Modular Recovery Control and Obstacle Avoidance for a UAV-Payload System in Constrained Indoor Environments

A non-thesis project report submitted to the

Graduate School

of the University of Cincinnati

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Department of Aerospace Engineering and Engineering Mechanics

of the College of Engineering and Applied Sciences

by

Akshay Elangovan

B.Tech in Mechanical Engineering, VIT University Chennai Campus

July, 2023

Research Advisor: Catharine McGhan, Ph.D.

ABSTRACT

Unmanned Aerial Vehicles (UAVs) have become increasingly popular over the last decade with the advent of multirotors for applications such as reconnaissance, surveying, photography, and transportation of small packages. A multi-UAV application that has become quite common is of payload transportation, where several multirotor systems work collaboratively to transport a single payload. While existing control systems are equipped to handle tasks involving path planning and navigation for multi-UAVs, they are not equipped to handle failure cases such as loss of team members. In a multi-UAV slung payload system, suitable recovery methods are important because if a team member fails, the stability of the platform reduces. Considering an extreme failure case where a multi-UAV slung payload system fails down to a single UAV with the payload attached to it, disturbance to the system with the swinging payload could be enough to destabilize the platform and risk collision with nearby obstacles. In situations like this, it is necessary to have a recovery control method that can help the UAV arrest the swinging and stabilize the system while avoiding any obstacles.

This research work implements and tests a control architecture for recovery of a destabilized UAV-slung payload system by characterizing the different behaviour-actions required for stabilization after a failure down to a single-UAV in an indoor environment scenario. The control architecture used in this research is based on fuzzy logic and uses genetic fuzzy systems to train action-primitive behaviours into a fuzzy rule base that can be equipped with behaviours for a resilient failure recovery and transport mission for a UAV-payload system that is detached from a multi-UAV team. The steps to design, train and simulate such a modular system are presented in this report along with a detailed discussion about validation, robustness testing, and future work.

TABLE OF CONTENTS

1. INTRODUCTION

Over the last two decades, unmanned aerial vehicles (UAVs) specifically rotorcrafts have undergone a significant engineering transition from being a complex flight control problem to being used for modern commercial applications such as reconnaissance [1], surveillance and monitoring [2], agriculture [3], videography [4], and even recreational activities like Flying First-Person View (FPV) racing [5]. With miniaturization of technologies meant for rotorcraft, the advent of multirotor systems such as quadrotors and hexacopters was noted. Multirotor UAV systems presented simpler dynamics compared to traditional helicopter platforms and were much more stable along with the ability to hover compared to fixed wing aircrafts. The versatility of control applications such as path planning and manipulation made multirotor UAVs more suitable than fixed-wing UAVs for low-speed applications. While the multirotors used in these applications are usually small in size (less than 12 inches in diameter mostly) to have good agility, maneuverability and speed, there can be some benefits from using a physically larger design. The most obvious one is increased flight time, thanks to better battery technology (e.g., Li-Ion). The second advantage is stronger thrust abilities, which allows increased payload carrying capacity for additional gadgetry or instruments, such as cameras, spraying equipment, manipulator arm [6] or just payload transportation in general. Payload transportation using multirotors have outdoor and indoor applications which vary in terms of complexity. Many companies have already tested VTOL (Vertical Take-Off and Landing) capable multirotor systems for payload delivery, with a focus on replacing ground vehicles for last mile services. With some commercial experiments for payload delivery limits to small payload capacities of 10 lbs. or less, UAVs are currently being designed for payload transportation of medical first-aid apparatus and e-commerce goods in outdoor environments [7][8][9]. Examples of indoor applications of such payload carrying multirotors include warehouses that can benefit from a large-scale use to organize or move packages, and/or to complement automated inventory scanning capabilities [10]. This can potentially bring end-to-end autonomy for warehouses where several ground robots and multirotors can interact and manage

inventory collaboratively. Recent advances in control theory for drone applications have made it accessible to control a drone and add robotic intelligence such as sense-and-avoid, path planning, and target tracking.

In the context of payload transport, a theme that is recurrent is interest in the use of multi-UAV systems for a variety of reasons, including increasing payload capacity, handling larger sized objects, and collaborative robotics applications, to name a few [11]. While existing research that handles multi-UAV and payload control, path planning, and tracking are significant, it becomes evident that we need to have some failure recovery systems in play before rolling out these systems into the real world. Failures such as loss of a member UAV in a multi-UAV team (e.g., due to collision, system wear, or equipment failure) should not mean risking the rest of the team along with the payload. This greatly necessitates a back-up controller or recovery system that can handle the effects of failure, restore the system to stability, and continue its original mission.

1.1 Motivation

Existing systems have demonstrated slung payload as a favored segment in payload transportation such as [12][13]. While these designs are good in theory and have been demonstrated in a lab space, they should be tested for expected failures during flight and have defined strategies to recover from failure for redundancy and mission resilience. The advantage of robustness in multi-UAV teams led us to investigate multi-UAV slung payload systems. When a multi-UAV team suffers the loss of a team member, the rest of the team would have to counter the sudden loss of stability or thrust. The reactive actions that the remaining team members must perform are quite intuitive when we replace these members with a human analogy of people carrying furniture. But in a slung payload UAV system, such reactions could be hard to program as they are very dependent on the physics of the model. We hypothesized that using genetic fuzzy systems (GFS), those intuitive motions can be learned. To retain the idea of intuitiveness, the UAV's basic control system will not receive any inputs other than its own position and velocity.

Figure 1: Human analogy of a two-team payload transport

[Image is free to use; https://www.pexels.com/photo/a-couple-lifting-a-couch-7218600/]

Literature has many good examples of genetic algorithms (GA) being used to train fuzzy control systems setting up the right conditions for specific applications [14][15][16]. This thesis draws motivation from neural networks that can store information through their several neurons and learn based on objectives [17]. By imagining the fuzzy parameters as neurons, we wished to see if a similar effect can be achieved. As this document progresses, we will define the scope and constraints for this idea to set the expectations for this graduate level research endeavor.

1.2 Objective

The primary objective of the research study is to propose a methodology for autonomous recovery when a multi-UAV payload system fails down to a single UAV with a slung payload. Recovery may include but is not limited to stabilization of the UAV and payload, while using awareness of surroundings to avoid any collisions. The actions expected from the control scheme during recovery include:

a) Recover a disturbance to the system by stabilizing the payload and returning the UAV to hover

b) Track waypoints to navigate the UAV-payload system as necessary

c) Avoid obstacles / hazards in a constrained indoor environment

An early milestone for this thesis is to identify a blueprint fuzzy control system that will be used to train different modular behavior actions necessary for the objective. Using genetic fuzzy training, behaviors to stabilize the payload and track waypoints will be 'learned' by the chosen control system. Initial training would verify the capacity of genetic fuzzy systems to learn desired behaviors and the trained system's limitations. Then, policies that would help access these behaviors in the context of obstacle avoidance were developed. The control scheme was tested in different use case scenarios and validated for reasonable performance. Through these tests, the viability of having different behaviors trained into fuzzy controllers was analyzed.

1.3 Problem Statement

A UAV-slung payload system is a popular control problem for aerial small payload delivery and has been studied and developed to a significant extent [18][19][20]. As the payload is suspended from the UAV, obviously swinging can occur because of the dragging motion of the transport. To reduce such swinging, control algorithms have been developed where the UAV uses the payload's position as in [21][22] and make the necessary maneuvers to dampen oscillations. While these systems maintain fair control over the slung payload, researchers have considered multi-UAV slung payload systems for certain advantages. Multi-UAV teams offer more thrust to increase payload capacity, increase robustness in case of single UAV failure and have the payload connected to a stable platform that limits swinging oscillations. While from a safety standpoint, it may not be reasonable to encourage multi-UAV slung payload system to increase payload capacity; in cases where the failure of a team member UAV would mean failure of the entire system, it does make sense to use a multi-UAV team for slung systems in order to have more stability over the platform. Some failures that may cause this are a single system failure, cable breakage, etc. In such a system, we assume that each UAV can support the payload weight and that the multi-UAV arrangement under normal circumstances prevents the payload's swing.

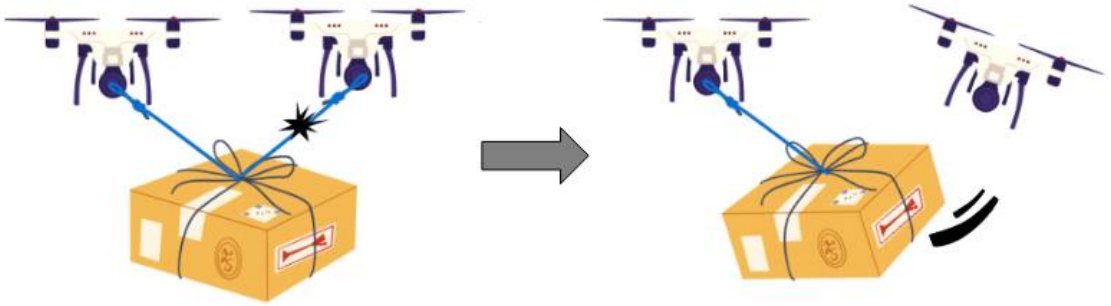


Figure 2: Failure of a 2-UAV-slung payload system to a single UAV-slung payload system

In this study, we consider a larger scenario of multi-UAV payload transport where multiple teams are transporting parts in an indoor environment such as warehouses and corridors. UAVs are given a trajectory to stay within and necessary information for navigation such as location of waypoints and obstacles. We are not dealing with scheduling or initial trajectory planning of this large system but choose to focus on a single team that happens to suffer a loss of team members. While prevention is key in drone safety, we consider an extreme failure case that would cause all but one UAV in the team to fail, which would likely cause the remaining single UAV and payload to swing out of control without a mitigation strategy. Since the remaining UAV is now possibly destabilized with a payload that is swinging out of its control, this is a failure that needs to be dealt with. As explained under the initial discussion, with a high-safety case requirement it is reasonable to assume that the single UAV would have enough thrust to support itself and the payload by design. So, a backup control system that can help with this type of failure is required which can: (1) stabilize the payload and UAV; (2) avoid any obstacles in the vicinity that the system could possibly run into because of the destabilized condition; and (3) continue to follow waypoints and navigate to safety, or even continue the original transport mission.

Nonlinearity in physics poses an interesting control challenge. Fuzzy logic offers inherent robustness and modularity as advantages that are suitable for this study. But the fuzzy rule base for desired behaviors may be too complex to manually deduce and hence genetic fuzzy systems are used. Genetic fuzzy systems that can work to recover the UAV-payload system in the given scenario are the expected outcome of this research. Payload stabilization and obstacle avoidance are of priority and will be demonstrated with

thorough testing. A description of the training conditions and their results along with sufficient testing for validation of the proposed modular system in the given scenario will be discussed.

1.4 Assumptions

Because this study involves experimenting with fault recovery and using genetic fuzzy systems in a method that is not conventionally intended, some assumptions were made to simplify the control problem. The payload is assumed to be a point mass, and the cable that connects the UAV to the payload is assumed to be massless while being attached at the center of gravity of the UAV. This makes the problem easy to model as the dynamics would resemble a single pendulum attached to the UAV. Other UAVs pre-failure are not modelled and assumed to not interfere with the single-UAV payload system under consideration. The cable is assumed to be always taut (cable tension > 0), keeping the slung payload condition true. This is reasonable given that the payload is suspended from the UAV and the UAV is not expected to make sudden maneuvers that might cause the tension in the cable to hit zero. The environment considered for this study is indoors so external disturbances that are likely in an outdoor setting (such as birds, wind gusts, rotor downwash, etc.) are assumed to not exist. The indoor environment also makes it likely to control or prevent other disruptive variables (such as a sudden air draft or unknown obstacles) that might be hazardous to the system, thus are hence neglected within reason. Finally, during training and testing simulations, we assume that ideal sensor data is available without any noise and the odometry is available to be used by the controllers with no dropout. A separate validation test case is used to evaluate the system under the influence of gaussian sensor noise. As discussed in the previous section, a single quadrotor in the team is assumed to exert enough thrust to keep itself and the payload airborne. This remaining UAV is assumed to not have any mechanical issues that could make it incapable of stabilizing the payload during the simulated test runs.

2. BACKGROUND

2.1 Payload Carrying UAVs

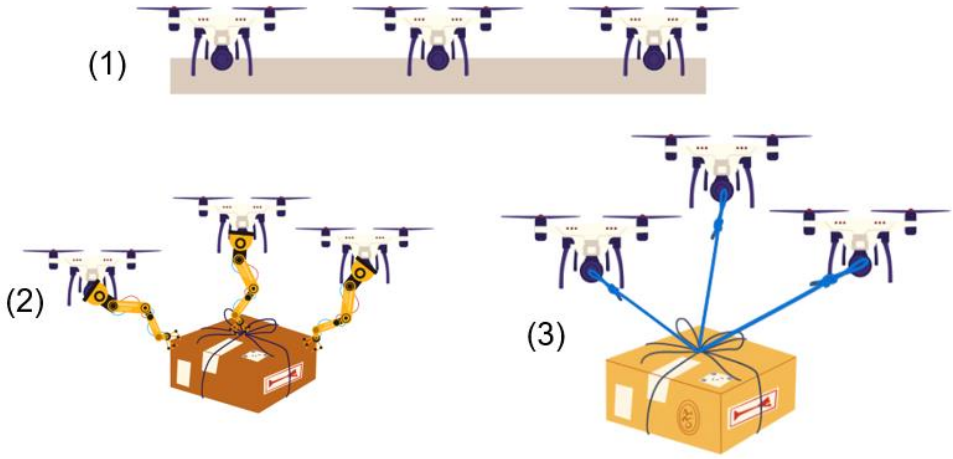


Figure 3: Various Configurations of UAV(s)-payload system: (1) Physical attachment; (2) Manipulator attachment; (3) Cable attachment

In the larger scenario of payload transportation, the payload can be moved by either a single UAV or multi-UAV system, but the payload can be attached to a UAV system one of two different ways: by a direct physical attachment to the UAV body itself, or by linkages such as manipulators or cables as shown in Fig. 3. Since a UAV's flight dynamics heavily rely on rotating about its axes to achieve translational motion, the method of physically attaching a payload to a single UAV increases the system's moment of inertia, so is only efficient for smaller payloads. Also, a larger UAV is relatively more expensive than a multi-UAV team comprising of smaller and less powerful UAVs working in unison. Alternately, the use of direct physical attachment for a multi-UAV system can allow for the transport of larger and potentially heavier packages, thanks to the total thrust generated by the UAVs. If the UAV team is arranged such that each individual UAV's thrusts act at different points along the payload's body, and the thrusts can be manipulated to control the rotation of whole system, translation can be achieved. But unless such systems work with accurate collaboration, the problem could soon be over-constrained, with reaction forces damaging some

fixture or drone itself. Also, while a larger UAV or a multi-UAV team can be used for larger payloads, the room required for safely flying and landing such an aircraft could be quite limited in indoor environments.

Similarly, manipulators add complexity in terms of control due to the increased degrees of freedom and the resulting reaction forces onto the UAV platform. It is worth mentioning that in the case a manipulator joint fails, constructing a recovery strategy becomes more complex. A multi-UAV team holding the same payload through manipulators could become over constrained, and in practice this could lead to harm of a manipulator joint or the payload itself. On the other hand, using a cable to connect a UAV to a payload would neither inhibit mobility like the physically attached example, nor possess as many moving parts as a manipulator, making it mechanically simple to model. The downside to using cables is that the payload can be dragged about the UAV like a pendulum and the swinging motion may not be ideal in constrained environments.

What all this means for the use case of payload transportation is scalability: either the number of drones in flight space would increase or the amount of payload we can transport using drones would increase. The former has major limitations because of its potential to crowd air space, the required maintenance of a large fleet of drones, and the logistics that come with their operation. On the other hand, increasing the payload capabilities of each single-UAV would ease some of the issues stated above. Cooperative drones can be designed to transport a large payload that each drone could not transport on its own. For safety reasons, there should be a careful balance between what each single drone UAV in these drone teams can carry, versus the total weight the combined team is allowed to carry; erring on the side of caution leads to a requirement of the maximum drone team payload being equal to what a single payload can safely lift by itself, with a team consisting of at least three members to allow constraint of the package motion in all three dimensions.

Due to the limitations stated for physically attaching a payload to a drone or using manipulators, using cables seems like the best choice for the immediate future, adding functionality without a great increase of complexity.

2.2 Prior Work

Multi-UAV systems that can carry a slung payload cooperatively have been implemented by researchers in the past. While the system as a whole is constrained to some level, the control methods used rely on several assumptions. These assumptions reduce the number of control methods that need to be tested for system destabilizing failures (e.g., of a team member or cable). The need for resilience in multi-UAV operations, as discussed in [23] and [24], is relevant when realizing warehouse autonomy where human-system integration would be required. While fault detection and isolation systems exist, such as [25], not many control methods are designed for or test the condition where the system may lose a team member. The closest related work to this research is a research thesis [26], which used genetic fuzzy logic to design a decentralized control method for multi-UAV slung payloads. In that study, the resilience of the method was demonstrated by shutting down one team member imitating a condition of drone power loss and showing that the rest of the team were able to hold hover, adapting to a new cable force-torque distribution. This early work showed in-place stabilization of the team and payload after failure in one drone while at least four other drones remain in the team. Our work draws inspiration from this and attempts to handle an extreme failure case not discussed in [26]: failure down to one UAV-slung payload where the oscillation of the payload is non-trivial. Bisig's work [26] also doesn't attempt to handle waypoint following during or after failure, nor does address obstacle avoidance; we are interested in addressing these issues while continuing to use Genetic Fuzzy System (GFS) techniques as well.

2.3 Fuzzy Logic and Genetic Fuzzy Systems

Fuzzy logic is a many-valued or non-binary approach to imprecise data to compute decisions based on approximate reasoning as opposed to a binary 'true/false' or '1/0' Boolean Logic. Fuzzy Logic was introduced by Dr. Lotfi Zadeh in his 1965 proposal [27] and presents two main concepts: linguistic variables and a fuzzy if-then rule. Linguistic variables are variables whose values are words rather than numerals. A simple way to put it is that linguistic variables are answers to 'what' and 'how' questions as a human would answer them. We answer questions like how tall somebody is by using our perspective and in relation with something than a scale to give the absolute answer. Fuzzy offers the tolerance to be imprecise with how the

systems perceive data. They no longer need to be absolutes that are quantized crisply. Instead, they can be seen as “degrees of truth”, or how much do they resemble two different classifications while solely belonging to none. The fuzzy rule base is propositions made using linguistic variables to linguistic values.

In traditional approaches, data must be absolute and void of noise for the computer to use it as an input. The noisier it is, the more unreliable is the output of the computation. Also, data is quantized by absolutes and is treated that way throughout the process. But realistically, this is rarely how data behaves. Fuzzy through its linguistic variables and “degrees of truth” approach offer a solution to be imprecise with data. Linguistic variables are characterized by their membership functions which allow the values to be fuzzy quantized or granulated. For example, we cannot choose an absolute barrier between hot and cold temperatures. In such a case, fuzzy systems will be able to handle partial truths and use that data for decision making. Unlike traditional approaches, which are based on rigid mathematical rules, fuzzy logic offers to be adaptive to imprecision in the data and in many ways mimic how we think. We summarize details and quickly use them to reason and to make decisions. Similarly, fuzzy systems can work with partial truth data and use them for logical reasoning without needing finer details or absolute perception as most other approaches require by design.

To reiterate, linguistic variables as discussed are variables that take words or sentences as values. For instance, ‘hot’ and ‘cold’ can be linguistic values for the variable of ‘temperature’. The weather is a perfect example, taking on values such as warm, sunny, windy, or cloudy. While daily weather can vary as a combination of the above, the degree of agreement of each condition would dictate which weather is dominant. Every linguistic variable in fuzzy logic is then comprised of a set called a fuzzy set characterized by a membership function. The degree of agreement from the weather example is chosen by this membership function. In systems, a membership function can be chosen to be (a) standard triangles or (b) trapezoids in order to label different levels of agreement or membership. From such classifications arises the idea of granulation, which is a form of fuzzy quantization. Granulation is the process of classifying data into fuzzy sets where the value of a variable can be anything in the fuzzy set and determined by a probability

distribution. This makes elastic constraints on variables available for use, instead of requiring absolute or deterministic constraints. It makes the fuzzy if-then rule base possible and convenient (tolerable) to work with in an imprecise environment. Linguistic variables, along with the fuzzy rule base, allow for data compression (maybe inherent with losses) through granulation. These concepts allow for fuzzy logic's tolerance to imprecision and ability to reason based on determined propositions for the linguistic variables and their values. Based on how the outputs of fuzzy systems are calculated, two of the most well-known kinds of Fuzzy Inference Systems (FISs) are Mamdani and Takagi-Sugeno systems. When Takagi-Sugeno uses a linear combination of its inputs to derive an output, Mamdani systems [28] allow for the output to also be fuzzified, which means the output can be given a range that is not dependent on the inputs. For this reason, we used Mamdani-style Fuzzy Inference Systems in this study. MATLAB's fuzzy toolbox also aids to setup up Mamdani systems neatly which was helpful for prototyping implementation.

Fuzzy rules are composed of if-then statements that match combinations of two or more inputs using 'AND' or 'OR' operator to a particular output membership function. This definition is specific to Mamdani systems, which is what was used in this study, and hence the background is limited to Mamdani type fuzzy systems. While expert knowledge can be used to derive the membership functions and rules for any application, this exhaustive process can be replaced by evolutionary learning algorithms such as genetic algorithms. By choosing the elements that make up the building blocks of a fuzzy controller as chromosome elements that can be manipulated in a population through generations based on a fitness function, fuzzy systems can be trained or optimized. Examples of such genetic fuzzy systems trained for different applications can be found in [29-32]. Using fuzzy logic to control quadrotor UAVs has been studied and demonstrated in [32], which gives the base controller that we chose for training individual behaviors into. In this study, we chose to use genetic fuzzy systems to train the stabilization and waypoint tracking behaviors. The linguistic variables are predefined with triangular membership functions and only the exhaustive rule base, i.e., IF-THEN combination consequents will be learnt by the genetic algorithm run by

simulating the UAV-payload system and calculating a fitness based on the simulation results. Further implementation of this genetic fuzzy method for this study is explained in Chapter 4.

3. METHODOLOGY

This chapter provides a complete description of the system, the considered environment and the general principles involved in solving the problem scenario given in section 1.3: how to handle a sudden team member detachment from a multi-UAV payload system where only one UAV remains attached to the slung payload in a constrained environment.

Multi-UAV systems using cables to handle the payload directly offer mainly two advantages: more control over the payload platform to limit the swinging oscillations and additional thrust to increase payload capacity. Failure cases in such multi-UAV platforms can reflect the robustness of the team. If the multi-UAV team was designed for increased thrust capabilities, failure of team members may result in insufficient thrust to keep the payload aloft. Alternately, if every UAV is capable of carrying the payload itself, and a multi-UAV team was chosen for added control and stability, this resembles a high safety case. If we consider a situation where a UAV in that team is facing failure, to lower risk to the team (e.g., non-collaborative forces, possible inflicted damage) it would be safest to release itself from the payload and move away. An identical failure scenario is where a cable linkage connecting the payload to a single UAV breaks, while leaving the payload still attached to the remaining UAVs in the team. Consideration of the number of UAVs left in the team is a simple factor to understand the resulting effects of detachment. For example, if a 4-member team becomes a 3-member team, it is a high safety case that would not result in much swinging of the payload. If a 3-member team fails down to a 2-member team the payload would now swing out of plane from the two remaining UAVs. A 2-member team failing down to a single-UAV payload system would be an extreme failure case, as the payload is now able to swing about the UAV largely unconstrained and may cause the UAV or payload to run into any nearby obstacles if a mitigation strategy is not implemented in order to prevent this. So long as the single UAV can generate enough thrust to lift

the payload, as assumed in the safety case above, a different controller may be needed to handle the hazardous swinging and help the system with this kind of failure.

The primary objective of the controller for the explained situation is to (a) arrest the swinging behavior in the now single UAV-payload system after the exit of the team member, and (b) stabilize the payload. It would be nice if the system can continue to follow waypoints and do other things that the initial team was scheduled to do, but at the very least we require some kind of controller on board that can handle this fail scenario and stabilize the payload effectively. To stabilize the payload, it is obvious that some kind of maneuver is necessary and that raises another objective, which is to avoid running the UAV or payload into nearby obstacles as the payload is being stabilized. We hypothesize that by setting up a controller as discussed in the following sections, we can have modular controllers that can stabilize the payload and then follow waypoints.

3.1 Environmental Design and System Characteristics

The intended application environment is a larger scenario of moving payloads inside a factory or warehouse, where multi-UAV teams are transporting parts/payloads in a queue inside an interior space. Each part is carried by a separate UAV team operating independently, with their own trajectory and corridor limits to move within, similar to a group of drones transporting payloads in a warehouse with many shelves, machines, and corridors. This research study focuses on a single UAV team in such an environment, where the team is only given its path and obstacles to be wary of from a high-level monitoring and scheduling system. The scheduling and initial trajectory planning of this high-level system are not discussed here (being outside the scope of this work), but are assumed as present for the initialization of our problem scenario. In the chosen scenario, a multi-UAV team was given an initial trajectory and carrying a payload, when in-flight failure of a team member happens in such a way that the team has failed down to a single UAV-payload system as shown in Fig. 4. The dotted and dashed circle is meant to visualize an imaginary boundary around the system maintained free of any obstacle for safety (the safe flight corridor). The boundary before failure is smaller due to the increased control in the platform. However, to ensure safety

after failure, the (expected new required safe) radius is increased (shown by the dashed line in Fig 4B) to compensate the swinging payload, and this change in radius (and potentially recentred new flight corridor) may require the UAV to move before performing payload stabilization.

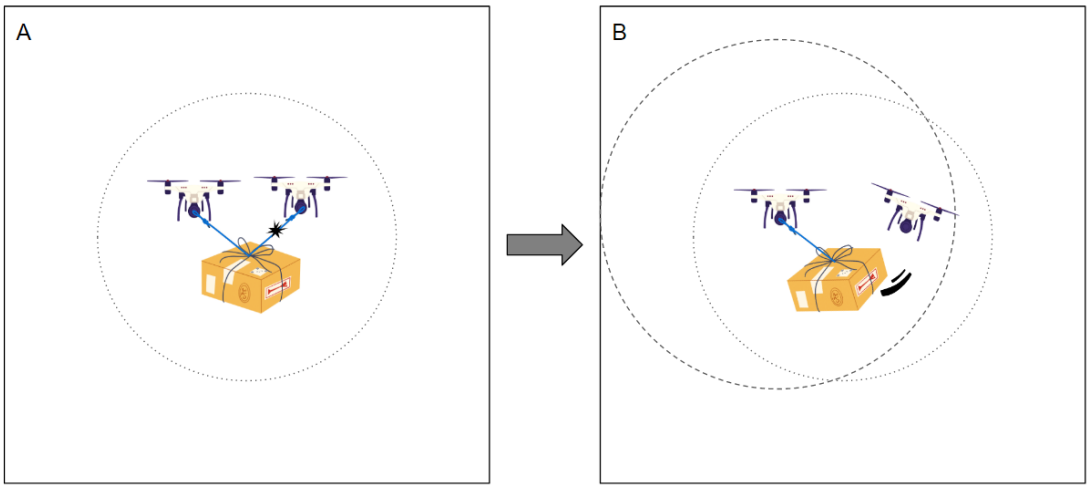


Figure 4: Failure of a 2-UAV-slung payload system (A) to a single UAV-slung payload system (B) showing safe boundary inside a constrained environment

The obstacles in a warehouse environment are assumed to be tall enough and rectangular enough that they can be modelled as walls (limits) that the UAV system is commanded to maintain a distance from. This idea (while a very conservative estimate) gives the problem an easily and quickly defined finite corridor for the system to stay within during transport and eventual payload stabilization post-failure. For this study, a 2-UAV team carrying a payload that has its general waypoints and locations of walls (Fig. 4A) fails to a single-UAV payload system that now has a swinging payload (Fig. 4B). The single UAV is modelled to have sufficient thrust to maintain itself and the payload as per the high safety case reasoning explained before.

For a quadrotor of unit mass, the payload mass satisfying this condition can be calculated as follows:

$$Thrust - weight\ ratio = \frac{Total\ Thrust\ of\ all\ motors}{Total\ weight\ of\ model}$$

For racing drones, very high thrust-weight ratios such as 4:1 and above are chosen while for most normal drones 2:1 is recommended. Since a 2:1 ratio leaves very little room for additional payload and delivery drones ought to have high thrust, a 3:1 ratio is chosen as a minimum (for lack of a better word, probably not what I want to say, bare minimum metric, etc) in this calculation.

Payload Capacity = (Motor thrust * Number of motors * Hover Throttle %) –The weight of the model itself.

For an agile aircraft, 50% hover throttle or lower is recommended. The product of motor thrust, and the number of motors is simply total thrust. For a quadrotor weighing 1 kg,

Payload Capacity = (Total Thrust of all motors * Hover Throttle %) - Weight of the model

= (3*0.5) - 1 = 0.5 kg

For a quadrotor weighing 2 kg,

Payload Capacity = (Total Thrust of all motors * Hover Throttle %) - Weight of the model

= (6*0.5) - 2 = 1 kg

Maintaining the model-to-payload ratio as 2:1, the UAV and payload characteristics were chosen as shown in the following table (Table 1). The fuzzy error divisor and fuzzy velocity divisor are meant for pre-processing the inputs to the fuzzy controllers which operate on a [-1, 1] range. The range extremities were set as 1 and are scaled by user input to make the control scheme modular. An example of this implementation is shown below.

$$Error\ in\ x = \frac{x_{goal} - x_{UAV}}{Fuzzy\ Error\ Divisor}$$

$$Error\ in\ \dot{x} = \frac{\dot{x}}{Fuzzy\ Velocity\ Divisor}$$

Similarly, the outputs of the fuzzy controllers also must be scaled with max permitted thrust and torques accordingly as the fuzzy force output is in range [0 1.5] and fuzzy torque output range is [-1, 1].

$$Force = Fuzzy\ force\ output * Hover\ Thrust = Fuzzy\ force\ output * (M + m) * g$$

$$Torque = Fuzzy\ torque\ output * Max\ permitted\ Torque$$

This means that the FIS controllers below are effectively being trained on unitless values, and thus should theoretically work for any payload and UAV system that has the same mass and length ratios.

Quadrotor Mass (M)	2 kg	Payload Mass (m)	1 kg
Thrust to weight ratio	3	Cable length (l)	1 m
Quadrotor Diameter (2r)	1 m	Max Permitted Thrust (F_{max})	1.5 x Hover Thrust
Moment of Inertia (J_x)	0.17 kgm^2	Max Permitted Torque (τ_{max})	0.5 Nm
Moment of Inertia (J_y)	0.17 kgm^2	Fuzzy Error Divisor	3
Moment of Inertia (J_z)	0.085 kgm^2	Fuzzy Velocity Divisor	3

Table 1: System Characteristics

In the set experiments, the UAV and payload are set with a cable angle of 45 degrees away from the closest wall as it would be in a 2-UAV team when the UAV the farthest from the wall would release from the team, making the payload swing towards the wall. The problem is modelled as recovery being the primary objective rather than high-performance control over position, as the latter would put additional strain on the training process and would lead to poor performance when both goals cannot be prioritized equally.

3.2 Characterization of Fail Scenarios

During a cooperative payload transport mission, several failures may occur that may endanger the payload or the team of drones themselves. As mentioned in Bisig's work [26], it is possible that a member may lose power and fall while being attached to the rest of the team. While this can be foreseen by a monitoring network, the shutdown mechanic can be used as a safety mechanism in other not-so-obvious failures, such as one or more rotor failures, where the member may try to fly rashly pulling the team or fly toward the team or payload causing in a crash. But this solution poses the problem of resilience to the team if the power is cut too late. So, having a cable release mechanism can be beneficial for unforeseen drone failures, where the failing drone detaches itself from the system and flies away from them (much like the Skycrane in NASA's Mars 2020 Mission). We consider the extreme case of a two-quadrotor team carrying a slung payload failing to a single quadrotor to study the challenges and verify the proposed solution before extending it to a more generalized multi-quadrotor failure solution. In the set of experiments used in this study, one quadrotor in a two-quadrotor team carrying a payload detaches from the payload during a hover and moving case. In both cases, the swing of the payload is not desired as it not only risks the quadrotor being airborne with its oscillating disturbances but also the payload contents. The payload and quadrotor also have the hazard of crashing into a wall or object in the vicinity. The recovery (or a need for avoidance) of the detached quadrotor is not considered in this study; the recovery of the remaining quadrotor and the attached payload is considered and prioritized. Failure cases that are expected to be handled are summarized as follows:

1. Inability of the drone to hold its position due to the swinging payload
2. Swinging of the payload that can damage its contents
3. Collision of payload or drone with any obstacle in the vicinity
4. Inability to continue the transport mission or navigate to a service point

3.3 Modelling

A model of a quadrotor UAV with a cable suspended payload as shown in Fig 5 is considered. The system was modeled under the assumption that the quadrotor is rigid and symmetrical, and the payload is modelled

as a pendulum is a point mass suspended at the end of a massless rigid rod attached to the quadrotor's center of gravity. The symbols in Fig. 5 are described as UAV quadrotor mass M, payload mass m, rope length l, pitch angle θ, roll angle φ and yaw angle ψ of the quadrotor, payload swing angle α and payload rotation angle β, respectively.

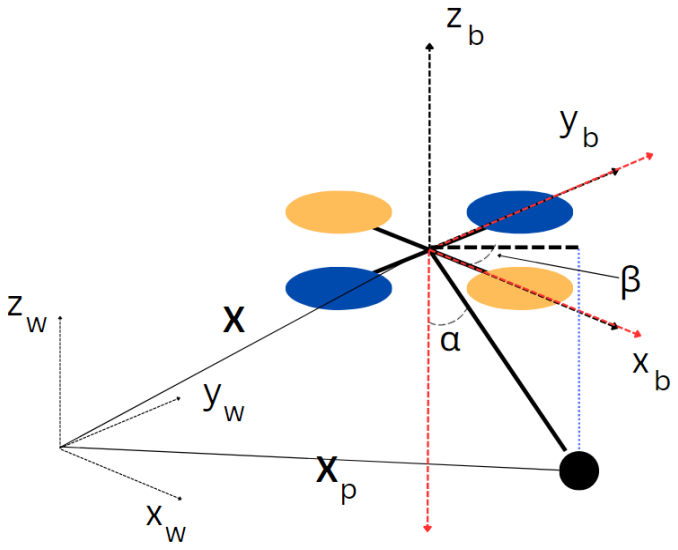


Figure 5: Quadrotor transporting a payload in w frame

The generalized coordinates defining the configuration of the system are q = [x, y, z, θ, φ, ψ, α, β]^T, where x, y and z are the positions of the center mass of the quadrotor in a global/world fixed frame of reference w. The payload coordinates can be denoting as a function of the generalized coordinates as follows:

$$x_p = x + l\cos(\beta)\sin(\alpha)$$

$$y_p = y + l\sin(\beta)\sin(\alpha)$$

$$z_p = z - l\cos(\alpha)$$

The kinetic (K) and potential (U) energy of the system are

$$K(q,q') = \frac{1}{2}Mx'^2 + \frac{1}{2}My'^2 + \frac{1}{2}Mz'^2 + \frac{1}{2}J_\theta\theta'^2 + \frac{1}{2}J_\varphi\varphi'^2 + \frac{1}{2}J_\psi\psi'^2 + \frac{1}{2}mx_p'^2 + \frac{1}{2}my_p'^2 + \frac{1}{2}mz_p'^2$$

$$U(q) = Mgz + mgz_p$$

where J_θ, J_φ, J_ψ are the moments of inertia of the quadrotor in pitch, roll and yaw axis and g is the acceleration due to gravity. For the Lagrangian function $L(q, q') = K(q, q') - U(q)$, the system's motion equations are derived using the Euler-Lagrange's equation:

$$\frac{d}{dt}\left(\frac{\delta L}{\delta q'}\right) - \frac{\delta L}{\delta q} = u$$

where input vector u is given by $u = \left[f_x, f_y, f_z, \tau_\theta, \tau_\varphi, \tau_\psi, 0, 0\right]^T$. These motion equations for the system in 3D by solving the Euler-Lagrange Equation for $q = [x, y, z, \theta, \varphi, \psi, \alpha, \beta]^T$ are shown below:

$$\ddot{x} = \frac{f_x\left(M + m\cos^2\beta\cos^2\alpha + m\sin^2\beta\right) - f_y\left(m\cos\beta\sin\beta\sin^2\alpha\right) + f_z\left(m\cos\beta\cos\alpha\sin\alpha\right) + Mml\left(\dot{\alpha}^2 + \dot{\beta}^2\sin^2\alpha\right)\cos\beta\sin\alpha}{M(M+m)}$$

$$\ddot{y} = \frac{-f_x\left(m\cos\beta\sin\beta\sin^2\alpha\right) + f_y\left(M + m\sin^2\beta\cos^2\alpha + m\cos^2\beta\right) + f_z\left(m\sin\beta\cos\alpha\sin\alpha\right) + Mml\left(\dot{\alpha}^2 + \dot{\beta}^2\sin^2\alpha\right)\sin\beta\sin\alpha}{M(M+m)}$$

$$\ddot{z} = \frac{f_x\left(m\cos\beta\sin\alpha\cos\alpha\right) + f_y\left(m\sin\beta\sin\alpha\cos\alpha\right) + f_z\left(M + m\sin^2\alpha\right) - Mml\left(\dot{\alpha}^2\cos\alpha + \dot{\beta}^2\sin^2\alpha\cos\alpha\right)}{M(M+m)} - g$$

$$\ddot{\theta} = \frac{\tau_\theta}{J_\theta}$$

$$\ddot{\varphi} = \frac{\tau_\varphi}{J_\varphi}$$

$$\ddot{\psi} = \frac{\tau_\psi}{J_\psi}$$

$$\ddot{\alpha} = -\frac{1}{Ml}\left(f_x\cos\beta\cos\alpha + f_y\sin\beta\cos\alpha + f_z\sin\alpha\right) + \dot{\beta}^2\sin\alpha\cos\alpha$$

$$\ddot{\beta} = \frac{f_x\sin\beta - f_y\cos\beta - 2Ml\dot{\alpha}\dot{\beta}\cos\alpha}{Ml\sin\alpha}$$

To simplify these equations into 2D (XZ plane), the elements $y, y', y_p, y_p', \varphi, \varphi', \psi, \psi', \beta$ can be substituted as 0 and the resulting equations are given below:

$$\ddot{x} = \frac{M + m\cos^2\alpha}{M(M+m)} f \sin\theta + \frac{m \sin\alpha \cos\alpha}{M(M+m)} f \cos\theta + \frac{ml\dot{\alpha}^2 \sin\alpha}{M+m},$$

$$\ddot{z} = \frac{m \sin\alpha \cos\alpha}{M(M+m)} f \sin\theta + \frac{M + m\sin^2\alpha}{M(M+m)} f \cos\theta - \frac{ml\dot{\alpha}^2 \cos\alpha}{M+m} - g,$$

$$\ddot{\alpha} = -\frac{\cos\alpha}{Ml} f \sin\theta - \frac{\sin\alpha}{Ml} f \cos\theta,$$

$$\ddot{\theta} = \frac{\tau}{J_\theta}.$$

In a planar representation, we are assuming that UAV is controlled by four thrusters however only the thrusters that are in the XZ plane passing through the UAV's center of gravity are responsible for the corresponding torque causing roll.

3.4 Control Scheme

To reiterate the context, we want to deal with a multi-quadrotor cable system for package transport that fails down to a single quadrotor slung payload system in a constrained indoor environment such as warehouses or corridors. Fuzzy logic controllers sequenced for position control scheme in [32] were adapted as shown in Fig 6. Using GFS technique, these FIS sequences are trained offline prior to use, for situations that have the quadrotor payload system continue to track waypoints despite the initial disturbance (~45˚) after detaching from other quadrotors in a multi-quadrotor team. The inputs here are divided by the divisors from Table 1, to scale down the inputs for the fuzzy inference system. Similarly, the thrust and torque outputs are scaled by hover thrust and max permitted torque values respectively.

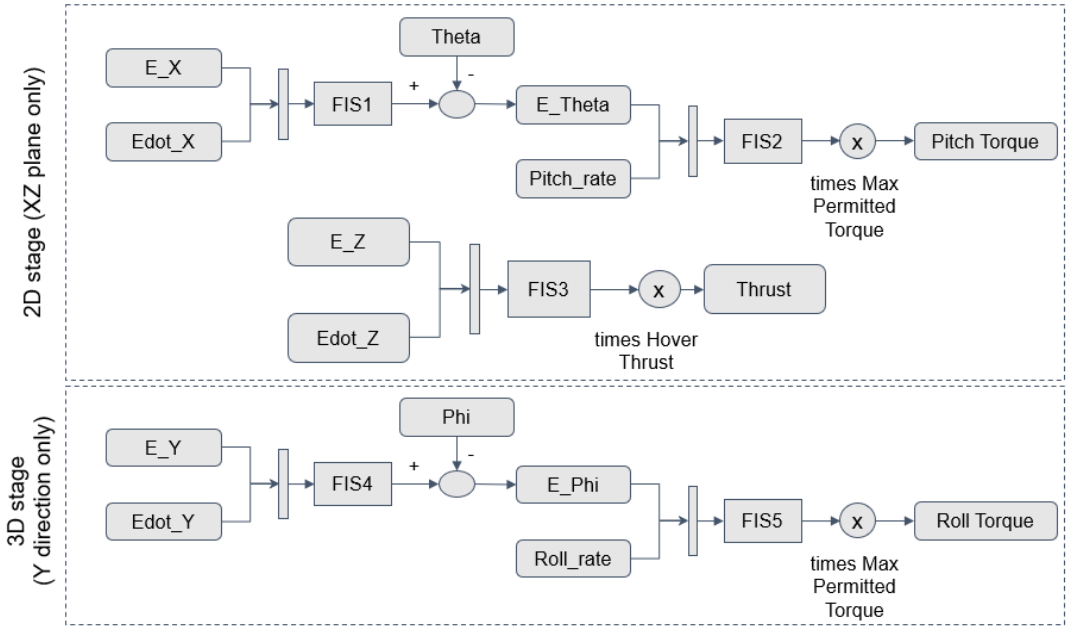


Figure 6: General FIS setup

This setup is used for both behaviors with the only difference being the rule base that will be trained and stored into them. While FIS1, FIS2 and FIS3 are used for control in the XZ plane and are trained using the 2D equations to reduce complexity by staging the learning process, FIS4 and FIS5 (which help in Y direction translation) are trained later in a 3D environment and augmented to the previously trained FISs for full 3D control. In addition to the state feedback, these FISs need the commanded waypoints to be given to them for calculating the errors and which are given to the system as predefined waypoints by the user. For obstacle avoidance, the waypoints may be modified by an obstacle avoidance FIS prior to passing to the behavior trained FIS sequences. The control scheme layout can be summarized as shown in Fig 7. A switching policy is used which determines what FIS sequence is to be engaged and accordingly passes the control inputs from the correct (selected, active) FIS sequence to the plant.

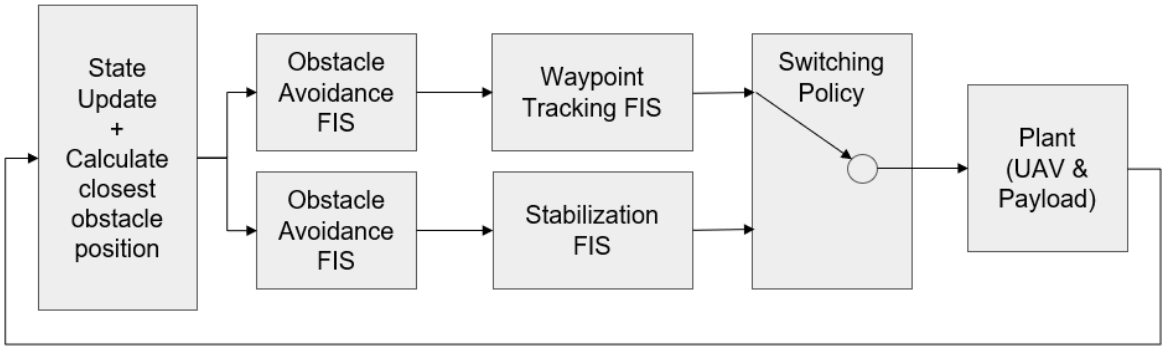


Figure 7: Proposed Control Scheme

3.5 Training Objectives

Breaking down the objectives explained in section 1.2, the behaviors (e.g., action primitives) that are required are (1) payload stabilization and (2) waypoint tracking. Obstacle avoidance and navigation after recovery can be achieved as a combination of these two behaviors as well. We hypothesized that simple behavior actions can be learned into the rule bases of a fuzzy control scheme through GA, enabled by suitable penalty functions. These behaviors can be swapped between as necessary by a switching policy at appropriate times. While the FIS setup used may be intuitive for waypoint tracking and position control, having no cable angle input may raise doubt in the payload stabilization behavior. We theorized that this formulation of fuzzy rules could be trained to stabilize the swinging payload given that the UAV position errors (impacted by the singing payload) are fed as inputs to the FIS setup. By learning this dependency, the intuitive and human like response to stabilize payload can be trained. Earlier in this experiment, we observed that the genetic algorithm struggled to learn multiple behaviors together and took a lot of time because of the highly non-linear equations. To note, there was no convergence after 200 generations and training time was impractical. So, the training process was staged into 2D, where XZ controllers were trained, and then by adding Y controllers, 3D control was achieved. This was possible due to the symmetricity of the quadrotor that was listed as one of the assumptions for this study.

For payload stabilization, the control sequence was expected to learn how to arrest the swing when the cable is released from a 45˚ cable angle offset while not straying far away from the initial position. Any stray offset can be eliminated by use of the waypoint tracking controller, but keeping position offset reduction minimal and prioritizing the settling of the payload and cable swing is chosen as a tradeoff, considering system safety. This system is trained in a hover condition as the switching policy will engage the waypoint tracking to slow the system to near hover if failure occurs during motion. For waypoint tracking, the quadrotor is expected to travel and reach waypoints no more than 3 meters away; this value was chosen by the set fuzzy input ranges for displacement errors. By training individual directions of X, Y and Z, waypoints that require the combination of the three directions will also be trained.

3.6 Switching Policies and High-Level Planners

After observing the performance of the controllers responsible for waypoint tracking and payload stabilization behaviors, it is necessary to explore how these controllers can fit into a larger robust control system that can sense the environment, and plan current and/or future actions based on strategic combinations of trained behaviors. Let us consider an instance of two quadrotors carrying a payload through a corridor when one quadrotor detaches from the team due to failure. The remaining quadrotor now has a swinging payload destabilizing its flight, which could also be at the risk of colliding with a wall in the corridor. A higher-level planner now has to choose between using the payload stabilization control to return to hover vs. the waypoint tracker to avoid the wall. To demonstrate that such swapping is possible, a variety of switching policies were chosen. These policies were used as proof-of-use only and are assumed to be suboptimal. (The optimization of these policies is not within the scope of this study.)

For obstacle avoidance scenarios, the switching policy considered the distance to the closest obstacle and compared it against a predefined safe distance. If the former was lesser than the safe distance, then the motion-tracking FISs are engaged to move the quadrotor payload system to a safe position calculated by a manually tuned obstacle avoidance FIS. Once a safe distance from the obstacle is reached, the stabilization FISs are engaged to return the quadrotor and payload to hover. Realistically, return to hover is not

considered as the end goal, instead, the quadrotor should then be commanded to land when it's safe to do so or carry on with the transport if possible. For this reason, we considered moving the quadrotor payload system to multiple waypoints as a possible action which is discussed in the following sections.

The switch between waypoint tracking to stabilization in a multiple-waypoint scenario is quite rudimentary because of how the controllers were trained. While the motion-tracking FISs were trained to achieve a waypoint, it doesn't do so without restricting swing for efficiency reasons. On the other hand, the payload stabilization controllers were trained to arrest swinging with minimal stray from the original position but not zero offsets. Considering the trained behaviors, it is necessary to adapt the switching policy to work as we intend, while acknowledging that the waypoint is only reached with the minimal stray of the stabilization controller as the maximum tracking error. The switching policy chosen for the multiple waypoints following demonstrations engages the stabilization FISs when the distance to the waypoint is below the maximum tracking error observed during testing. While these policies are suboptimal, they demonstrate how these FIS control systems trained to do specific tasks can be used modularly depending on primitive actions or desired behaviors. It also shows that new behaviors can be trained on similar FISs and be utilized without any change to the previously trained controllers but with an adjustment to the switching policy layer or a policy addition to the high-level planner.

4. IMPLEMENTATION

The methodology discussed was implemented in MATLAB 2021a with the help of MATLAB’s Fuzzy Logic Toolbox and Parallel Computing Toolbox. The desktop computer used for running software simulations is an Ubuntu 20.04 LTS operating system with an Intel Core i5-9600K (CPU) processor @ 3.70 GHz x 6 and 16GB of memory. A Genetic Algorithm program by Seyedali Mirjalili [33] was customized to optimize FIS rule bases using genetic functions such as crossover, mutation, inversion, and elitism, and to save the final population along with best chromosome upon exist. MATLAB’s parfor function was utilized to improve runtime. The following sections will delve into the environment setup for

training, controller development for payload stabilization, waypoint tracking, and training implementation followed by development of an obstacle avoidance controller and refining trainings.

4.1 Environment Modelling

This study deals with a failure situation resulting in a “down-to-one” quadrotor to which the payload is attached and is in a state of swing because of the egress from the other quadrotors. The aimed environment for the system is indoors where wind gusts should not be a cause for concern, but limited flight spaces are. To help envision such application environments, we considered warehouses and hallway corridors. For the model characteristics presented in section 3.4, it is necessary to understand the limits such as the room/space required for safe stabilization and translation. Hence, an empty 8x8 unit was chosen for stabilization and waypoint tracking training cases. Ideally, the quadrotor should be able to stabilize itself within a unit radius from the initial position and be able to traverse 3 units in any direction as the corridors that will be considered later will be a 4x8 unit space. With these spaces acting as training and testing zones for 2D training, when advancing to a 3D system, we accordingly switch to an 8x8x8 unit warehouse and a 4x8x8 unit corridor as shown in Fig 8. The centroid of these spaces was chosen as the origin for a world inertial frame of reference w, which would help calculate the position of obstacles, walls, and waypoints. A waypoint in this inertial frame w regardless of whether specified by an obstacle avoidance controller or a path planner would be the local frame of reference L for the controller to estimate the quadrotor position and errors. The attitude of the quadrotor and cable are calculated with respect to a body centered frame B fixed at the quadrotor’s center of mass. All three frames of reference are oriented parallel to each other and the edges of the test space room.

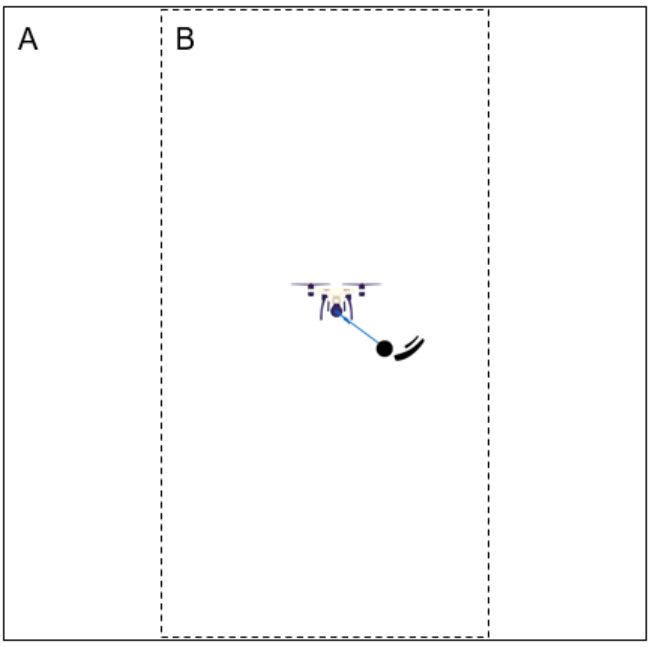


Figure 8: Training environment: (A) Empty room 8x8x8m; (B) Corridor4x8x8m

4.2 Base Controller Design

The idea behind controller development was that several behaviors can be trained individually into identical fuzzy controllers and swapped in a modular way depending on the situation at hand. Since, the combination of membership functions and rule base dictate what is the behavior we ought to see from a controller, we hypothesized that by tuning the same combination we can have the fuzzy controller do other things, such as learn actions that can help stabilize the payload and follow waypoints. To execute this, a traditional position and attitude control scheme for Fuzzy Logic Controllers (FLC) as discussed in [30] is chosen as the standard. The FIS implementation in MATLAB is done by FLC. The membership function design for FIS1, FIS2 and FIS3 are shown in Fig 9-11 respectively; where the first row is input 1 and input 2 and second row is the output. The rules are listed as all combinations of input 1 and input 2, which for 7 and 5 membership functions respectively gives 35 rules. The consequent of the rule is left as membership function 1 of the output and will be tuned by the GA. The behaviors desired to be encoded into the controllers can be done by tuning the rules and membership function parameters, often by both simultaneously. Since multiple FLCs would be trained simultaneously and each FLC has up to 35 rules, the variables to be tuned

would increase if membership function parameters were included. Moreover, for the behaviors desired, the type of membership functions and parameter differences would cause reduced explainability. For example, an output that may be fuzzified as small for one behavior may not be small when swapped to a different controller. Also, each training might not define all rules but only those that influence a particular desired behavior. So, a standard triangular membership function was chosen, and they were defined to cover identical ranges.

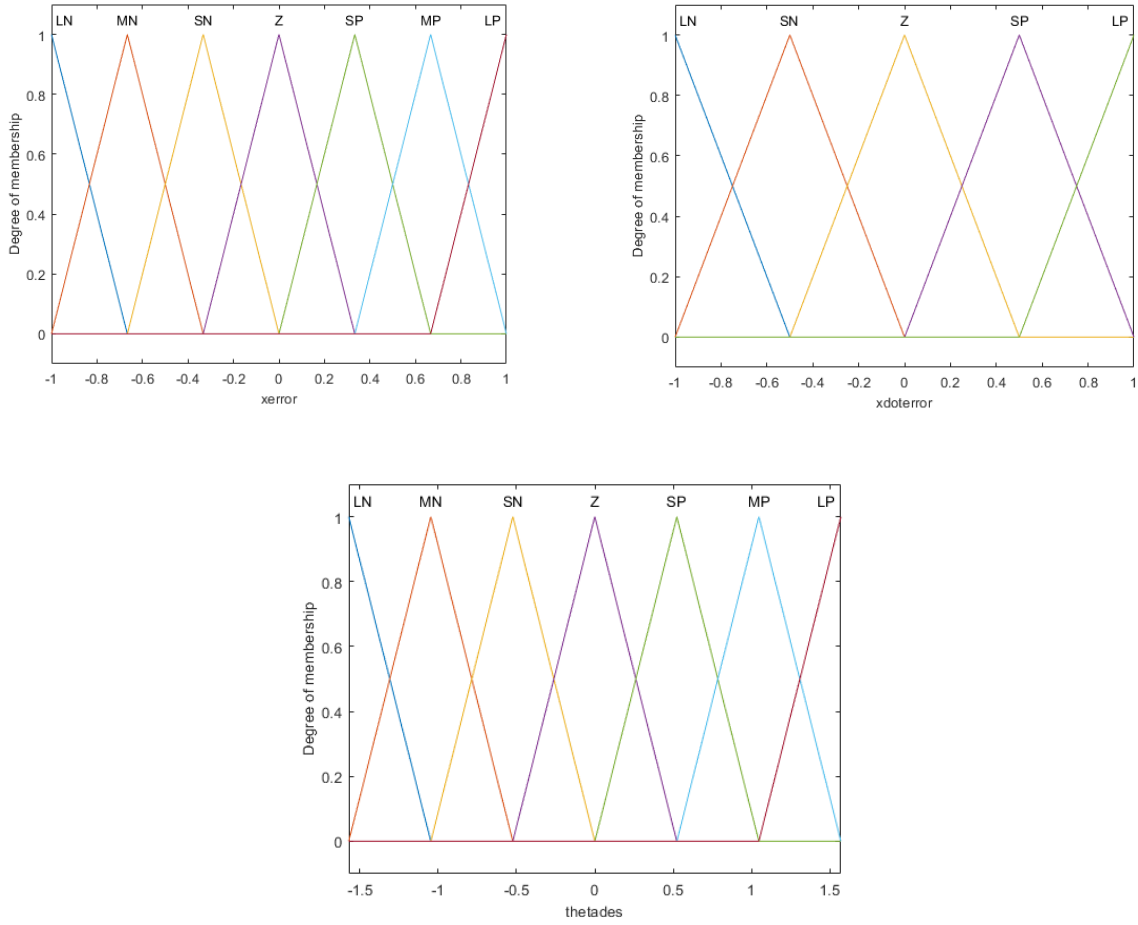


Figure 9: FIS1 membership functions

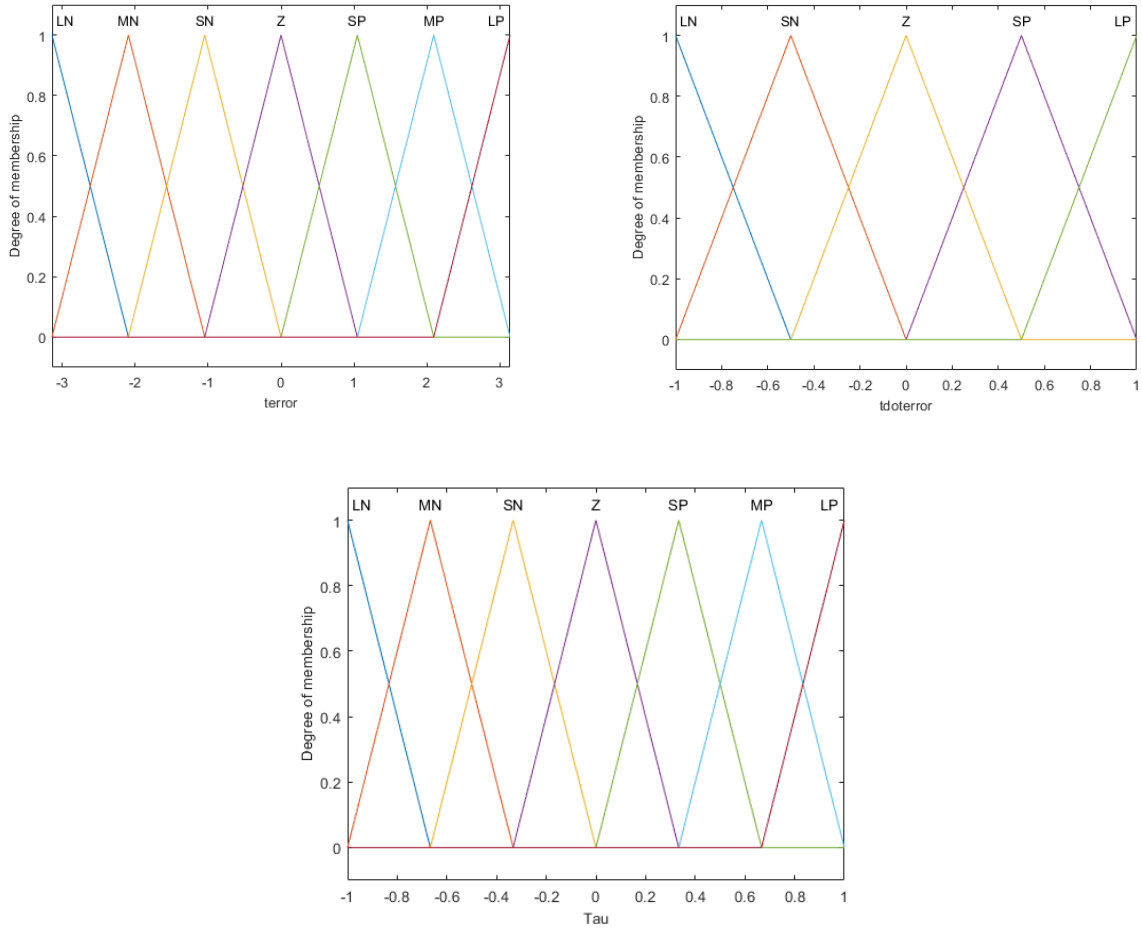
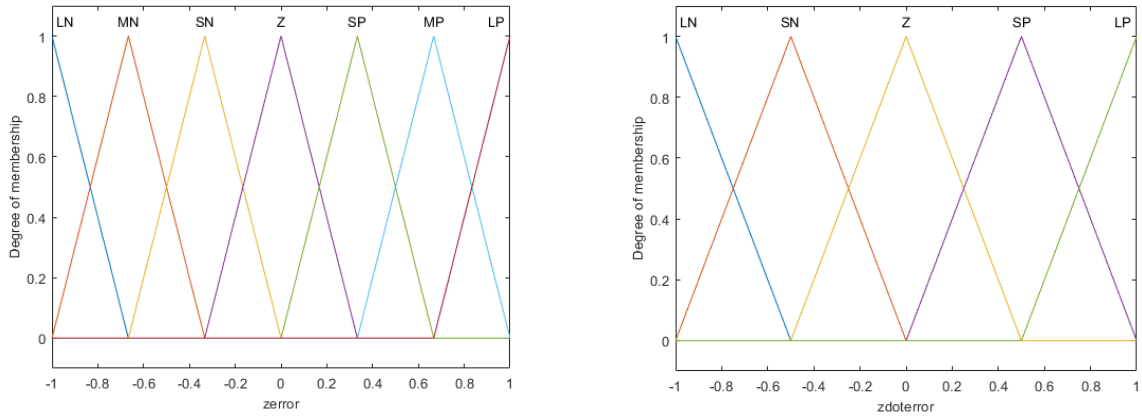


Figure 10: FIS2 membership functions



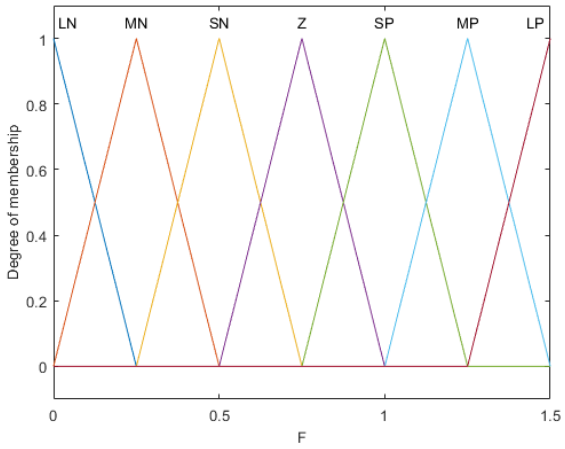


Figure 11: FIS3 membership functions

The rule base that must be trained will ultimately decide the performance of the system as the membership functions are kept fixed. The rule base is encoded as a string of numbers with each number specifying the output membership function. The location of the number determines the consequent of a predefined combination of antecedents in the FLC rule base table. So, the rule base of each FLC is encoded in a string of length 35 and the chromosome used for the genetic algorithm would contain all the rule base strings concatenated in order.

4.3 Training Implementation

A set of FLC controllers were trained together to learn a particular maneuver in different training scenarios. The set of controllers must perform the following tasks: i) stabilize the attitude of the payload and quadrotor; ii) control position by manipulating the attitude of the quadrotor. Each of these tasks requires the associated FIS system to be trained in different scenarios with different initial conditions.

4.3.1 Payload Stabilization Training

This scenario was designed with the failure case that if a quadrotor and payload are egressed from a two-quadrotor slung payload team during hover, (1) the payload would swing and likely destabilize the remaining quadrotor, and (2) the only controllable device is the quadrotor that has to fight the swing and stabilize the entire two-body system while also making sure that the payload is returned to a neutral position with a minimal swing. With fuzzy logic, we know that complex behaviors for human-like actions can be

encoded from [15]. But since the destabilization maneuver is complex and it would take complex experiments for even an expert to form all the rules necessary, a genetic algorithm can be used to minimize a cost function that emulates our desired end conditions. However, due to the nature of the maneuver and the finite data that can be gathered by the algorithm to have influenced the actions to be learned, it should be noted that (1) the rule base would only have rules that are relevant to the action-trained if possible, and (2) the non-participant rules are random as they don't affect the outcome as described by the chosen fitness function.

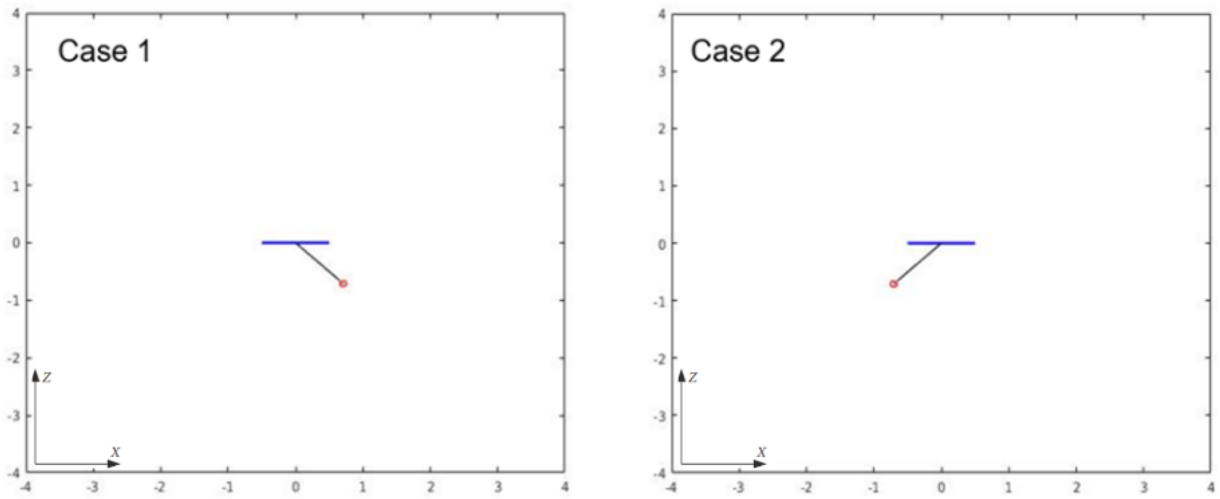


Figure 11: Payload stabilization training cases in XZ

For initial training in 2D, the quadrotor was positioned at (0, 0) which is the centroid in a hover position and zero velocities. The payload cable was set to + and -45 degrees with the vertical neutral position along -Z direction as this could be the positions that the payload would egress from when the other quadrotor detaches from the team as shown in Fig 11. The algorithm would initialize a random rule base and run both initial conditions, calculate the fitness and sum them together for the overall fitness of a rule base. The fitness function should contain all terms that may be important to minimize in this situation, these include minimizing cable angle velocity as soon as possible, and minimizing straying from the initial position while doing the former. To avoid straying the goal position was set to (0, 0), the same as the initial position. The

action to be learned is the recovery of the system in or close to the initial position. The fitness function used to capture this behavior is as follows:

$$Fitness_{PS} = -\sum_{i=0}^{n}\left(\acute{\alpha}_i^2 + (10 * (|\acute{\alpha}_i| + |xe_i| + |ze_i|) * dt)\right)$$

The negative sign contributes to minimization as the genetic algorithm used is a maximization algorithm. The cable velocity term, $\acute{\alpha}$ is used instead to keep the algorithm scalable in behavior and minimizing the cable angle to zero might be too harsh a goal to realize and may not be true in situations where the failure is not down to one. The xe and ze terms minimize the error from the goal position and hence keeps the quadrotor close as possible to the initial position. The dt term multiplied acts as an integral of the errors and ensures that the errors are minimized as soon as possible while the weight of 10 ensures that the difference in magnitude between the terms doesn't make them insignificant.

For the 2D case in XZ plane, three FISs were trained simultaneously, fis_x, fis_roll, and fis_z. This makes the number of variables to be tuned to be 35 x 3 = 105. The population size suggested in literature [32] is 5 to 10 times the number of variables. For 105 variables, the population was set to 5 times the number of variables and rounded off to the closest multiple of 10. The number of generations was set to 35 arbitrarily and confirmed to be reasonable once the test results were obtained. The entire system was simulated for a time span of 6 seconds using MATLAB's ode45 for each initial condition to calculate the fitness during the genetic algorithm.

The initial training results showed a behavior of reducing cable velocity within the given time span and returning to hover. The quadrotor moved to either direction along with the payload to swiftly bring it to rest but this maneuver makes the quadrotor come to rest around 0.4m away from the initial position. This is acceptable considering the payload characteristics, and gives a sense of how much room is needed to recover the payload safely when walls or obstacles are nearby. If there are any walls nearby that endanger the system with collision, then the system can be commanded to move to a safer waypoint and then perform

payload stabilization. Since this experiment showed that recovery is possible to be learned and encoded in the simple control scheme, we wanted to explore how much information could be stored in the rule base – for example, commanding to move to a new waypoint. Upon further training, the system while retaining the stabilization behavior did not learn any new rules required for waypoint tracking. This indicated that there is a rule overlap between the two actions, and one action would need to be overwritten on rules necessary for the other in the current FIS setup. So modular controller switching was proposed, and this led to separate waypoint tracking controllers being trained as discussed in the following section.

When we scaled the system up to 3D, we introduced two new FISs namely fis_y, and fis_pitch responsible for the YZ plane. The controllers previously trained in 2D performed satisfactorily in a 3D environment which meant no changes were necessary. However, the same controllers couldn't be used for YZ as the new controllers had to also learn to work with the XZ plane controllers. So in 3D, similar training was repeated for the new YZ controllers while fixing (/ not allowing training to change) the XZ controllers, i.e., FIS1, FIS2 and FIS3. The initial conditions used in this training run were having the quadrotor start at (0, 0, 0) and setting the cable angle as + and - 45 degrees with the vertical neutral position along -Z direction but in the YZ plane. The commanded position was set as (0, 0, 0), similar to the previous training, and so was the timespan allocated to simulate the system (set to 6 seconds). Since now only two controllers are being trained, the number of variables is 35 x 2 = 70; the population size was set as 5 times the number of variables, 350 and the number of generations was set to 35. The fitness function on the other hand was modified to include the ye term to emulate the same behavior in a 3D environment as shown below:

$$Fitness_{PS} = -\sum_{i=0}^{n}\left(\acute{\alpha}_i^2 + \ (10 * (|\acute{\alpha}_i| + \ |xe_i| + |ye_i| + |ze_i|\ ) * dt)\right)$$

Since the fis_z controller is fixed, the new controllers have to adapt and learn based on output actions that the previous control scheme may output during the simulation. The training results showed a similar recovery maneuver minimizing the cable velocity swiftly now in the YZ plane. There was an offset of 0.4m

in the Y direction observed, but this is accepted as a tradeoff for payload stabilization. The training conditions are summarized in Table 2 below.

Scenario	Environment	Case	UAV start position	UAV commanded position	Cable Angle
Stabilization	2D (XZ only)	1	(0, 0)	(0, 0)	+45° in XZ plane
		2	(0, 0)	(0, 0)	-45°in XZ plane
Stabilization	3D (XYZ)	1	(0, 0, 0)	(0, 0, 0)	+45° in YZ plane
		2	(0, 0, 0)	(0, 0, 0)	-45° in YZ plane

Table 2: Payload Stabilization Training Conditions

4.3.2 Waypoint Tracking Training

In the failure scenario described earlier, having the ability to transit from the point of detachment to a waypoint that is away from any potential obstacles or could be deemed a safe zone to stabilize the swinging payload could be vital in the act of recovery. This would also add robustness to the system if the quadrotor is able to resume the original transport action without having to switch back to a new control system, or at the very least act as an interim support. One of this study's aims was to illustrate the modularity of fuzzy logic systems, more specifically how multiple smaller behaviors can be trained onto identical and modular fuzzy logic systems, with the option to add more behaviors with no reconfiguration to the existing systems. Hence, the waypoint tracking controller was also trained on the same blueprint as the stabilization controller, with relevant training scenarios and fitness functions.

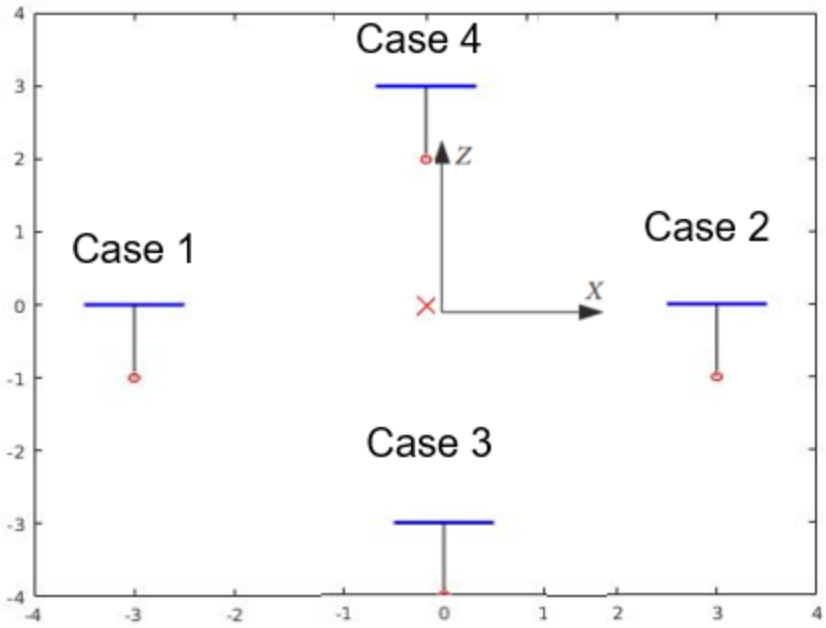


Figure 12: Waypoint tracking training cases in XZ

To check feasibility, initial training was conducted using 2D equations of motion in the XZ plane. The waypoint reached by the quadrotor was defined as (0, 0). The quadrotor was positioned at a distance of 3 units from (0, 0) on X axis and Z axis, i.e, (3, 0); (-3, 0); (0, 3) and (0, -3) with zero velocities as shown in Fig 12. This choice is equivalent to having the quadrotor positioned at the (0, 0) and passing a goal waypoint 3 units to its left, right, below it and above it. This combination of initial conditions allows training the rule base to encompass all waypoints within a 3 unit radius from the quadrotor with the radius value itself chosen as the maximum error we defined for the control system. Similar to the stabilization training, the algorithm would initialize a random rule base and run all four initial conditions, calculate the fitness and use the sum fitness as overall fitness of a particular chromosome rule base. For waypoint tracking, we ought not penalize cable velocity as it would have the quadrotor move slowly and impractically. Considering this tradeoff, the modified fitness function is as follows:

$$Fitness_{WT} = -\sum_{i=0}^{n}(10 * (|xe_i| + |ze_i|) * dt)$$

The desired action to be learned is having the quadrotor accelerate towards the waypoint and stop at the waypoint. To encode this, we use error between the quadrotor position and goal position coordinates over time where i = 0, implies when time t = 0 and i = n implies time t = T. T is the timespan that we command MATLAB's ode45 to simulate our system and it is chosen as 8 seconds considering the observation time that other researchers used for similar tracking cases. The dt term multiplied acts as an integral of the errors and ensures that the errors are minimized as soon as possible while the weight of 10 is multiplied to scale the fitness for readability. Three FISs, fis_x, fis_roll and fis_z will be trained simultaneously for this 2D experiment. Extending the previous reasons, the number of variables is 105 and the population size was chosen as 530. The number of generations was set to 35 arbitrarily and confirmed to be reasonable once the test results were obtained.

The training results showed that the rise and sink quadrotor actions were the first behaviors to be trained as they are simple in terms of physics and the relevant controller, fis_z has very direct rules that would enable this. The quadrotor's action of traveling left or right would require the controller to tilt in the direction of motion while also balancing thrust to avoid flying off. Though this action is relatively complex, the algorithm was able to tune sufficient elements in the rule base to learn this action. In all training conditions, the quadrotor was able to reach the goal waypoint well within the given timespan but there does exist a maximum overshoot of 0.1m. This is acceptable given that there is a swinging payload that is not being accommodated by the controllers.

Scenario	Environment	Case	UAV start position	UAV commanded position	Cable Angle
Waypoint Tracking	2D (XZ only)	1	(3, 0)	(0, 0)	0°
		2	(-3, 0)	(0, 0)	0°
		3	(0, 3)	(0, 0)	0°
		4	(0, -3)	(0, 0)	0°
Waypoint Tracking	3D (XYZ)	1	(0, 3, 0)	(0, 0)	0°
		2	(0, -3, 0)	(0, 0)	0°

Table 3: Waypoint Tracking Training Conditions

Scaling the system to 3D, the controllers responsible for Y-direction, fis_y and fis_pitch should be trained while no changes are necessary to the previously trained XZ controllers. Similar to the 2D training, the goal position was set as (0, 0, 0) while the quadrotor are positioned at (0, 3, 0) and (0, -3, 0) with zero initial velocities. The XZ controllers are fixed and the YZ controllers have to rely on and adapt to the existing fis_z controller's efforts. For the two controllers being trained, the number of variables is 70 and hence the population size is set as 350 and the number of generations is 35. The system is simulated over a timespan of 8 seconds. The fitness function for 3D training is modified as shown below:

$$Fitness_{WT} = -\sum_{i=0}^{n}(10 * (|xe_i| + |ye_i| + |ze_i|) * dt)$$

By this experiment, waypoint tracking controllers for XYZ were trained. Staging the training as shown, reduces the time to learn all actions significantly and is recommended for convergence in complex systems. Both the payload stabilization and waypoint tracking controllers created so far have a commanded waypoint that is being predefined and passed to the controller. Also, there is no input of the surrounding environment to these controllers yet, and they at best imitate low-level position and attitude control (such as PID) but

are able to perform specific maneuvers if necessary. To make this system autonomous, for waypoint following, a waypoint planner could be used. For autonomous recovery, a basic waypoint planner can be created that senses the obstacles nearby and commands a waypoint to move away to maintain a potential safe distance. A manually tuned fuzzy control was created for this reason and is discussed in the following section.

4.4 Obstacle Avoidance Controller

The obstacle avoidance controller's objective is to calculate a commanded waypoint for the previously created controller based on the distance to the closest obstacle. If the distance is less than the safe radius we wish to maintain, then a waypoint that has a safe distance is commanded. For this purpose, we propose an Obstacle Avoidance Fuzzy Inference System or OAFIS. This FIS takes in three inputs of the spherical coordinates of the obstacle as seen from the body fixed frame B at the center of mass of the quadrotor which has all of its axes aligned with the world frame w. The spherical coordinates of the obstacle as seen by the quadrotor fixed body frame B; i.e., radius, polar and azimuthal angle are linguistically named as "distance_to_obstacle", "polar_angle" and "azimuthal_angle" for the fuzzy inputs. The expected output of OAFIS is a commanded waypoint, it has three fuzzy outputs namely "X_goal", "Y_goal" and "Z_goal". The outputs from OAFIS are corrections that have to be made to the quadrotor positon to avoid an obstacle and hence the quadrotor position is added accordingly to give the waypoint coordinates in G frame of reference. This way we could incorporate a waypoint planner could still pass waypoints to the control system while the OAFIS works to make corrections if any obstacles are withing a safe distance. The membership functions for the inputs and outputs are shown in Fig 13 and Fig 14 respectively. The definitions and nomenclature for these membership functions were chosen manually and are straightforward.

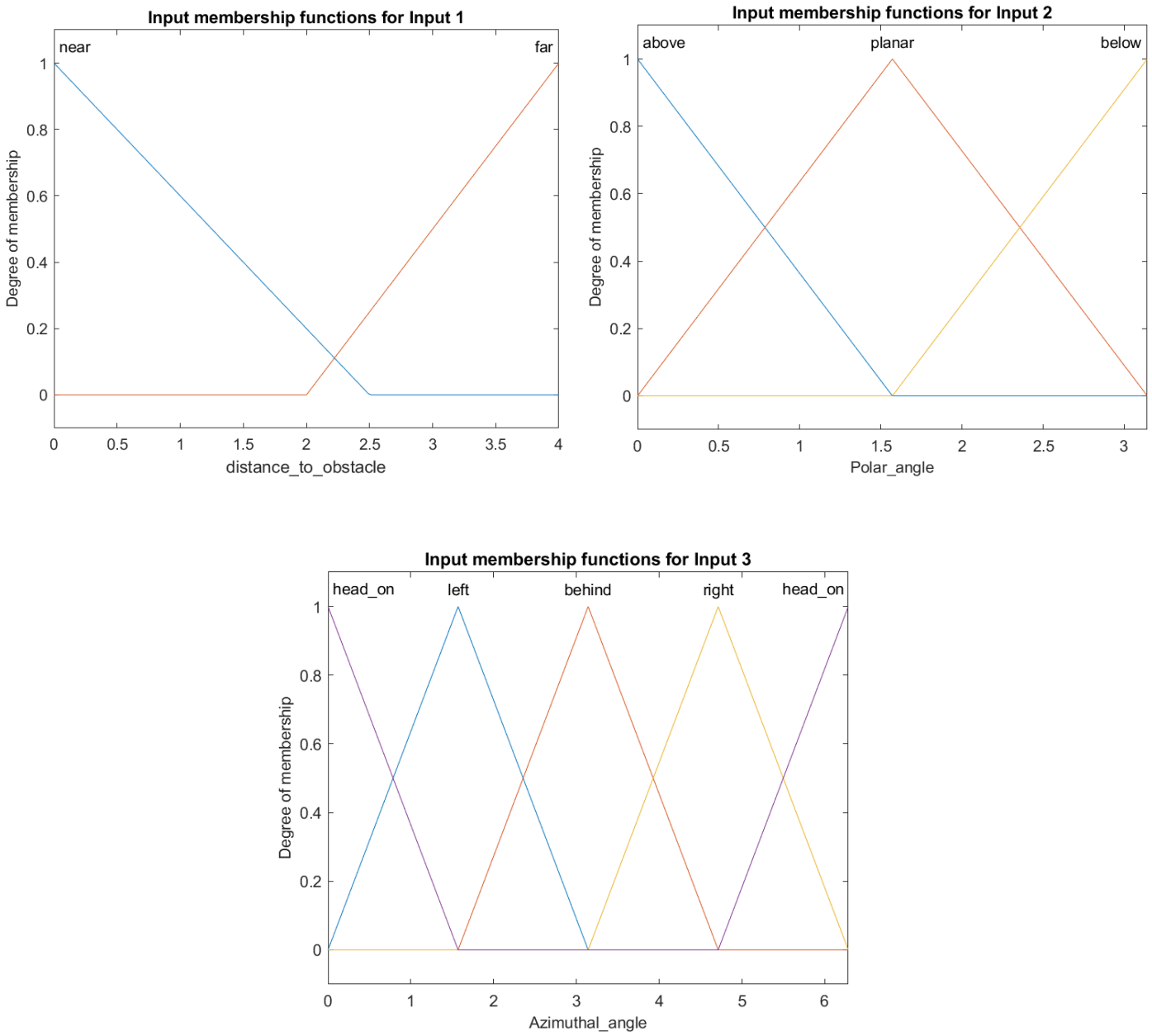


Figure 13: Membership functions for OAFIS inputs

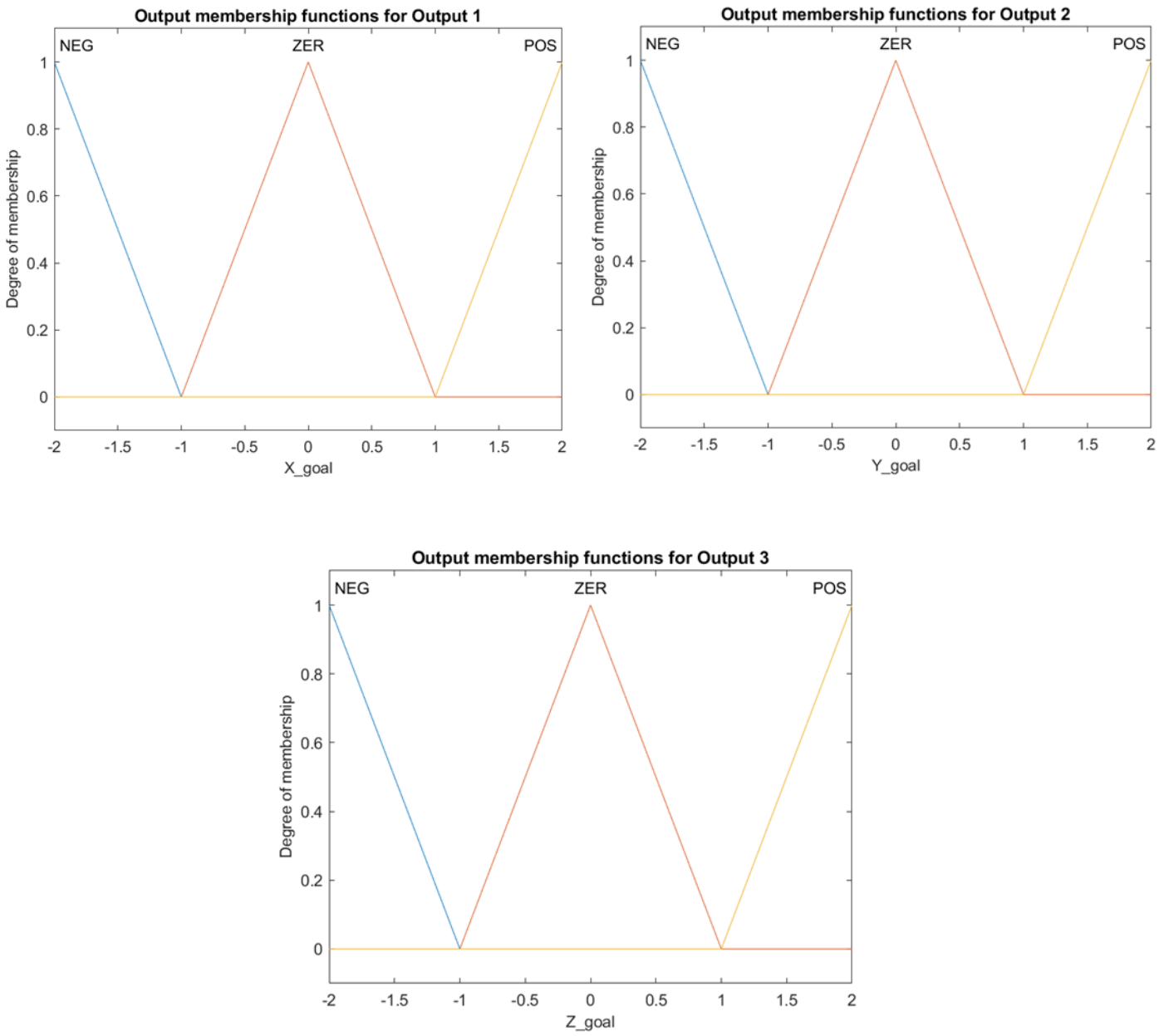


Figure 14: Membership functions for OAFIS outputs

The rule base for OAFIS was developed intuitively as application of the rules in different for the given scenario is straightforward. If no obstacle is within the defined safe radius, no new waypoint is necessary. If an obstacle is near, the output is made to move the quadrotor away from it. As opposed to the complete

combination of the complete rule base as done for the genetically trained controllers, the following IF-THEN rules served sufficiently to achieve the obstacle avoidance objective:

1. If DISTANCE_TO_OBSTACLE is FAR then X_GOAL is ZER, Y_GOAL is ZER and Z_GOAL is ZER
2. If DISTANCE_TO_OBSTACLE is NEAR and POLAR_ANGLE is ABOVE then X_GOAL is ZER, Y_GOAL is ZER and Z_GOAL is NEG
3. If DISTANCE_TO_OBSTACLE is NEAR and POLAR_ANGLE is BELOW then X_GOAL is ZER, Y_GOAL is ZER and Z_GOAL is POS
4. If DISTANCE_TO_OBSTACLE is NEAR and POLAR_ANGLE is PLANAR and AZIMUTHAL_ANGLE is HEAD_ON then X_GOAL is NEG, Y_GOAL is ZER and Z_GOAL is ZER
5. If DISTANCE_TO_OBSTACLE is NEAR and POLAR_ANGLE is PLANAR and AZIMUTHAL_ANGLE is LEFT then X_GOAL is ZER, Y_GOAL is NEG and Z_GOAL is ZER
6. If DISTANCE_TO_OBSTACLE is NEAR and POLAR_ANGLE is PLANAR and AZIMUTHAL_ANGLE is BEHIND then X_GOAL is POS, Y_GOAL is ZER and Z_GOAL is ZER
7. If DISTANCE_TO_OBSTACLE is NEAR and POLAR_ANGLE is PLANAR and AZIMUTHAL_ANGLE is RIGHT then X_GOAL is ZER, Y_GOAL is POS and Z_GOAL is ZER

Before the OAFIS output is used, we need to figure which control system will be engaged because the payload stabilization FISs can only stabilize at the same waypoint and the waypoint tracking FISs cannot stabilize a swinging payload because of how they were trained. By consideration of when we would like to use these behaviors generally, a basic switching policy was proposed: ff distance to obstacle is less than a defined safe distance, then OAFIS outputs a corrected waypoint and waypoint tracking FISs are engaged;

else, OAFIS outputs zero correction and the payload stabilization FISs are engaged to stabilize to the payload around the current waypoint or quadrotor position.

We tested this system by placing the quadrotor less than the determined safe distance from an obstacle and giving it a initial payload angle of 45 degrees to imitate detachment from a team while next to a wall. The results observed showed that the quadrotor moved effectively to maintain the desired safe distance from the obstacle, stabilized the payload and returned to hover. Detailed results will be discussed in Chapter 6. The system works well for the designed objectives; however, due to the training conditions not including any initial velocities, the rule base elements required for recovering a moving quadrotor-payload system from detachment is not possible. We suspect this can be solved by a seeded training of the existing controllers to learn new rule base elements and no modification to the control structure, but leave this to future work.

4.5 Refinement Training of Controllers

Since the population after training the payload stabilization and waypoint tracking controllers were saved, it is possible to retrain the rule base initializing the new population from existing progress. Staging the learning process like this saves run time and helps the algorithm find new solutions while retaining existing behaviors. Since the payload stabilization FISs can be engaged after the waypoint tracking FISs bring the quadrotor system to near zero velocity at a desired waypoint, the payload stabilization FISs do not need to be retrained. The waypoint tracking FISs, on the other hand, should be retrained using the same fitness function and method, except with new initial condition scenarios along with the old, and initialized with the saved population from earlier training. Retaining the old initial conditions assures that the previously trained behaviors are not lost and the new initial conditions would encourage the algorithm to look for solutions that work well with all the given conditions. The new conditions are similar to the old except with a velocity that ‘throws’ them towards goal position. Since a max velocity of 2m/s was observed during previous testing, this velocity was used as the ‘throw’ velocity. By running the genetic algorithm for all 5 FISs involved in waypoint tracking in this manner, the waypoint tracking FISs are now refined to handle

non-zero velocities as well. To really understand the abilities and limitations of the control system, we developed the application scenarios as discussed in the following chapter where the control system was rather thoroughly tested.

4.6 Switching Policies

For obstacle avoidance case or immediately after failure, the switching policy used would engage the waypoint tracking FIS sequence unless the distance between the quadrotor and the nearest obstacle is more than a predefined safe distance of 2m. If the distance is more than 2m, the payload stabilization FIS sequence is engaged. (Given that the cable length is 1m and the quadrotor radius is 0.5, this amount was determined to be reasonable by trial.) For multiple waypoint cases, the switching policy will have waypoint tracking FIS sequence engaged unless the distance between the waypoint and the quadrotor is less than 0.4m. This distance is the stray distance that was observed during the payload stabilization trials. The assumption here was that if stabilization is engaged 0.4m from the desired waypoint, the quadrotor would settle near the desired waypoint. These switching policies were sufficient to observe the performance of the trained behaviors working cooperatively.

5. SCENARIOS

This section discusses different scenarios that were chosen to evaluate our control scheme's performance. The fuzzy control schemes with behavioral actions like payload stabilization and waypoint tracking were obtained by implementing the genetic fuzzy training methodology for a quadrotor-payload system as discussed in Chapter 4. The quadrotor is controlled by the FIS controllers while a goal waypoint is either predefined for evaluation of training results and some multiple waypoint tracking problems or is produced by the OAFIS for obstacle-laden scenarios demonstrating autonomous control. After careful observation of the control scheme's performance in the primary scenarios, secondary scenarios were developed to test its limitations and application bounds. The secondary scenarios include situations that the control scheme was not developed or trained for but should be able to perform reasonably by applying control policies. The detachment of a UAV payload from a 2-team UAV-payload system during hover would be the most basic

start to solve the recovery from the detachment problem. In fact, this scenario is what the payload stabilization controllers were trained on. After this scenario was studied, we proceeded to add complexities to each scenario with the single quadrotor-payload system. The subsequent test should check if the initial team was not hovering but moving at a particular speed. The primary scenario also raises concern about what would be the recovery action if there are obstacles or hazards nearby. Would the system crash into a wall during its recovery? What would be the room needed for performing the recovery action free of collisions? How would the UAV payload move to a safe zone to perform recovery if it is not available at the moment of detachment? These questions lead us to the next scenario to evaluate the waypoint tracking controller. having it track different waypoints in 3D space. The following scenarios would evaluate the performance when detachment occurs near a wall where the actions of Scenarios 1 and 2 have to be performed in a sequence depending upon switching policies. Finally, experimental scenarios to find the robustness and performance limits of the controller are discussed.

5.1 Payload Stabilization after detachment during hover

In this scenario, the UAV is given a payload angle offset that is bigger than from what it was trained on, i.e, 60 degrees while it is at hover and the result for the entire system is analyzed. The payload's offset is also set at 45 degrees with the XY frame so that there is an angular disturbance in both the XZ and YZ projected frames as shown in Fig 15. To note, all orthographic views generated in this study use azimuth angle of 30˚ and elevation angle of 30˚. This would engage all five of the trained FISs to observe how well they collaborate despite being trained separately. The goal point passed to the payload stabilization FISs is the initial position of the UAV which is set as the origin of world frame W. The factors that will be useful to determine if the payload stabilization is working as intended are the settling time of the cable's angular velocity vs time graph and the stray of the UAV with respect to its initial position. Ideally, we want the stabilization to occur as soon as possible, which can be achieved by moving the UAV in the direction of the payload's motion. But since there is a constraint of minimum stray that was placed in the fitness function, the stabilization motion would be a result of decelerating oscillation about the given waypoint. While this tradeoff between stabilization time and stray distance exists, having the payload stabilize within

6 seconds, the time allowed during training and having the stray less than a meter, which is the predefined length of the payload and cable can be defined as acceptable performance. While zero cable angle is a hard goal to achieve, near zero cable angle velocity and UAV returning to hover at the same altitude are the end states desired from the control maneuvers.

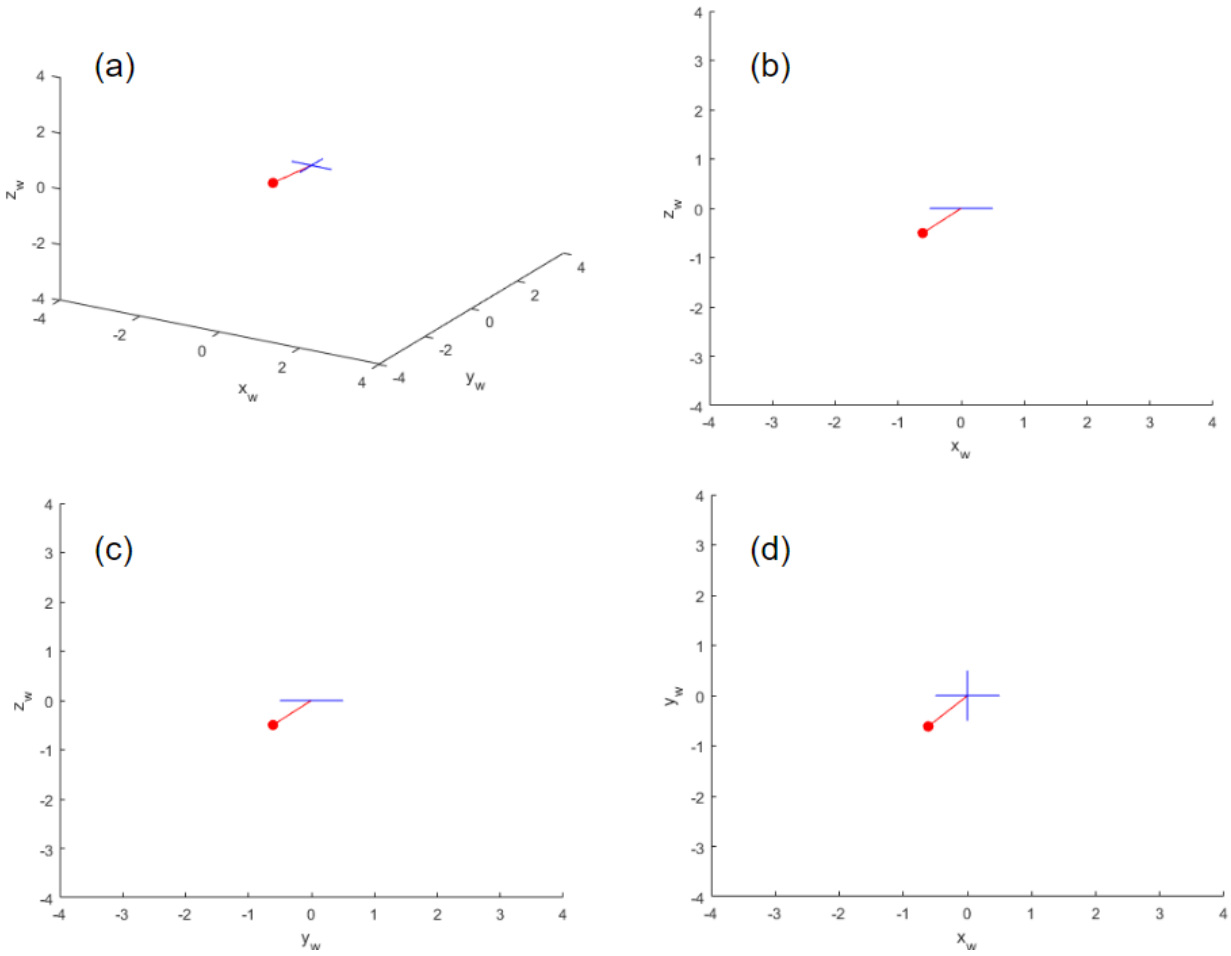


Figure 15: Scenario 1: Payload released when α=60˚ and β=45˚; (a) Orthographic View; (b) Front View (XZ); (c) Side View (YZ); (d) Top View (XY)

5.2 Waypoint Navigation

In this scenario, the UAV is given a desired waypoint to reach in a sequence and the experiment is repeated for different directions of travel. The initial position of the UAV is set as the origin in the world frame w. While the waypoints that the FISs were trained on were placed linearly on the XYZ axes of world frame w,

this means that only 3 FISs were actively used during training simulations. The waypoint (3,3,3) is supplied in a sequence and to engage all 5 FISs to work collaboratively as shown in Fig 16. Payload stabilization is not a task that the waypoint tracking controllers can handle, as previously explained in section 3.7. Hence, the initial cable velocity is set as zero, similar to the training cases. The error history between desired waypoint and current position can be used to analyze rise time, settling time, and overshoot percentage. If these values are within acceptable ranges as compared to a PID controller or a comparable fuzzy controller for a UAV, the performance of the control scheme is considered validated. Given that the control scheme does not take the payload's mass or position as input at this low-level, the performance expectations are admittedly only aimed to be comparable to traditional controllers.

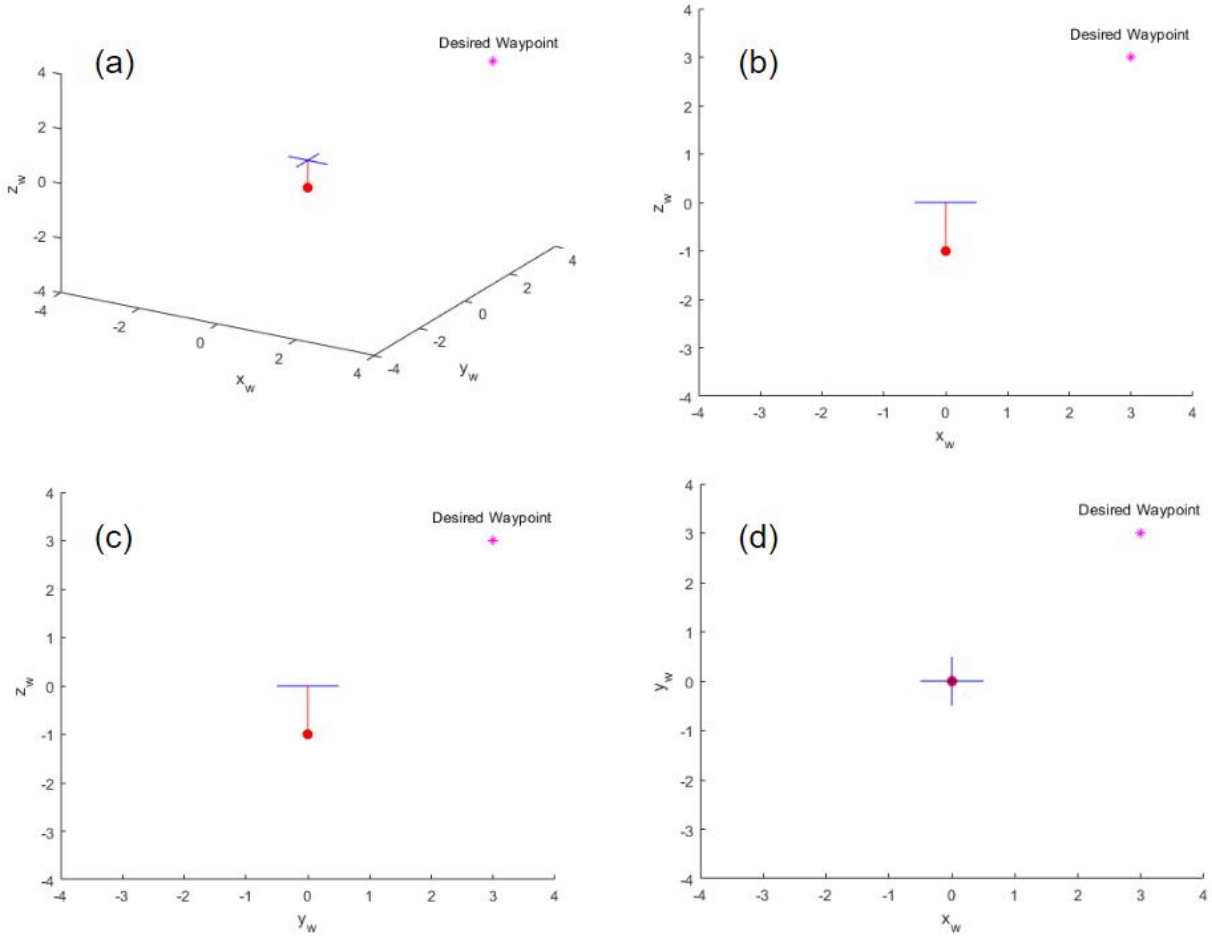


Figure 16: Scenario 2: Waypoint navigation from (0,0,0) to (3,3,3) with no initial payload disturbance; (a) Orthographic View; (b) Front View (XZ); (c) Side View (YZ); (d) Top View (XY)

5.3 Payload Stabilization after detachment during motion

In this control task, the UAV with some initial velocity is given a goal waypoint to reach along with a payload disturbance angle to replicate detachment from a 2-UAV team during motion as shown in Fig 17. The control system must reach the waypoint and then switch over to the payload stabilization FISs to stabilize the payload. Since the payload stabilization FISs do not have any history of training with velocities, the motion tracking FISs are responsible for braking near the goal waypoint. A heuristic switching policy was used to swap control over to the stabilization FISs the motion tracking FISs reach the goal waypoint and hover with a swinging payload. Due to how the stabilization training was set-up, stabilization is only doable when the current position is close to the goal waypoint according to the fuzzy definition. Reduction of cable angle velocity to near zero with minimal stray from the goal position are the desired end states of this experiment. Acceptable stray distance is defined by the maximum stray distance as observed in Scenario 1.

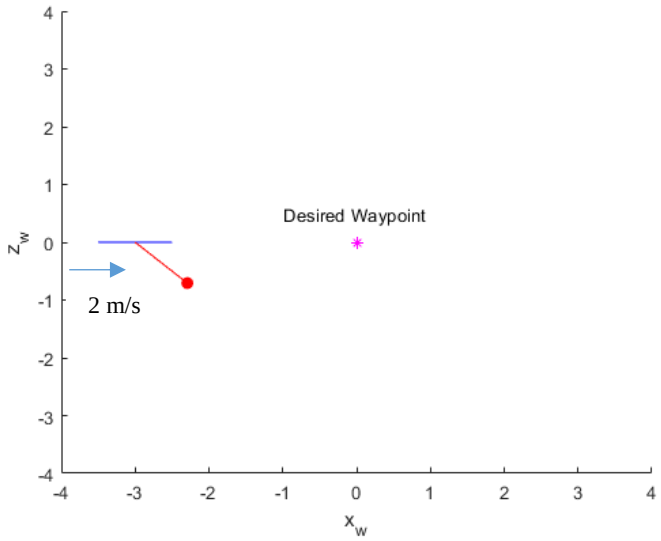


Figure 17: Scenario 3: Waypoint navigation moving right on the X axis from (-3,0,0) to (0,0,0) with cable disturbance of 45˚

5.4 Payload Stabilization after detachment near wall/hazard

In this scenario, the UAV is positioned within 1m, at (2.2,0,0) from a wall with zero velocity as shown in Fig 18. The cable is positioned in a way that the payload is to swing into the wall. OAFIS is used to determine goal positions in this scenario and is provided a way to calculate the closest obstacle position from the UAV body-fixed frame of reference B. Based on position of the obstacle relative to the UAV, OAFIS would provide a waypoint that navigates the UAV away from the hazard. A switching policy is used to switch from waypoint tracking to payload stabilization FISs once the UAV is at least 2m away from all neighboring obstacles. The distance to obstacle from the UAV and payload and settling of cable angle velocity are used as factors to evaluate performance of the control scheme. If there is a collision between the UAV or payload and the obstacle or if the cable is not stabilized, then the control scheme would need further modifications until the objectives are met.

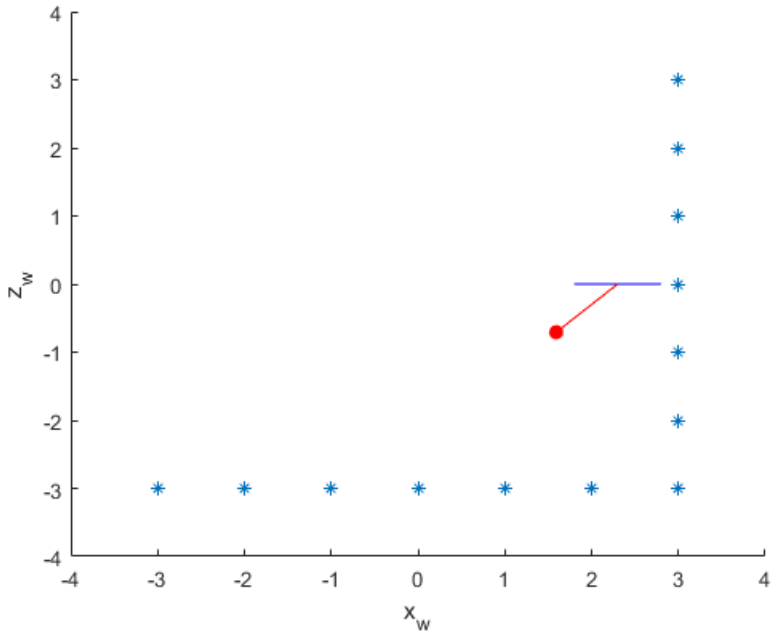


Figure 18: Scenario 4: Failure next to wall

5.5 Robustness Check using off-training system characteristics

In this scenario, we iterate scenario 4 with the payload mass (m = 2 kg) and cable length (l = 1.5m) slightly different from the parameters that the system was initially trained on. Though any changes as per the ratios

chosen shouldn't affect behavior, off-ratio parameters are chosen for this scenario. The observations from this experiment will show how well the controllers can perform without retraining. While similar performance is expected to some level, this experiment will also show how the control system can be modularly used with relevant policies.

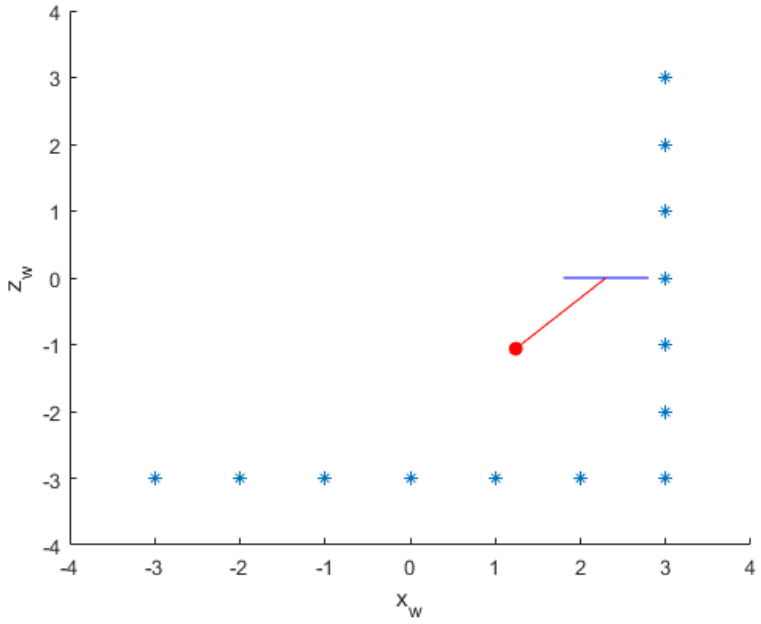


Figure 19: Scenario 5: Failure next to wall with larger mass and cable length

5.6 Multiple Waypoint Navigation

Consider a situation where the UAV should continue tracking a set of waypoints after payload stabilization to an extent where it can potentially continue the previous transport mission (without switching to using another algorithm). To verify how far can we utilize the same control system beyond stabilization, this scenario gives the UAV a set of waypoints to reach in a sequence and the results are analyzed. UAV position is compared with the desired position against time. The waypoint sequence is given in Table 4 and the UAV's initial position is F, such that the sequence ends where it starts and Fig 20 shows the same visually.

Waypoint	A	B	C	D	E	F
Coordinates	(1,0,0)	(1,3,0)	(1,3,3)	(1,0,3)	(-2,0,3)	(-2,0,0)

Table 4: Waypoint sequence for Scenario 6

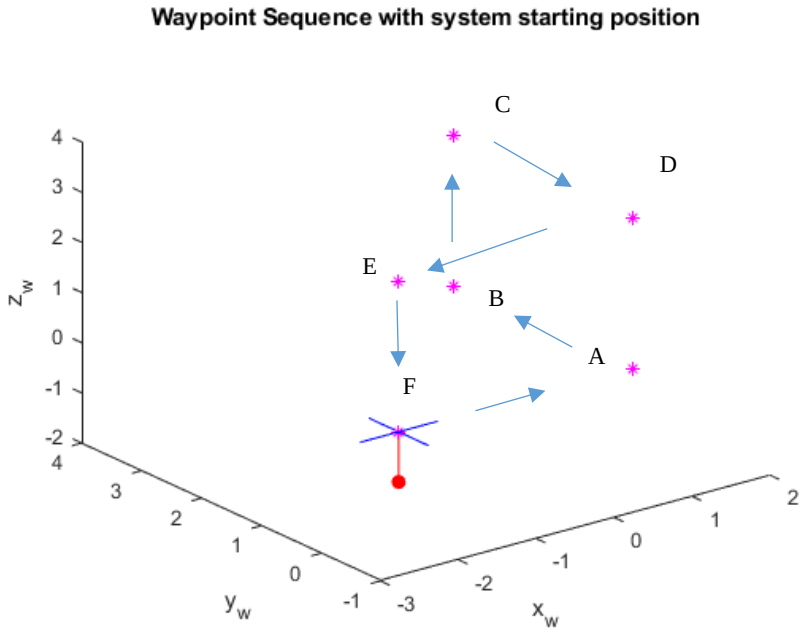


Figure 20: Scenario 6: Waypoint sequence with system starting position

5.7 Dynamic Waypoint Tracking

In this scenario, the UAV is made to track a dynamic waypoint, starting from rest. The path is traced by the equations $x_{goal} = 1 + t; y_{goal} = 0; z_{goal} = 0$. The UAV's ability to follow the waypoint, cable angle, and velocities are observed. While this scenario is different from what the controllers were trained to do, the limitations at which the control scheme fails can be observed in these tests.

The described scenarios help outline the abilities of the proposed control scheme, as well as the bounds within which it has to operate to avoid failure. The simulation results from these test scenarios are discussed in the next chapter along with failure case observations.

6. RESULTS

The simulations are run by solving the dynamic equations using MATLAB's ode45 and the solution is used to plot graphs and animate the motion for observation. This section is constructed to detail the training scenarios first, then to get an understanding of the performance of the control system, the scenarios previously discussed are simulated.

6.1 Training Results

Prior to discussing the results of the scenarios, it is necessary to understand what the behaviors were trained on and what were specifically learned by applying the methodology. The controllers were trained in two stages: training the controllers responsible for the XZ plane in 2D, and then training the controllers responsible for the YZ plane in 3D. The results of FIS1 to FIS3 or fis_x, fis_roll, and fis_z will be first discussed, followed by FIS4 and FIS5 or fis_y and fis_pitch for payload stabilization, and waypoint tracking training setup are shown here; particulars about the training implementation were already discussed in previous chapters.

6.1.1 Payload Stabilization

The payload stabilization training case involves releasing the payload attached cable at an angle of 45 degrees with the global vertical axis, while the UAV is given its own initial position to hold and hover at. Due to the disturbance of the swinging payload, the UAV may be dragged from the initial position and the repeated oscillations should be arrested at the earliest to stabilize the system and prevent potential collisions with a nearby hazard. Since the distance required for the stabilization maneuver is unknown prior training, hazards were omitted, and the objectives as formulated by the penalty function are to reduce cable velocity as well as the position errors between goal position and UAV's position. The stabilized state that is desired is simply defined as the immediate reduction of swinging velocity and the UAV's return to hover as close to the initial/goal position as possible. Additional constraints were not placed to return the UAV to initial position as stabilizing swing is prioritized and the finite position error or stray is acceptable as a trade-off.

For the initial training stage in 2D (XZ plane only), the GA was set up with a population of 530 for (5 suggested multiplier x 35 rules consequents x 3 controllers = 5 x 105 parameters = 530 (after roundoff).

The chromosome with the best/lowest penalty value was used as the training solution which will be verified. To verify the solution, the system dynamics was simulated using the chromosome solution loaded into the relevant FISs; fis_x, fis_roll and fis_z. Initial UAV position was passed as the desired waypoint while the payload was given an initial angle of 45 degrees and the UAV has zero velocity in all directions. The results from the simulation were then used to animate the UAV-payload system and generate graphs.

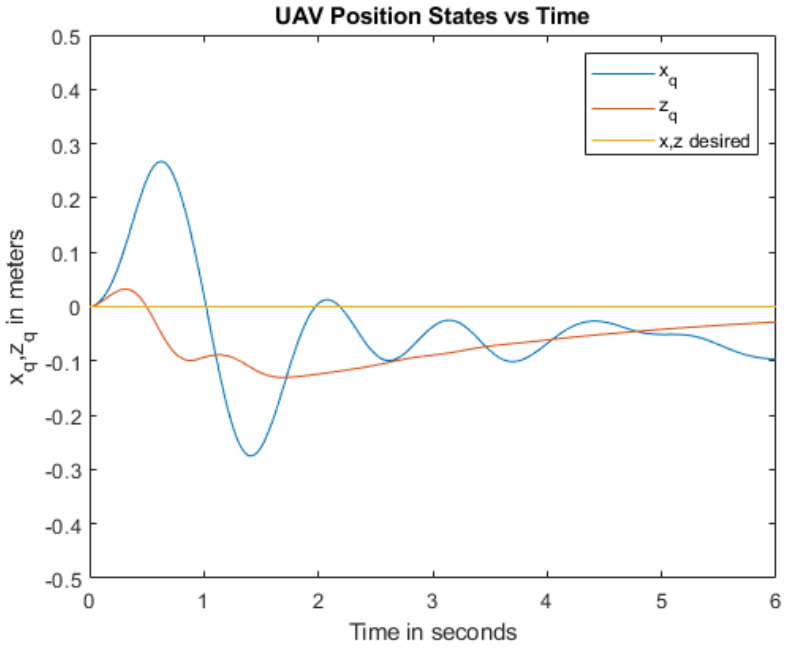


Figure 21: UAV position states vs Time

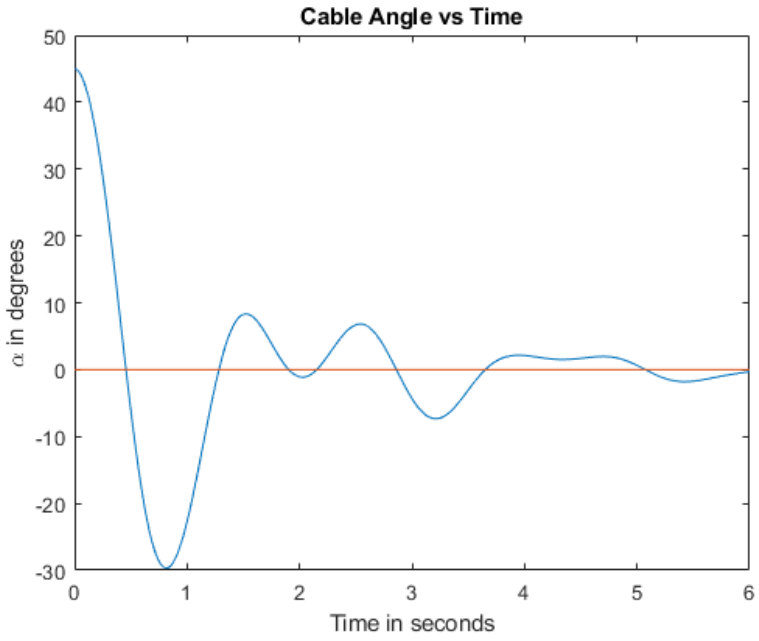


Figure 22: Cable angle vs Time

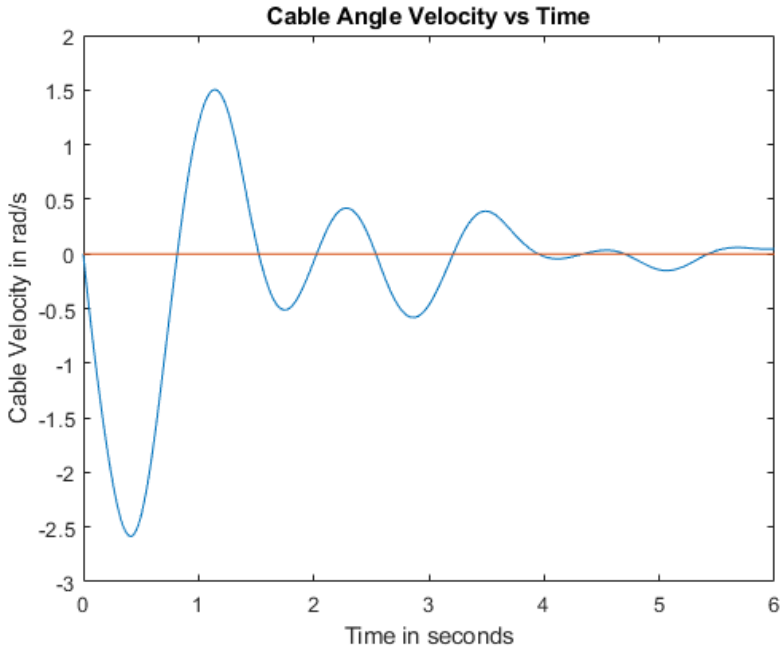


Figure 23: Cable angle velocity vs Time

As seen in Fig 23, the cable velocity quickly converges to zero but not aggressively as an aggressive maneuver here would create a large difference between the given waypoint and UAV position. It can also

be observed that the maximum position error while performing the stabilization maneuver is 0.1 meters. Considering that the length of the UAV and cable is 1 meter and the designed corridor for this application is at least 4 meters wide, this stray can be neglected. The stray from the commanded waypoint can also be compromised considering that a fast cable velocity convergence is obtained. The UAV position change while performing the maneuver for stabilizing the payload is shown in Fig.21. It should be noted that neither payload position nor cable angle were served as inputs during training for this maneuver, and hence the actions performed by the controller are a result of UAV position and velocity errors that occur as a result of the swinging payload. This is not to say payload information was completely omitted during training, as the cable velocity terms in the penalty function played a huge role in the learning of this behavior. The swift minimization of cable velocity brings the cable angle close to zero with small oscillations that dampen as time passes.

Having verified successful training for payload stabilization, it was shown that maneuvers or behaviors can be encoded into rules of simple fuzzy controllers as allowed by the physics of the system. To test this system in 3D, the controllers responsible for YZ plane (fis_y and fis_pitch) were also trained in a similar manner but with the trained controllers fis_x, fis_roll and fis_z being active. Initial position was passed as the desired waypoint while the payload is now set to swing from an angle of 45 degrees in the YZ plane. This training session utilized the 3D version of system dynamics. Since only two controllers are being trained for, the GA was set up with a population of 350 for (5 suggested multiplier x 35 rules consequents x 2 controllers = 5 x 70 parameters = 350 population size).

The best chromosome from training was then loaded into fis_y and fis_pitch to verify training. This second stage of training in 3D makes the two FISs work cooperatively with the previously trained controllers while also learning how to sense the payload disturbance and act to stabilize it. This can be observed by how the fis_z controllers are not part of the training but the two controllers for YZ utilize it based on its previous training. The verification of training was done by simulating the system with all 5 controllers using the trained chromosomes. The results observed from release of the payload from an angle of 45 degrees in the

YZ plane are shown in Fig 24 and Fig 25. The results show similar arrest of cable velocity, and a stray of 0.134 meters.

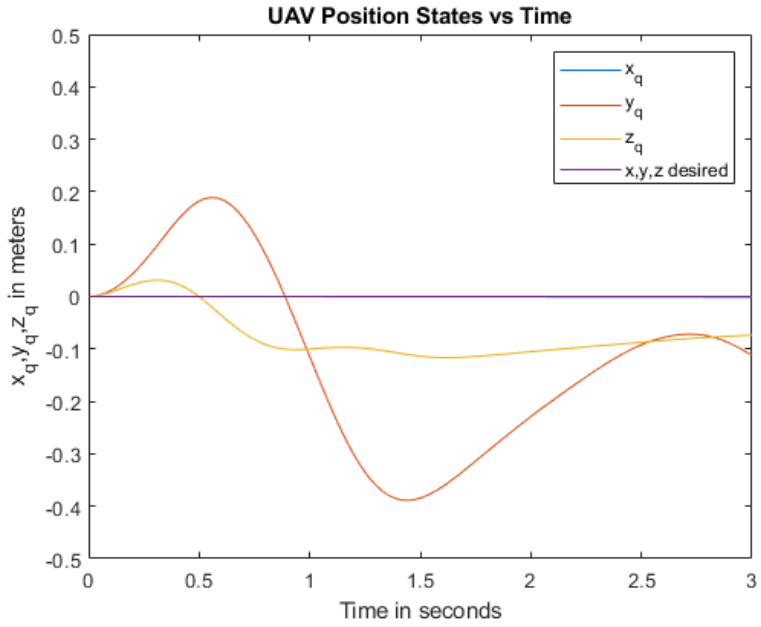


Figure 24: UAV position states vs Time

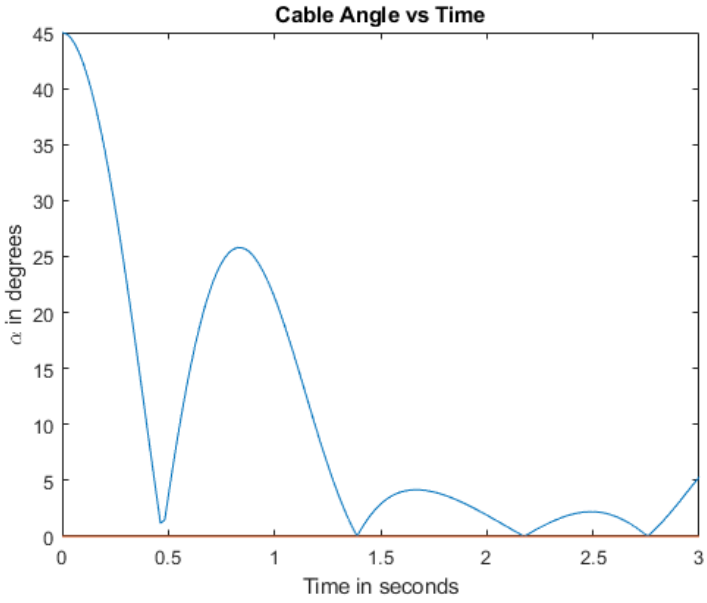


Figure 25: Cable angle vs Time

Only the rules that contribute to the disturbance sensing and corresponding action consequents are tuned for the given application. Non-contributing elements in the rule base may not be tuned and will have a random value assigned to them by the algorithm. This is important to note as it may be used to draw the limitations of the trained controllers. For example, for errors more than 2 meters, the control may not work as the genetic algorithm wouldn't have faced that scenario during the training simulations. But the objective of this study is not to create a 'for-all-scenarios' recovery system, but rather to encode behaviors that can be used appropriately for the scenarios considered in section 3.2 using policies. This means that actions such as release of payload during motion could require a different set of differently-trained controllers that can reach the waypoint, after which the payload stabilization controllers can be used to stabilize the system.

6.1.2 Waypoint Tracking

The objective for the waypoint tracking controllers is to reach a given waypoint within a reasonable time. In the scenario of detachment during motion, the remaining system is expected to reach its waypoint before stopping to stabilize the payload. This decision was made to navigate the destabilized UAV system to a safe zone (a place where other UAV teams that may be travelling in a queue may not be obstructed). In addition to that, the UAV system can also be commanded to track a set of waypoints after recovery to continue the original transport mission.

This controller set's primary training involves positioning the UAV-payload at hover and commanding to track a waypoint at the origin of the XZ plane in 2D. The implementation of this training is discussed in section 4.3.2. For the training stage in 2D, the three FISs being tuned are fis_x, fis_roll and fis_z. The GA was set up with a population of 530, i.e. 5 suggested multiplier x 35 rule consequents x 3 controllers = 530 (after roundoff).

Just like section 6.1.1, the chromosome with the lowest penalty value was used as the training solution and used for validation. The solution chromosome is loaded into the relevant FISs and all four training conditions were simulated using ode45. The first two training conditions demonstrate the UAV's ability to traverse left and right along the X-axis. As shown in Fig 26, the UAV converges to the goal waypoint within

5 seconds, but the response time is not as good as a PID controller. This could be improved by tuning the FIS's membership function parameters, but this study limits its scope to testing the possibility and limitations of these fuzzy systems for the payload recovery application. The overshoot distance and settling oscillations upon reaching the waypoint are ignored within reason, as the UAV must travel 3 meters in 8 seconds while dragging a payload. The payload does swing upon reaching the waypoint and a maximum of roughly 15 degrees is seen for starting from ideal hover condition as seen in Fig 27. This is because cable information was not included in the penalty equation and hence is not prioritized. Including the cable angle would add constraints upon convergence speed during training, and since the payload can be stabilized by the previously trained controllers, if payload swings too far, the controllers can be swapped by a switching policy.

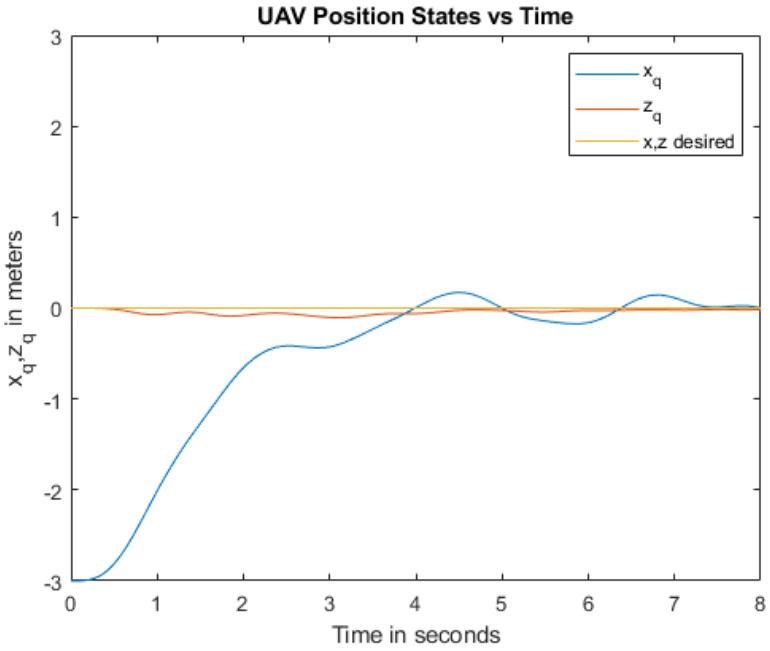


Figure 26: UAV tracking (0,0,0) from (-3,0,0)

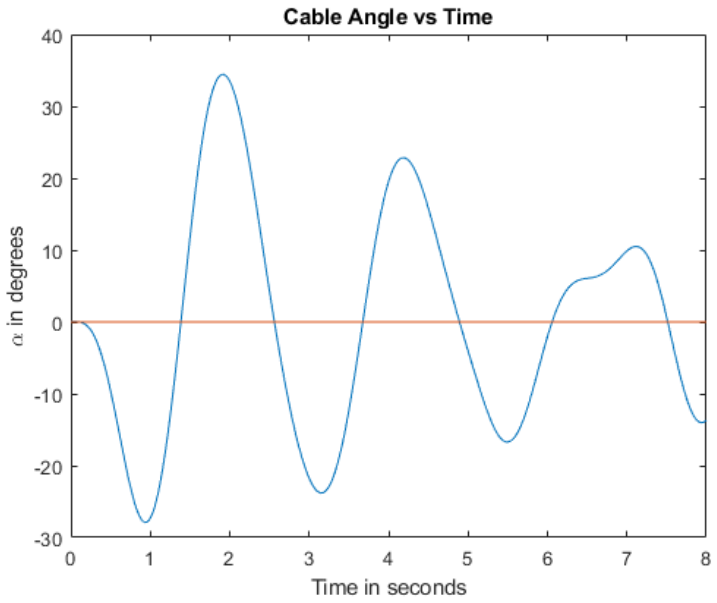


Figure 27: Cable angle vs Time

For vertical rise and sink training conditions, the results were straightforward with little overshoot distance and almost no oscillations. This is possible as thrust is handled directly by a single FIS, 'fis_z' with no payload oscillation throwing the system off balance. The simulation results for those training conditions are shown in Fig 28 and Fig 29.

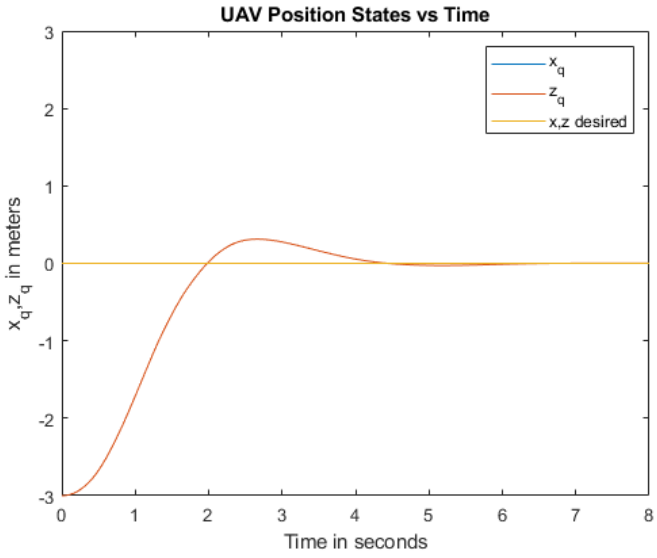


Figure 28: UAV rise from (0,0,-3) to (0,0,0)

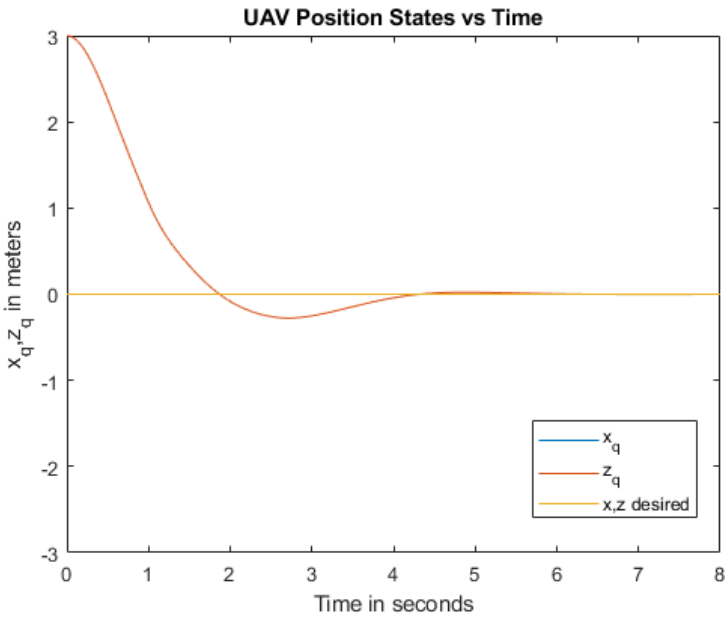


Figure 29: UAV sink from (0,0,3) to (0,0,0)

To scale the controllers to 3D, 'fis_y' and 'fis_pitch' were trained in a second stage of training keeping the previously trained controllers active on 3D system dynamics. Like the second stage of payload stabilization training, the controllers responsible for the YZ plane must rely on the previously trained 'fis_z' controller for thrust adjustments and must cooperate with 'fis_x' and 'fis_roll'. Since only two controllers are being tuned, the GA was set up with a population of 350.

The training was verified by running the training conditions with all five controllers using ode45. The results are shown in Fig 30. Just like the XZ plane tests, the convergence towards the waypoint was observed with an overshoot distance of 0.42 meters for traversing 3 meters along the Y-axis. This performance is acceptable, considering the constraints placed during training and the limited information that the control system utilizes. This concludes the training stages done for this study, as the behaviors desired were achieved with fair performance.

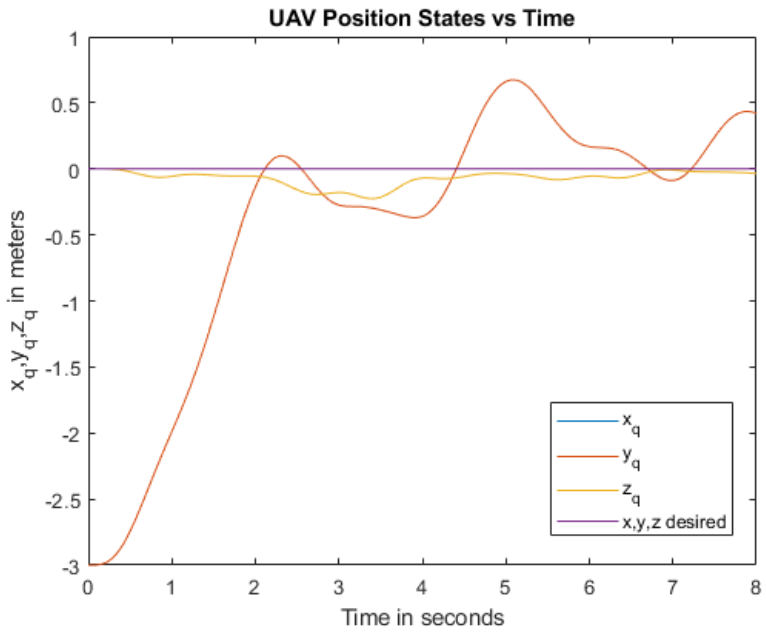


Figure 30: UAV tracking (0,0,0) from (0,-3,0)

While the controllers work well on the conditions that they were trained in, they need to be tested on the scenarios drawn from Chapter 5 to validate their performance. The next section discusses the results from different test cases conducted on developed payload stabilization and waypoint tracking controllers in addition to the OAFIS for obstacle avoidance.

6.2 Simulation Test Results

A series of test scenarios were constructed to test the abilities and limitations of the trained controllers. For some of these tests, the controllers must be swapped between to make use of the trained behaviors as the scenario requires using a switching policy. The switching policies used for these tests were not optimized to keep the primary focus on the developed controllers. The results for each scenario explained in Chapter 5 are discussed below.

6.2.1 Payload Stabilization after detachment during hover

This test was done to validate the payload stabilization FIS's ability to stabilize an angular disturbance that is different from its training conditions. To test all five controllers together, the test is conducted in 3D with the cable angle being released from -60 degrees and rotated 45 degrees about the Z axis so that the swinging

oscillation is not limited to XZ or YZ like the training stages. The UAV was positioned at the origin of the world frame w which is also the commanded waypoint. The results from simulation are shown in Fig 31 to 33. The cable velocity is shown to be arrested within 5 seconds with the UAV straying 0.4 meters from the initial position. After the maneuver, the UAV comes to a hover with the very small cable angle oscillations that disappear over time. But this end state should be good enough to swap over to the waypoint tracking controller and continue the original mission or navigate a new set of waypoints.

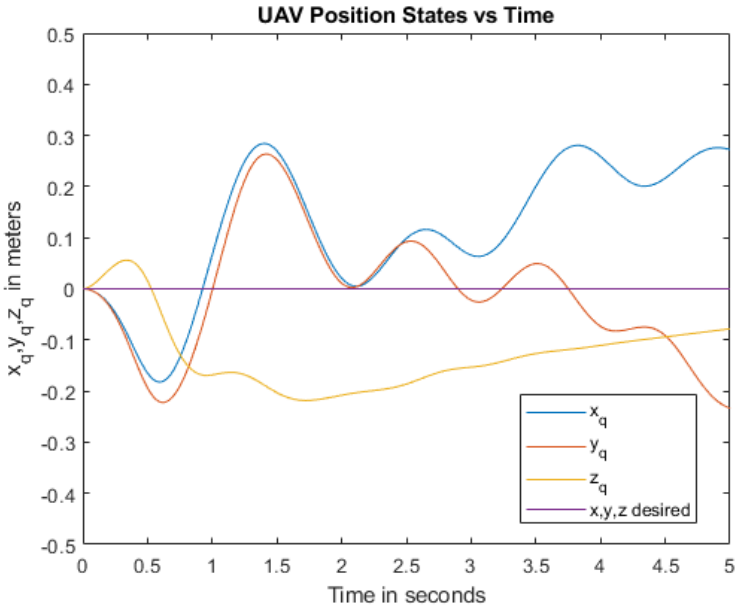


Figure 31: UAV position vs Time

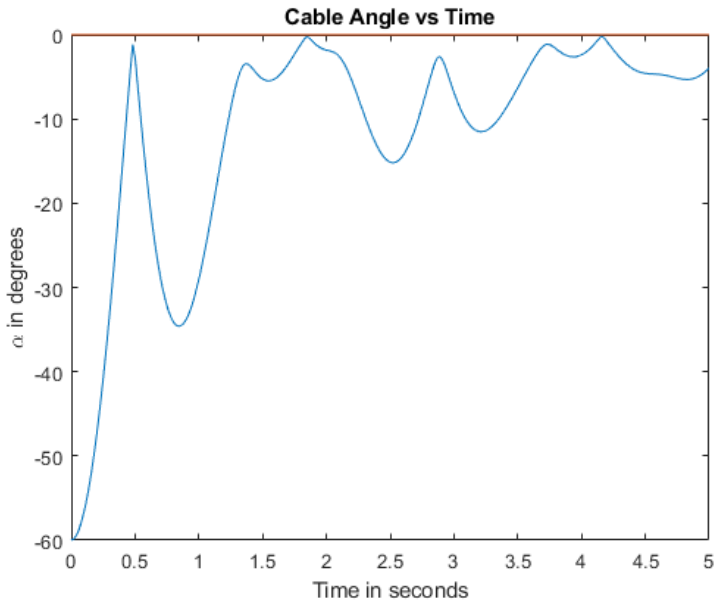


Figure 32: Cable Angle vs Time

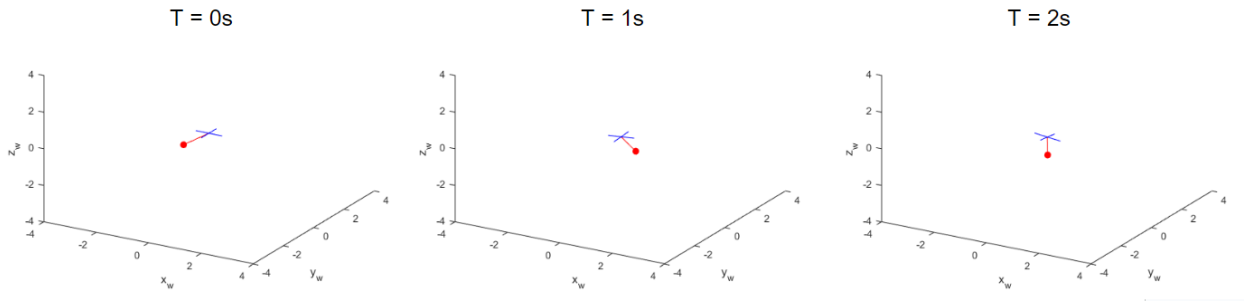


Figure 33: System Simulation Timestamps

6.2.2 Waypoint Navigation

In this scenario, a waypoint is passed and the controllers' ability to converge to a waypoint in diagonal direction is tested. Unlike the training conditions, the choice of these waypoints will engage all 5 FISs to work synchronously as the UAV has to fly diagonally. Fig. 34-36 shows the performance of the motion tracking controllers in a 3D space. It can be seen that the system shows convergence to the desired waypoints with small overshoot (0.5m) and fair response. As discussed in section 6.1.2, the response time and overshoot can be optimized for comparable performance with a PID controller but with the given learning constraints imposed on the controllers, this is acceptable. The increased overshoot is because of

the swinging payload that is not stabilized during these maneuvers and goes to show that ideally a swap to the stabilization controller should be done at the end of convergence at a waypoint as the motion tracking controller wasn't trained and hence unprepared to handle the oscillations.

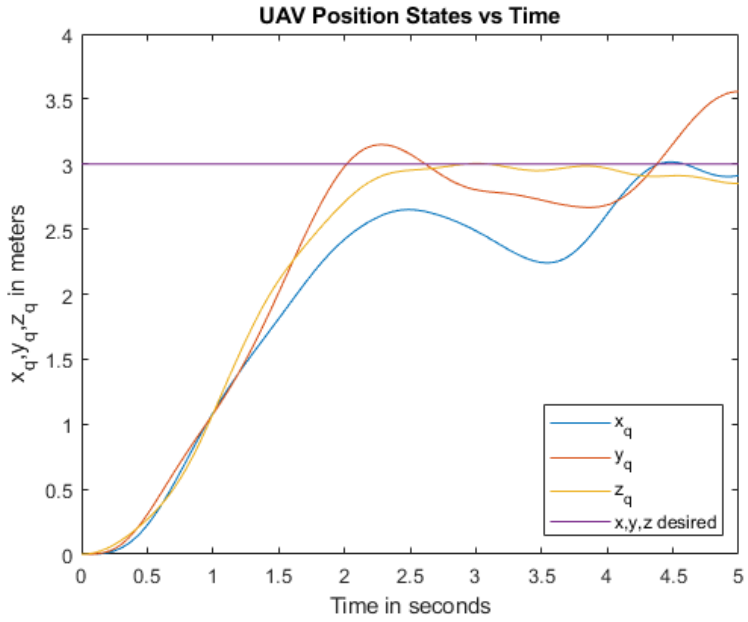


Figure 34: UAV position vs Time

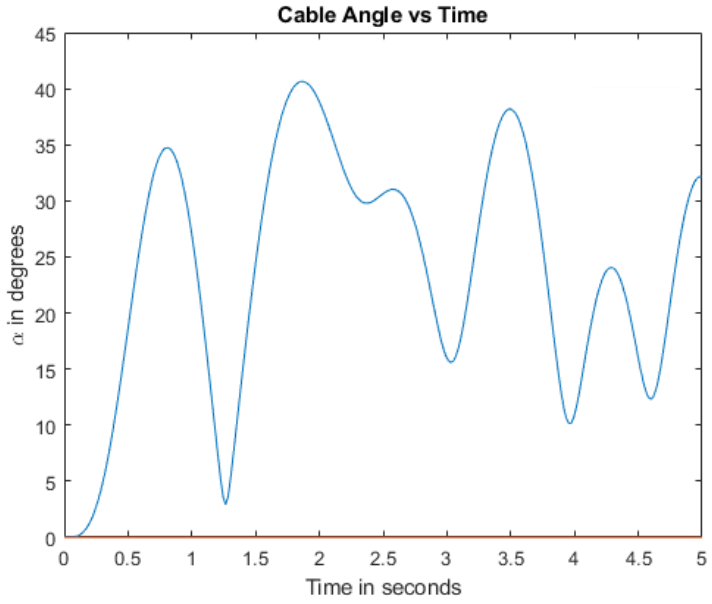


Figure 35: Cable Angle vs Time

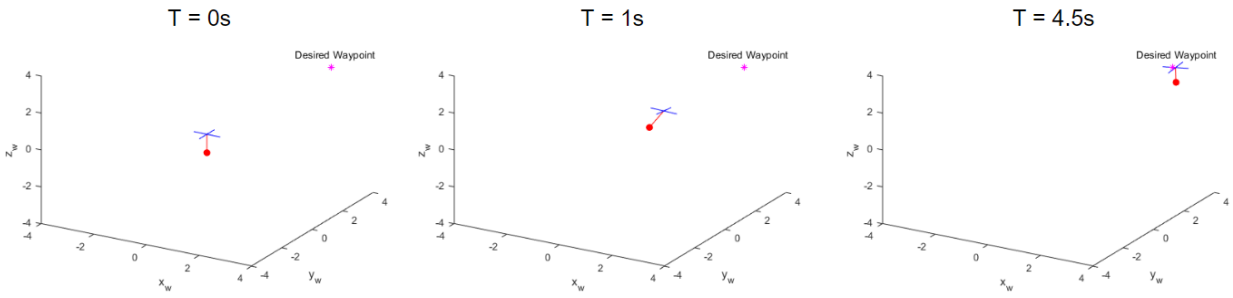


Figure 36: System Simulation Timestamps

6.2.3 Payload Stabilization after detachment during motion

In this test, a UAV-payload system is given an initial velocity in the direction of a waypoint, while the cable is set at 45 degrees to imitate detachment of a team member during motion. The goal waypoint is chosen the origin in world frame w, while the UAV is positioned at (-3, 0, 0) and given a positive velocity of 2 meters per second in the +X direction which is approximately the highest velocity observed when the UAV is commanded to travel 3 meters. Since the UAV is in motion and the stabilization FISs were trained that they can perform the maneuver only at the waypoint, a simple switching policy was used that engages the motion tracking controllers and switches to the payload stabilization controllers only after the former brakes and slows down the system at the given waypoint. The time taken to complete the braking and converging at the waypoint was observed and was used to switch to the payload stabilization controller. The results for this simulation are plotted in Figures 37-39. Though the switching policy is heuristic and non-optimal, it can be seen that the cable velocity is stabilized upon convergence within 6 seconds bringing the cable angle from a maximum observed 45 degrees down to 10 degrees. The positional error from the waypoint is 0.35 meters, which is not bad compared to the performance observed in previous tests. Given that the payload is stabilized near the vicinity of the given waypoint, and that the UAV-payload system can now be scheduled to move to another waypoint, this test demonstrates the payload recovery potential of the developed control system.

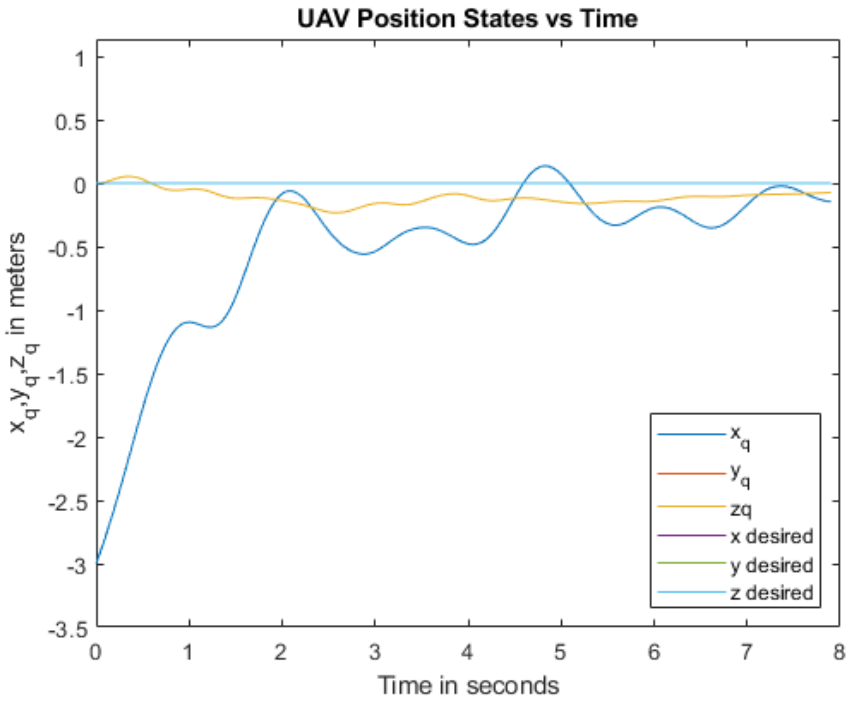


Figure 37: UAV position vs Time

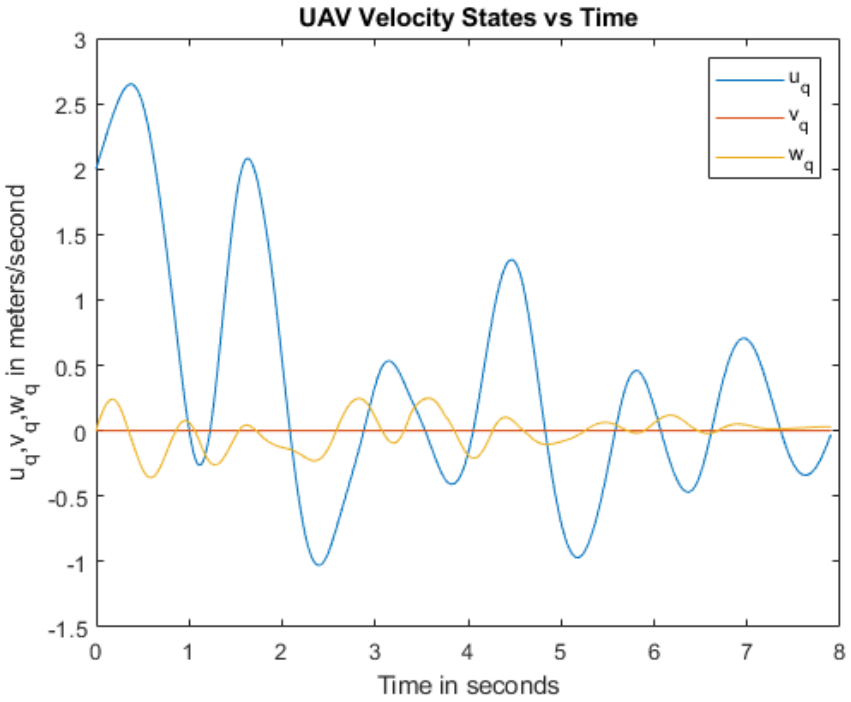


Figure 38: UAV linear velocity in X and Z vs Time

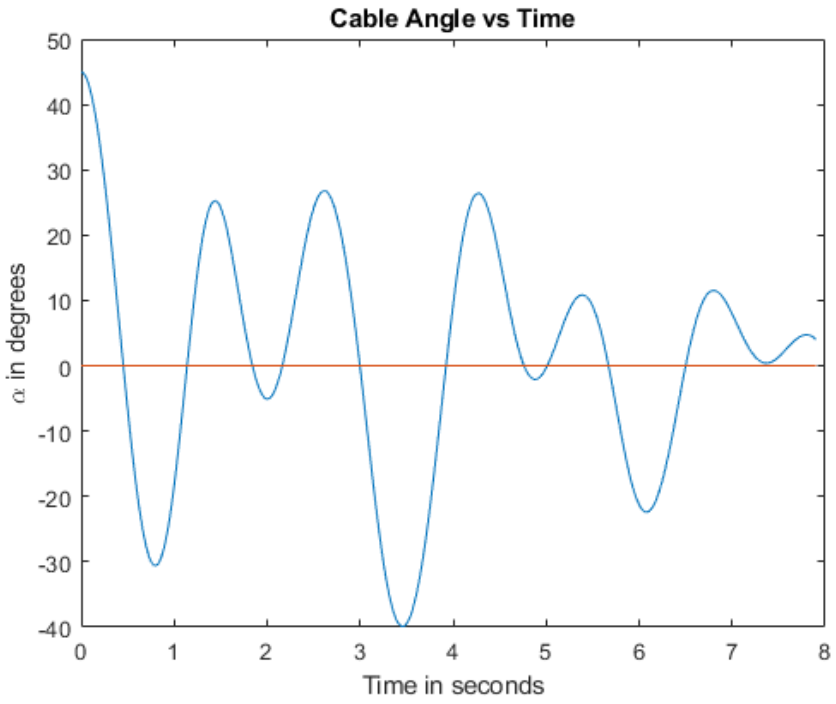


Figure 39: Cable Angle vs Time

6.2.4 Payload Stabilization after detachment near wall/hazard

This scenario was implemented by placing the UAV 0.7 meters away from a wall in the X direction and positioning its cable at 45 degrees with the global vertical away from the wall such that the payload will swing into the wall. The distances are chosen using simple trigonometry such that if the UAV hovers at the same position, the payload will collide with the wall. The UAV can sense its nearest obstacle in cylindrical coordinates from its body centered frame B. The cylindrical coordinates are then used by OAFIS to navigate the UAV to a new waypoint accordingly. Motion tracking FISs are engaged by default. When the distance to the closest obstacle is over a chosen safe distance of 2 meters, the control system swaps over from motion tracking to payload stabilization. In the simulation, the UAV moved away from the wall in time, navigated to maintain the safe distance and performed payload stabilization as shown in Fig 40-44.

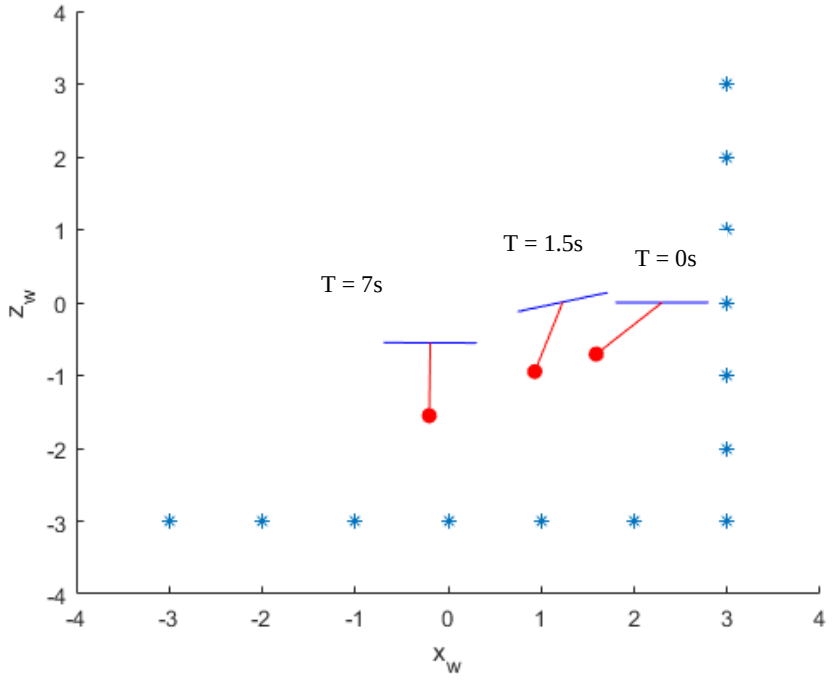


Figure 40: Timestamp images from payload stabilization near wall simulation in MATLAB

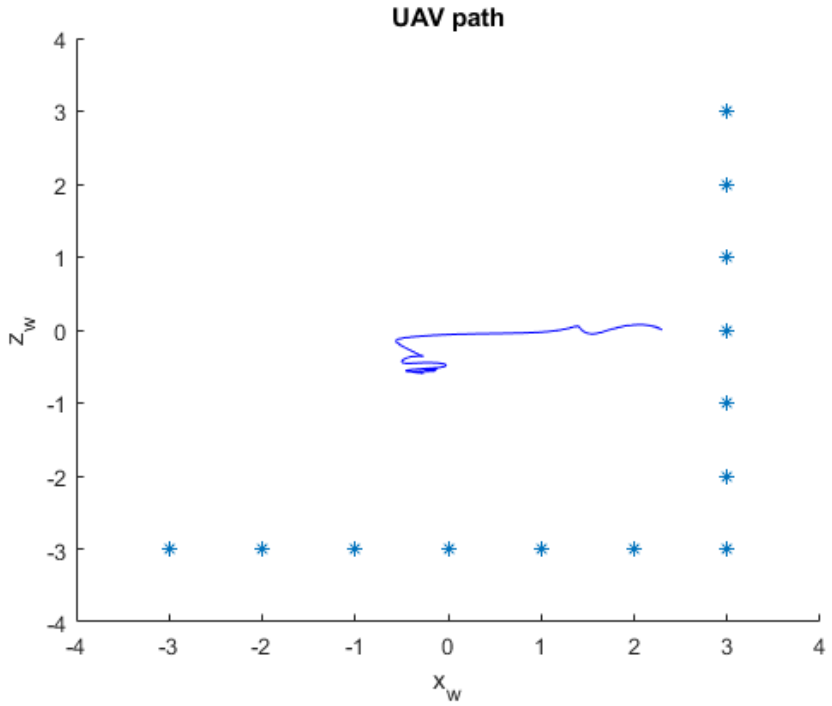


Figure 41: UAV path from payload stabilization near wall simulation

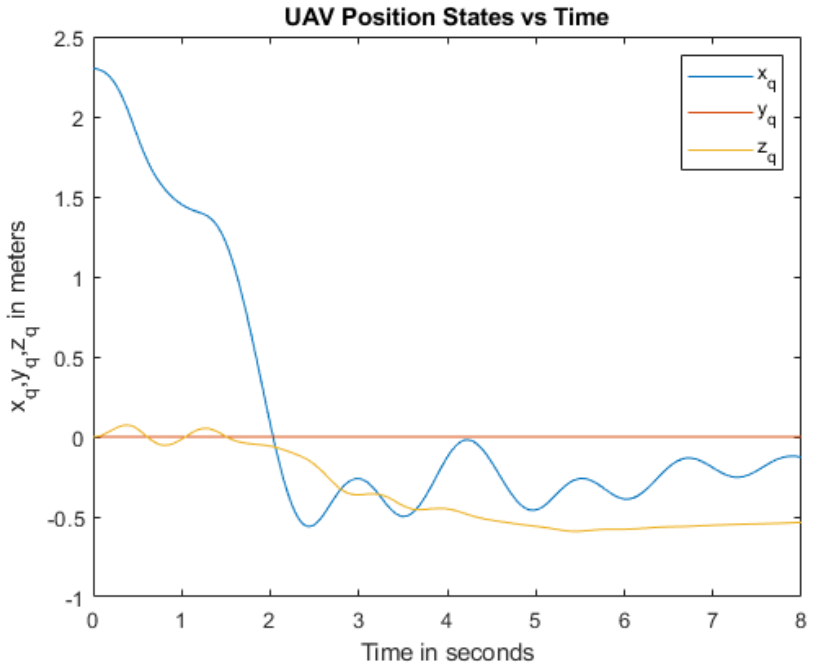


Figure 42: UAV position vs Time

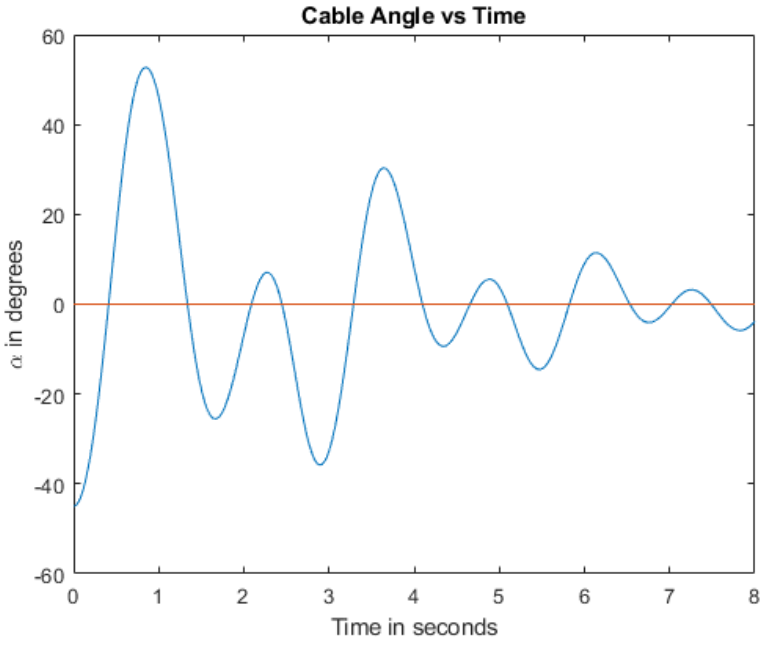


Figure 43: Cable angle vs Time shows dissipation of swinging over time

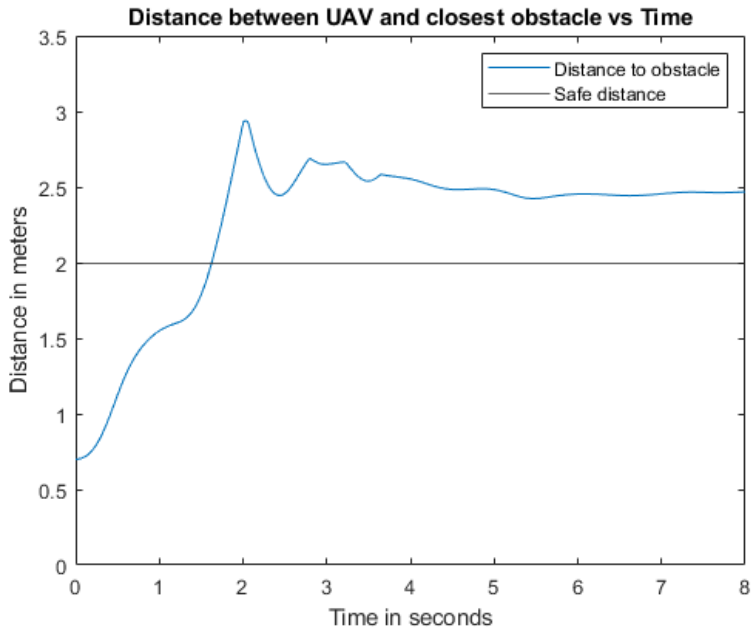


Figure 44: Distance between UAV and closest obstacle

From the simulation results, it is clear that OAFIS effectively navigates the UAV before and collision with the wall and maintains the specified safe distance as specified in our switching policy. The payload stabilization controllers also work to reduce the cable velocity effectively bringing the oscillation down to less than 0.2 radians (<10°) which dissipates as time progresses. These results demonstrate how the modular controllers developed can be effectively used to handle payload stabilization in the vicinity of obstacles when their positions are known.

6.2.5 Robustness Check using off-training system characteristics

In this simulation, the payload mass and cable length are increased, for a different ratio of unitless characteristics from what the system was trained on. The payload mass is set as 2 kg and the cable length is also set as 1.5 m. While these increments are not in accordance with the ratios discussed in Chapter 3, the fuzzy controllers should still apply similar maneuvers to bring the system to a waypoint and stabilize the payload. The results from the simulation are shown below in Fig 45-49.

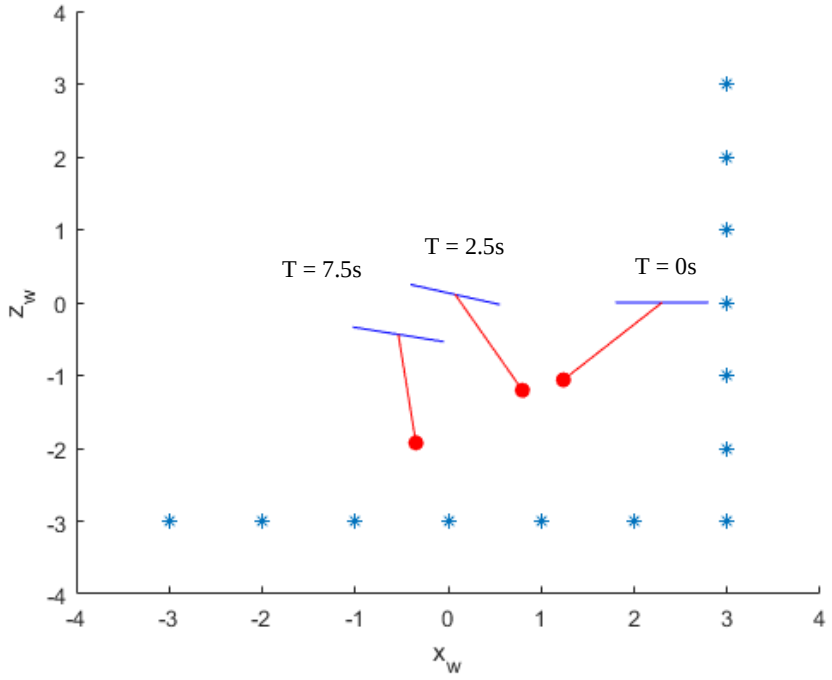


Figure 45: Timestamp images from payload stabilization near wall simulation in MATLAB

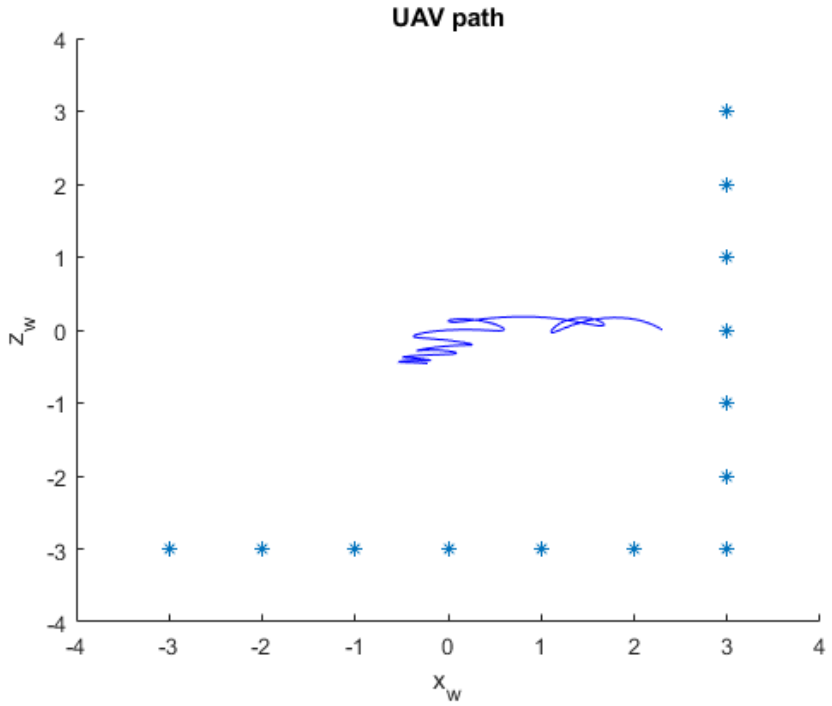


Figure 46: UAV path from payload stabilization near wall simulation

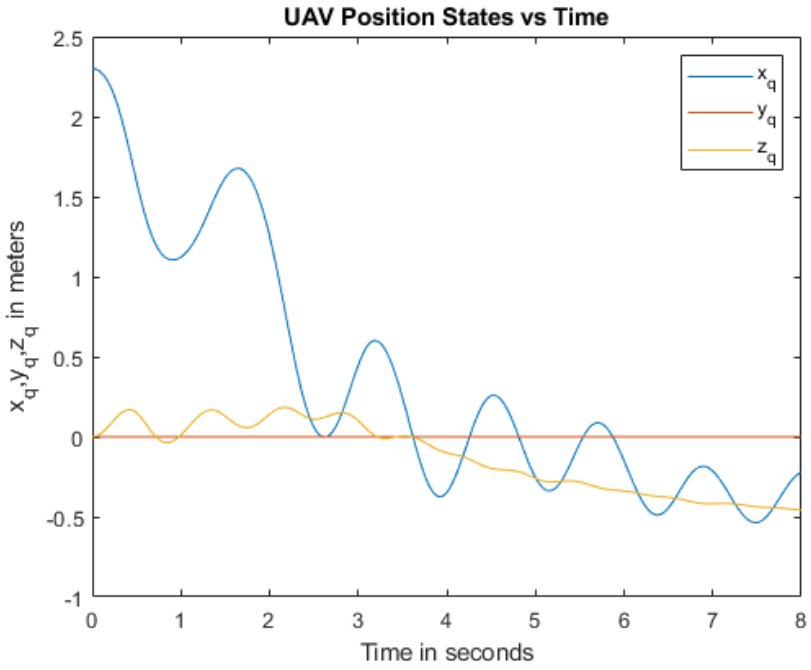


Figure 47: UAV position vs Time

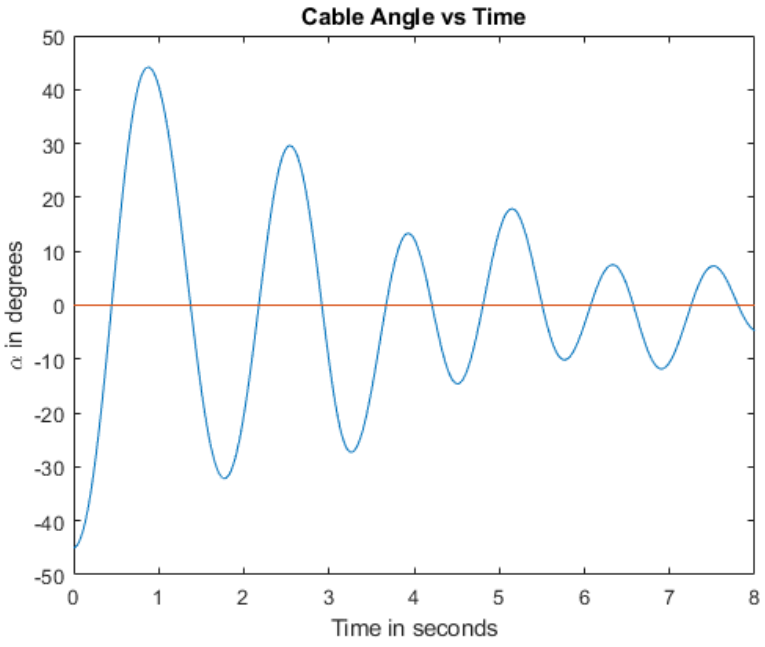


Figure 48: Cable angle vs Time shows dissipation of swinging over time

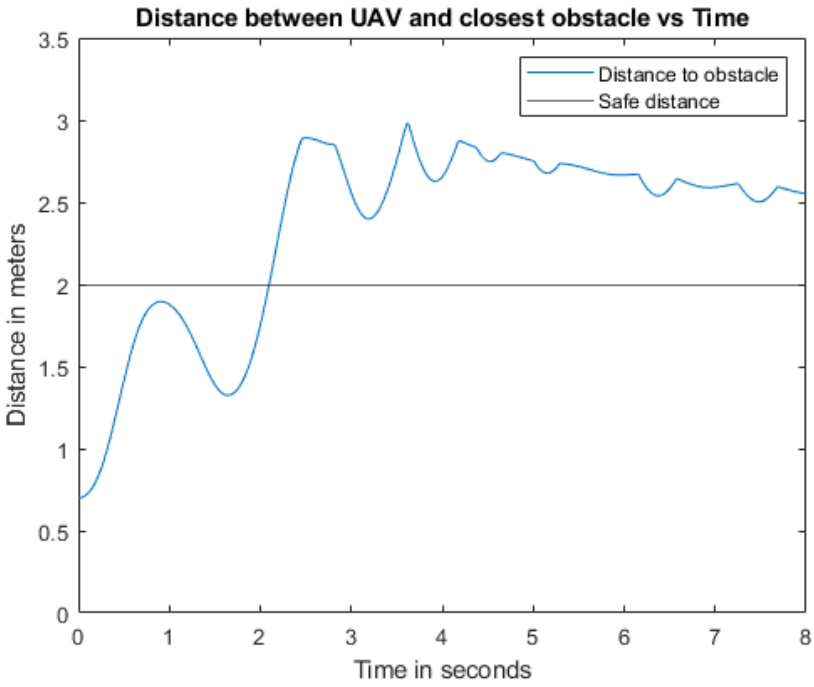


Figure 49: Distance between UAV and closest obstacle

The results showed that the navigation to a safe distance and payload stabilization took a longer time as compared to the previous results, but the UAV was still able to converge at a safe distance and stabilize the payload using the unaltered controllers. Increasing the payload mass is done so under the assumption that the UAV can still generate enough thrust to hover at 50% thrust and under this assumption, the fuzzy classification of thrust doesn't change and hence the rules do not require retraining. While this test demonstrates the adaptability of the controller, the performance is still non-ideal compared to the performance under training characteristics. Thus, while the system is robust to significant divergences between the actual system parameters vs. estimated model parameters trained on, the controllers should be retrained to the best-known specific UAV parameters, if possible, when the unitless values are off-ratio.

6.2.6 Multiple Waypoint Navigation

In this simulation, the UAV was positioned at (-2, 0, 0) and starts from hover with cable angle set to 0 degrees. The UAV is then given a set of waypoints that would require it to move in the +X direction, followed by +Y, +Z, -Y, -X and –Z such that it reaches the initial position. The exact sequence of waypoints

fed is given in Table 5. The waypoints are passed after a time interval of 8 seconds. All coordinates are described in world frame w.

Waypoint	A	B	C	D	E	F
Coordinates	(1,0,0)	(1,3,0)	(1,3,3)	(1,0,3)	(-2,0,3)	(-2,0,0)

Table 5: Waypoint Sequence for Scenario 6

Upon reaching a waypoint the UAV must swap to payload stabilization mode to arrest payload swing before receiving the next waypoint. The switching policy used here was 'if the UAV is within 0.4 meters of the waypoint', which is the average stray observed in scenario 1, 'the payload stabilization controllers are engaged'. The simulation results showed that the control system could traverse all dimensions in 3D while also stabilizing the payload in intervals. The path traced by the UAV and payload are shown in Figure 50.

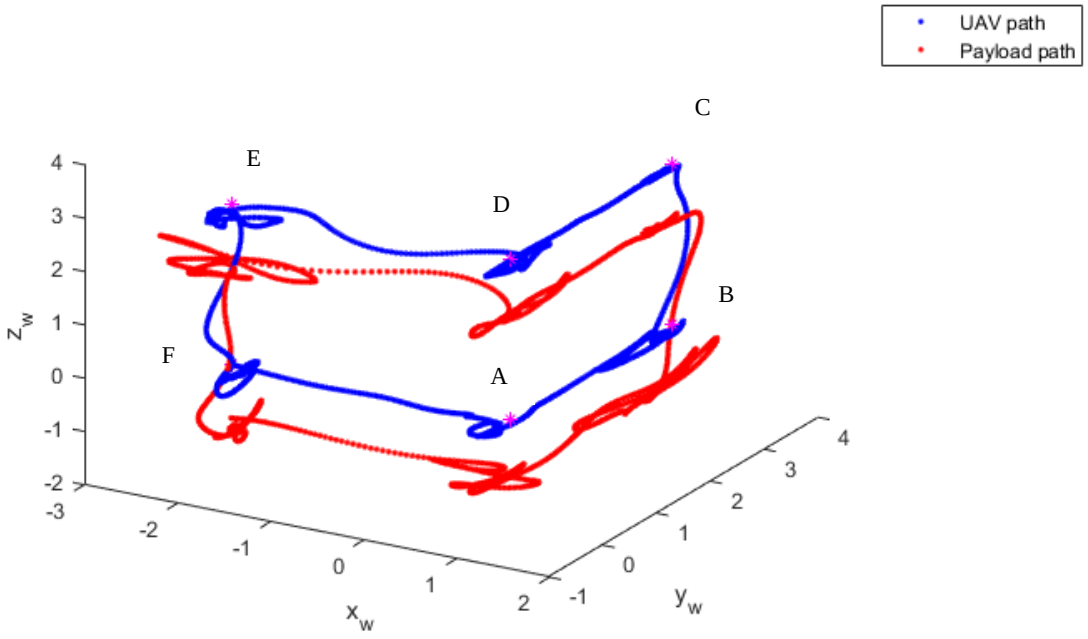


Figure 50: Path traced by UAV (blue) and payload (red)

The results show that the UAV does waver from the waypoint upon convergence to stabilize the payload, but the errors are kept under 0.4 meters because of the switching policy. The attitude response shows convergence, but not without overshoot or oscillations that need to be optimized in future training. While the performance is not satisfactory, this is still very good considering that the controllers were not trained for this act but could potentially be forced to serve such an application by using a suitable switching policy.

6.2.7 Dynamic Waypoint Tracking

In this test, the UAV is made to track a moving waypoint. The results from a linearly moving waypoint is shown below. Figure 51 shows a smooth response of the UAV to track and finally converge at the waypoint after it stops. The results show that the controllers handle this task with a smoother response as compared to a station waypoint that is far where it accelerates and then decelerates close to the goal position.

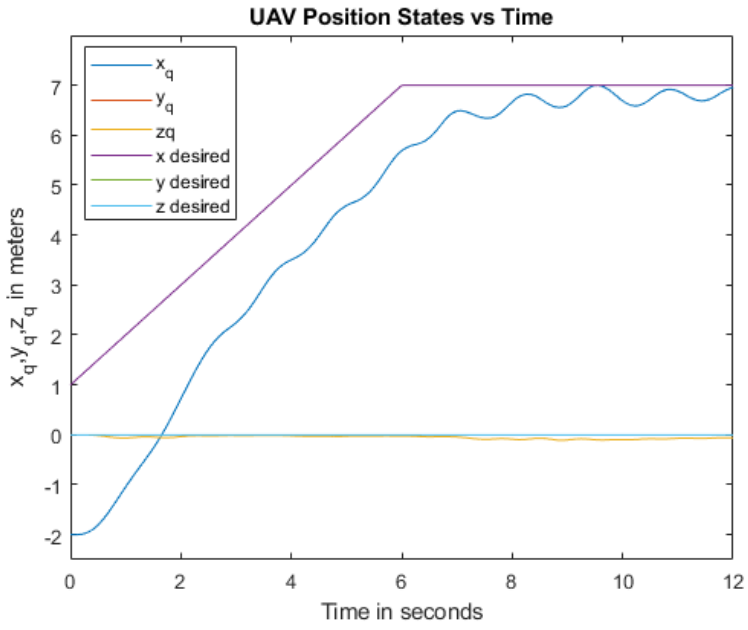


Figure 51: UAV position vs Time

Following the linear waypoint test, we were curious to see how the system would behave if the waypoint was made to follow a circular path. While the UAV seems to follow a path that is close to the path traced by the waypoint, it is not circular as the UAV tries to accelerate linearly towards the current waypoint position. Then as time progresses, the velocity builds up, causing the UAV to destabilize as the fuzzy

controllers were designed for a maximum velocity of 3 m/s. Thus, a limitation of the system is that position errors and velocity should be monitored such that they don't increase beyond the fuzzy controllers' described ranges.

6.2.8 Tests with induced sensor noises

In this test, scenario 4 is repeated but with added sensor noise. The quadrotor is given sensor noise corresponding to +/-1% error and +/-5% error by adding a Gaussian distribution generated value to each input with a mean of zero and standard deviation of 1/3 of the FIS input range. This experiment is in correspondence with Bisig's work [26]. The results for Scenario 4 with 1% and 5% noise is shown in Fig 52 and 53 respectively. 15% noise was tested and found that the control system cannot handle it.

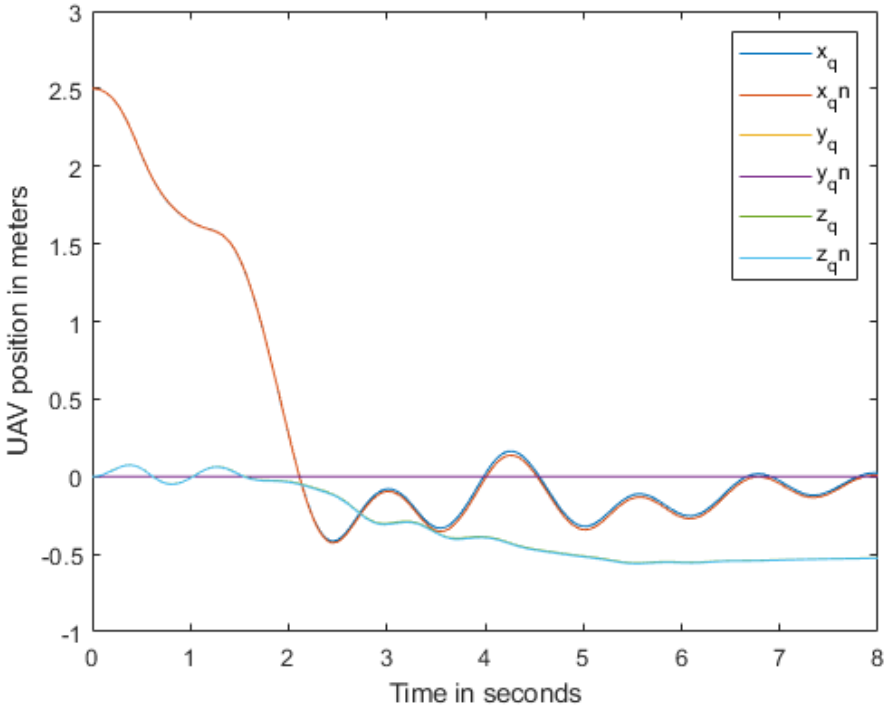


Figure 52: Scenario 4 with 1% noise

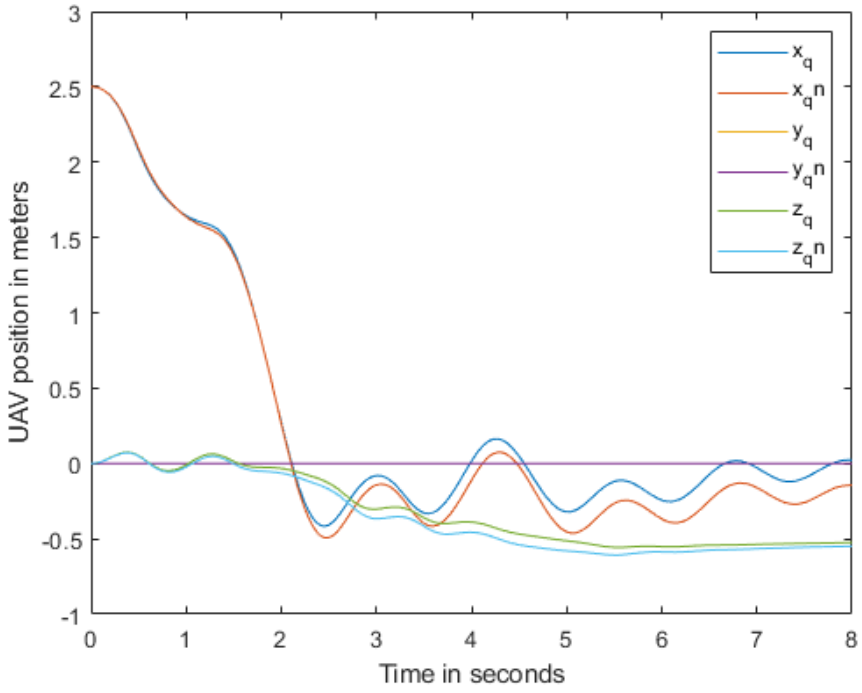


Figure 53: Scenario 4 with 5% noise

7. CONCLUSIONS & FUTURE WORK

While the control segment of aerial vehicles is traveling more towards autonomy and intelligent methods, this thesis presented an unconventional method of adding modular behaviors through genetic fuzzy systems. The application of fault recovery set a very interesting objective for the type of behavior to be learned – a maneuver to stabilize a payload. Using a general fuzzy scheme that was already proven for controlling position and achieving stability in UAVs, the rule base was then treated as a pseudo-neural network that can tune and store information for the UAV to perform payload stabilization even without a payload angle input. This was possible because of the physics of the swinging payload, which caused detectable effects that can be sensed through the UAV position and velocity error inputs defined in the fuzzy controllers. With simulation training using genetic algorithms, the newly tuned rule base for this application repurposed a position control fuzzy system for payload stabilization. To assess the possibility of such learning within constraints of rule base optimization only, the system was first trained in 2D and scaled up to a full-state

space 3D system. The test results on payload stabilization showed quick arrest of the cable swinging velocity and stabilization of the system to hover but with a small stray away from the commanded position as a trade-off. While the performance of the payload stabilization does not represent the best degree of system oscillation being dissipated (e.g. like a PID controller), this method does not guarantee optimal control of the system, but shows the possibility of optimization and tailoring rule bases of standard control systems for modular learnable actions. This demonstration showed the ability of the system to recover in case of failure down to one UAV and payload scenario, under the assumption that the UAV has sufficient thrust to support itself and the payload. The robustness of the payload stabilization controllers in the event of increasing payload mass and cable length was also observed.

For failures resulting in down to one UAV in indoor payload transports, the vicinity may not be free of hazards such as walls or other drones. In such cases, the UAV or payload should ideally not swing into nearby obstacles and given that the trained payload stabilization maneuver requires a free zone on its own, waypoint tracking behaviors were trained. Using the same scheme as the payload stabilization controllers, the penalty function was modified for the new learning objective: tracking waypoints. Since the cable or payload information was excluded from the training to avoid placing too many constraints on the learning process, the payload did oscillate upon convergence at the goal point because of the velocity ramp up and ramp down on the UAV. But since this oscillation can be mitigated by the previously trained payload stabilization controllers, and due to the fact that the UAV is not given access to payload state information, the performance, while it's not perfect, is reasonable.

These controllers still need a waypoint being passed to them for obstacle avoidance. Hence, a fuzzy controller that outputs a waypoint depending upon the perceived position of the closest obstacle, dubbed OAFIS, was developed. This manually tuned controller was simple in idea and design, and provided a waypoint based on the current UAV position only if a hazard was within a fuzzified ‘near’ definition. The simplicity of OAFIS made it suitable for static and dynamic singular obstacles. The OAFIS can be further

refined; for this graduate level research work, we limited the scope to handling walls and slow-moving UAVs.

To demonstrate that these modularly trained behaviors can be swapped effectively, several test scenarios were chosen and simulated using a switching policy based on the observations made during validation tests. While these switching policies are not promised to be optimal, it was sufficient to demonstrate the application of modular control systems and action-behaviors. If more behaviors are to be trained and utilized, further research into a 'smart'er switching policy is necessary. For example, if the distance of stray during the stabilization maneuver is predictable based on the cable swing angle, then the waypoint being used to converge at the commanded waypoint can be adjusted in a way that the following stabilization could bring the UAV to the desired waypoint. This study takes the first steps towards genetic fuzzy systems used in a modular repurposed fashion that could be scheduled and utilized as necessary by a higher policy system.

Given that the control schemes proposed in this thesis were implemented in simulations, the next step would include implementing and validating these schemes in controlled experimentations. Doing so would give rise to new challenges such as sensor noises, wind gust from an air draft, size and shape of the payload affecting dynamics which were not modelled due to the assumptions and scope of this study. Additionally, the control scheme can be tested for in a two-UAV team, which could work because of how the payload information is not included but would require additional constraints to prevent the UAVs from running into each other, not identifying each other as an obstacle and working cooperatively. For the actual control scheme itself, a more complex fuzzy control system that can stabilize while continuing to track a waypoint would be ideal and might not require the UAV to repeat a stop-go motion during multiple waypoints. One of the issues that we ran into during experimentations was having the fuzzy inputs go beyond the input range because of untrained regions in the rule base or unsuitable switching policies. As stated, the switching policies were tailored for the scenarios chosen and are not representative of a final switching policy suitable for other applications beyond the test scenario. To get the best use of the developed controllers, further study into 'smart'er and/or more advanced switching policies using those is required.

Although the current work demonstrates training methodology and set up for UAV-payload recovery after detachment from a team, the study does only test failure down to one UAV and does not provide the developed systems as a scalable back up strategy for any number of UAV teams. So further improvement to the methodology would involve increasing the scope of failure ranges. This could either mean increasing the application range of the repurposed stabilization controllers by testing it on multi-UAV systems, or exploring the limits of repurposing other control systems developed for scalable multi-UAV systems. The role that this thesis plays is to act as proof of work that genetic fuzzy systems can be used as pseudo-neural networks that can learn simple behavior actions through simulations, and that such trained behavior can be accessed in a modular way to create an intelligent control system when combined with a switching control policy. Such a system shows possibility of growth by the addition of new behavior actions and modification of the policy(/ies) that are used to access these behaviors. Lastly, reducing assumptions made for this study to make the system more realistic would be a significant step in future work.

ACKNOWLEDGEMENTS

The author would like to acknowledge many constructive discussions and guidance from Dr. Catharine McGhan of the Department of Aerospace Engineering, University of Cincinnati.

This work is available online on GitHub under the BDS-3 Clause License (link). The simulation videos are available in the YouTube Playlist (link).

REFERENCES

1. Subodh Bhandari, Steven Viska, Harsh Shah, Callie Chen, Guiseppe Tonini and Scott Kline. "Autonomous Navigation of a Quadrotor in Indoor Environments for Surveillance and Reconnaissance," AIAA 2015-0717. AIAA Infotech @ Aerospace. January 2015.
2. Yang, Hong & Abousleiman, Rami & Sababha, Belal & Gjioni, Ermal & Korff, Daniel & Rawashdeh, Osamah. (2009). Implementation of an Autonomous Surveillance Quadrotor System. 10.2514/6.2009-2047.

3. Rahman, Mohammad Fatin Fatihur, Shurui Fan, Yan Zhang, and Lei Chen. 2021. "A Comparative Study on Application of Unmanned Aerial Vehicle Systems in Agriculture" *Agriculture* 11, no. 1: 22. https://doi.org/10.3390/agriculture11010022
4. Ke Xie, Hao Yang, Shengqiu Huang, Dani Lischinski, Marc Christie, Kai Xu, Minglun Gong, Daniel Cohen-Or, and Hui Huang. 2018. Creating and chaining camera moves for quadrotor videography. ACM Trans. Graph. 37, 4, Article 88 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201284
5. Mujahid Abdulrahim, Justin Dee, Gene Thomas and Garry Qualls. "Handling Qualities and Performance Metrics for First-Person-View Racing Quadrotors," AIAA 2018-0293. 2018 AIAA Atmospheric Flight Mechanics Conference. January 2018.
6. Medhi, Jishu. "Modular Architecture for Intelligent Aerial Manipulators." Master's thesis, University of Cincinnati, 2019. http://rave.ohiolink.edu/etdc/view?acc_num=ucin1573811910421278
7. https://www.dhl.com/global-en/home/insights-and-innovation/thought-leadership/trend-reports/drones-logistics.html
8. Jung, Sunghun & Kim, Hyunsu. (2017). Analysis of Amazon Prime Air UAV Delivery Service. Journal of Knowledge Information Technology and Systems. 12. 253-266. 10.34163/jkits.2017.12.2.005.
9. https://www.fox19.com/2021/10/01/krogers-drone-delivery-program-showing-success-early-stages-company-says/
10. Moura, André & Antunes, José & Dias, A. & Martins, Alfredo & Almeida, J.M.. (2021). Graph-SLAM Approach for Indoor UAV Localization in Warehouse Logistics Applications. 4-11. 10.1109/ICARSC52212.2021.9429791.
11. Multiple UAV Systems: A Survey, Georgy Skorobogatov (1Computer Architecture Department, Universitat Politècnica de Calatunya, C. Jordi Girona, 1-3, Barcelona, Spain), Cristina Barrado (1Computer Architecture Department, Universitat Politècnica de Calatunya, C. Jordi Girona, 1-3,

Barcelona, Spain), and Esther Salamí (1Computer Architecture Department, Universitat Politècnica de Calatunya, C. Jordi Girona, 1-3, Barcelona, Spain); Unmanned Systems 2020 08:02, 149-169

12. L. Qian, S. Graham and H. H. . -T. Liu, "Guidance and Control Law Design for a Slung Payload in Autonomous Landing: A Drone Delivery Case Study," in *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1773-1782, Aug. 2020, doi: 10.1109/TMECH.2020.2998718.
13. B. Xian, S. Wang and S. Yang, "An Online Trajectory Planning Approach for a Quadrotor UAV With a Slung Payload," in *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 6669-6678, Aug. 2020, doi: 10.1109/TIE.2019.2938493.
14. Sathyan, A., & Ma, O. (2019). Collaborative Control of Multiple Robots Using Genetic Fuzzy Systems. Robotica, 37(11), 1922-1936. doi:10.1017/S0263574719000353
15. Ernest, Nicholas. "Genetic Fuzzy Trees for Intelligent Control of Unmanned Combat Aerial Vehicles." Doctoral dissertation, University of Cincinnati, 2015. http://rave.ohiolink.edu/etdc/view?acc_num=ucin1427813213
16. Arnett, Timothy. "Iteratively Increasing Complexity During Optimization for Formally Verifiable Fuzzy Systems." Doctoral dissertation, University of Cincinnati, 2019. http://rave.ohiolink.edu/etdc/view?acc_num=ucin156387481300899
17. Korte, Christopher. "A Preliminary Investigation into using Artificial Neural Networks to Generate Surgical Trajectories to Enable Semi-Autonomous Surgery in Space." Doctoral dissertation, University of Cincinnati, 2020. http://rave.ohiolink.edu/etdc/view?acc_num=ucin1595499765813353
18. Tolba, M., Shirinzadeh, B., El-Bayoumi, G. et al. Adaptive optimal controller design for an unbalanced UAV with slung load. Auton Robot 47, 267–280 (2023). https://doi.org/10.1007/s10514-023-10090-z

19. J. Wehbeh, S. Rahman and I. Sharf, "Distributed Model Predictive Control for UAVs Collaborative Payload Transport," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 11666-11672, doi: 10.1109/IROS45743.2020.9341541.
20. Graham, Silas. “Development of a Quadrotor Slung Payload System by Silas Kevin Isaac Graham A thesis submitted in conformity with the requirements.” (2019).
21. F. A. Goodarzi, D. Lee and T. Lee, "Geometric stabilization of a quadrotor UAV with a payload connected by flexible cable," 2014 American Control Conference, Portland, OR, USA, 2014, pp. 4925-4930, doi: 10.1109/ACC.2014.6859419.
22. Byung-Yoon Lee et al., "Study on payload stabilization method with the slung-load transportation system using a quad-rotor," 2015 European Control Conference (ECC), Linz, 2015, pp. 2097-2102, doi: 10.1109/ECC.2015.7330849.
23. Ordoukhanian, Edwin & Madni, Azad. (2016). Resilient Multi-UAV Operation: Key Concepts and Challenges. 10.2514/6.2016-0475.
24. Ordoukhanian, Edwin & Madni, Azad. (2018). Human-Systems Integration Challenges in Resilient Multi-UAV Operation. 131-138. 10.1007/978-3-319-60384-1_13.
25. Valenti, Mario & Bethke, Brett & Fiore, Gaston & How, Jonathan & Feron, Eric. (2006). Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery. AIAA Guidance, Navigation, and Control Conference and Exhibit. 10.2514/6.2006-6200.
26. Bisig, Caleb. "Modular Decentralized Genetic Fuzzy Control for Multi-UAV Slung Payloads." Master's thesis, University of Cincinnati, 2021. http://rave.ohiolink.edu/etdc/view?acc_num=ucin1617106491512366
27. Zadeh, L. A., "Fuzzy Sets," Information and Control, Vol. 8, pp. 338-353, 1965
28. Mamdani, E. H., "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," International Journal of Man-Machine Studies, Vol. 7, Issue 1, pp. 1-13, 1975
29. Ernest, Nicholas & Carroll, David & Schumacher, Corey & Clark, Matthew & Cohen, Kelly & Lee, Gene. (2016). Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial

Vehicle Control in Simulated Air Combat Missions. Journal of Defense Management. 06. 10.4172/2167-0374.1000144.

30. Chin, Cheng Siong & Lin, Wei. (2018). Robust Genetic Algorithm and Fuzzy Inference Mechanism Embedded in Sliding-Mode Controller for Uncertain Underwater Robot. IEEE/ASME Transactions on Mechatronics. PP. 1-1. 10.1109/TMECH.2018.2806389.
31. Torres-Salinas, Hugo, Juvenal Rodríguez-Reséndiz, Edson E. Cruz-Miguel, and L. A. Ángeles-Hurtado. 2022. "Fuzzy Logic and Genetic-Based Algorithm for a Servo Control System" *Micromachines* 13, no. 4: 586. https://doi.org/10.3390/mi13040586
32. FNU, Vijaykumar Sureshkumar. "Autonomous Control of A Quadrotor UAV Using Fuzzy Logic." Master's thesis, University of Cincinnati, 2015. http://rave.ohiolink.edu/etdc/view?acc_num=ucin1428049378
33. Mirjalili, Seyedali. “Introduction to Genetic Algorithms: Theory and Applications”, Udemy (2020). https://www.udemy.com/course/geneticalgorithm/