Implementation of a Multi-UAV Flight System

Elijah Chen, Christopher Endres, Joseph Grigus, and Rachit Singhvi
University of Illinois at Urbana-Champaign

The objective of this design project was to implement a multi-UAV flight operations system. The goal was to be able to get two of the UAV's to fly at the same time and have flight data of both UAVs recorded by the MOCAP system, and sent to both independent ground stations. This was done using ground stations assigned with specific tasks. For the follower UAV, the current position of the leader UAV was fed as the desired value to implement a tracking system. Two separate on-board planners and ground stations were implemented along with tests done using state feedback controller with, and and without, integral action. The test of this system was done as a formation flight, where one UAV was following a predetermined path while the other UAV is following the lead UAV at an offset distance of 1m. During the experiment, it was found that the following UAV was able to follow the lead UAV at an offset distance on a predetermined flight path, however steady-state error was observed, particularly in the z-direction. The on-board controller of the follower UAV was tuned and altered in a attempt to minimize this error.

I. Nomenclature

UAV	=	Hummingbird Quadrotor
$o_{1,2,3}$	=	x, y, and z position of the UAV [m]
$v_{1,2,3}$	=	x, y, and z velocities of the UAV [m/s]
$\theta_{yaw,pitch,roll}$	=	yaw, pitch, and roll angles [rad]
$\omega_{1,2,3}$	=	yaw, pitch, and roll angular velocities [rad/s]
R_1^0	=	transformation matrix between body and inertial frames
N	=	transformation matrix used to convert angular rates to angular velocities
m	=	mass of the UAV (with battery) [kg]
$J_{1,2,3}$	=	moments of inertial for yaw, pitch, and roll angles of the UAV [kg m^2]
$f_{1,2,3}^0$	=	total force acting on the UAV in the x, y, and z directions [N]
$f_{1,2,3}^1$	=	force acting on the UAV from the motors in the x, y, and z directions [N]
$\tau_{1,2,3}$	=	torques applied by the UAV in the x, y, and z directions [Nm]
x	=	state variables
$u_{1,2,3,4}$	=	state inputs
k_F	=	force coefficient of motors
k_M	=	moment coefficient of motors
l	=	length of UAV arm [m]
α	=	first order spin rate coefficient
β	=	zeroth order spin rate coefficient
$\mu_{1,2,3,4}$	=	motor command value, an integer between 1 and 200
$\sigma_{1,2,3,4}$	=	spin rate of motor [rad/s]
$\theta_{yaw,pitch,roll_{desired}}$	=	desired yaw, pitch, and roll angle of the UAV [rad]
$o_{1,2,3_{desired}}$	=	desired x, y, and z position of the UAV [m]
A, B	=	state space matrices
Q, R	=	gain matrices for lqr calculation
K	=	state feedback matrix
$k_{1,2,3,4,5,6,7,8,9,10,11,12}$	=	controller gains

II. Introduction

UAV's have been becoming increasingly prevalent in civilian use cases. From crop monitoring, to home deliveries, to private surveillance, to entertainment, UAV's could have a massive impact on everybody's daily life in the near future. However, one UAV on its own can only do so much. UAV's have a limited range, maximum payload capacity, and battery life. In order to make UAV's more commercially viable, multiple, if not an entire fleet, of UAV's are needed.

There are many more challenges involved with controlling multiple UAV's. For one, even the same model and type of UAV may have varying features that may make the control of the individual UAV's different. Even with the same code and gains, each UAV needs to be tested to make sure it is stable in flight with minimal error. Additionally, the UAV's need to be able to know where other UAV's are in space. The more UAV's that are introduced into a system, the more likely they are to collide. Collision avoidance between each UAV needs to be accounted for.

This lab focused on a small scale application: getting two UAV's to fly in the same space and fly in a simple formation. This serves as the basis for most applications, as for any monitoring or entertainment application, the UAV's need to be able to fly in a set formation to be used most effectively. To make this possible, several changes needed to be made to the previously implemented controller and ground-station. The on-board code for the second (aka "following") UAV needed to be adjusted so it would properly identify itself from the first (aka "leading") UAV, and the following UAV needed to be tested to ensure its stability. After this, the UAV's needed to be able to recognize each other and needed to be capable of avoiding collisions. Finally, for the formation flight, the planner on the following UAV needed to be tuned to quickly mirror the movements of the leading UAV.

III. System Architecture

In order to get both UAV's to operate at once, a more robust system architecture was needed. The system consisted of two UAV's communicating to two two independent ground stations. Both of these ground stations were independent and running separate planners. The planners were controlling he motion of the UAV's, either telling them where to move or what object they needed to follow or avoid. The ground stations communicated with the ground stations to tell them where the UAV's or objects were located.

A. UAV and on-board controller

The two UAV's that were used featured an on-board PD controller, as well as an XBee to communicate with their respective ground stations. The UAV's had five IR balls mounted on them to make them distinguishable to the MOCAP system. Both of the UAV's were very similar in build, both featuring silver-nut propeller blades. This distinction is important as the black and silver nut UAV's vary widely in flight characteristics. The UAV's were selected on the basis on similarity in their characteristic parameters through trial and error. This made it so the same on-board code could be run on both UAV's, although better results were seen when each UAV was given its own unique gain matrix.

However, the state feedback controllers operated the same way. They both run at 1000Hz and serve to update motor commands based off of the IMU and assisted by the MOCAP data received at 50Hz. Further, the code also logs the UAV's position and orientation variables from the on-board IMU and sends the variables to the ground station at 50Hz. This is done in the same way the UAV receives it's commands, by matching the hex values of the corresponding variables in the on-board code and ground station code. The flow diagram for the interaction between a single UAV, Obstacle (in our case the second UAV), and the ground station is shown in Fig. 1. The only caveat to our case would be the introduction of a second ground station that only receives/sends data to the leader UAV.

B. MOCAP

The MOCAP system consists of an array of multiple IR cameras distributed throughout the testing space. These cameras track reflective markers attached to the UAV and obstacle, where unique marker orientation and layout produces a non-ambiguous three dimensional object in the MOCAP software. Each object in the testing space, such as the UAV and an obstacle, are identified by a user in the MOCAP software, where the user creates what is called a "rigid body". These rigid bodies are needed to be initialized with the correct name before running the ground station. The primary UAV, either the one moving or the leader, was always assigned Rigid Body 1 and the second UAV was assigned Rigid Body 2. This allowed the ground stations to distinguish which UAV they were controlling and which one they were following or avoiding.

When running both UAV's, it was imperative to reduce error in the tracking system and following UAV position. The test floor needed to be as clear as possible, and the members holding the fishing poles had to try to stay out of line

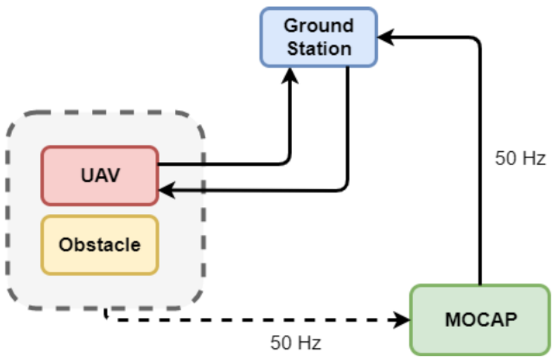

Fig. 1 Distance from the UAV to the object

of sight of the cameras. This was done because at certain times, one of the UAV's could partially block the other from being seen by the MOCAP. It needed to be ensured that maximum sight lines were allowed in this case so both UAV's could remain controllable.

C. Ground Stations

Each ground station was responsible for controlling one of the UAV's. The ground stations served as the planner for the UAV's, as well as the feedback where they received MOCAP data via Ethernet cable from the MOCAP computer and transmitted that data along with planner commands to the UAV's via an XBee.

In order to differentiate the UAV's in the MOCAP data, the ground station IDs needed to be adjusted. For the primary UAV, the ground station remained the same labeling the UAV as ID 1 and the obstacle as ID 2, where the ID number is the rigid body number. On the second ground station, this was reversed. This meant that both of the ground stations saw their UAV as the "UAV" and the other UAV as an "obstacle," allowing them to direct their UAV to or away from the other UAV.

The planner on both of these ground stations is what really carried the weight of this experiment. They told their UAV whether they were hovering, moving, avoiding, or following. The mission profile of each UAV was programmed into the planner to make the UAV's work together in one environment.

IV. Control Design and Implementation

A. Model

In order to control both of the UAV's, a model was made describing the motion of the UAV's. These equations of motion were derived by looking at the kinematic and dynamic equations associated with the UAV. The kinematics of the UAV model were expressed by Equation 1 while the dynamics were expressed by Equation 2.

$$\begin{bmatrix}\dot{o}_1\\ \dot{o}_2\\ \dot{o}_3\end{bmatrix} = \begin{bmatrix}v_1\\ v_2\\ v_3\end{bmatrix} \qquad \begin{bmatrix}\dot{\theta}_{yaw}\\ \dot{\theta}_{pitch}\\ \dot{\theta}_{roll}\end{bmatrix} = N\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \tag{1}$$

$$\begin{bmatrix}\dot{v}_1\\ \dot{v}_2\\ \dot{v}_3\end{bmatrix} = \left(\frac{1}{m}\right)\begin{bmatrix}f_1^0\\ f_2^0\\ f_3^0\end{bmatrix} \qquad \begin{bmatrix}\dot{\omega}_1\\ \dot{\omega}_2\\ \dot{\omega}_3\end{bmatrix} = \begin{bmatrix}J_1 & 0 & 0\\ 0 & J_2 & 0\\ 0 & 0 & J_3\end{bmatrix}^{-1}\left(\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix} - \begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}\begin{bmatrix}J_1 & 0 & 0\\ 0 & J_2 & 0\\ 0 & 0 & J_3\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}\right) \tag{2}$$

Likewise, inputs for the UAV also needed to be defined. The inputs were defined based off of the four motor commands, which control the force applied to the UAV by each motor. These motors each produce a force and a torque on the UAV. Their model is shown in Equations 3, 4, and 5.

$$f^1 = \begin{bmatrix} 0 \\ 0 \\ -k_F\sigma_1^2 - k_F\sigma_2^2 - k_F\sigma_3^2 - k_F\sigma_4^2 \end{bmatrix} \qquad \tau = \begin{bmatrix} -k_F l\sigma_3^2 + k_F l\sigma_4^2 \\ k_F l\sigma_1^2 - k_F l\sigma_2^2 \\ -k_M\sigma_1^2 - k_M\sigma_2^2 + k_M\sigma_3^2 + k_M\sigma_4^2 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -k_F l & k_F l \\ k_F l & -k_F l & 0 & 0 \\ -k_M & -k_M & k_M & k_M \\ k_F & k_F & k_F & k_F \end{bmatrix} \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_3^2 \\ \sigma_4^2 \end{bmatrix} \qquad k_F = 7.49e-6 \qquad k_M = 1.309e-7 \qquad l = 0.1725 \tag{4}$$

$$f^1 = \begin{bmatrix} 0 \\ 0 \\ -u_4 \end{bmatrix} \qquad f^0 = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R_1^0 f^1 \qquad \tau = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \qquad g = 9.81\frac{m}{s^2} \tag{5}$$

With these equations, a state space model was formed. The state space model was an important tool in controller design and allowed for a set of linear functions to substitute for the equations of motion found in 1 and 2. The state space form was defined as $\dot{x} = Ax + Bu$ where x, $\dot{x}$, and u are defined in Equation 6.

$$\begin{aligned} x &= \begin{bmatrix} o_1 & o_2 & o_3 & \theta_{yaw} & \theta_{pitch} & \theta_{roll} & v_1 & v_2 & v_3 & \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T \\ \dot{x} &= \begin{bmatrix} \dot{o_1} & \dot{o_2} & \dot{o_3} & \dot{\theta_{yaw}} & \dot{\theta_{pitch}} & \dot{\theta_{roll}} & \dot{v_1} & \dot{v_2} & \dot{v_3} & \dot{\omega_1} & \dot{\omega_2} & \dot{\omega_3} \end{bmatrix}^T \\ u &= \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T \end{aligned} \tag{6}$$

B. Design

After the model of the 12 states was defined, the model was linearized and implement into a controller. Linearizarion of the model was important for quickly computing proper motor commands the UAV needed to execute in order to preform its mission.

In order to implement a closed loop controller, a point to linearize about was chosen. Both of the UAV's were linearized about the same point since they would be making the same maneuvers. The point about which the equations of motion were linearized was chosen such that hover would be achieved at that point, where velocities and angular velocities were zero, and all of the orientation angles were chosen to be zero. The hover point was chosen to be $\begin{bmatrix} 0.5 & 0.5 & -1 \end{bmatrix}$. The only non-zero input associated with this condition was the force of gravity acting on the drone, mg. The values for the state variables at equilibrium are shown in Equation 9.

$$x_e = \begin{bmatrix} 0.5 & 0.5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \qquad u_e = \begin{bmatrix} 0 & 0 & 0 & mg \end{bmatrix}^T \tag{7}$$

The Jacobian of the equations of motion, Equations 1 and 2, was taken with respect to x_e and u_e to produce A and B, respectively. The resultant open loop system is shown in Equation 8, where A and B represent the matrices calculated by the Jacobian.

$$\dot{x_c} = Ax_c + Bu_c \tag{8}$$

The open loop controller found in Equation 8 needed a feedback loop to form a closed loop controller. A feedback loop of $u = -Kx$ was added to the the open loop controller to implement state feedback which is implemented as $\dot{x} = (A - BK)x$.

At this point it is important to mention that along with implementing a state feedback controller on the UAV's, the follower UAV was also tested with an on-board state feedback controller with integral action. During testing, it was observed that the quality of the on-board controller was more important for the follower UAV than for the leader UAV as the follower UAV showed a steady state error in the Z direction. It is important to minimize this error as a dynamical system which is implemented for transportation or other purposes that would require a very accurate system in order to avoid collision.

In this regard, effort was made to implement a state feedback controller with integral action which resulted in the state space to be augmented to a 15x1 matrix as follows:

$$x_c = \begin{bmatrix} x_{er} & y_{er} & z_{er} & \theta_{yaw_{er}} & \theta_{pitch_{er}} & \theta_{roll_{er}} & \dot{x}_{er} & \dot{y}_{er} & \dot{z}_{er} & \omega\dot{1}_{er} & \omega\dot{2}_{er} & \omega\dot{3}_{er} \\ \int x_{er} & \int y_{er} & \int z_{er} \end{bmatrix}^T \tag{9}$$

The K matrix consisted of gains defined by using the MATLAB function LQR. This function was defined using the A and B matrices defined in 8. The LQR was used to design a linear state feedback controller of the form $u_c = -Kx_c$ which was a simple PD controller. The LQR function produced a $12x4$ gain matrix ($15x4$ for with integral action). These gains were calculated to minimize the weighted sum of squared errors and inputs. Using the MATLAB function $K = lqr(A, B, Q, R)$, where Q and R were weighted penalties defined as diagonal matrices shown below. Their values were determined through iterative testing. This function produced the K matrix found in Equation 10.

```
Q = diag(75, 75, 1150, 5, 8, 8, 15, 15, 600, 8, 8, 1); // diag(...., 8, 8, 1, 0.01, 0.01, 0.01) (15x15)
R = diag(150, 150, 150 ,150);
K = lqr(A,B,Q,R);
```

$$K = \begin{bmatrix} 0 & 707.1 & 0 & 0 & 0 & 1734.0 & 0 & 591.6 & 0 & 259.2 & 0 & 0 \\ -707.1 & 0 & 0 & 0 & 1734.0 & 0 & -591.6 & 0 & 0 & 0 & 259.2 & 0 \\ 0 & 0 & 0 & 182.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 97.9 \\ 0 & 0 & -2768.9 & 0 & 0 & 0 & 0 & 0 & -2815.6 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

It should be noted that in case of integral action, the K matrix was obtained similarly, however, weights also had to be put on the error integral terms. Low weights were used as it is proven that integral action introduces instability to the system. These gains were found after iterative testing aimed at reducing the sum-squared error of the UAV position and yaw over a standardized flight-path. Initially, both UAV's used the same gains. This was due to both UAV's being of the silver nut variety, meaning that they had similar flight characteristics. In later experiments, a controller with a better tuned gain matrix was obtained from the group which primarily worked with that particular UAV, and had optimized their gains for that specific UAV. The gains shown above were tuned for the first, leader UAV. This served as a crude way to observe smaller steady state error.

C. Implementation

In order to implement a linear state feedback, the current state estimates were received from the MOCAP system. Based off the error, the control input u_c was calculated, which is shown in Equation 11.

$$x_c = x - x_e \qquad u_c = -Kx_c \qquad u = u_c + u_e \tag{11}$$

The motor commands μ_1, μ_2, μ_3, and μ_4 were observed to understand the the behavior of the controller. These motor commands are the inputs that are implemented on the on-board controller based on the estimated error. The motor commands are calculated using the squared spin rates that are determined using Equation 12.

$$\begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_3^2 \\ \sigma_4^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -k_F l & k_F l \\ k_F l & -k_F l & 0 & 0 \\ -k_M & -k_M & k_M & k_M \\ k_F & k_F & k_F & k_F \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{12}$$

Motor commands μ_1, μ_2, μ_3, and μ_4 are then calculated using α and β, which were derived from a previous experiment, as shown in Equation 13.

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\sigma_1} - \beta}{\alpha} \\ \frac{\sqrt{\sigma_2} - \beta}{\alpha} \\ \frac{\sqrt{\sigma_3} - \beta}{\alpha} \\ \frac{\sqrt{\sigma_4} - \beta}{\alpha} \end{bmatrix} \qquad where\ 0 < \mu \leq 200,\ \alpha = 3.8467,\ and\ \beta = 100.5216 \tag{13}$$

The motor commands were bounded between zero and two hundred as these were the minimum and maximum values the on-board computer could pass to the motors. With the closed loop controller implemented and motor commands calculated, the UAV was able to track and achieve a goal position with very little steady state error.

Just as the motor commands were capped, it was also necessary to stop the integral action when controller was implemented on tests if the error was not low enough. This was done so that the integral terms do not go above a certain threshold risking instability of the system. The implementation is shown in the code snippet below:

```
if (xc[2] < 0.25){                  //integral action to only implement when error is small
            integral[0] = integral[0] + 0.01*(xc[0] + xc_prev[0]);// trapezoidal method
            integral[1] = integral[1] + 0.01*(xc[1] + xc_prev[1]);
            integral[2] = integral[2] + 0.01*(xc[2] + xc_prev[2]);
            xc[12] = integral[0];
            xc[13] = integral[1];
            xc[14] = integral[2];
        }
        else{
            integral[0] = 0;                // if error larger than threshold, reinitialize integral
            integral[1] = 0;
            integral[2] = 0;
            xc[12] = 0;                     // to stop I-action
            xc[13] = 0;
            xc[14] = 0;
        }
```

The distance threshold was determined using the error in the z direction as that was the state with the highest error.

Disclaimer: Even though the system was tested with integral action controller on the follower drone, the final system did not include it as there was not enough time to tune the gains for our purposes. This means that the UAV's in both of the flights analyzed in Section VI use only the PD controller described earlier.

V. Planner Design and Implementation

The planner function is responsible for the motion planning of the UAV. A planner function may be designed to perform several tasks such as obstacle avoidance, trajectory following or even decision making in some cases. In our case, two simple planner functions were implemented. On the leader UAV, the planner function was responsible only for way-point navigation, whereas, on the follower UAV, the planner function was responsible for maintaining a certain trajectory based on the position values obtained from the leader UAV.

A. Planner Design

1. Leader UAV

For this UAV, no obstacle-responsive planner function was implemented on the ground station. Instead, the trajectory was set in the main function itself and the UAV was made to navigate through way-points. Keeping in mind future improvements to the model of the system, and the implementation of integral action on the UAV's, the way-points to be navigated through were changed minimally through the time steps. This was done in order to prevent the error term to initialize at a large value and cause the integrator to have a high value resulting in immediate instability.

That being said, it is possible to allow a planner function for obstacle avoidance on this UAV given that the obstacle is given a different ID than the follower UAV.

2. Follower UAV

A simple responsive flight planner was implemented on the follower UAV. This planner was responsible for setting the desired value of the follower UAV based on current value of the leader UAV. This was a minor change in the sense that since the follower UAV views the leader UAV as an "Obstacle", however instead of avoiding it, it is made to follow it. This way-point setting planner function was implemented as follows:

```
void planner(State6_f QuadCurrent, State6_f ObstCurrent, PosYaw_f *QuadDesired, PosYaw_f
QuadGoal)
{
    QuadDesired->Pos.x = ObstCurrent.Pos.x - 1.00;
    QuadDesired->Pos.y = ObstCurrent.Pos.y;
    QuadDesired->Pos.z = ObstCurrent.Pos.z;
}
```

Where, QuadDesired is the struct variable that stores the desired x,y and z position of follower UAV. This variable is updated with a new value at 50 Hz due to the fact that the MOCAP system determines the value of the obstacle (position of the leader UAV) at 50 Hz, and this value is then fed as the desired value for the follower UAV. However, since setting the desired value of the follower UAV equal to the current value of the leader would result in collision, an offset of 1m was introduced for the follower in the x-direction. Initially, an obstacle avoidance algorithm was also implemented by setting higher threshold values for the sphere of influence around the two UAV's, however due to the erratic behaviour of the drones, some collisions and instability between the two UAV's was noticed and a simpler algorithm was implemented in lieu of the obstacle avoidance.

VI. Experiment Results and Analysis

The goal of this design project was to create a multi-UAV system which is safe and controllable. By implementing the controller and planner from Sections IV and V, a multi-UAV system was tested and analyzed over several tests. A standard flight-plan utilized in previous experiments, as well as simple hover were explored to characterize the accuracy of our system. For the follower UAV, in hover the standard deviation of the entire flight was 0.1723m, and the analyzed standard flight-plan had a standard deviation of 0.5824m.

A. Hover

To test the multi-UAV system, two different flight plans where run, one being a simple hover. As explained in Section V, the following UAV tracked the leading UAV, and set it's desired position to be equal to the leader UAV with an offset in the x-direction of -1m. Testing this system with the hover, the data was plotted to show the error in the x, y, and z direction over time and the frequency of the mean sum squared error of the position of the following UAV. These plots can be seen in Figs. 2 and 3 respectively.

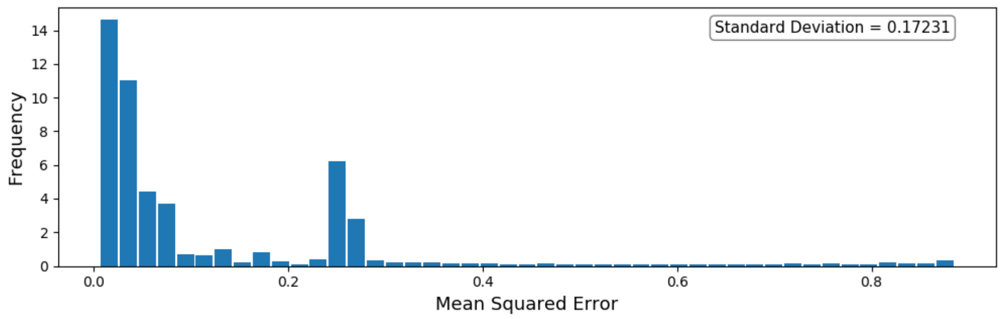

Fig. 2 Frequency of the mean squared error of follower UAV for hover test flight

It can be seen in Fig. 3 that the positional error demonstrates a relatively large steady state error in the z-direction. This error prompted the experimentation of integral action with state feedback in the following UAV, described in IV, and the modification of the follower UAV's gains. The x and y directions have a relatively low average error, but an oscillatory error curve shows the responsiveness of the following UAV to the leading UAV's position is inherently unstable. This is mostly due to the fact that the leader UAV also has some error in it's own flight-plan which is trying to be corrected at the same time, and the un-tuned follower UAV gains. In Fig. 2, it can be noticed that the highest frequency mean sum squared is very low, hinting to a relatively accurate system overall. Given that the most major issue seen in this test flight was the steady-state error in the z-direction, the group then tested the standard flight-plan from previous experiments.

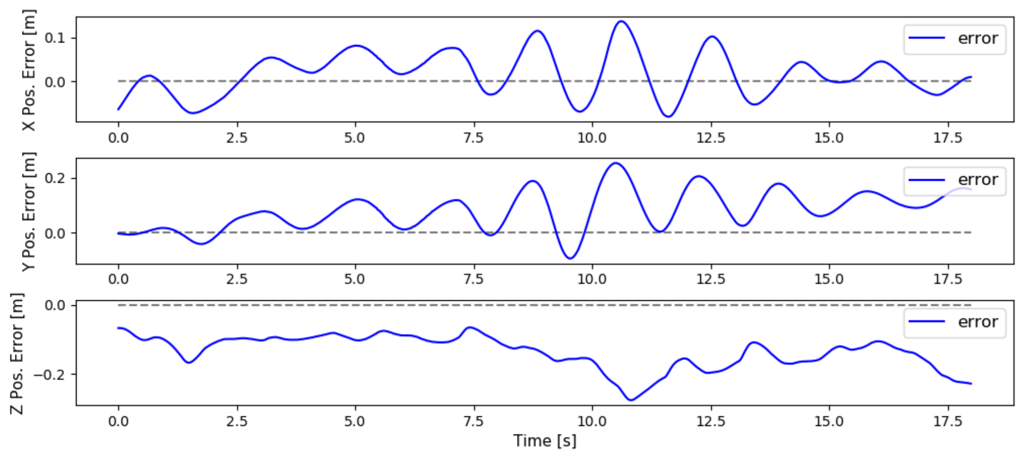

Fig. 3 Positional error of follower UAV over time for hover test flight

B. Standard Flight-plan

The next flight-plan which was tested was the standard flight-plan, which contains 8 points in the UAV flight-space which resemble a box. The first half of this flight-plan was used, as the following offset of the following UAV made flight on the second half of the flight-plan difficult as the UAV was nearly out of the view of the MOCAP system. Similar analysis of the standard flight-plan was done as was done for the hover, where the mean sum squared frequency and the error in position over time were plotted in Figs. 4 and 5 respectively.

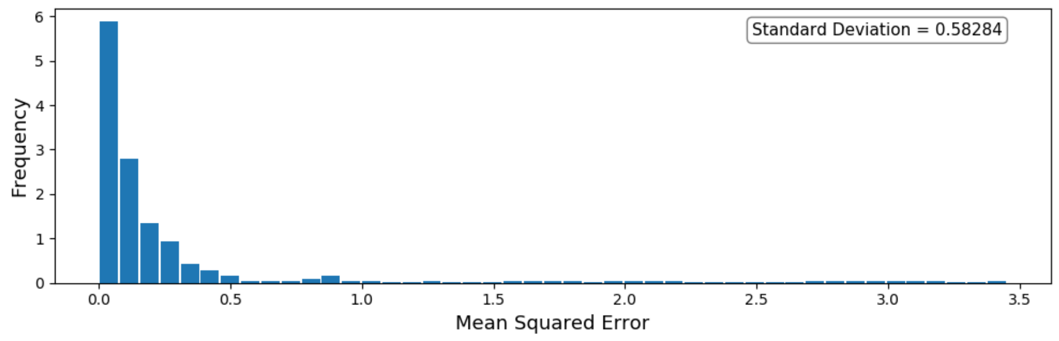

Fig. 4 Frequency of the mean squared error of follower UAV for standard flight-plan test flight

Seen in Fig. 5, the positional error in the x, y, and z position is again minimal, and in Fig. 4, the higher frequency of the mean squared error is small, just like the hover test. These two figures paint a clear picture of the accuracy of the implemented planner and controller in the follower UAV. The next measurement of the fidelity of the implemented planner can be deduced by analyzing Fig. 6, where the position of the following UAV and the leading UAV with an offset, represented as the "desired position".

In Fig. 6, a small steady-state error can be noticed in the z-position, and a significant "lag" in the x, y, and z positions. This lag was considered to be acceptable, as the follower and leader UAV never got close enough for a collision during flight. The flight of the follower UAV can also be considered mildly "erratic", where during the flight, the following UAV "jumps". These jumps can be seen in the small peaks in the following UAV's position over time. This erratic behavior was not able to be pinpointed to a singular cause, but it is theorized to be due to the implementation of the planner of the follower UAV having no filter. It is also theorized that the inclusion of state feedback integral action in the follower drone would be able to minimize these small perturbations during flight.

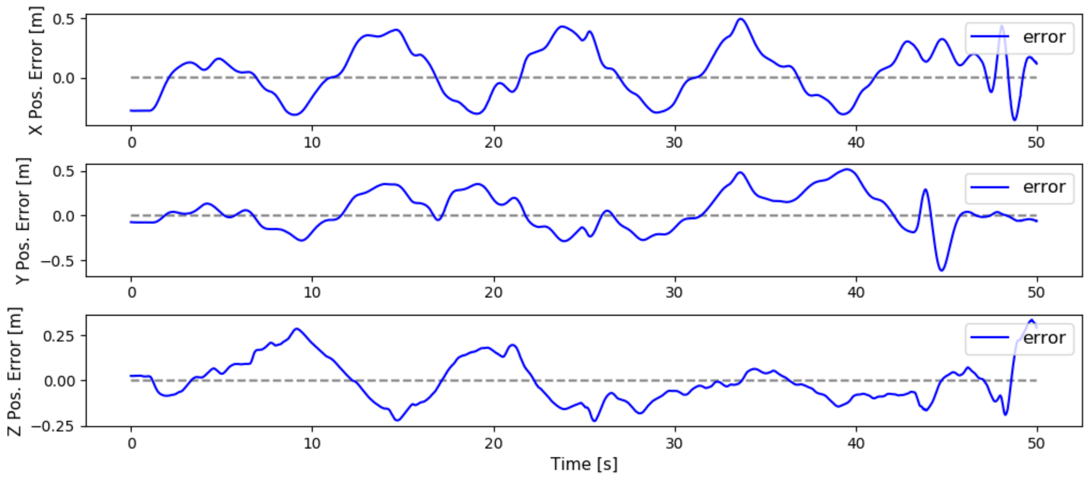

Fig. 5 Positional error of follower UAV over time for standard flight-plan test flight

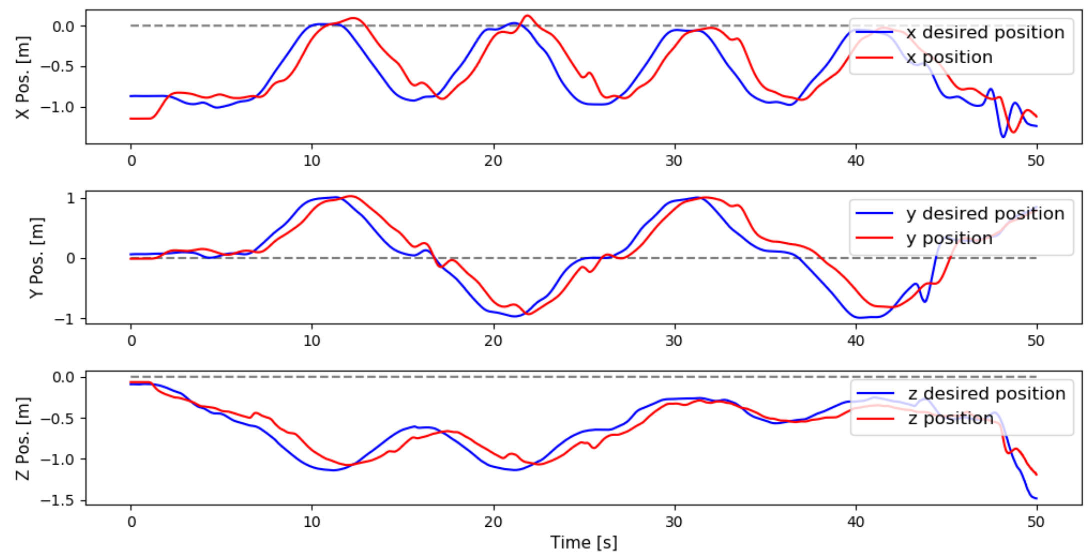

Fig. 6 Position follower UAV over time for standard flight-plan test flight

VII. Conclusion

In this experiment, two UAV's were made to fly at the same time and in formation. First, the two UAV's were made to hover together to make sure that the MOCAP system was able to keep track of both UAV's. Then, one UAV was made to hover while the other followed a preset flight pattern while implementing obstacle avoidance. It was found that the drone doing obstacle avoidance was able to recognize the second hovering drone and avoid it. The leader UAV was then set to follow a pre-determined flight path. The follower UAV was then set to follow the leader drone along the flight path. It was found that the follower UAV was able to follow the leader UAV along the flight path.

Although the experiment was successful in that the follower UAV was able to follow the leader UAV along a flight path, improvements could have been made to better improve results. During testing it was found that whenever the two UAV's would get close to each other, the UAV's would start to exhibit steady state error. In order to reduce the steady state error with the current PD controller, gains could've been adjusted better for each individual drone. Another way to compensate for this would have been to implement integral action with the PD controller. This would allow the controller to get rid of the steady state error that was encountered by the system. Although some of integral action was done during the experiment, more trials experimentation with this involving both UAV's would've made the experiment better.

Acknowledgments

We would like to thank Prof. Timothy Bretl for giving us a great introduction to the world of aerial control systems, and for giving us the opportunity and independence to apply the knowledge through this project. We would also like to extend our thanks to Dan Block, Shubhendra Chauhan, Marc Akiki, Noel Brindise, and Guisy Falcone for their relentless support throughout the lab hours. Lastly, we would like to thank the University of Illinois for providing us with state-of-the-art facilities and numerous opportunities to develop into mature students, researchers, and citizens.