UNSYS digital

International Journal of Robotics and Mechatronics

Autonomous Formation Flight Algorithm and Platform for Quadrotor UAVs

Ahmad Drak, Mohammad Hejase, Mohammad ElShorbagy, Addy Wahyudie, Hassan Noura

UAE University, United Arab Emirates

***Abstract*—**There are many advantages for having a formation flight of multiple Unmanned Aerial Vehicles (UAVs) over a single UAV. In this paper, a simple proposed formation flight algorithm is proposed. The algorithm consists of the path planning and the Hierarchical Leader-Follower algorithm. In the Hierarchal Leader-Follower formation, the leader has more autonomy, while the follower is completely dependent on the behavior of the leader. The path planning and Hierarchical Leader-Follower algorithm are both implemented and tested on a formation of two quadrotors flying over a single axis. A Graphical User Interface (GUI) was developed to implement this algorithm.

***Keywords*—**Formation flight, quadrotor, UAV, graphical user interface (GUI), path planning.

I. Introduction

RECENTLY, there has been a vast increase in the research on formation flight of UAVs, mainly due to the new applications and benefits that result from having multiple UAVs fly and coordinate simultaneously. Among the most notable advantages are the faster and broader acquisition of data, ability to complete a mission even if a UAV were to fail, division of tasks among UAVs, and attaching a variety of payloads to the system of UAVs [1 - 2].

In this study, a formation flight of quadrotors is considered. A quadrotor is a Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicle (UAV) that is lifted by four propellers each driven by a separate rotor. Quadrotors have become a very popular type of UAV and have been extensively utilized in various applications. Such applications include surveillance, imaging, hazardous environments, indoor navigation, and mapping. The popularity of UAVs is mainly due to the certain features they possess, such as low dimension, good maneuverability, simple dynamics and payload capability [3].

The cross configuration of a quadrotor is depicted in **Figure 1**. In order to eliminate the need of a tail, two pairs of opposite propellers rotate in opposite directions. Increasing or decreasing the speed of the four propellers simultaneously allows ascending or descending. Vertical rotation is achieved by creating an angular speed difference between the two pairs of opposite rotors. Horizontal motions are achieved by tilting the vehicle through changing the speed of a pair of rotors [4-5]

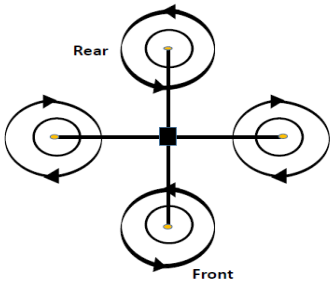

Figure 1 Quadrotor configuration

As the complexity of the formation flight architecture for UAVs increases, the need for a Graphical User Interface (GUI) emerges. The authors of Ref. [6 - 8] use a GUI for monitoring a single or multiple UAVs. The proposed GUI in this paper was implemented on MATLAB's Graphical User Interface Development Environment (GUIDE). GUIDE is a MATLAB based development environment, which aids the user in designing UI's that perform a certain task such as controlling real-time hardware or simply manipulating, editing and plotting data. It consists of several tools that simplify the process of creating complex GUI's. It provides a "point-and-click" control of software and hardware applications; this eliminates the need of resorting to methods requiring extensive and tedious programming [9].

The MATLAB GUI created through GUIDE is a self-contained MATLAB program, in other words, it constitutes a complete and independent unit. This feature allows the user to either run the GUI from MATLAB or develop it into a separate independent standalone executable. GUIDE provides certain tools for developing a UI, and has the ability of converting the UI into MATLAB code, which can be easily modified by the user to control the behavior of the UI based on a desired application [9 - 10].

In this study, a formation flight algorithm is considered, implemented and tested on two quadrotors. The need of a formation flight rises from the various limitations that hinder the performance and flexibility of individual UAVs when undergoing certain tasks. These limitations could be a result of hardware/software faults or even a limitation on the range of operations a single UAV could undergo. The proposed

Corresponding author: Addy Wahyudie (e-mail: addy.w@uaeu.ac.ae).
This paper was submitted on February 28, 2014; revised on November 20, 2014; and accepted on November 25, 2014.

formation flight algorithm is based on a hierarchal leader-follower scheme similar to the ones described in Ref. [11] and Ref. [12], where the leader tracks a series of waypoints representing a path, and the follower uses the most updated leader position for tracking. The authors of Ref. [11] implemented a hierarchal class leader-follower formation flight controller for the control of two quadrotor UAVs over 2 axes; the leader was fed a path to follow while the follower used the leader's position as its tracking reference. While the authors of Ref. [12] used a hierarchal class leader-follower controller for the control of two quadrotor UAVs, a path was defined for the leader to track, while the follower used the leader's position and angle. Moreover, the algorithm utilizes the use of a communication system, a localization system and a path planning scheme. A GUI which allows for parameter initializations, mission planning, flight control, quadrotor deployment, real-time monitoring, and flight data analysis was designed and implemented. The proposed GUI in this paper is implemented to give the operator more freedom when controlling the flight of the quadrotors in real-time, and to simplify the process of deployment. Additionally, it limits the usage of typed commands and text-based interfaces, making it easier to manage and control the overall system. A similar GUI is presented in Ref. [13]. The GUI almost has the same functionality as the one proposed in this paper, however, it was not developed on MATLAB GUIDE. The proposed algorithm, along with the GUI, could be developed on another platform, such as Robot Operating System (ROS), as presented in Ref. [14]. Due to hardware requirements and the higher complexity associated with prototyping and development in ROS environments, the MATLAB platform was favored. Developing the formation flight algorithm along with the path planning and the accompanying GUI on the MATLAB platform, constitute the originality of this paper.

The outline of this paper is described as follows: In section II, a proposed method for a formation flight scheme will be discussed, along with the GUI. Section III will go over the experiment setup used to test the formation flight. The real flight test results and discussion will be covered in section IV, and finally, the conclusion will be presented in section V.

II. PROPOSED METHOD

A. Formation Flight Algorithm

When selecting a formation flight scheme, it is necessary to select the class of the formation flight controller to be used. The most common types of formation flight classes are centralized, decentralized, and hierarchal. Centralized controllers utilize a central node to perform all calculations and generate control commands for quadrotors [15], they generally require a large amount of computations [16]. In decentralized controller classes, all quadrotors in the formation are equipped with their own onboard controllers and have the same level of autonomy [17], this leads to a loosely coupled formation [15]. Hierarchal class schemes involve assigning different levels of autonomy to the quadrotors in the formation [18], a hierarchal class formation flight scheme was favorable due to it being computationally lighter than a centralized scheme, and more optimal than a decentralized scheme [19].

The most common types of formation flight are behavioral formations, virtual center formations, and leader-follower formations [20 - 21]. Leader-follower formation types are the ones most closely associated with hierarchal class schemes. In a leader-follower formation, it is enough to specify a path for the leader to fly on in order to dictate the path taken by the formation [22].

In this section, the architecture of a system supporting a hierarchal leader-follower Formation scheme for two quadrotors will be discussed. One quadrotor acts as a leader, while the other acts as a follower. A ground control station, also referred to as the 'Mission Server' was designed.

It is essential for the success of a formation flight to have the following components: a communication system, a localization system, a formation flight scheme. These components were designed and implemented on a MATLAB environment, and then linked together in a user friendly interface by designing and implementing a GUI.

Communication System

A wireless communication system was used (WiFi 802.11n). Each of the mission server, leading quadrotor, and following quadrotor were issued a unique IP address. A QUARC Simulink block supporting TCP/IP protocol was used to facilitate communication between each of the mission server, leading controller, and following controller.

Localization System

As for the localization system, Optitrack positioning cameras were used, this allows for a workspace of 2m x 2.5m x 1.5m (these values were found by manual testing). Even though the workspace is not large, it is enough to test and verify the functionality of the designed system. To ensure proper tracking, the leader and follower were fitted with reflective markers arranged in unique asymmetrical geometric shapes to allow detection by the optitrack cameras. The limitation of such a localization system is its inability to function in a real word scenario. However, the Optitrack provides high accuracy positions, which is ideal for an indoor flight test of the proposed algorithm and the GUI. The authors of Ref. [23] used a stereo camera embedded on the quadrotor for indoor positioning. The merits of such a localization scheme, as detailed by the authors, is that it's cheap to implement and is applicable in real life outdoor scenarios. However, it requires more processing power from the controller and does not provide accurate positions when compared to the Optitrack.

Due to the limited workspace, the implementation of the formation was tested over the x-axis only, with the y-axis and the z-axis being kept constant for each quadrotor.

Path Planning

An offline path planner that aims at constructing an array of waypoints from a starting to an ending point, both specified by

the operator was implemented. Although the formation is carried out over a single axis, the designed path planner algorithm also works for a formation flight over two axes. The user is required to specify the waypoint separation (resolution), and select the initial and final waypoints before the path planner algorithm is executed. Due to the absence of obstacles in the workspace environment, it is sufficient to have a straight path connecting the starting and ending points. Using a basic line equation between two points and a distance equation between the waypoints, the path planner calculates all intermediate waypoints between the selected starting and ending points.

Figure 2 below shows the initial and final waypoints along with intermediate waypoints. The mathematics behind the operation of the path planner can also be seen below.

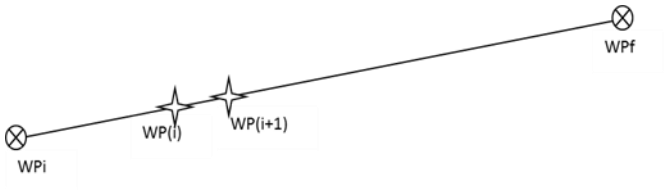

Figure 2 Path between Initial and Final Points

Let WPs be the initial point chosen by the user and WPf be the final one. WP(i) is a generated waypoint, while WP(i+1) is the next generated waypoint.

Each waypoint can be thought to consist of an X coordinate and a Y coordinate.

$$\mathrm{WPs} = [\mathrm{Xs}, \mathrm{Ys}] \quad (1)$$

$$\mathrm{WPf} = [\mathrm{Xf}, \mathrm{Yf}] \quad (2)$$

$$\mathrm{WP(i)} = [\mathrm{X(i)}, \mathrm{Y(i)}] \quad (3)$$

$$\mathrm{WP(i+1)} = [\mathrm{X(i+1)}, \mathrm{Y(i+1)}] \quad (4)$$

The slope of the trajectory in the figure is equal to rise/run.

$$\mathrm{Slope} = \frac{\mathrm{Rise}}{\mathrm{Run}} = \frac{\mathrm{Yf-Ys}}{\mathrm{Xf-Xs}} = \frac{\mathrm{Y(i+1)-Y(i)}}{\mathrm{X(i+1)-X(i)}} \quad (5)$$

$$\mathrm{Distance\ between\ waypoints} = \mathrm{D} = \sqrt{\left(\mathrm{X(i+1)} - \mathrm{X(i)}\right)^2 + \left(\mathrm{Y(i+1)} - \mathrm{Y(i)}\right)^2} \quad (6)$$

Using the slope and distance equations above, the X and Y intermediate waypoint equations can be derived to the following recursive equations:

$$X(i+1) = X(i) + \left(\frac{Xf-Xs}{|Xf-Xs|}\right)\sqrt{\frac{D^2}{slope^2+1}} \quad (7)$$

$$Y(i+1) = Y(i) + \left(\frac{Yf-Ys}{|Yf-Ys|}\right)(Slope(X(i+1) - X(i)) \quad (8)$$

A user friendly interface is designed on MATLAB in order to be able to select the initial and final points for waypoint generation. Equations (7) and (8) are used to calculate and plot the generated waypoints. **Figure 3** below shows the executed path planning file. The background picture in the GUI of **Figure 3** represents the actual workspace available. Any point chosen in the GUI represents the actual position of the quadrotor in the real setup. This is ensured by entering the workspace limits, as seen in **TABLE I**.

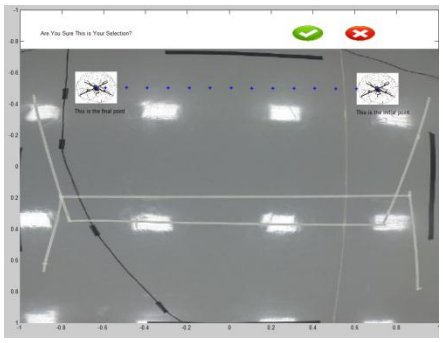

Figure 3 Results of Path Planning Execution

Formation Flight Scheme

The authors of Ref. [24] propose a system performing complex maneuvers; such a system requires velocity and acceleration matching to achieve an optimal performance. The quadrotors used were of the same brand and type, and the same single flight LQR control system was used for the both of them. This would yield very similar performances in terms of velocity and acceleration for both the leader and the follower. Additionally, the distance of travel is very short, which means that the resulting difference in velocities is tolerable. Since the system is indoors, the experiment is conducted over a controlled environment, so external disturbances are kept minimal. For the reasons above, it was deemed unnecessary to match the velocities of the quadrotors. Hence, a velocity matching control system is not needed for that, it is sufficient for the follower to calculate its next position based on the current position of the leader being run through a basic algorithm. **Figure 4** depicts the algorithm used by the follower to acquire its next waypoint.

Before designing the system architecture, it is necessary to define certain parameters that need to be initialized to ensure the safety and success of the mission. The initialized parameters are stored in the MATLAB workspace. This allows Simulink to access the values of these parameters. **TABLE I** show the parameters that were initialized beforehand.

Figure 4 Algorithm for Follower Waypoints

System Logical Flowcharts

Before programming the system on Simulink, logical flowcharts explaining the functions of each of the mission server (**Figure 5**), leader controller (**Figure 6**), and follower controller (**Figure 7**) were designed and accordingly constructed.

TABLE I
PARAMETERS TO BE INITIALIZED

Parameters	Description	Values
Controller Parameters	Stores control parameters in MATLAB workspace for usage in control blocks of leader controller and follower controller	Parameters calculated for LQR controller. Fed to MATLAB Workspace via "setup_qball_x4.m"
URIs	Sets the URIs of the mission server, leader controller, and follower controller	Mission Server: 182.168.1.11 Leader Controller: 182.168.1.95 Follower Controller: 182.168.1.97 Fed to MATLAB Workspace via "setup_vehicle_uri.m"
Workspace Limits	The range of visibility of the optitrack system.	X: -1 to 1 Z: -0.75 to 1 Y: 0 to 1.3 Fed to MATLAB Workspace via "setup_workspace.m"
Path Planning	The array of waypoints to be fed for the leader to follow	Depends on the selection done by user. Fed to MATLAB Workspace via "setup_workspace.m"
Leader & Follower Heights	The constant heights for the leader and follower to maintain during the mission	Leader: 1m Follower: 1m
Leader & Follower Z positions	The constant Z positions for the leader and follower to maintain during the mission	Leader: -0.45m Follower: 0.85m
Following Distance	The distance the follower should follow the leader with.	Since formation is done over 1 axis, it was set to 0m. (follower is required to match the leader X position)

System Architecture and Data Flow

After designing the logical flowcharts and setting up the architecture of the system, MATLAB Simulink was used in order to realize and eventually test the system. The designed architecture can be seen below in **Figure 8**. The figure shows the flow of data between the different controllers and responsibilities of each. **TABLE II** below provides a brief explanation over the function of each of the blocks seen in the figure. The resulting system was made up of three main Simulink models, and multiple MATLAB files. In a real time test, managing such a large number of files all at once is not practical.

TABLE II
BLOCK FUNCTION EXLANATIONS FOR SIMULINK PAGES

	Block Name	Description
Mission Server	Data to leader	Send Leader Waypoints Send Leader Position Data Send Whether Leader is Armed/Unarmed Send Workspace Limits Send Mission Status
	Data to Follower	Send Joystick Armed or Unarmed (optional feature) Send Follower Position Data Send Whether Follower is Armed/Unarmed Send Formation Mode (Linear or Planar) Send Workspace Limits Send Direction of Leader
	Optitrak Measurements	Receives Leader & Follower Position Data from Optitrack System
	Logical Subsystem	Setting of Minimum Height Needed for Qball's before Issuing Waypoints Checking if Mission Started Issues Waypoints when Qball's at Required Height & Mission Started
	Trajectory Generation	Receive waypoints from MATLAB Workspace Issues Next Waypoint to Leader Specifies Leader Z position for Linear Formation Calculation of Leader Direction for Linear Formation Generate Message Upon Completion of Mission
	Workspace Limits	This block is used so that the operator can visually check and make sure the workspace limits are correct. A mistake in setting the limits could lead to crashes and costly mistakes.
	Data Displays	For testing purposes, allows the observation of data values while system is running.
	Operator Duties	Toggle the start/stop switch. Select the formation algorithm to be put in use. Arm/Unarm the leader Qball. Arm/Unarm the follower Qball.
Leader Controller	Position Commands	Setting Qball Height Landing of Qball Upon Mission Completion Using Leader Position and Waypoint Data to Store them in their Respective Variables Once Qball is at Appropriate Height Ensures Waypoint Data is Within Workspace Boundaries
	Data From Host	Receives Next Waypoint Receives Position Data Receives if Leader is Armed/Unarmed Workspace Limits Mission Status
	Data to Follower	Sends Leader Position Data to Follower Sends Mission Status to Follower
	Control Blocks	Control the flight of the quadrotor
	Black Box	Log Flight Data for Analysis
Follower Controller	Data From Leader	Receives Leader Position Data to Follower Receives Mission Status to Follower
	Landing	Lands Qball when Mission is Complete & Follower Within 10 cm of Leader Position
	Mode Selected	Follower Waypoints Issued Depending on Formation Mode Selected by User in Mission Server
	Plane Following	Contains Algorithm for Planar Following
	Line Following	Contains Algorithm for Linear Following Setting Constant Z position of Follower
	Control Blocks	Control the flight of the quadrotor
	Black Box	Log Flight Data for Analysis
	Operator Duties	Select Line Following Distance Select Plane Following Distance

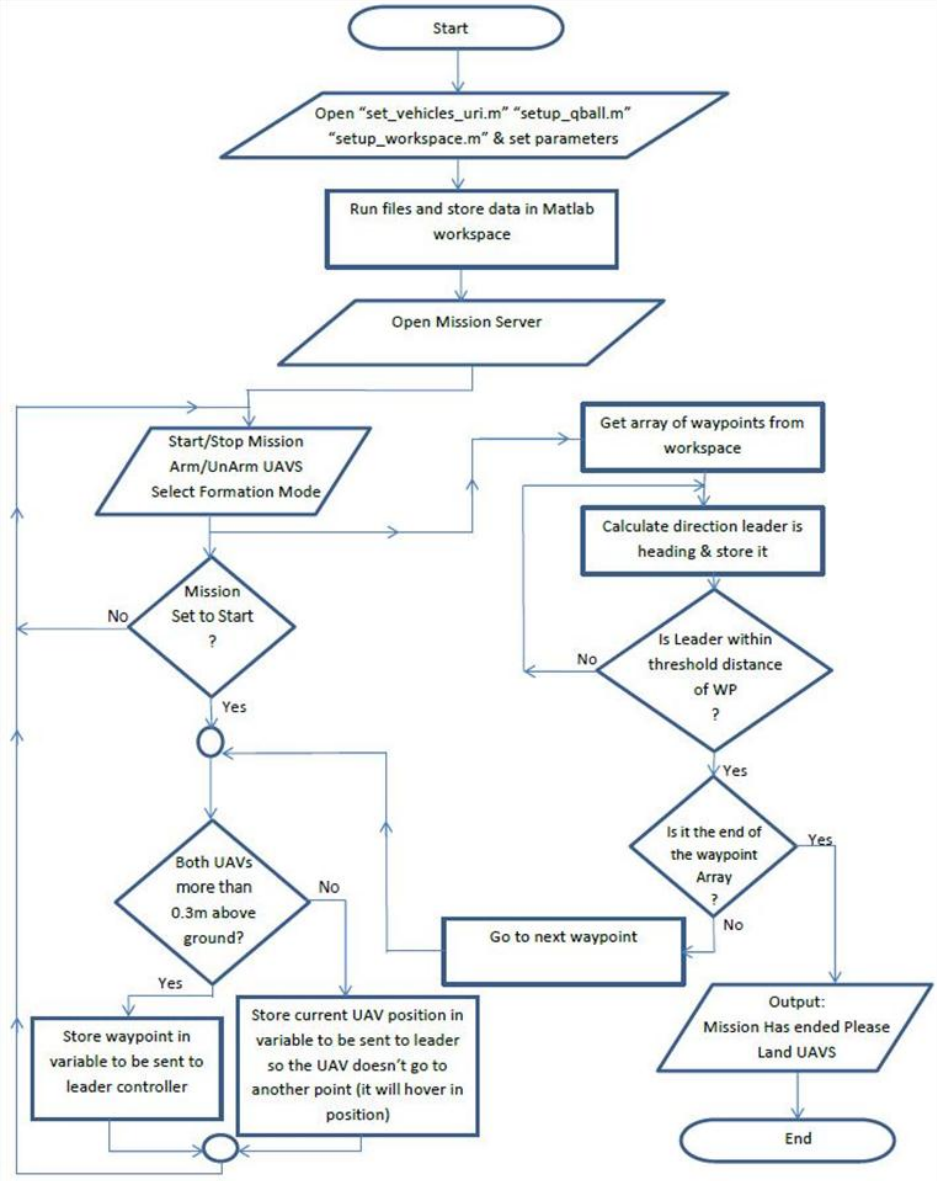

Figure 5 Logical flowchart of mission server

B. Graphical User Interface (GUI)

Design of the GUI

It is imperative to identify what is to be controlled from within the GUI before starting the design process. The three main Simulink model files needed to achieve formation flight will be controlled from within the GUI, in addition to running the required m-files. The mission server, leader control and follower control Simulink model files, all have to be built, started and stopped at the convenience of the user or when needed, hence it is essential that these parameters be controlled form the GUI.

The m-files are to be executed first, as they initialize certain required parameters, such as controller parameters and vehicle Universal Resource Identifier (URI). Moreover, for the convenience of the user, the trajectory that the quadrotors have to follow will be generated from within the GUI. This is done by running an m-file, which in return launches another GUI for choosing the initial and final waypoints for the formation in a graphical and professional way, rather than entering the points manually in the corresponding Simulink file.

Several real-time plots were also added to the GUI as to allow the user to graphically and precisely monitor the trajectory of the leader and follower quadrotors. After flight analysis data, such as pitch, roll and PWM can also be plotted directly from the GUI after the mission has ended. This allows the user to investigate the inflight data recorded and analyze it for further improvements if needed. The end product is a GUI which allows the management and control of a formation flight mission from start to finish, this makes testing and performance assessment easier and more practical for the operator.

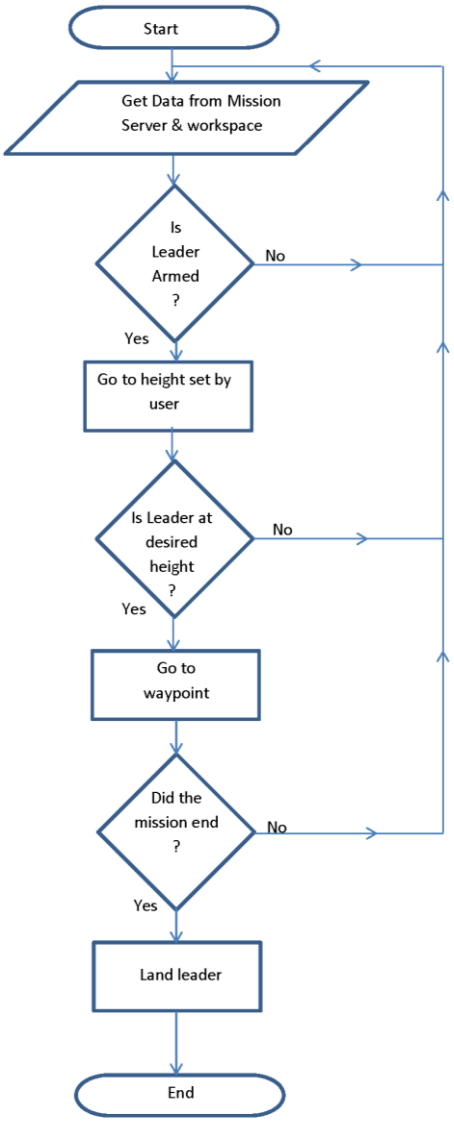

Figure 6 Logical flowchart for leader controller

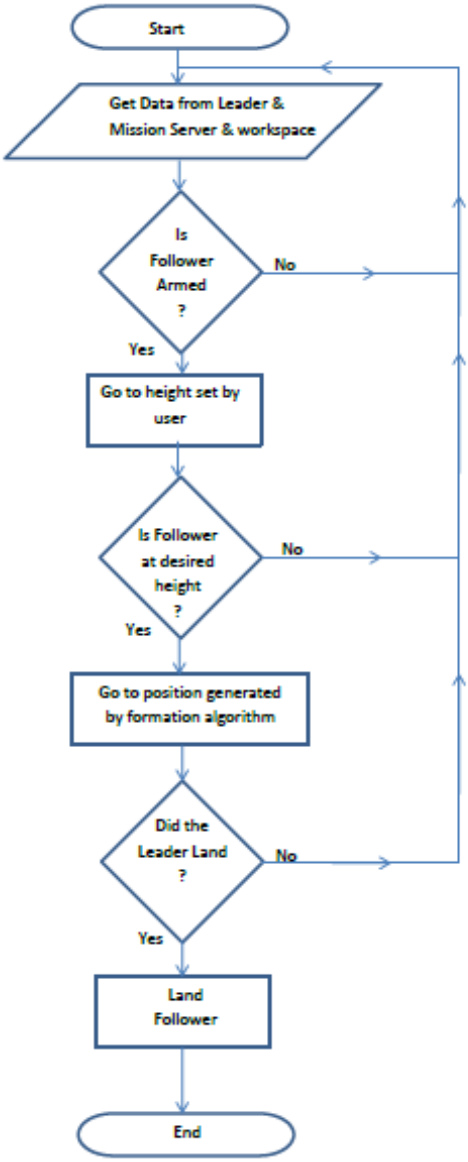

Figure 7 Logical flowchart for follower controller

Implementation of the GUI

The required GUI objects were chosen from the objects palette and then dragged and dropped into the Layout Editor area. The properties of the objects were edited accordingly as to meet our design needs. Upon saving the GUI figure, GUIDE will automatically generate an m-file containing the code of the GUI objects. The code was edited to fit the needs of the design Built in MATLAB and QUARC functions were added to the generated m-file as to manipulate the actions of each GUI object.

QUARC is a rapid prototyping platform for real-time control. It integrates with Simulink to allow the models to be run in real-time on different targets. Some features of QUARC are real-time performance, real-time plotting and code generation for multiple targets [25].

Figure 9 is used for the following discussion. The push buttons in the "Initial Setup" panel ("Setup Qball", "Setup Vehicle URI" and "Trajectory Generation"), in addition to the "Plot" and "Calculate" buttons in the "After Flight Data Analysis" panel, all use the same ideology of code. Each of the buttons has a separate function but perform the same task. A similar approach is used for the build, start and stop buttons in the "Mission Server", "Leader Control" and "Follower Control". These buttons will build, start or stop the corresponding Simulink model.

The Start and Stop Mission push buttons found in the "Main Control" panel, will control the start and stop parameters found in the Mission Server model file. The methodology here is that when a value of '0' is issued to the "start_stop" local variable in the Simulink file, the mission will be stopped and when a value of '1' is issued, the mission is to start. It should be mentioned

here that 'Value' property of a push button is '0' when not pressed and '1' when pressed.

As for the "Real-time Data Plotting" panel, the three axes have no callback functions. Rather, the needed axes name is specified in the 'Figure Block' used from within the Simulink model file.

A MATLAB m-file was done where upon pressing the "Plot" button in the "After Flight Data Analysis" panel, that m-file will be called and run and in return displays five different plots of several recorded data while inflight. Similarly, upon pressing the "Calculate" button in the same panel, another MATLAB m-file will be called and run and in return, it calculates the value of the error between the desired trajectory and the actual trajectory. This error value is based on the root mean square calculation. The final GUI after running it can be seen in **Figure 9**.

III. Experiment Setup

The QUANSER platform currently available in the United Arab Emirates University (UAEU) was utilized in conducting an experiment to test the proposed formation flight algorithm through the designed GUI. The platform mainly consists of six Optitrack cameras (**Figure 10**) used for localization, a PC that acts as the ground station and development platform, and Qball-X4 quadrotors (**Figure 11**). The quadrotors workspace is enclosed within a safety net to protect the operator(s) and avoid fatal crashes of the Qball-X4, as seen in **Figure 12**.

Each of the Qball-X4 has onboard a control module called QECM (QUANSER Embedded Control Module). This control module is comprised of a data acquisition card (HiQ QUARC), a Gumstix target computer and a wireless module which allows for communication between the ground station and the Qball-X4. The Gumstix is utilized for rapidly deploying the Qball-X4's developed controllers [26].

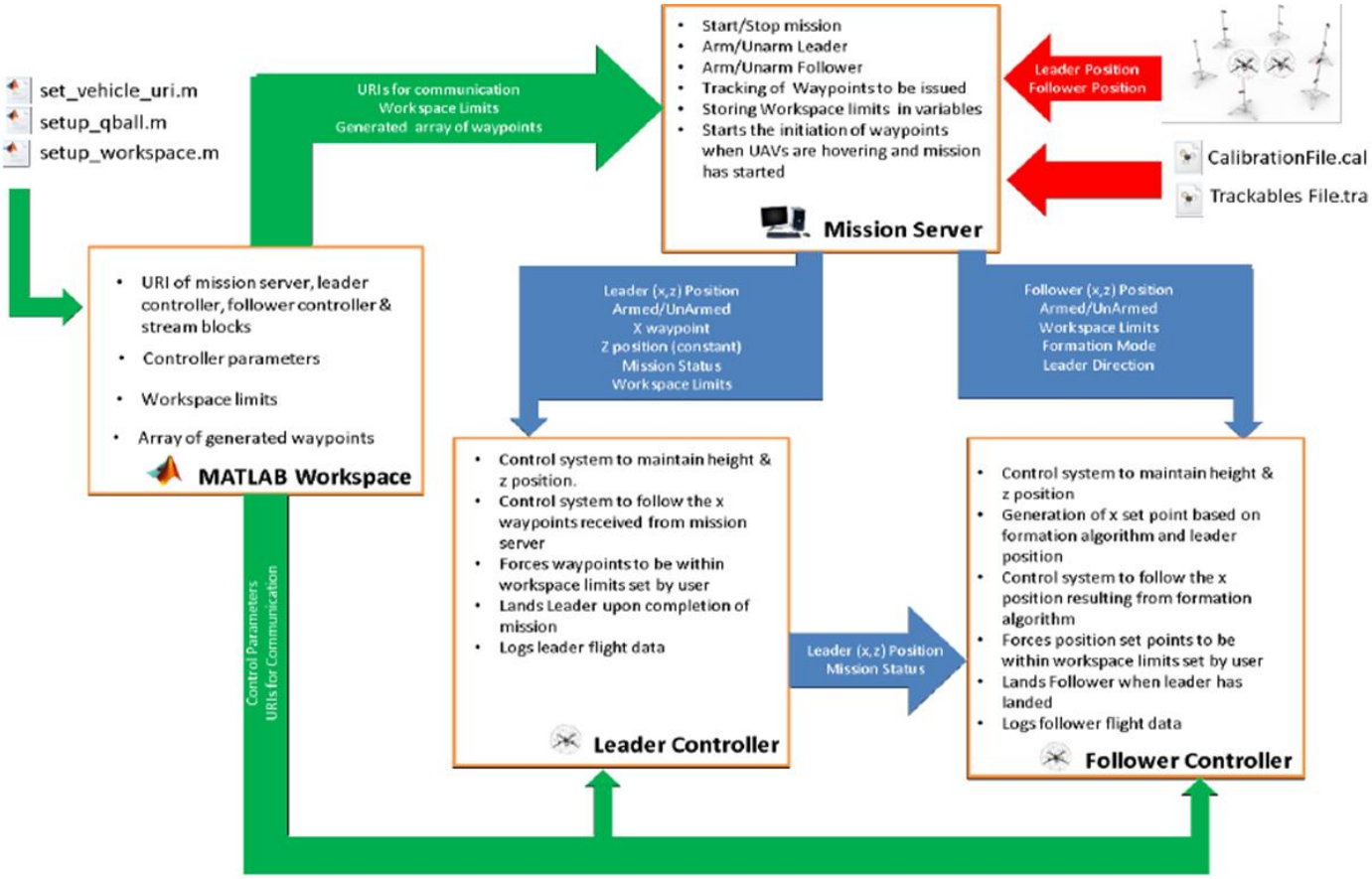

Figure 8 Formation flight architecture

The PC ground station has QUARC (Quantum Architectures & Computations toolbox) embedded with Matlab/Simulink, which is our development platform. QUARC integrates with Simulink for accelerated control prototyping and hardware-in-the-loop testing [25].

The data acquisition card has a gyroscope, accelerometer, magnetometer IMU sensor and PWM outputs to drive the brushless motors. Once the developed controller is downloaded onto the Qball-X4 and is running, it allows for real-time sensor measurements, data logging and parameter tuning [26].

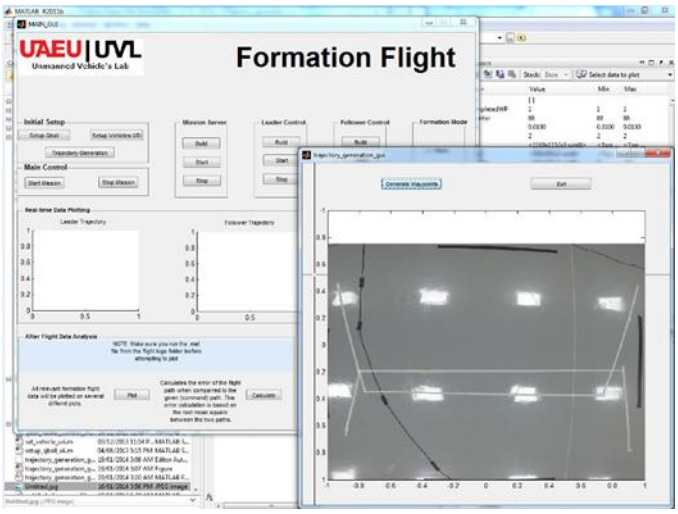

Figure 9 Main GUI and trajectory generation GUI

Figure 10 Optitrack camera

Figure 11 QUANSER's Qball-X4

IV. Results and Discussion

The objective of this test is to assess the performance of a formation flight using two quadrotors running in the designed platform. The path that the quadrotors followed was generated from within the GUI by clicking on the initial and final points, as seen in **Figure 3** in section II. Due to limited workspace and insufficient number of cameras to cover large spaces, the formation flight was implemented on one axis only (X axis) and

Figure 12 Safety net enclosing the workspace

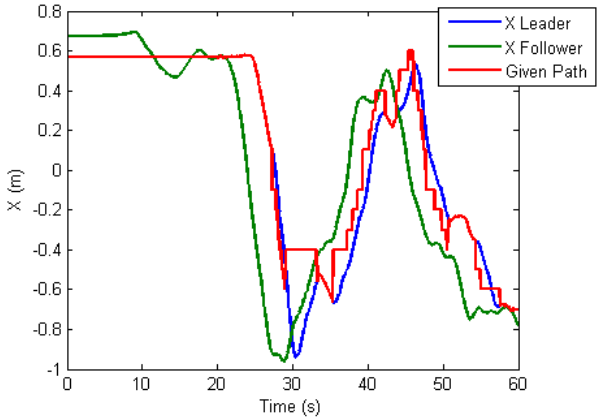

Figure 13 Leader and follower quadrotors paths on the X axis

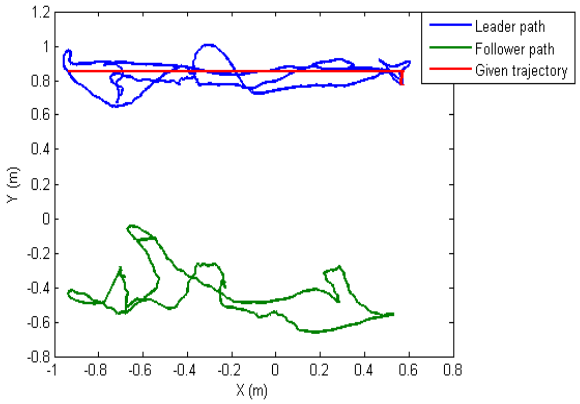

Figure 14 Leader and follower quadrotors paths on the XY plane

the Y axis was kept constant. The generated path extended from +0.56m to -0.93m on the X axis. The quadrotor started at the initial point (+0.56) and covered the generated path three times, covering a total distance of 4.47m, it landed after it has reached the final waypoint (-0.93). The Y axis was kept constant at +0.85m for the leader quadrotor, and -0.43m for the follower quadrotor. The Z axis (height) for both, leader and follower was kept constant at 1m. The total flight time was 60 seconds.

Figure 13 depicts the both the leader and the follower X axis paths along with the desired path. It is clear that both the

leader and the follower quadrotors were capable of following the generated path.

The path taken by both quadrotors in the XY plane can be seen in **Figure 14**. Despite the presence of some oscillations when tracking the given path, the quadrotors were able to maintain their formation and stay on course until they reached their final destination. From the displayed results above, it can be seen that the proposed formation flight algorithm is to an extent lacking in accuracy of tracking the given generated path. Upon running the error calculation m-file (based on root mean square) from the GUI, it was found that the leader quadrotor had a mean deviation of 0.1436 (14.36cm) from the given path, while the follower quadrotor had a mean deviation of 0.2617 (26.17cm). The equation used for the root mean square error is as follows:

$$\sqrt{\frac{\sum_{t=1}^{n}(Xd,t-Xt)^2}{n}}$$

It is possible that the lack of Optitrack localization cameras is one of the reasons behind the inaccuracy in tracking. This was seen visually through the behavior of the quadrotors during the experiment. Additionally, the use of more robust controllers for the quadrotors, along with a velocity matching formation flight controller would yield better results.

V. Conclusion

In this paper, a pre-existing hierarchal leader-follower formation flight algorithm for quadrotor UAVs was implemented on the MATLAB platform, along with a GUI for ease of deployment and monitoring. The architecture of the formation flight algorithm was discussed, implemented, and tested on one axis. A graphical offline path planner was designed and utilized in order to generate waypoints by simply clicking on a GUI representing the actual workspace available. The selected initial and final waypoints on the GUI represented the exact points on the workspace. The GUI also accounted for flight control of both quadrotors, and real-time monitoring of their paths. The flight test results validated the practicality of the designed GUI and formation flight algorithm. Moving forward, the system can still be further improved through the use of more robust control systems, and a larger workspace to allow for flight in multiple degrees of freedom.

References

[1] B. Alessandro, M. Adriano, M. Riccardo, L. Sauro and M. Mauro, "A Modular Framework for Fast Prototyping of Cooperative Unmanned Aerial Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 507-520, 2012. CrossRef

[2] D. Long, C. Haiyang, H. Jinlu and C. YangQuan, "Cognitive Multi-UAV Formation Flight: Principle, Low-Cost UAV Testbed, Controller Tuning and Experiments," *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Washington, DC, 2011.

[3] T. Bresciani, *Modelling Identification and Control of a Quadrotor helicopter*, Lund: Lund University, 2008.

[4] S. Bouabdallah, *Design and Control of Quadrotors with Application to Autonomous Flying*, EPFL, 2007.

[5] D. Schmdit, *Simulation and Control of a Quadrotor Unmanned Arial Vehicle*, University of Kentucky, 2011.

[6] W. Etter Jr, P. Martin and R. Mangharam, "Cooperative Flight Guidance of Autonomous," 2011.

[7] A. Zul Azfar and D. Hazry, "Simple GUI design for monitoring of a remotely operated quadrotor unmanned aerial vehicle (UAV)," in *Signal Processing and its Applications (CSPA)*, Penang, 2011.

[8] D.-M. Kim, D. Kim, J. Kim, N. Kim and J. Suk, "Development of near-real-time simulation environment for multiple UAVs," in *Control, Automation and Systems, 2007. ICCAS '07*, Seoul, 2007.

[9] I. The MathWorks, "Creating Graphical User Interface," 2000-2013. [Online]. [Accessed 23 December 2013]. VIEW ITEM

[10] S. T. Smith, MATLAB: Advanced GUI Development, Dog Ear Publishing, 2006.

[11] A. Mercado, "Quadrotors flight formation control using a leader-follower approach," in *European Control Conference (ECC) 2013* , 2013.

[12] M. Abas, D. Pebrianti, S. Ali, D. Iwakura "Circular Leader-Follower Formation Control of Quad-rotor Aerial Vehicles," *Journal of Robotics and Mechatronics 25*, pp. 60-71, 2013.

[13] Nonami, K., Kartidjo, M., Yoon, K. J., & Budiyono, A, Autonomous Control Systems and Vehicles: Intelligent Unmanned Systems, Springer, 2013. CrossRef

[14] Weaver, J. N., Arroyo, D. A., & Schwartz, D. E., "Collaborative Coordination and Control for an Implemented Heterogeneous Swarm of UAVs and UGVs," in *Proc. Florida Conference on Recent Advances in Robotics*, 2013.

[15] Giulietti, F., Pollini, L., & Innocenti, M., "Autonomous formation flight," in *Control Systems*, 2000.

[16] Merino, L., Caballero, F., Ferruz, J., Wiklund, J., Forssén, P. E., & Ollero, A, "Multi-UAV Cooperative Perception Techniques," *Multiple Heterogeneous Unmanned Aerial Vehicles*, pp. 67-110, 2007.

[17] Keviczky, T., Borrelli, F., & Balas, G. J, "A study on decentralized receding horizon control for decoupled systems," in *American Control Conference, 2004*, 2004.

[18] Phan, C., & Liu, H. H, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *System Simulation and Scientific Computing*, 2008.

[19] W. Zhao, "Model predictive based UAV formation flight control: formulation, extension and experiment," 2012.

[20] Zhao and Weiha, "Model Predictive Based UAV formation Flight Control: Formulation, Extension and experiment," 2012.

[21] Ren, Wei and Randal, "Decentralized Scheme for Spacecraft Formation Flying via Virtual Structure Approach," *Journal of Guidance, Control and Dynamics*, vol. 27, no. 1, pp. 73-82, 2004. CrossRef

[22] F. Fahimi, "Towards full formation control of an autonomous helicopters group," in *Aerospace Conference*, 2007.

[23] Azrad, S., Fadhil, M., Kendoul, F., & Nonami, K, " Quadrotor UAV Indoor Localization Using Embedded Stereo Camera," *Applied Mechanics and Materials*, vol. 629, pp. 270-277, 2014. CrossRef

[24] G. Cai, C. Ben and L. Tong, *Unmanned Rotorcraft Systems*, pp. 1-11, 2011. CrossRef

[25] QUANSER, "Quanser Real-Time Control SOftware (QUARC)," August 2013. [Online]. [Accessed January 2014]. VIEW ITEM

[26] QUANSER, "Quanser Qball-X4 : User Manual," [Online]. [Accessed 5 January 2014]. VIEW ITEM

[27] C. Da, J. Sun and S. Wu, "UAVs Formation Flight Control Based on Behavior and Virtual Structure," in *AsiaSim 2012*, Shanghai, Springer Berlin Heidelberg, 2012, pp. 429-438.

[28] M. Turpin, N. Michael and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, vol. 33, no. 1-2, pp. 143-156, 2012. CrossRef

[29] P. Ogren, E. Fiorelli and N. Leonard, "Formations with a mission: stable coordination of vehicle group maneuvers," in *Proc. of Intl. Sym. on Mathematical Theory Networks and Systems*, Notredame, 2002.

[30] J. Desai, J. Ostrowski and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Transactions on Robotics*, vol. 17, no. 6, pp. 905-908, 2001. CrossRef

[31] R. Beard, L. Jonathan and H. Fred, "A Coordination Architechture for Spacecraft Formation Control," *Control Systems technology*, vol. 9, no. 6, pp. 777-790, 2001.