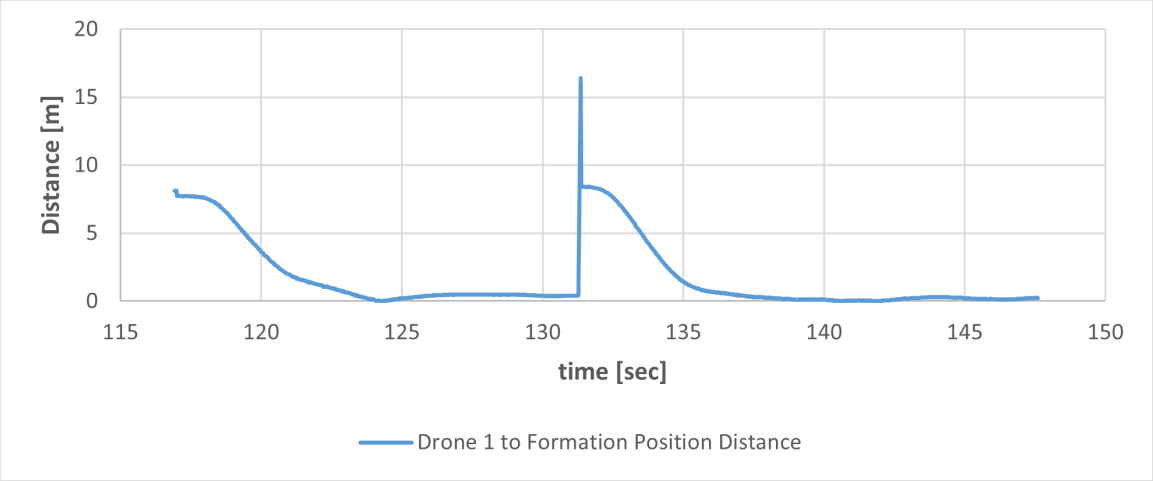


Figure 9.6. Position error of UAV 1 to the assigned formation position.

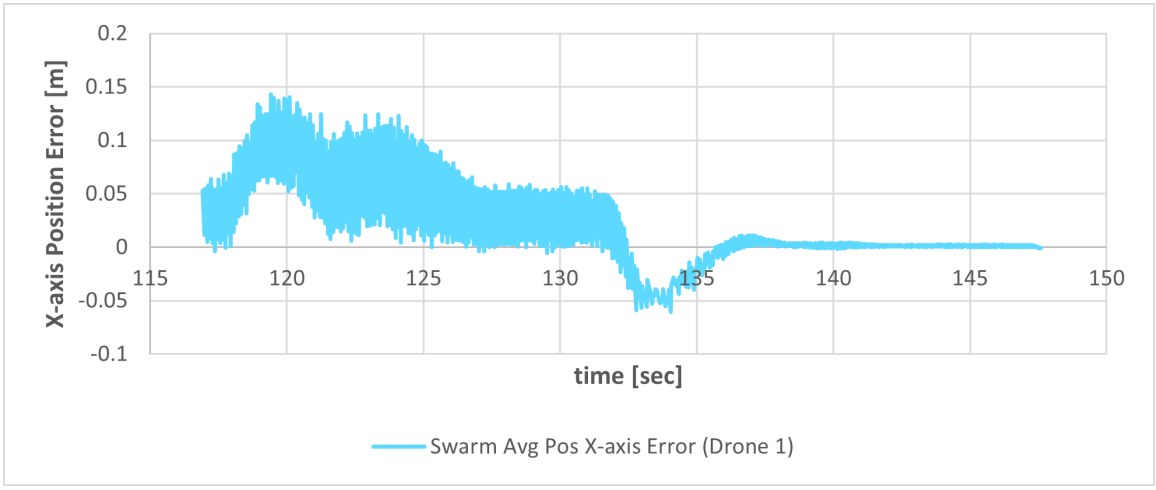


Figure 9.7. Error in calculated swarm average position by UAV 1 compared to their actual average position along the Xaxis.

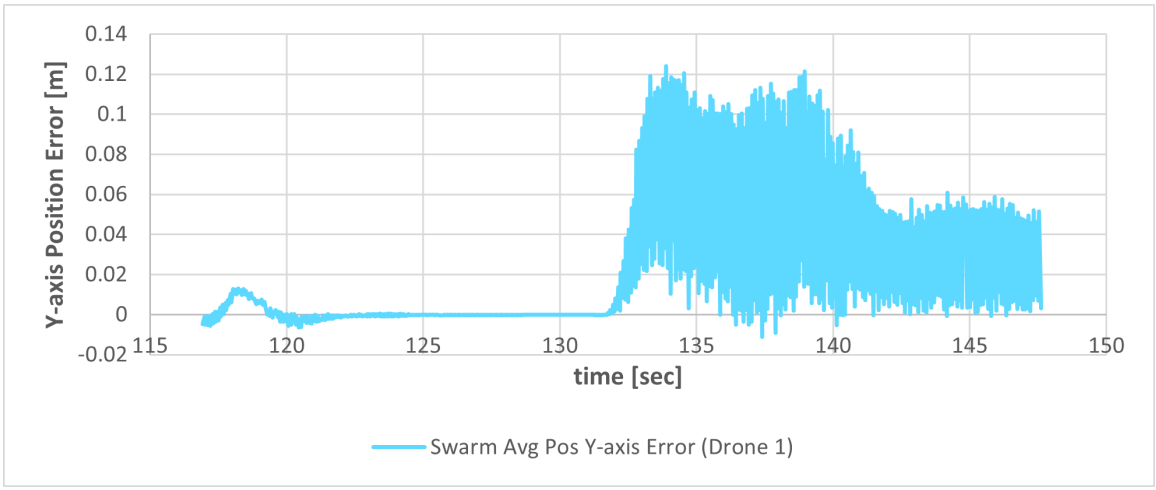


Figure 9.8. Error in calculated swarm average position by UAV 1 compared to their actual average position along the Yaxis.

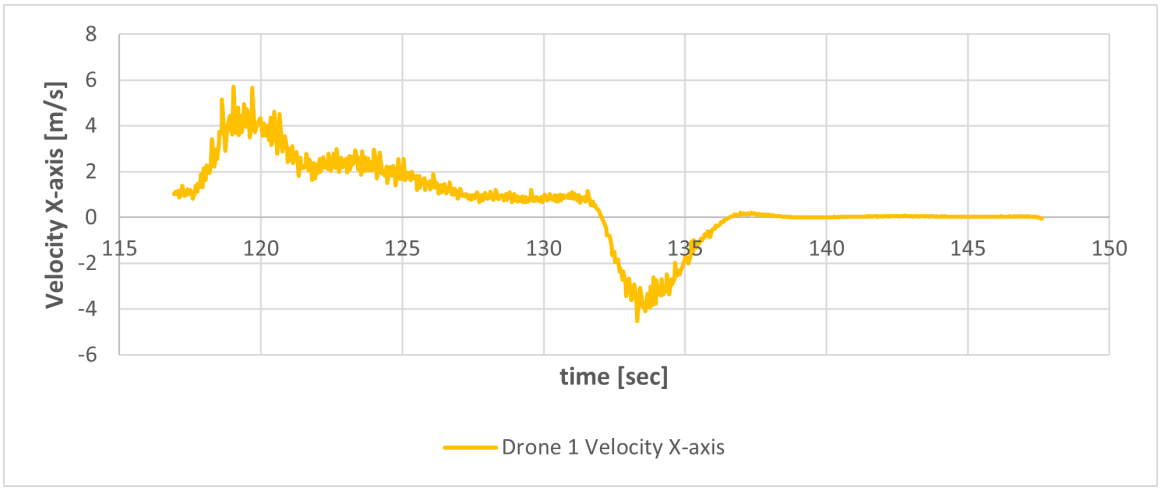


Figure 9.9. UAV 1 velocity along the X-axis during the reformation test (fig. 9.4.

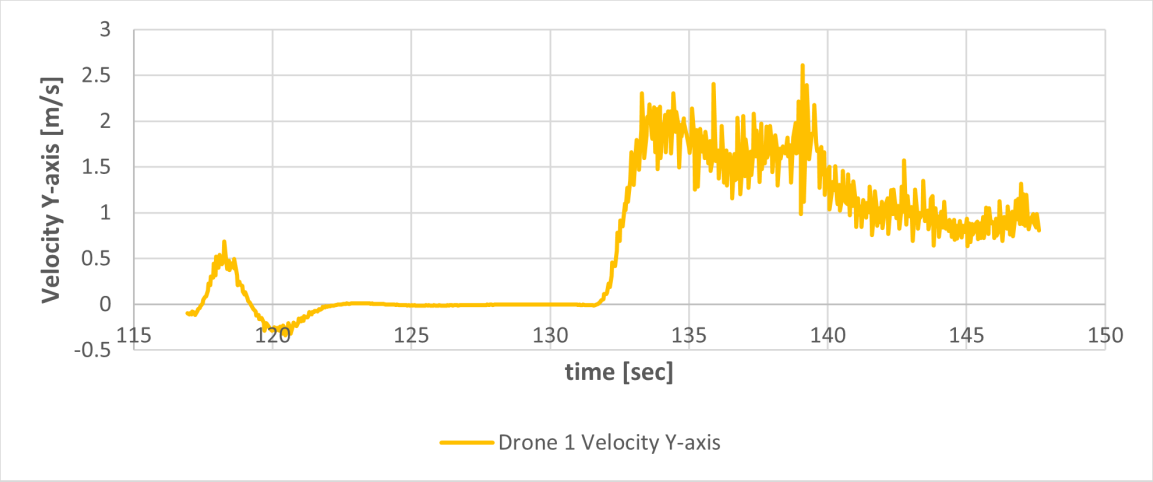


Figure 9.10. UAV 1 velocity along the Y-axis during the reformation test (fig. 9.4.

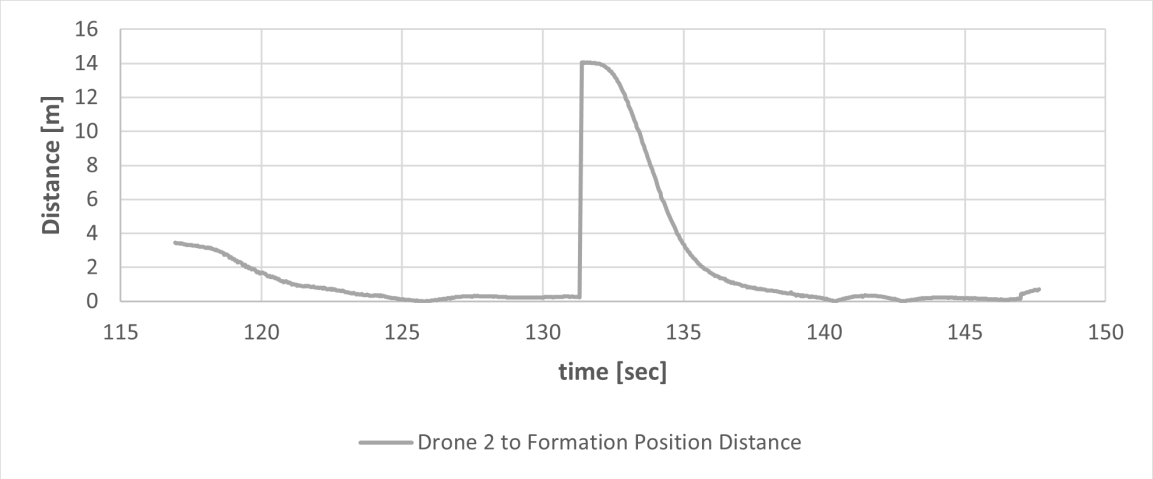


Figure 9.11. Position error of UAV 2 to the assigned formation position.

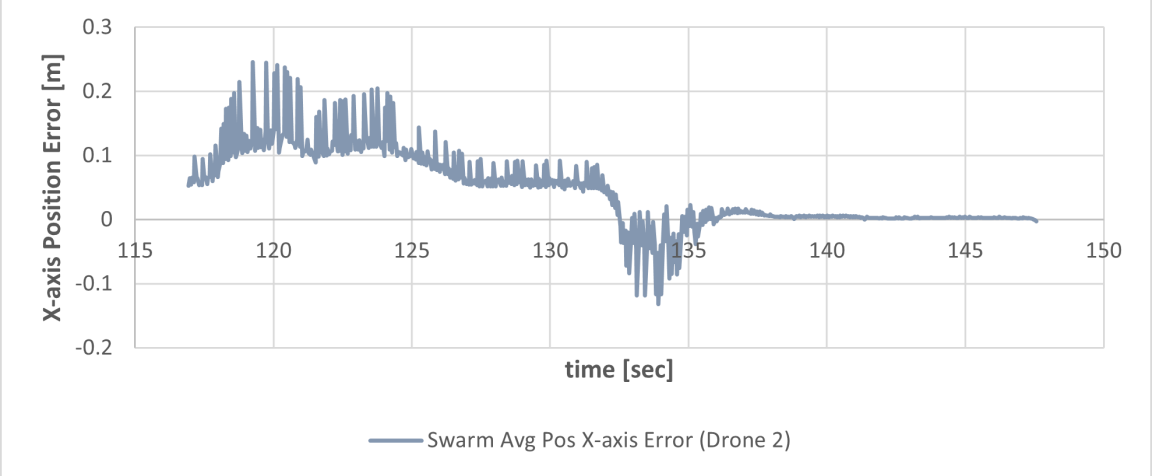


Figure 9.12. Error in calculated swarm average position by UAV 2 compared to their actual average position along the Xaxis.

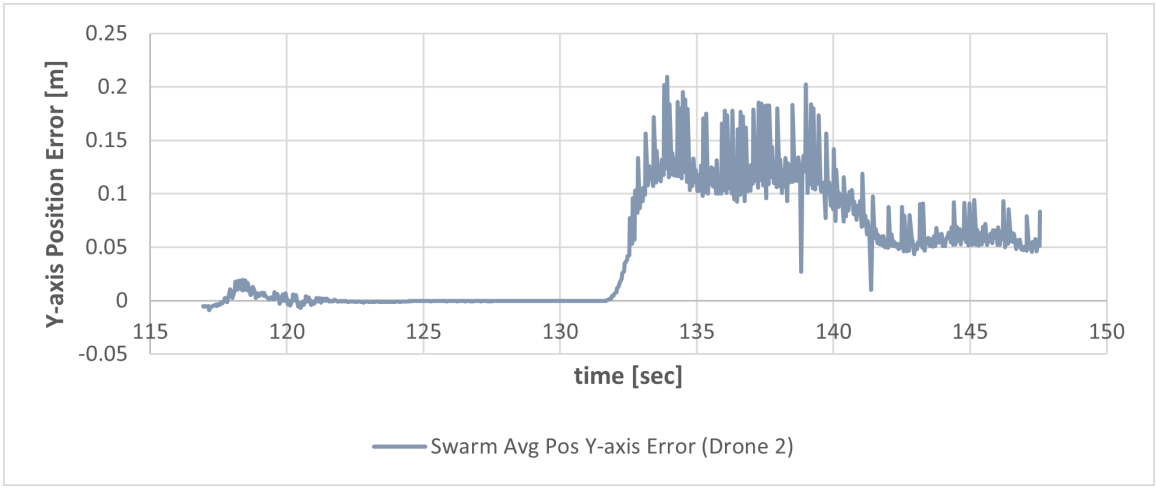


Figure 9.13. Error in calculated swarm average position by UAV 2 compared to their actual average position along the Yaxis.

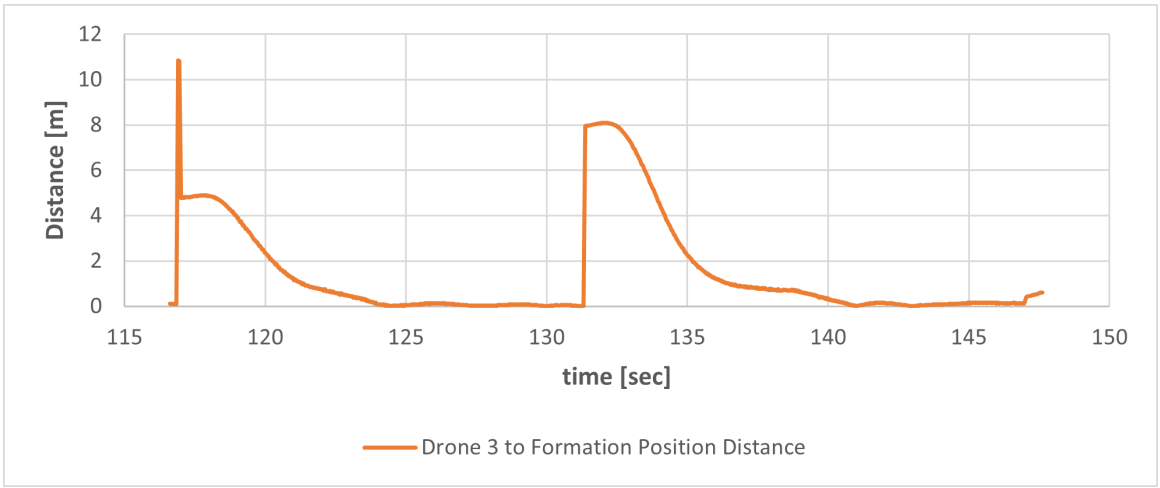


Figure 9.14. Position error of UAV 3 to the assigned formation position.

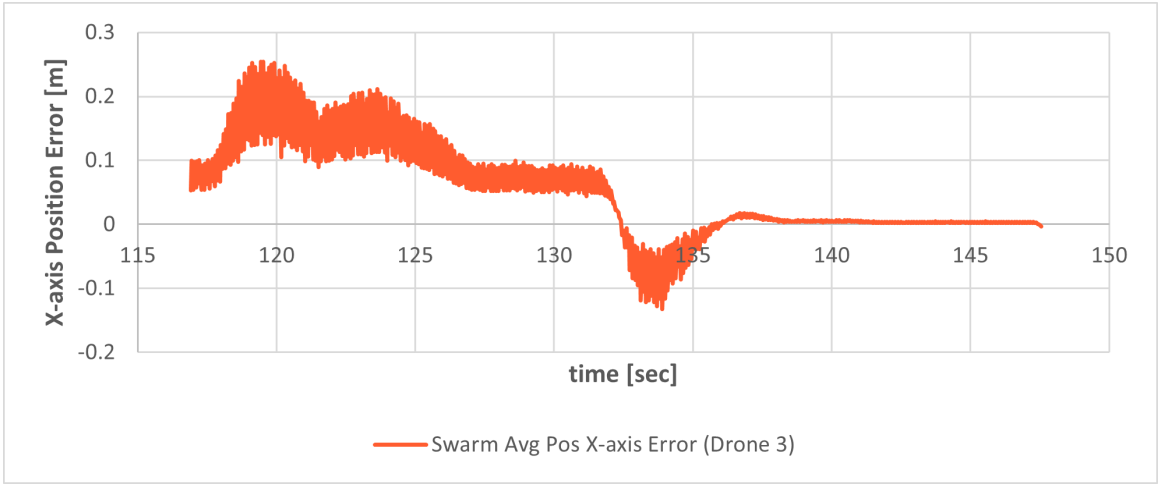


Figure 9.15. Error in calculated swarm average position by UAV 3 compared to their actual average position along the Xaxis.

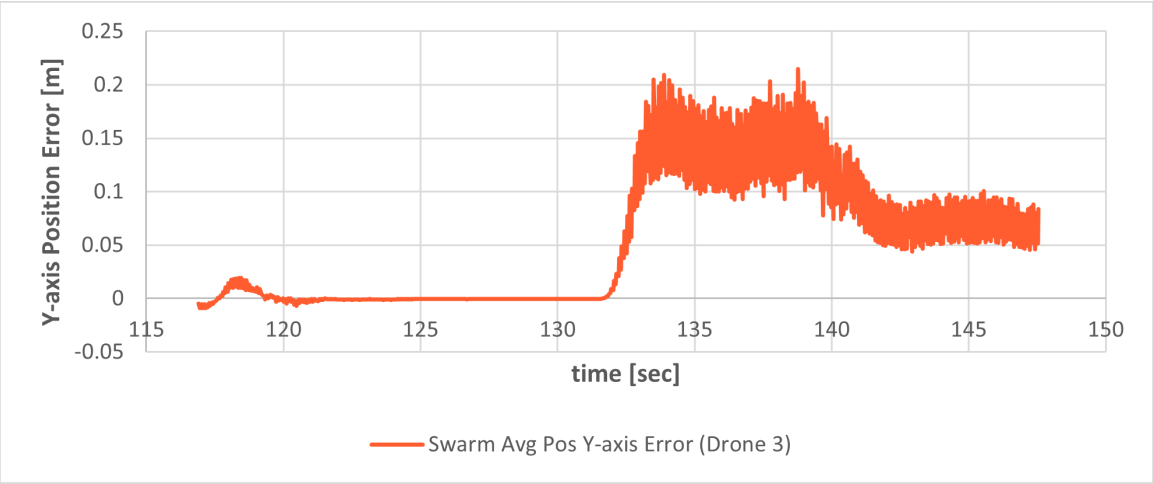


Figure 9.16. Error in calculated swarm average position by UAV 3 compared to their actual average position along the Yaxis.

9.1.2 Formation Reconfiguration: Simulation 2

Looking at more simulation runs of the formation reconfiguration test outlined in section 9.1.1, some insights on collision avoidance can be made. This development simulation run was conducted prior to *simulations 1 and 3*. In this simulation run, collision avoidance minimized both formation and goal-seeking behaviors but maintained a bug where the goal-seeking behavior was minimized to zero during a calculated potential collision at collision times exceeding 3 seconds which is later discussed.

Simulation 2 of formation reconfiguration is a simulation run utilizing formation assignment from the *gate node* (UAV 1) (reference section 8.2.1), but in this case, the *gate node* does not reassign formation ID's after the initial formation assignment, exposing the a failure mode for centralized formation assignment. As a result, during significantly different formation changes, such as a *"circle* formation to *"single-file row"* formation shown in fig. 9.20, the UAV take inefficient paths to the new formation positions. Additionally, the UAVs cross paths, leaving the potential for collisions. In this scenario, we examine the collision avoidance behavior with the behavior settings shown in table 9.3.

Having the collision avoidance weight ($w_{c-avoid}$) set to a value of one (table 9.3), meant the UAV would try to move a distance of one meter prior to colliding. Since the collision boundary is set to 75% of the formation spacing, this results in a very reactive collision avoidance at large distances, as can be seen in fig. 9.20. Figures 9.17, 9.18, and 9.19 show how the collision behavior activated in the instances of potential collisions, cross-referencing the timestamp shown in fig. 9.20.

Around $t = 158\ sec$, "Drone 1's" and "Drone 3's" collision avoidance behavior activates (see figs. 9.17 and 9.19) as they head towards each others general direction, but then they drastically avoid each other as seen in fig. 9.20. The effect of the collision avoidance behavior with the formation behavior created overly reactive reactions during reformation, which was later reduced in proceeding simulation tests.

In previous development simulations, just reducing the formation and goal-seeking behaviors when within the collision avoidance behavior boundary, minimized collisions in the same exact scenario. Which indicated that a minimized collision avoidance reaction would be best for future simulation runs. Following "Drone 2" in fig. 9.20 displays how a pure reduction in the main behaviors such as goal-seeking effects formation reformation. "Drone 2" quickly tries to get into formation and heads for the goal since they are the active behaviors initially (fig. 9.18), until about $t = 157.7$ where the formation behavior is reduced due to a potential collision to "Drone 3". Then "Drone 2" veers to the right explained by the stronger formation weight from about $t = 157.8\ sec$ with the collision avoidance behavior reacting again for a moment. Then at about $t = 159.7\ sec$ the collision avoidance behavior is active, but due to the fact that the time to collision exceeds $3\ seconds$, the goal behavior is minimized, resulting in "Drone 2" prioritizing getting into formation.

Additionally, in this simulation case, since there was no formation re-assignment due to a failure of the *gate node* to create and pass these new assignments the potential for collisions increased. With three UAV, the effects were minor, but if the formation possessed a greater number of agents, unavoidable collisions may have been more likely.

w_{goal}	w_{form}	w_{alt}	$w_{c-avoid}$
0.5	1.0	1.0	1.0

Table 9.3. Behavior weight settings and minimum value allowed for calculated time to collision in collision avoidance behavior.

Collision Avoidance Settings	
$t_{collision_{MIN}}$	Minimized behavior weights
0.1	w_{goal} : 0.0

Table 9.4. Collision avoidance settings adjusted for the current simulation case. $(t_{collision_{min}})$: Minimum value allowed for calculated time to collision in collision avoidance behavior. $Minimized\ behavior\ weights$: The minimum value set for other behaviors when below minimum collision avoidance boundary.

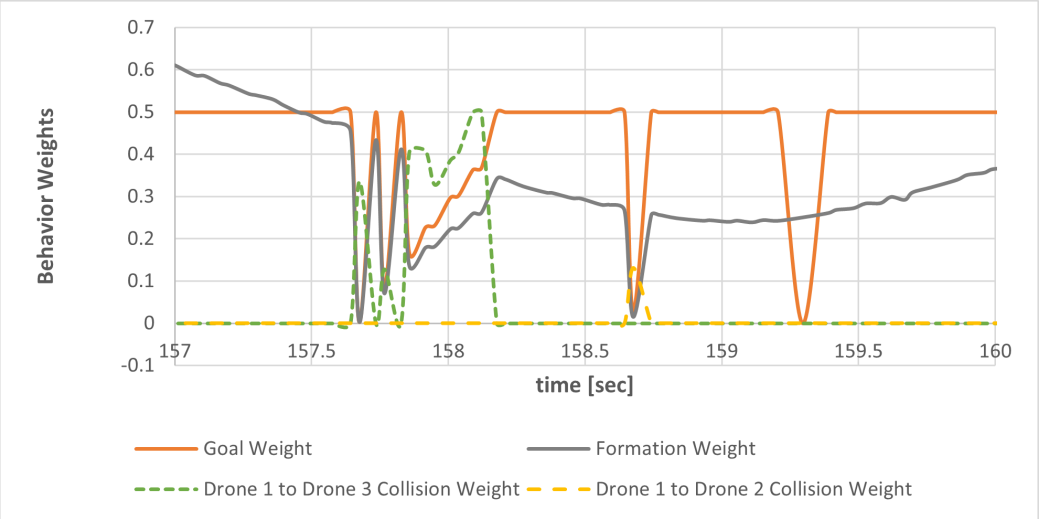


Figure 9.17. Behavior weights (f) of UAV 1 during reformation from circle formation to single-file row formation, showing collision avoidance behavior in effect.

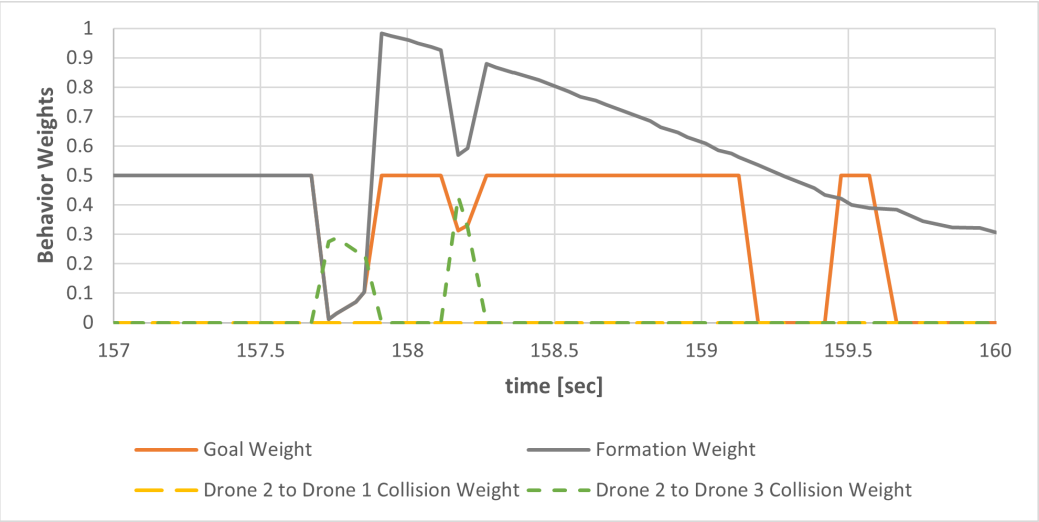


Figure 9.18. Behavior weights (f) of UAV 2 during reformation from circle formation to single-file row formation, showing collision avoidance behavior in effect.

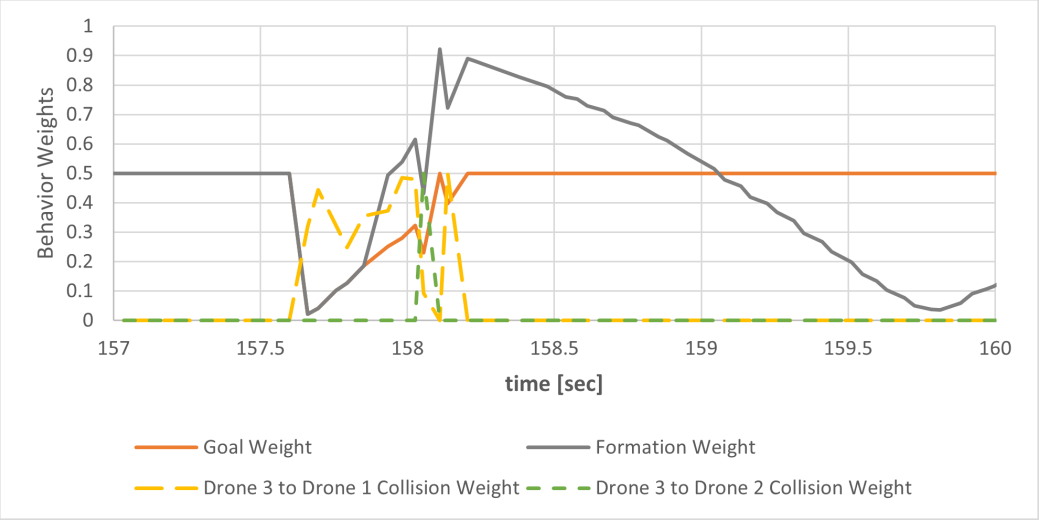


Figure 9.19. Behavior weights (f) of UAV 3 during reformation from circle formation to single-file row formation, showing collision avoidance behavior in effect.

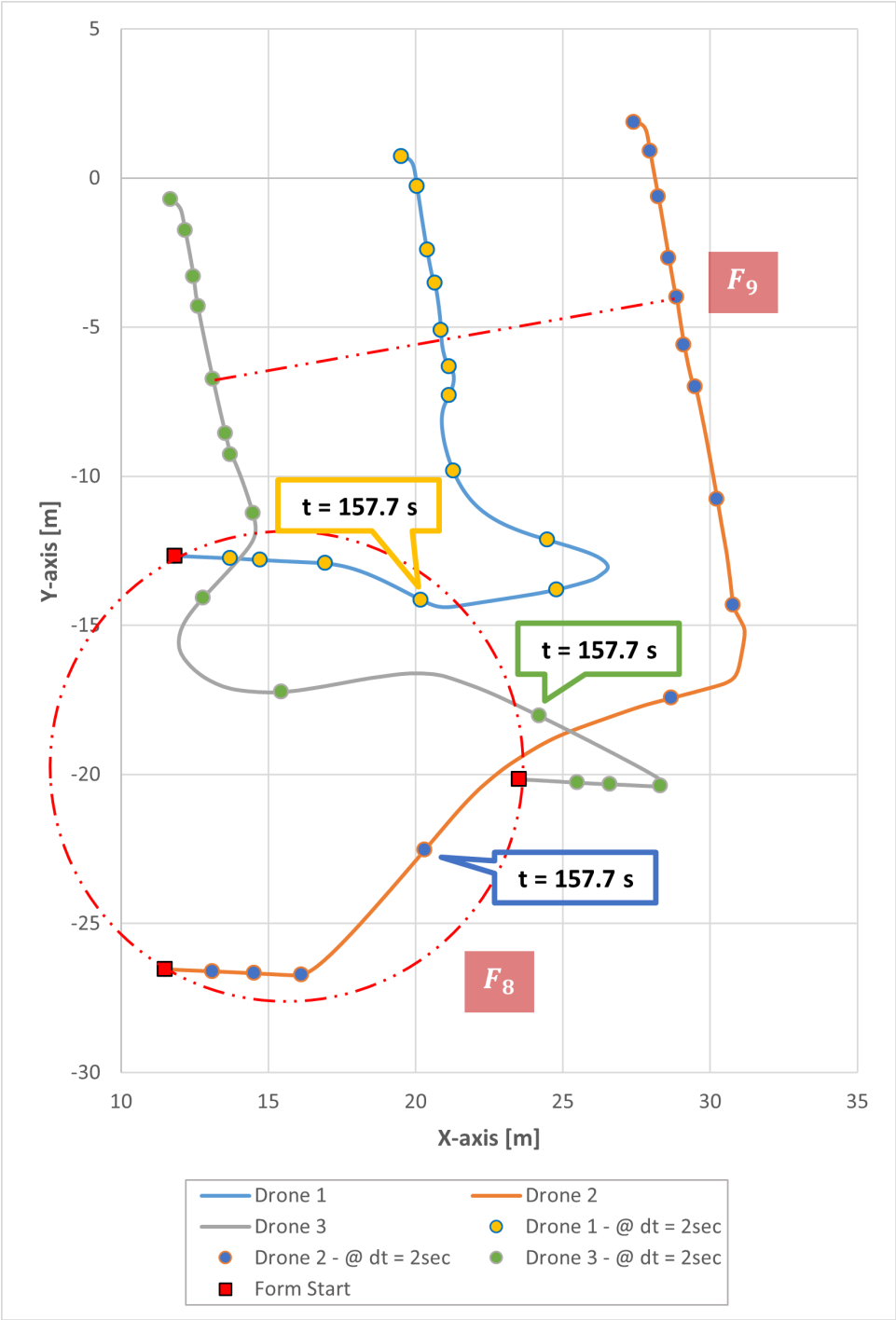


Figure 9.20. Reformation test, as discussed in section 9.1.1, zoomed in on waypoints 7-9 (fig. 9.1). UAV formation moves in circle formation to the waypoint at (20,-20), then reconfigures into a single-file row formation without changing formation ID assignments. The data depicts the starting point of the data displayed with red markers. To visualize speed of motion, the drone positions are marked for every 2 seconds of elapsed time. The drone position markers have a time callout at time = $157.7sec$ to compare the behaviors in figs. 9.17 to 9.19.

9.1.3 Formation Reconfiguration: Simulation 3

Again, Looking at another simulation run of the formation reconfiguration test outlined in section 9.1.1, more information is derived regarding collision avoidance. Here we focus on the initial formation formulation, with the behavior settings shown in tables 9.6 and 9.7.

Simulation 3 presents another exemplary case of collision avoidance during the test scenario, the swarm was assigned their formation positions before getting to altitude, but each drone gets to altitude depending on when it was spawned into the Gazebo simulator world (see table 9.5) and the result is UAV 3 needing to overtake UAV 2 to get into formation. Because the first commanded formation is the “single-file” formation, UAV 2 and the lead formation position UAV 3 is trying to get to are nearly colinear. Looking at fig. 9.21, UAV 3 is shown trying to get to the lead formation position while avoiding UAV 2 and UAV 1 (figs. 9.22 to 9.24). The collision avoidance behavior is applied during after determining collision course while reducing the formation and goal-seeking behaviors. The goal of the collision avoidance behavior is to prevent collisions without an overreactive reaction since more collisions could occur, which may lead to controller instability. In this case, collision avoidance successfully guided the UAV to the formation positions. Before collision avoidance, for the same situation, UAV 3 would collide with UAV 2 head-on. This scenario also displays the limitations of this collision avoidance algorithm in close proximity. The reaction at close proximity is not strong enough with the settings placed on collision avoidance, but increasing the settings, while it does improve close proximity reaction, causes excessive reactions at farther distances from each other (see fig. 9.20). The greatest benefit to the collision avoidance algorithm is the reduction of other behaviors to the bare minimum. Ensuring the UAV substantially slows down enough to make weaving through other UAVs possible. It may be the case that simply changing the settings of the collision avoidance behavior based on UAV-to-UAV distance would improve collision avoidance at close proximities. Or in addition to the current collision avoidance algorithm, integration of a distance-based collision avoidance weight below a certain UAV-to-UAV distance as it was presented in [6], [7] for improved close proximity avoidance.

This issue, where assigned formation positions were inefficient due to the current positions of the UAV compared to before reaching altitude was later resolved by ensuring formation assignments occurred once the UAV reached the goal altitude.

	$UAV\ 1$	$UAV\ 3$	$UAV\ 3$
timestamp [sec]	10.296	10.334	10.163

Table 9.5. Timestamps for each UAV upon reaching the goal altitude.

w_{goal}	w_{form}	w_{alt}	$w_{c-avoid}$
0.5	0.85	1.0	0.2

Table 9.6. (w_i): Behavior weight settings for the current simulation case.

Collision Avoidance Settings	
$t_{collision_{min}}$	Minimized behavior weights
0.5	w_{goal} : 0.1 w_{form} : 0.1

Table 9.7. Collision avoidance settings adjusted for the current simulation case. $(t_{collision_{MIN}})$: Minimum value allowed for calculated time to collision in collision avoidance behavior. $Minimized\ behavior\ weights$: The minimum value set for other behaviors when below minimum collision avoidance boundary.

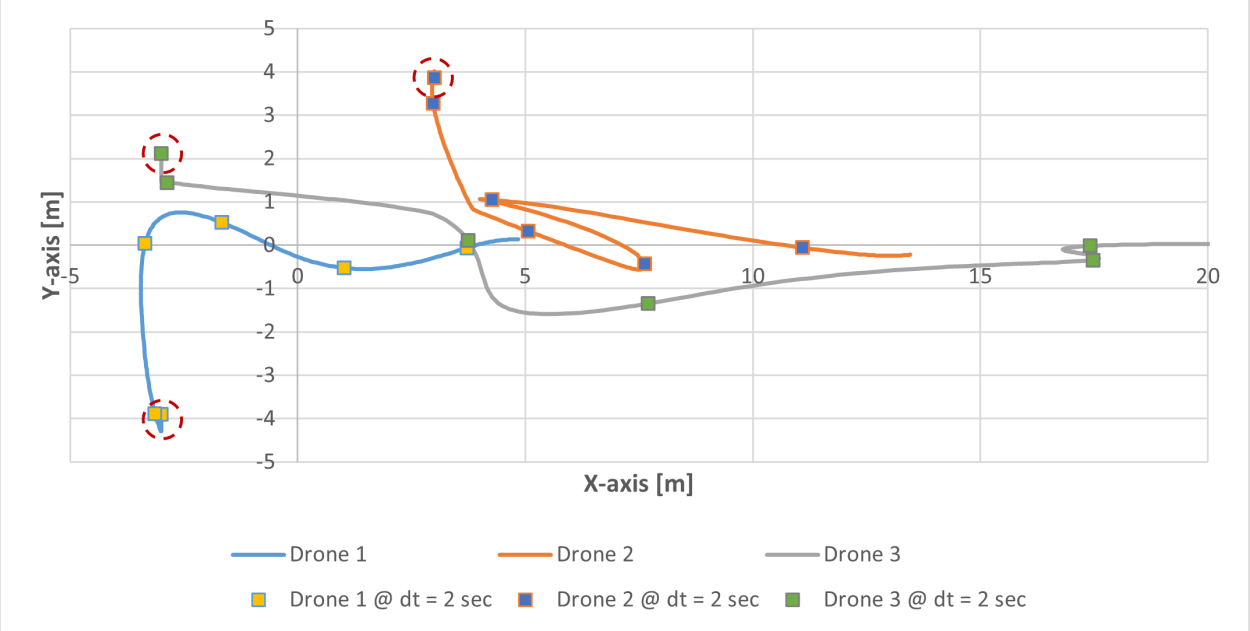


Figure 9.21. Reformation test, as discussed in section 9.1.1, zoomed in on waypoints 7-9 (fig. 9.1). Dashed circles indicate the UAV's spawn location into the simulation world. To visualize speed of motion, the UAV positions are marked for every 2 seconds of elapsed time.

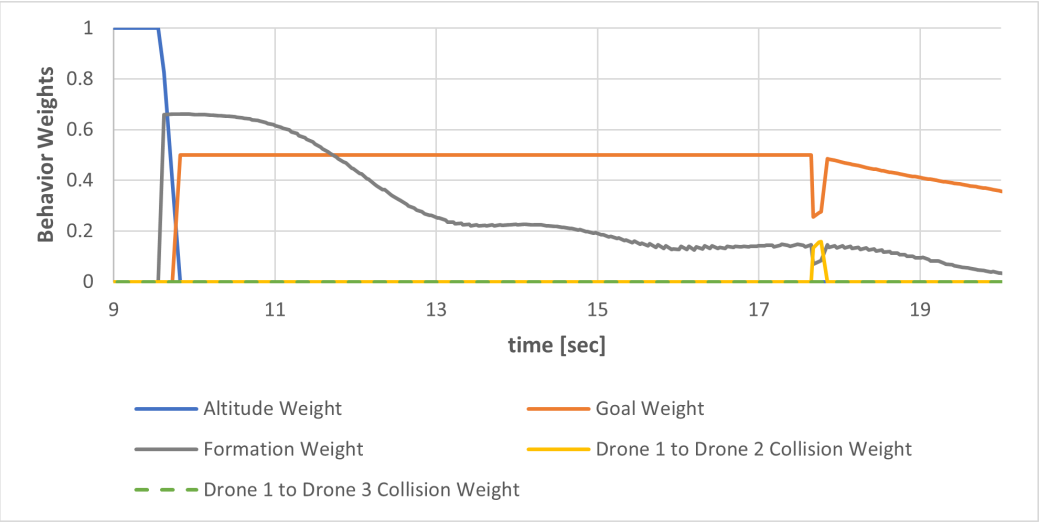


Figure 9.22. Behavior weights (f) of UAV 1 during initial formation formulation upon reaching goal altitude, showing collision avoidance behavior in effect.

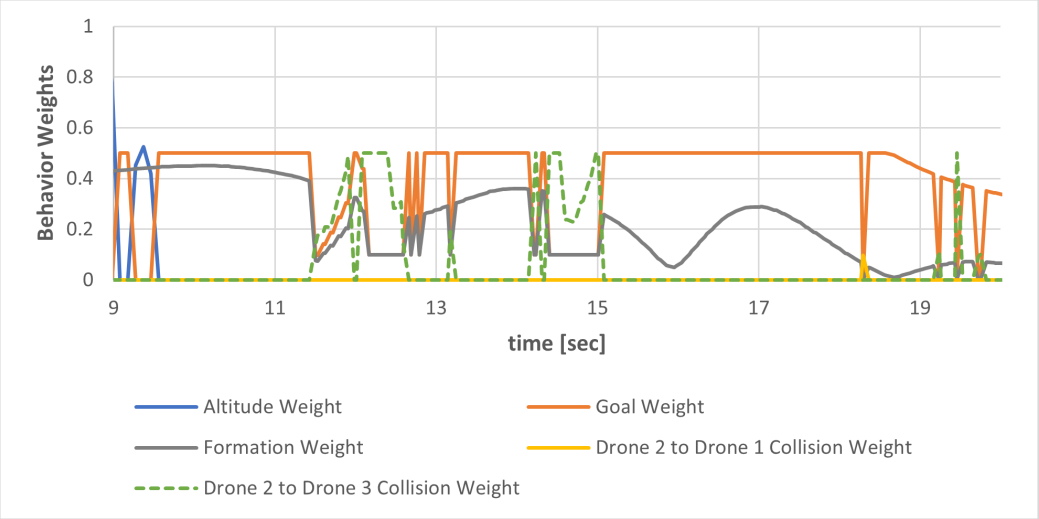


Figure 9.23. Behavior weights (f) of UAV 2 during initial formation formulation upon reaching goal altitude, showing collision avoidance behavior in effect.

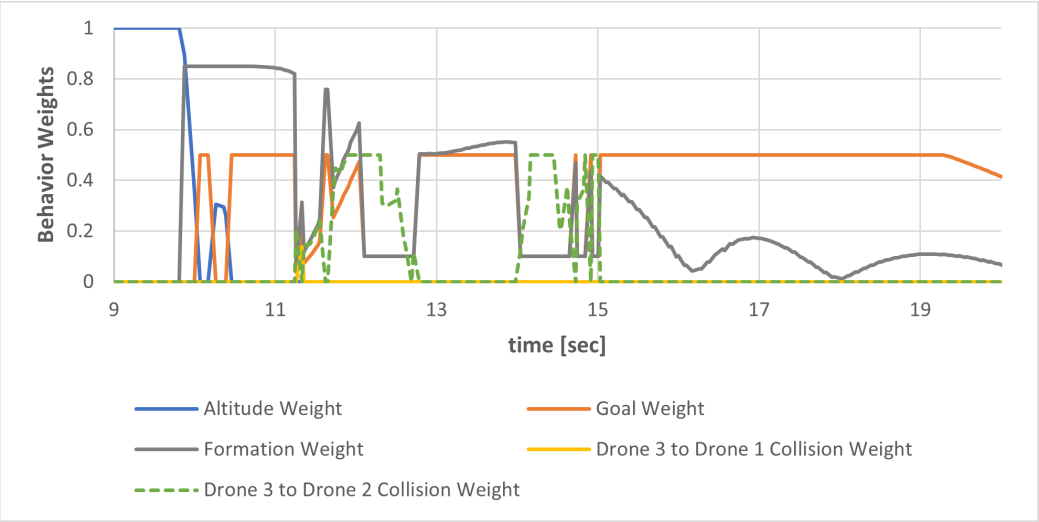


Figure 9.24. Behavior weights (f) of UAV 3 during initial formation formulation upon reaching goal altitude, showing collision avoidance behavior in effect.

9.2 UAV Failure within Formation

An often-discussed benefit of strictly decentralized swarms is not having a single point of failure. But, to maintain formations, a unifier is needed. In this thesis, shared knowledge of each UAV'S position with a formation center reference is used. Without failure handling procedures, the difference between a centralized formation is the survivability of each UAV. Instead of all UAVs failing or stranded, the UAVs could continue the mission or return to the emergency base. However, investigating and addressing the various failure modes is outside the scope of this thesis. In general, failures that affect the formation:

- Total mechanical failure.
- Partial failure, drift from goal formation position
- Communication failure

The current design of the swarm does not have failure response procedures. Tests for partial and total mechanical failure were conducted to examine swarm response.

To simulate total mechanical failure (section 9.2.1), at $time = 35\ sec$, the motor speed commanded is set to zero (fig. 9.25). The simulation result shows UAV 2 and 3 oscillating between maintaining formation and heading towards the waypoint (fig. 9.26). Looking at the behavior outputs, the goal-seeking behavior remains constant while the formation maintenance behavior keeps the UAV near the formation position with reference to the swarm center, including the downed UAV (fig. 9.27).

For addressing total UAV failures, once at operating altitude, any UAV in the formation that is outside or a set boundary minimum for altitude could be ignored from the formation. Then the planner can take the initialization data for the mission and adjust the mission with the available set of UAV.

To simulate partial mechanical failure (section 9.2.2), at time $time = 35\ sec$ motors 1 and 2 of UAV 1 are set to 25% of commanded motor speed. The simulation results show UAV 1 drifting farther from the waypoint, and the other UAV maintain formation between each other and slowly drifting towards the moving formation center (including the faulty UAV) (fig. 9.28). Looking at the behavior outputs for UAV 3, it can again be seen that the formation behavior overtakes the goal-seeking behavior (fig. 9.29).

To address this failure mode, a separate solution from checking altitude is necessary. A proposed solution would be to look for the outlier of all the UAV in the swarm using the Z-score. Each formation would require a different solution due to the geometry of the formation. If any single UAV deviates from the formation greater than a set Z-score value, then the UAV can be ignored from the formation. Again, the planner can then take the initialization data for the mission and adjust the mission with the available set of UAV.

9.2.1 Simulation 1: Total Failure of UAV 1

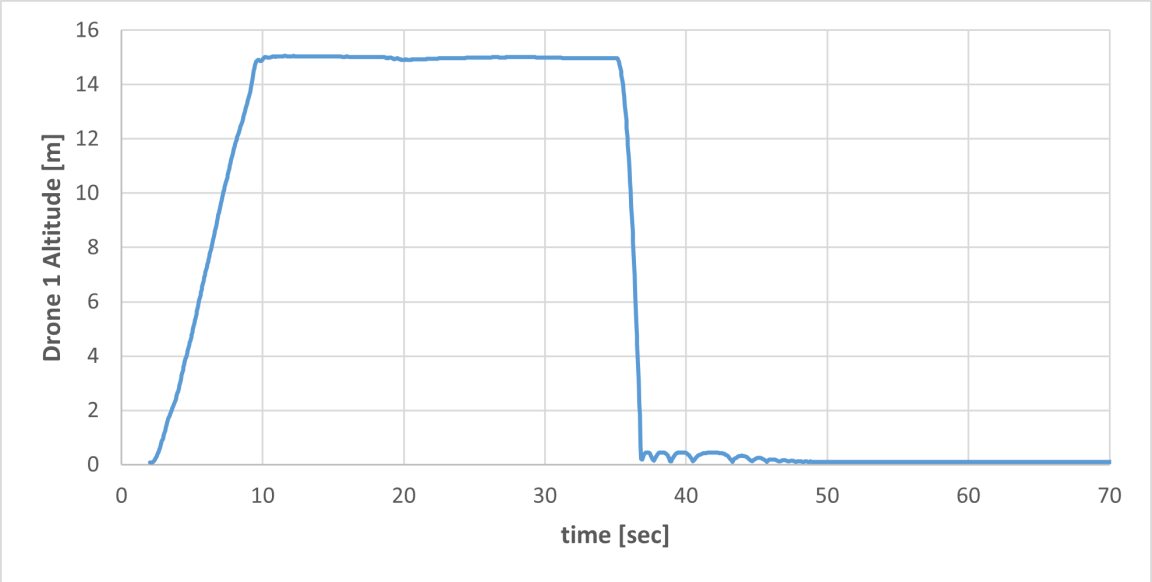


Figure 9.25. Altitude of UAV 1 with respect to time. The moment when total failure of all motors for UAV 1 is exhibited.

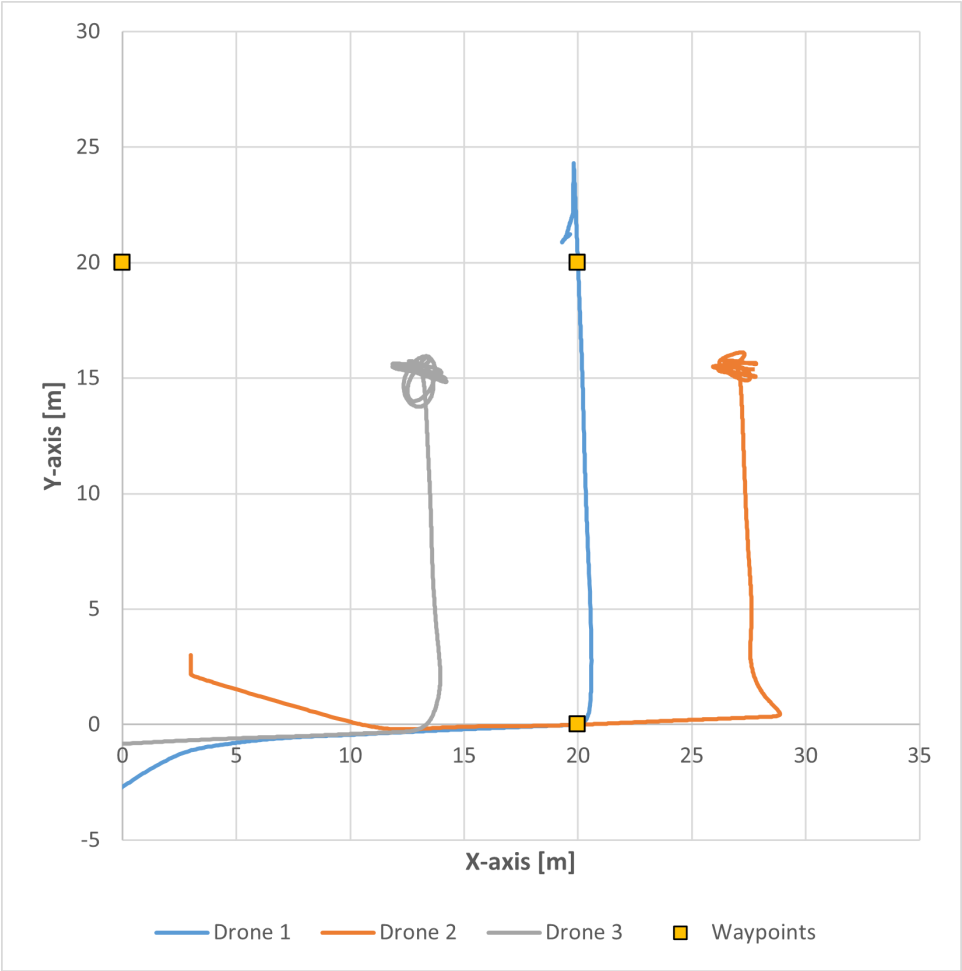


Figure 9.26. Failure test with induced total failure of UAV 1 during reformation test sequence (fig. 9.2) at time $t = 35\ sec$. All UAV positions are exhibited during this simulation.

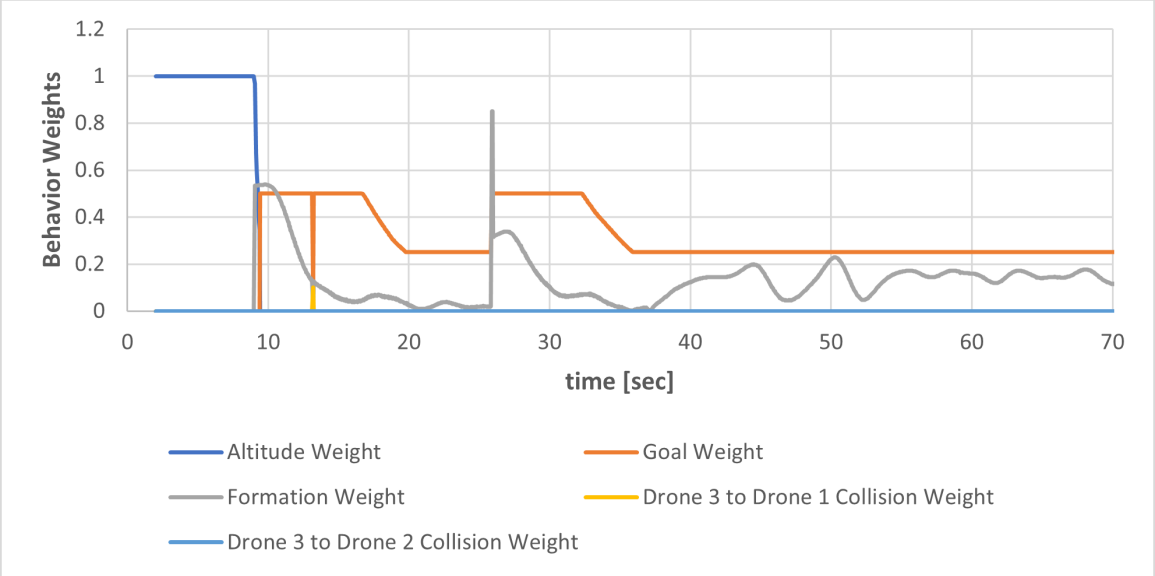


Figure 9.27. Behavior outputs of UAV 3 during failure test (fig. 9.26) with induced total failure of UAV 1 during reformation test sequence (fig. 9.2) at time $t = 35\ sec$.

9.2.2 Simulation 2: Partial Failure of UAV 1

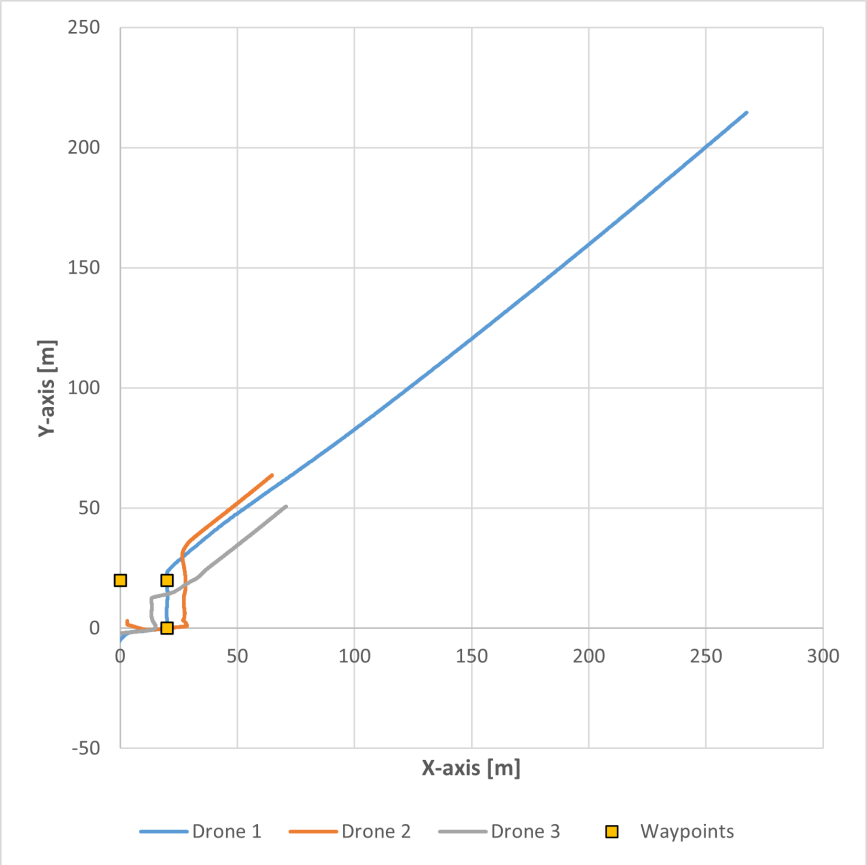


Figure 9.28. Failure test with induced 25% motor output (M1 & M2) failure of UAV 1 during reformation test sequence (fig. 9.2) at time $t = 35\ sec$. All UAV positions are exhibited during this simulation.

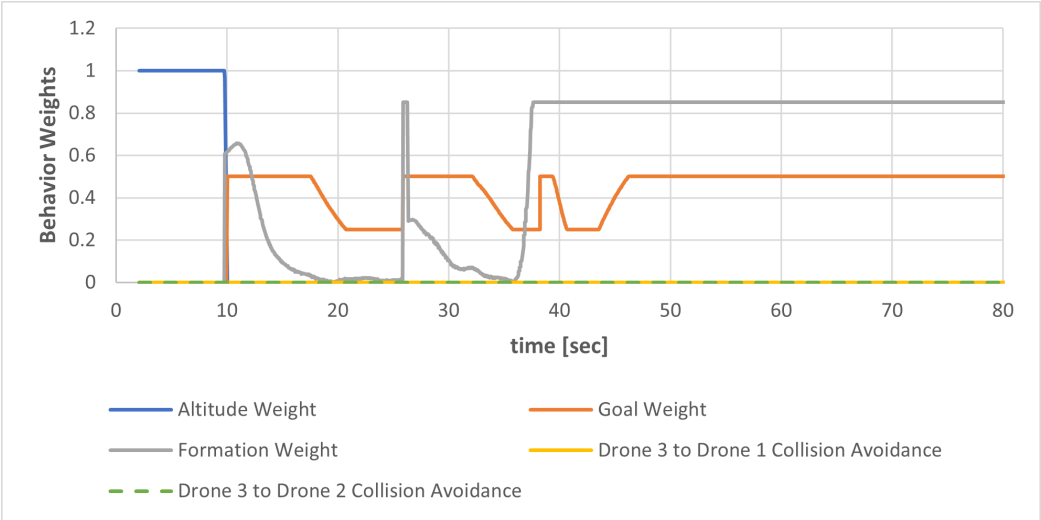


Figure 9.29. Behavior outputs of UAV 3 during failure test (fig. 9.28) with induced 25% motor output (M1 & M2) failure of UAV 1 during reformation test sequence (fig. 9.2) at time $t = 35\ sec$.

9.3 Surveying Mission

The scenario of this survey mission is that an earthquake has occurred in Los Angeles, California and first responders need to develop situational awareness of the area. To get visual feedback on the situation on the ground, photogrammetry is used as a first pass for surveying the area.

To make use of photogrammetry, while using the DJI P1 camera with a 25mm lens [11], a flight altitude is selected based on the amount of detail desired per pixel on the images taken. While higher altitudes are desired for covering more ground, smaller details could be lost. Additionally, at flight altitudes at/below $60.96\ meters$ $(200\ ft)$, motion blur on images taken can occur [12].

Another factor for consideration in photogrammetry surveying is the required amount of overlap between images taken. There is a balance between leveraging UAV flight time and taking enough images to not undersample features, so the recommended front and side overlap is 70%.

Lastly, the camera can be pitched to capture more 3-D features. The result is the requirement to capture more images to ensure enough image samples for properly reconstructing tall features [1] [12].

Multiple simulations were run, where from the local origin, a $250\ m$ by $250\ m$ located at $(100, 100)$ is mapped. A comparison is made between running the scenario with a single UAV, compared to a formation of UAV. Additionally, the UAV formation scenario is run with various behavior weight settings to study the effects on completing the mission scenario.

9.3.1 Simulation 1 (Single UAV)

In this simulation case, a single quadrotor UAV is running the behavior weights specified in table 9.8. This case will be used for comparison against the use of multi-agent UAV formation.

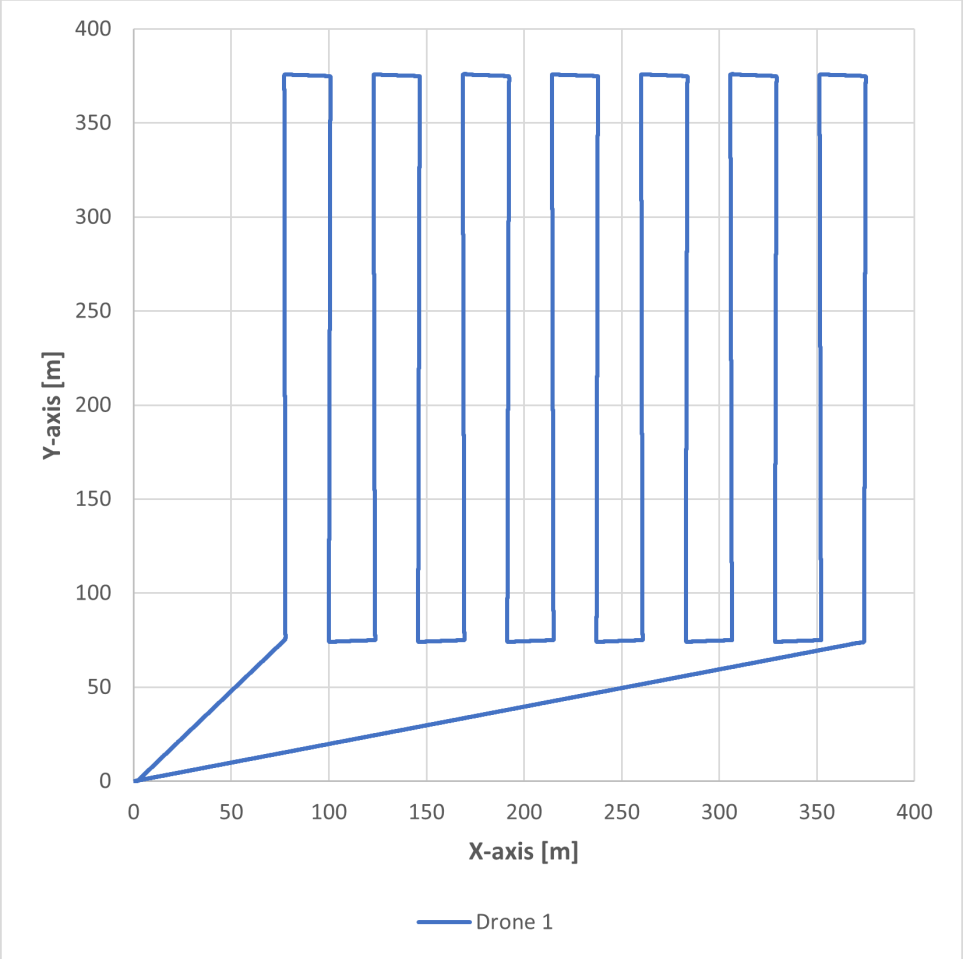


Figure 9.30. Surveying of 250m x 250m area located at (100,100) with a single UAV. Each corner of the UAVs path is a waypoint.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
0.5	N/A	N/A	1.0	46.33

Table 9.8. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.2 Simulation 2 (UAV Formation)

In this simulation case, a formation of quadrotor UAVs are running the behavior weights specified in table 9.9. This case is used for a direct comparison of completing the mission versus the single drone case.

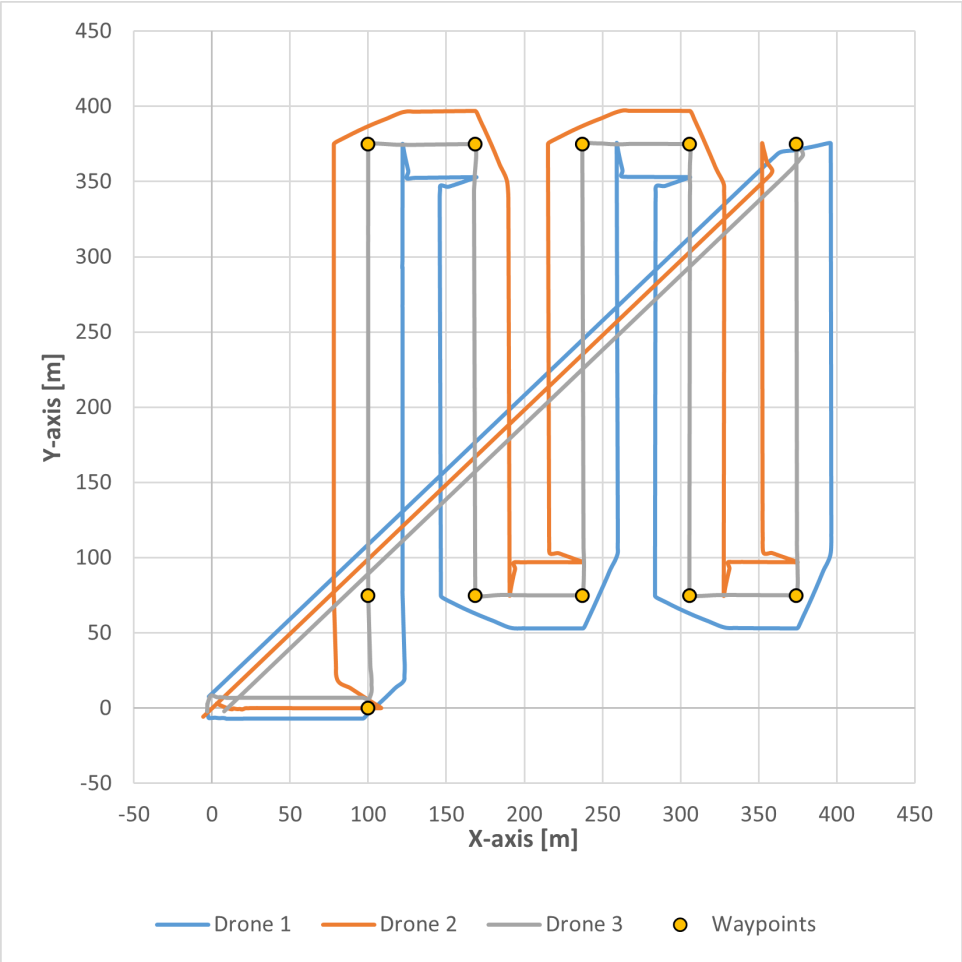


Figure 9.31. Surveying of 250m x 250m area located at (100,100) with a UAV formation.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
0.5	0.5	0.2	1.0	24.23

Table 9.9. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.3 Simulation 3 (UAV Formation)

In this simulation case, a formation of quadrotor UAVs are running the behavior weights specified in table 9.10. This case explores the increase of the formation weight for stricter adherence to formation.

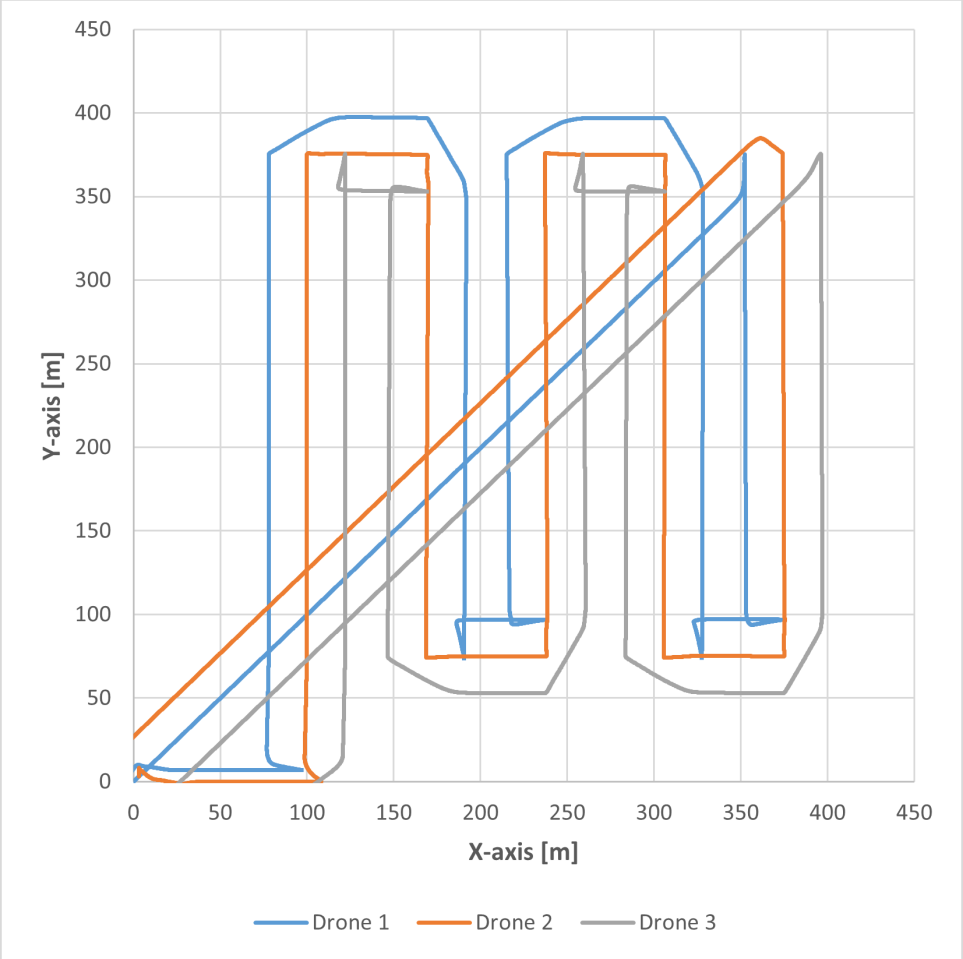


Figure 9.32. Surveying of 250m x 250m area located at (100,100) with a UAV formation.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
0.5	0.85	0.2	1.0	23.65

Table 9.10. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.4 Simulation 4 (UAV Formation)

In this simulation case, a formation of quadrotor UAVs are running the behavior weights specified in table 9.11. This case explores reducing the formation weight to zero to see the effect of the formation behavior on the time taken to complete the mission.

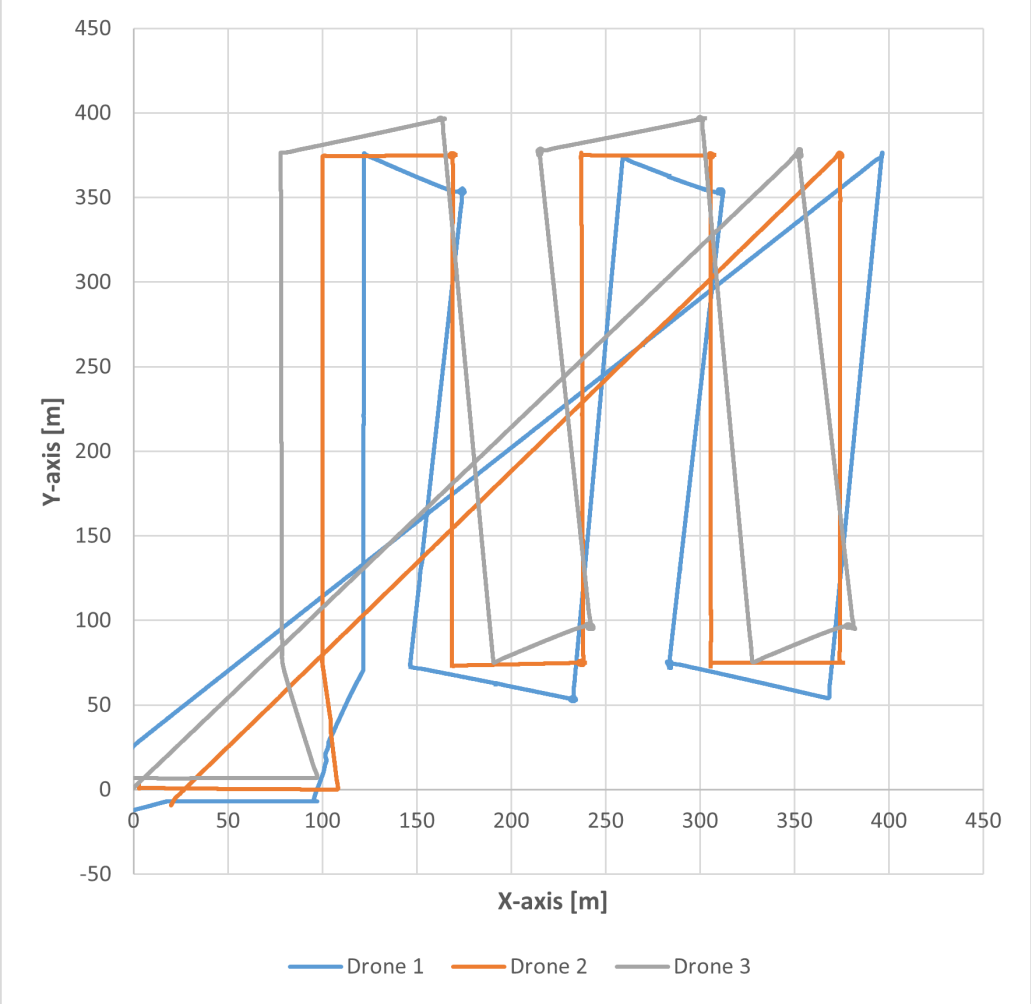


Figure 9.33. Surveying of 250m x 250m area located at (100,100) with a UAV formation.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
0.5	0.0	0.2	1.0	18.11

Table 9.11. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.5 Simulation 5 (UAV Formation)

In this simulation case, a formation of quadrotor UAVs are running the behavior weights specified in table 9.12. This case explores a minimized formation weight to compare the time to complete the survey mission to *simulation 5*.

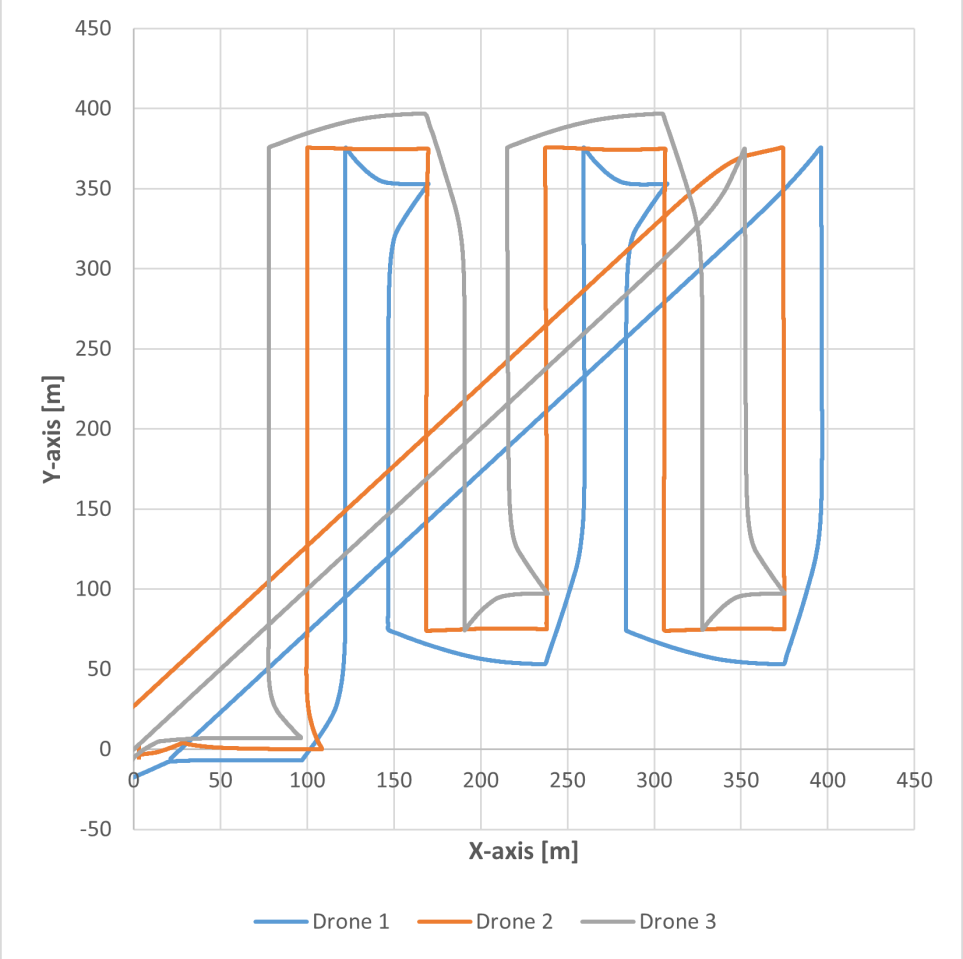


Figure 9.34. Surveying of 250m x 250m area located at (100,100) with a UAV formation.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
0.5	0.25	0.2	1.0	23.70

Table 9.12. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.6 Simulation 6 (UAV Formation)

In this simulation case, a formation of quadrotor UAVs are running the behavior weights specified in table 9.13. This case explores increasing the goal-seeking behavior to the previously determined ideal weights set in *simulation 2*.

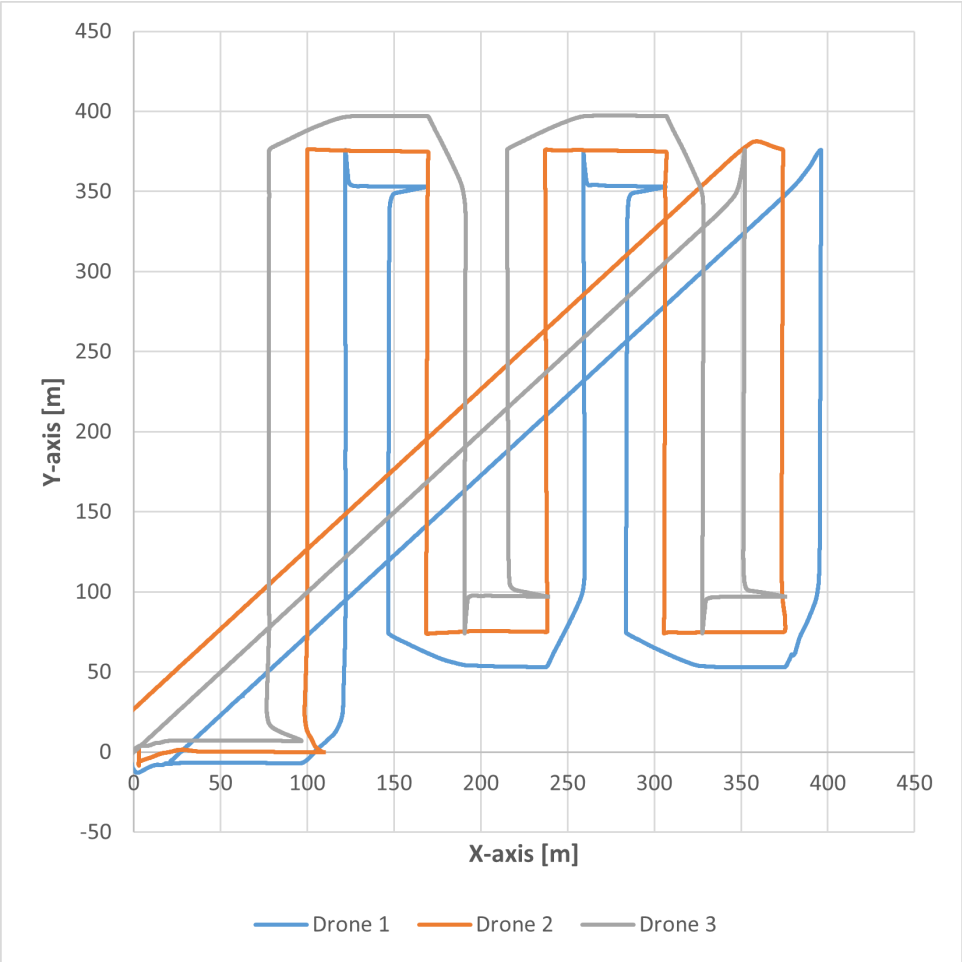


Figure 9.35. Surveying of 250m x 250m area located at (100,100) with a UAV formation.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
0.75	0.85	0.2	1.0	16.22

Table 9.13. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.7 Simulation 7 (UAV Formation)

In this simulation case, a formation of quadrotor UAVs are running the behavior weights specified in table 9.14. This case explores further increasing the goal-seeking behavior and is compared to results between *simulation 2* and *simulation 6*.

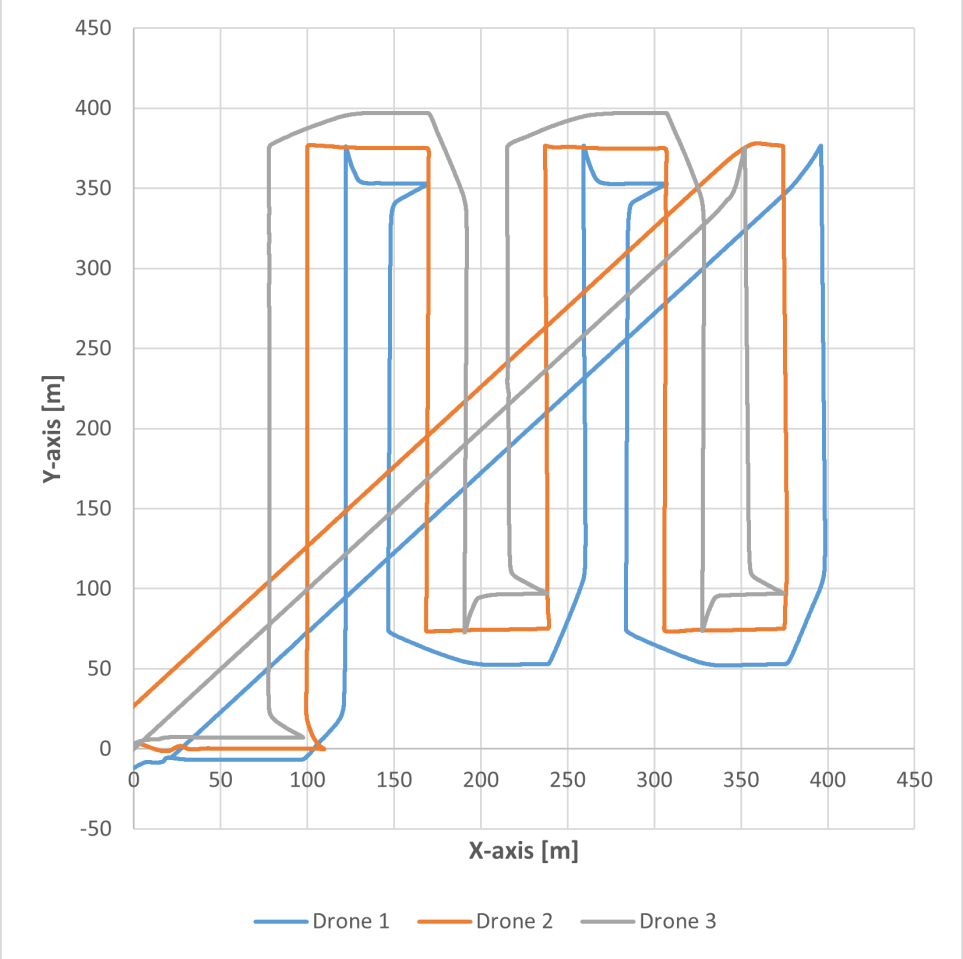


Figure 9.36. Surveying of 250m x 250m area located at (100,100) with a UAV formation.

w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
1.0	0.85	0.2	1.0	11.94

Table 9.14. Behavior weight settings and time to complete the survey mission with applied behavior weight settings.

9.3.8 Survey Mission Results

	$\#\ of\ UAV$	w_{goal}	w_{form}	$w_{c-avoid}$	w_{alt}	$Survey\ Time\ [min]$
$sim.\ 1$	1	0.5	N/A	N/A	1.0	46.33
$sim.\ 2$	3	0.5	0.5	0.2	1.0	24.23
$sim.\ 3$	3	0.5	0.85	0.2	1.0	23.65
$sim.\ 4$	3	0.5	0.0	0.2	1.0	18.11
$sim.\ 5$	3	0.5	0.25	0.2	1.0	23.70
$sim.\ 6$	3	0.75	0.85	0.2	1.0	16.22
$sim.\ 7$	3	1.0	0.85	0.2	1.0	11.94

Table 9.15. Survey mission results comparing single UAV to swarm with adjustments to formation and goal behavior weights.

Time taken to conduct the survey mission between *simulation 1* and *simulation 2* (table 9.15) had a 47.7% reduction in time. The expectation when running 3 UAVs to cover the same area was to have about a 66.67% reduction in time.

While running the simulation, an emergent behavior of waiting for straggling UAVs can be noted. This emergent behavior could be the reason for not meeting the expected time to complete the survey mission. So to investigate this, *simulation 5* (table 9.15) was run with a 50% reduction to the formation behavior. When compared to *simulation 2* the time taken to complete the survey was minimal (2.19% difference).

Since the changes were minimal, the effect was further investigated, and a reduction of the formation behavior weight (w_{form}) was set to 0.0 for *simulation 4*. Comparing *simulation 4* to *simulation 2*, there was a time reduction of 25.28%, and compared to *simulation 1* (single UAV) there is a 60.92% reduction in time, bringing it closer to the expected result. Some efficiency in getting to the waypoints may be lost while trying to maintain formation.

The problem with merely providing waypoints in formation patterns without maintaining formation is the necessity for additional waypoints to ensure a straight line path across the survey area. Any obstacle could throw the UAVs off their path producing unusable data points in mapping. Even when the formation behavior weight is increased from 0.25 (*simulation 5*) to 0.85 (*simulation 3*), the time to complete the survey mission is marginal in difference, with a 48.8% reduction in survey time for a formation weight of 0.25 in *simulation 5* and a 49.0% reduction in survey time for a formation weight of 0.85 in *simulation 3* when compared to the time to complete the mission with a single UAV *simulation 1*.

An additional factor to consider with *simulation 1* (table 9.15), the total run-time of a single UAV for surveying an area of 250m by 250m exceeds the expected amount of time the UAV can fly provided a fresh set of batteries and the camera payload [13]. Whereas the formation of UAVs quickly performs the task within the operating time detailed in the specifications.

In terms of completing the survey mission quickly, the goal-seeking behavior weight could be increased. Setting the weight of the goal behavior (w_{goal}) to 1.0 (*simulation 7*), speed up the survey mission completion from $23.65min$ (*simulation 3*) to $11.94min$ (*simulation 7*), but while trying to maintain the formation, the UAVs are constantly oscillating due to competing behaviors for reaching the waypoint and keeping the formation to the average center of all UAVs in the swarm. Additionally, formation adherence during turns is less adhered to. Reducing the goal-seeking behavior weight (w_{goal}) to 0.75 reduced the amplitude of the oscillation for the UAV motion. Smooth flight is a key aspect in producing useful photogrammetry data [28][1][12], therefore this greatly affects the number of usable images. In this survey mission scenario, a leader-follower approach may be best for smoother flight at higher speeds. Optionally, adjusting the behaviors to introduce some damping with each respective behavior as it is done for [2]. For surveying scenarios at lower altitudes, slower flight speeds are necessary, which this thesis' framework can accommodate.

Section 10. Conclusion

This thesis aims to provide a piece of the complete picture of a swarm of UAVs used for multiple objectives in an emergency scenario. The thesis focuses on a swarm of decentralized UAVs cooperating to get into formation and complete surveying tasks.

This thesis based the swarm with the DJI Matrice 300 RTK due to its commercial availability and current use in mapping. But as it was pointed out in [1], mapping could be accomplished with lower-cost equipment at the cost of quality. The limitations would be the same regardless, which is the flight time with the necessary payload. Using a decentralized control scheme for the swarm does limit the cost reductions with the necessity for all UAVs to possess a transmitter/receiver and enough computing ability.

Limitations imposed by sensor selection would require some adjustment to the settings of the implemented controllers. The control scheme this thesis provides is simple in complexity and implementation. PID gains for the UAV quadrotor can be set up discretely. An example of the order of setting gains:

- Altitude Controller
- Roll and Pitch Controller
- Yaw Controller

Then the behavioral controller gains can be set discretely as well, passing data to the altitude and attitude controllers.

With this thesis' implementation of the behavioral control scheme, the UAVs were able to independently maintain formation with each other while only communicating position and velocity data to each other. The error in formation maintenance was at a maximum of 0.1 meters in the direction of travel, which would not be enough to affect mapping through photogrammetry. Some careful consideration is required when changing formation before mapping an area since the formation travels from its formation center to the waypoint. Reformation is not guaranteed to settle with the formation center along the axis between the two waypoints, therefore the formation would deviate from the expected trajectory. Therefore, space must be given to the UAV formation to reformulate formation into the formation for surveying (single-row in this thesis' example)(see fig. 9.31).

When considering chaotic situations, the UAVs might need to rely on the collision avoidance behavior. But there must be consideration of the behavior weights set with the set formation spacing since collisions are more likely the more dense a formation is, and if weights are set high enough with collision avoidance detection boundaries reduced, would result in collisions between the UAVs.

When comparing the specifications of this controller scheme to that of the commercial specifications of the DJI Matrice 300 RTK, the maximum velocities the UAV could fly at were

less than 25% of the commercial option. But in photogrammetric mapping, images taken at altitudes higher than 61 meters (200 ft) can be taken at high velocity (variability dependent on actual setup)[28]. But once tall features and/or the flight altitude is below 61 meters (200 ft), flight speed must be reduced to prevent image blurring [28]. So, for these situations, a formation of UAVs can cover significantly more area than a single UAV.

A major consideration for the use case of a formation of UAV is the lack of any post-processing software for photogrammetry with a formation of UAV commercially available. Software would have to be tailored to the use case of a UAV formation taking images and the challenges that come with locating images taken by each UAV and relating them within tolerable accuracy for use cases such as mapping a city in a post-earthquake emergency scenario.

10.1 Future Work

Due to the aim of this thesis being a small part of the greater scheme of a multi-faceted flexible emergency response drone swarm, many aspects can be expanded from this thesis.

To address differences in capability to the specification of the DJI Matrice 300RTK, future work can be done on the integration of other controllers for the UAV altitude and attitude, such as short horizon MPC (Model Predictive Control) or LQR (Linear Quadratic Regulator).

Expanding upon this work, stationary and moving obstacle avoidance can be introduced for surveying UAV formation. For moving obstacles, a similar approach used for collision avoidance can be extended for moving-obstacle avoidance.

To further improve on failure modes, health monitoring, and failure procedures could be introduced to all UAVs to further isolate failures in the formation of the sole problematic UAV. This thesis has already commented on some proposed solutions for failures that produce drift from formation and total failure (sections 8.2.2 and 9.2). Expanding on the capability from purely surveying in formation, the UAV swarms can be set up where a single or a couple of UAVs leave the formation to perform a higher priority task while the rest of the formation continues surveying the area as defined by the mission requirements.

The possibilities for expanding upon this thesis are seemingly endless as other works expand the capability of quadrotor UAVs and develop solutions for decentralized cooperation.

References

[1] F. Nex and F. Remondino, "UAV for 3D mapping applications: A review", en, *Appl Geomat*, vol. 6, no. 1, pp. 1–15, Mar. 2014, Company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 1 Publisher: Springer Berlin Heidelberg, ISSN: 1866-928X. DOI: 10.1007/s12518-013-0120-x. [Online]. Available: https://link-1springer-1com-1000096ut365a.eczyt.bg.pw.edu.pl/article/10.1007/s12518-013-0120-x (visited on 08/28/2023).

[2] J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers", eng, *IEEE transactions on robotics and automation*, vol. 19, no. 6, pp. 933–941, Place: New York, NY Publisher: IEEE, ISSN: 1042-296X. DOI: 10.1109/TRA.2003.819598.

[3] W. Shyy, E. Atkins, A. Ollero, A. Tsourdos, and R. Blockley, *Unmanned Aircraft Systems*, eng. New York: John Wiley & Sons, Incorporated, 2017, ISBN: 978-1-118-86645-0.

[4] J. Zhang, J. Yan, and P. Zhang, "Fixed-Wing UAV Formation Control Design With Collision Avoidance Based on an Improved Artificial Potential Field", eng, *IEEE access*, vol. 6, pp. 78 342–78 351, 2018, Place: PISCATAWAY Publisher: IEEE, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2885003.

[5] G. Lee and D. Chwa, "Decentralized behavior-based formation control of multiple robots considering obstacle avoidance", eng, *Intelligent service robotics*, vol. 11, no. 1, pp. 127–138, 2018, Place: Berlin/Heidelberg Publisher: Springer Berlin Heidelberg, ISSN: 1861-2776. DOI: 10.1007/s11370-017-0240-y.

[6] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams", eng, *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998, Place: PISCATAWAY Publisher: IEEE, ISSN: 1042-296X. DOI: 10.1109/70.736776.

[7] D. Xu, X. Zhang, Z. Zhu, C. Chen, and P. Yang, "Behavior-Based Formation Control of Swarm Robots", eng, *Mathematical problems in engineering*, vol. 2014, pp. 1–13, 2014, Place: LONDON Publisher: Hindawi Publishing Corporation, ISSN: 1024-123X. DOI: 10.1155/2014/205759.

[8] *How Drones, Phones and Mini Cell Towers Can Benefit Public Safety*, en. [Online]. Available: https://enterprise.verizon.com/resources/articles/s/how-drones-phones-and-mini-cell-towers-can-benefit-public-safety/ (visited on 08/11/2023).

[9] K. Bush, "It's a Bird, a Plane... It's a Flying Cell Tower", en-US, *Wired*, Nov. 2018, ISSN: 1059-1028. [Online]. Available: https://www.wired.com/brandlab/2018/11/bird-plane-flying-cell-tower/ (visited on 08/11/2023).

[10] "Key technologies for safe and autonomous drones", en-US, *Microprocessors and Microsystems*, vol. 87, p. 104 348, Nov. 2021, Publisher: Elsevier, ISSN: 0141-9331. DOI: 10.1016/j.micpro.2021.104348. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0141933121005056 (visited on 08/11/2023).

[11] *Specs - Zenmuse P1 - DJI Enterprise*, en. [Online]. Available: https://enterprise.dji.com/zenmuse-p1/photo (visited on 08/14/2023).

[12] *How to Make Great Drone Maps and Surveys - Civil Tracker*, en-US, Section: Support, Mar. 2021. [Online]. Available: https://civiltracker.xyz/how-to-make-great-drone-maps/ (visited on 08/14/2023).

[13] *Specs - MATRICE 300 RTK - DJI Enterprise*, en. [Online]. Available: https://enterprise.dji.com/matrice-300/photo (visited on 06/18/2023).

[14] *E2000 - DJI*, en. [Online]. Available: https://www.dji.com/pl/e2000 (visited on 06/18/2023).

[15] Mike1024, *English: A diagram showing ECEF, ENU, Longitude () and Latitude () coordinates and the relationship between them.* Feb. 2010. [Online]. Available: https://commons.wikimedia.org/wiki/File:ECEF_ENU_Longitude_Latitude_relationships.svg?uselang=en#Licensing (visited on 06/20/2023).

[16] *Matrice 300 RTK - Industrial grade mapping inspection drones - DJI Enterprise*, en. [Online]. Available: https://enterprise.dji.com/photo (visited on 06/20/2023).

[17] O. Liang, *How to Choose FPV Drone Motors*, en-GB, Apr. 2023. [Online]. Available: https://oscarliang.com/motors/ (visited on 06/22/2023).

[18] *K&J Magnetics - Specifications*. [Online]. Available: https://www.kjmagnetics.com/specs.asp (visited on 06/22/2023).

[19] H. Elkholy and M. Habib, "Dynamic Modeling and Control Techniques for a Quadrotor", in Jan. 2015, ISBN: 978-1-4666-7387-8. DOI: 10.4018/978-1-4666-7387-8.ch014.

[20] W. Johnson, *Rotorcraft Aeromechanics* (Cambridge aerospace series), eng. New York: Cambridge University Press, 2013, vol. 36, ISBN: 978-1-107-02807-4. DOI: 10.1017/CBO9781139235655.

[21] *UBC ATSC 113 - Standard Atmosphere-Pressure*. [Online]. Available: https://www.eoas.ubc.ca/courses/atsc113/flying/met_concepts/02-met_concepts/02a-std_atmos-P/index.html (visited on 06/23/2023).

[22] T. Luukkonen, "Modelling and control of quadcopter", en, *Independent research project in applied mathematics, Espoo*, vol. 22, no. 22, 2011.

[23] M. Misin, "Model Predictive Controller of a UAV using LPV Approach", English, Ph.D. dissertation, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Jan. 2020.

[24] D. Lee, H. Jin Kim, and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter", eng, *International journal of control, automation, and systems*, vol. 7, no. 3, pp. 419–428, Place: Heidelberg Publisher: Institute of Control, Robotics and Systems and The Korean Institute of Electrical Engineers, ISSN: 1598-6446. DOI: 10.1007/s12555-009-0311-8.

[25] A. Eltayeb Taha, M. Rahmat, J. Mu'azu, and M. Musa, "Feedback linearization and Sliding mode control design for quadrotor's attitude and altitude", Apr. 2019.

[26] X. Chen, J. Tang, and S. Lao, "Review of unmanned aerial vehicle swarm communication architectures and routing protocols", eng, *Applied sciences*, vol. 10, no. 10, pp. 3661–, 2020, Place: Basel Publisher: MDPI AG, ISSN: 2076-3417. DOI: 10.3390/app10103661.

[27] D. F. Crouse, "On implementing 2D rectangular assignment algorithms", *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 4, pp. 1679–1696, Aug. 2016, ISSN: 0018-9251. DOI: 10.1109/TAES.2016.140952. [Online]. Available: http://ieeexplore.ieee.org/document/7738348/ (visited on 09/12/2023).

[28] *Taking Images for Photogrammetry*, en. [Online]. Available: https://resources.mapware.com/knowledge/taking-images-for-photogrammetry (visited on 08/14/2023).

Acronyms

BF Body-Fixed 28–33, 41

DARPA Defense Advanced Research Projects Agency 9

DGPS Differential Global Positioning System 6

ECEF Earth-Centered, Earth-Fixed 19

EF Earth-Fixed 28–30, 33, 34, 36, 37, 44, 45, 48, 56

ENU East-North-Up 19

IMU Inertial Measurement Unit 29

INS Inertial Navigation System 6

LQR Linear Quadratic Regulator 96

MANET Mobile Ad-Hoc Network 15

MPC Model Predictive Control 96

PPK Post Processing Kinematics 16

ROS2 Robot Operating System 2 17, 18

UAV Unmanned Aerial Vehicle 2–8, 11, 13–15, 17, 20, 29, 35–38, 40, 41, 43, 45, 47, 49–54, 56–72, 74–77, 79–96

UGV Unmanned Ground Vehicle 8–10

WSL Windows Subsystem for Linux 17