Validation and Verification Flight Tests of Fixed-Wing Collaborative UASs With High Speeds and High Inertias

A Ram (Bella) Kim, Iowa State University

Available at: https://works.bepress.com/arambella-kim/8/
Validation and Verification Flight Tests of Fixed-Wing Collaborative UASs With High Speeds and High Inertias

Aaron Blevins¹, A Ram Kim¹, Daksh Shukla¹, Shawn Keshmiri², and Weizhang Huang³,

University of Kansas, Aerospace Engineering Department, Lawrence, KS, 66045

The research objective for this work is to validate and verify guidance, navigation, and control algorithms that are designed for fixed-wing collaborative unmanned aerial systems (UASs) in unstructured environments. A biologically-inspired swarm control theory provides a framework to distribute sensor payloads between several smaller and less complex agents that have local interactions. Controller design and flight testing of large UASs with high speeds and high inertias holding a formation in a dynamically changing environment and in the presence of external disturbances is complex and requires advanced planning and safety measures. Verification and validation flight tests were conducted using a fixed-wing unmanned aerial system with 4 meter wing spans to investigate the robustness of the guidance, navigation, and control algorithms and also test the embedded morphing potential field collision avoidance logic.

Nomenclature

\[\nabla \text{mpf} = \text{gradient of morphing potential field}\\
\Gamma = \text{morphing factor}\\
\alpha, \beta = \text{airflow angles}\\
\sigma = \text{avoidance radius}\\
\phi = \text{bank angle}\\
\theta = \text{pitch angle}\\
\psi = \text{heading angle}\\
\eta = \text{relative angle between moving point velocity vector and relative position vector}\\
\xi_{\text{app}} = \text{approach heading relative to the trajectory path}\\
\xi_{\text{max}} = \text{maximum approach heading relative to the trajectory path}\\
\text{mpf} = \text{morphing potential field}\\
p, q, r = \text{body angular velocity}\\
\text{pf} = \text{potential field}\\
P_{\text{obj}} = \text{position in 3D space}\\
R_{\text{min}} = \text{minimum turn radius}\\
S = \text{shifting variable}\\
u, v, w = \text{translational velocity in body coordinate system}\\
\]

I. Introduction

The majority of airborne Earth Science missions require fixed-wing UASs with larger range, payload, and power capacity flying in unstructured environments. The related cost and complexity of large fixed-wing UASs are an order of magnitude higher than small, but mobile, UAS; however, the geometry, power, and payload limitations of small UAS reduce their functionality and practicality dramatically. Many works have focused on addressing this limitation by distributing tasks and science payload between a swarm of several smaller and less complex, payload driven, coordinated, and cooperative airborne sensors controlled by biologically inspired control laws and guidance logic.

Although several applications of discrete swarming robots or UAS have been done in recent years, swarm demonstrations are limited to simulation or applied to control a limited number of vehicles. The majority of cases have been either 2D or 3D swarms of small rotary-wing UASs assumed as point masses with no aerodynamic effects and extremely small range and payload capability, executing no practical science mission. Almost all cases have been tested in controlled environments with no external disturbances [1-4], where Earth Science missions and space

¹ Ph.D. Student, Department of Aerospace Engineering, University of Kansas, Lawrence, KS, 66045.
² Associate Professor, Department of Aerospace Engineering, University of Kansas, Lawrence, KS, 66045.
³ Associate Professor, Department of Mathematics, University of Kansas, Lawrence, KS, 66045.

Copyright © 2018 by Shawn Keshmiri. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.
explorations are predominantly conducted in unstructured and hostile environments with little or no a priori information. The work in [8] utilizes simple mathematical models of swarms given by the following three rules: birds must move in the same direction as their neighbors, remain close to their neighbors, and avoid collisions with their neighbors. Many other alterations and adaptations of interest to multi-agent UAS applications from this root research have surfaced such as flocking with a virtual leader, a virtual leader with a minority of uninformed agents, and a virtual leader with variable velocity, most of which have some inherent obstacle avoidance capabilities [5]. Even more recent research has been conducted on discrete swarm algorithms either in simulation or applied to a limited number of vehicles using virtual attractive and repulsive potential functions (VARP) [5,6,7,11]. One experiment of interest included small ground vehicles of up to three agents with fixed speed and turning radii following a virtual leader and experiencing variations in cohesion and stabilizations for different set of parameters of the potential function [11]. Another experiment conducted a 3D collaborative search with submarines in a miniature aquatic test bed [3]. Probably the most relevant application was a number of fixed-wing UAS flight tests conducted by Johns Hopkins University’s Applied Physics Laboratory (APL) where demonstrations included search, tracking, and classification of targets with two UAS and four unmanned ground vehicles (UGV) [5]. Though insightful, the majority of root research and robot or rotary wing UASs swarm investigations ignore, or dramatically simplify, the dynamics of UASs or robots and simulate agents as point mass particles. Point mass assumption can be held for small and slow motions; however, spatiotemporal requirements on distributed high speed and high inertia airborne sensors will often necessitate close formations where such assumptions cannot be made. The excellent fixed-wing work done by APL has significant relevance to the proposed research, though still neglects many of the adaptive qualities and tight formations required for this project.

The KU team proposed a novel guidance logic for multi-agent fixed-wing UASs using a moving mesh method. The moving mesh method is originally designed for use in the adaptive numerical solution of partial differential equations, where a high proportion of mesh points are placed in the regions of large solution variations and few points in the rest of the domain. Using the virtual moving point concept [13], UASs follow a virtual point position with the desired heading angle and velocity in 3D space [14]. This work presents the flight test procedure and validation and verification flight tests of two large fixed-wing collaborative UASs (DG-808) flying at 35 knots in close proximity.

II. Platform Characteristics and Summary of GNC

A. Guidance:
The guidance algorithms are used to compute the UAS state commands required to transition the system from the current position to the desired position. In this work, a moving point guidance algorithm is implemented that can be used for any arbitrary trajectory curve. This method computes necessary speed (\(V_t\)), roll angle (\(\phi\)), pitch angle (\(\theta\)) and sideslip angle (\(\beta\)) commands. The moving points indicate the desired position and velocity in 3-dimensional space in an inertial coordinate frame depicted by the following equation:

\[
o[k]: \{ p^o[k] = p^o_H[k], p^o_E[k], p^o_N[k]; \quad \dot{\theta}^o[k] = v^o_W[k], v^o_E[k], v^o_N[k]\}
\]

This is a discrete time (k) equation where \(o[k]\) indicates the moving point array consisting of the position and velocity vectors in inertial NEH frame, which are essentially different from vehicle’s position and velocity vectors (see Figure 1). The goal of this guidance is to asymptotically reduce the error in angles along the longitudinal and lateral planes to zero. To compute the state commands \(\theta_{cmd}^o\) and \(\phi_{cmd}^o\), the errors of the distance and velocity vectors are defined by geometric parameters shown in Figure 1. By updating the moving points, the line segment between point A and point B can be updated. Using line segment AB, the perpendicularly projected point of the aircraft position \(d\) can be determined. Distances \(d_{R_{lat}}\) and \(d_{R_{lon}}\) correspond to the distances between point \(d\) to the virtual reference points \(R_{lat}\) and \(R_{lon}\). Vectors \(L_{lat}\) and \(L_{lon}\) are defined from the aircraft position to the virtual reference points. Considering the profound differences between longitudinal and lateral directional dynamic modes
different values are used for the $L_{\text{Lat}}$ and $L_{\text{Lon}}$. Error angles $\eta_{\text{lon}}$ and $\eta_{\text{lat}}$ are defined between the velocity vector of the aircraft and the vectors $L_{\text{Lat}}$ and $L_{\text{Lon}}$.

The lateral guidance uses the lateral acceleration to determine the appropriate roll angle command ($\phi_{\text{cmd}}$). Shown in Figure 1, the radius $R$ is of the circular trajectory needed to reach the virtual reference point $\vec{R}_{\text{Lat}}$. The desired acceleration can be calculated from the following equation:

$$R = \frac{L_{\text{Lat}}}{2 \sin \eta_{\text{Lat}}}, \quad a_{s\text{cmd}} = \frac{V_T^2}{R} = \frac{2V_T^2 \sin \eta_{\text{Lat}}}{L_{\text{Lat}}}$$

This guidance method is used to gradually reduce error angles along the lateral-directional plane to zero. The lateral error ($\phi_{\text{cmd}}$) is calculated through the lateral acceleration by the following equation outlined from [15]:

$$\phi_{\text{cmd}} = \tan^{-1} \left( \frac{a_{s\text{cmd}}}{g} \right)$$

A PI controller approach was used to find $\theta_{\text{cmd}}$ as a function of the longitudinal error angle $\eta_{\text{Lon}}$.

$$\theta_{\text{cmd}} = \tan^{-1} \left( k_{a\text{Lon}} \cdot \eta_{\text{Lon}} \cdot \left( k_{p\text{Lon}} + \frac{k_{i\text{Lon}}}{s} \right) \right)$$

The proportional and integral gains $K_{p\text{Lon}}$ and $K_{i\text{Lon}}$ are used for the PI controller. $K_{a\text{Lon}}$ is typically an adaptive gain equal to $2V_T^2/L_{\text{Lon}}$, however this value was held constant for this work in order to reduce oscillations. The commanded sideslip angle command was maintained at zero throughout the flight, and the commanded airspeed was held constant at the aircraft trim condition.

**B. Navigation**

1. **LQ Path Planning**:

   For formation flight, a virtual reference point was used rather than designating a leader among the agents. This has the advantage of ensuring aircraft control if communication is lost between the agents. To accomplish autonomous formation flight, the virtual reference point was planned using an LQ guidance algorithm.

   LQ guidance methods were used for the virtual reference point, following procedure outlined in [14]. This guidance is used to reduce the position error, $d$, using linear quadratic regulator (LQR) control theory. LQR theory regulates the states based on the quadratic cost function as follows:

$$J = \int x^T Q x + u^T R u \, dt$$

Where $x$ is the state vector and $u$ is the control vector. The state vector consists of the aircraft position error and velocity error, $=[d, v_d]^T$, while the control vector $u$ is the lateral acceleration. Figure 2 shows the geometric definitions used for applying LQR control theory to aircraft path planning. The distance $d_{\text{lat}}$ is the crosstrack error.
between the moving point (virtual leader) and the projection point on the desired trajectory. This can be calculated from:

\[ d_{lat} = |\vec{V}| \sin(\psi - \psi_{desired}) \]

This cross track error is bounded by a maximum allowable error, \( d_{lat} \). The rate of the heading angle change is driven by the lateral acceleration of the aircraft:

\[ u = v\dot{\psi} \]

Where \( v \) is the speed of the moving point. Linear dynamics of the errors of distance and velocity error are used to determine the required acceleration.

\[ \dot{x} = Ax + Bu, \quad u^* = -R^{-1}B^T Px \]

Where \( P \) is the solution of the Riccati equation, shown below.

\[ A^T P + PA - PB R^{-1} B^T P + Q = 0, \quad Q \geq 0, \quad R > 0 \]

Where \( Q \) and \( R \) are the weighting matrices. \( R \) was set to 1, while \( Q \) is defined by:

\[ Q = \begin{bmatrix} q_1^2 & 0 \\ 0 & q_2^2 \end{bmatrix} \geq 0 \]

Distance and velocity error rates must be determined in order to complete the linear state. From Figure 3, it can be observed that:

\[ \dot{d}_{lat} = R_{lat} \sin(\psi_R - \psi_p) \]
\[ \dot{d}_{lat} = v \sin(\psi - \psi_p) \]
\[ \dot{v}_d = v(\psi - \psi_p) \cos(\psi - \psi_p) \]

\( \psi_p \) is set to 0 because the general desired trajectory is unknown. The above equations are linearized by applying the small angle assumption and can be expressed in state space as:

\[ \begin{bmatrix} \dot{d}_{lat} \\ \dot{v}_d \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_{lat} \\ v_d \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \]

After solving the Riccati equation, \( q_1 \) and \( u^* \) are determined by:

\[ q_1 = \left| \frac{d_b}{d_{b - d}} \right|, \quad u^* = -\left( \frac{d_b}{|d_b - d|} d + \sqrt{2 \frac{d_b}{|d_b - d|} + q_2^2 v_d} \right) \]

Where \( q_2 \) is the design parameter.

The significance of LQ guidance is that the moving points are planned by considering the initial heading angle, \( \dot{\psi} \), and converges to the desired heading smoothly. This is very profound since the planned path will not induce any abrupt changes in the dynamics of the vehicle.
2. Reference Point Formation Algorithm:
A reference point formation algorithm is used to assign the path for each agent. The position of each outer agent can be determined using the relative distances from the reference point. The desired velocity of each agent is based on the heading angle of the reference point’s velocity vector so that all agents in formation can maintain swarm coherency throughout the flight. Figure 4 shows this description.
Here, \( \psi_{ref} \) indicates the heading angle of the velocity vector for the reference point. The following equations present each agents’ position in an inertial NED frame by incorporating the position of the reference point and the relative distance from it:

\[
\begin{align*}
  p_{N_i} &= p_{N_{RP}} - |E_i| \sin \psi_{RP} + |N_i| \cos \psi_{RP} \\
  p_{E_i} &= p_{E_{RP}} + |E_i| \cos \psi_{RP} + |N_i| \sin \psi_{RP} \\
  p_{H_i} &= p_{H_{RP}}
\end{align*}
\]

Here, \( p_{N_i}, p_{E_i}, p_{H_i} \) are the north, east and height positions, while \( |E_i|, |N_i| \) are the desired relative distance along the east and north directions, respectively, for the \( i \)th agent.

\( p_{N_{RP}}, p_{E_{RP}}, p_{H_{RP}} \) are the north, east, and height positions of the reference point, while \( \psi_{RP} \) indicates the heading angle of the reference point. In this research, the formation is maintained in the longitudinal plane (North-East plane in inertial coordinates).

3. Morphing Potential Collision Avoidance:
A morphing potential algorithm is used to create a potential-based direction vector for the aircraft as a function of the relative distance and relative velocity between the agent and obstacles, outlined in [13]. In this work, other agents are considered as obstacles to be avoided. The general form of the potential is usually in a contour shape such as a circle. The potential function equation is defined as follows:

\[
p_f = A \exp \left( -\frac{d^2}{\sigma} \right)
\]

Here, \( p_f \) indicates the potential, \( A \) is the amplitude, \( d \) is the distance between the agent and the neighboring agent (obstacle), and \( \sigma \) is the radius of the obstacle. The unique feature of morphing potential is that its size and shape is varied based on the relative velocity vector between the agent and the obstacle, which enables path planning in advance. This characteristic makes it appropriate for systems with high speed and high inertia. The morphing potential function is defined as follows:

\[
mp_f = \exp \left\{ -\Gamma \frac{\lVert \vec{p}^{obs} - \vec{p}^{UAS} + \vec{S} \lVert^2}{\sigma^2} \right\}
\]

Where \( mp_f \) is the morphing potential, \( \Gamma \) is the morphing factor, \( \vec{p}^{obs} \) is the inertial position of the neighboring agent (obstacle), \( \vec{p}^{UAS} \) is the position of the agent, and \( \vec{S} \) is the shifting vector. The shifting term is used to create a more efficient potential field, by considering the position of the obstacle closer to the agent so that the actual path can be planned smoothly while avoiding unnecessary levels of cost with respect to off-track travel after the agent has passed the obstacle.

C. Control:
Using the generated guidance outputs \( (V_{cmd}, \phi_{cmd}, \theta_{cmd}, \beta_{cmd}) \), the role of the controller is to execute the desired control surface deflections to maneuver the vehicle. In this work, a decoupled LQR control was used, separating the lateral and longitudinal controllers. To formulate the LQR gain, the state space was obtained using the stability and control derivatives from the dynamic model and regulator terms that are the integration of errors in \( V_T, \phi, \theta, \) and \( \beta \). The sideslip angle (\( \beta \)) is continuously commanded to zero. The following equation shows the augmented linear state space and states for each LQR controller.
\[
\dot{x}_{\text{aug-lat}} = A_{\text{aug-lat}} x_{\text{aug-lat}} + B_{\text{aug-lat}} u_{\text{aug-lat}} \\
\dot{x}_{\text{aug-lon}} = A_{\text{aug-lon}} x_{\text{aug-lon}} + B_{\text{aug-lon}} u_{\text{aug-lon}}
\]

All states are perturbed states. For error calculation of regulator terms, total states are used.

\[
x_{\text{lat}} = \begin{bmatrix}
\beta \\
\phi \\
p \\
r
\end{bmatrix}, \quad u_{\text{lat}} = \begin{bmatrix}
\Delta \delta_a \\
\Delta \delta_r
\end{bmatrix}, \quad x_{\text{lon}} = \begin{bmatrix}
V_T \\
\alpha \\
\theta \\
q
\end{bmatrix}, \quad u_{\text{lon}} = \begin{bmatrix}
\Delta \delta_e
\end{bmatrix}
\]

Dynamic modeling for the DG808 aircraft was done in both Advanced Aircraft Analysis (AAA) and Athena Vortex Lattice (AVL) modeling software, shown below in Figure 5.

![Figure 5: AAA and AVL Dynamic Modeling for DG808 UAS](image)

Pertinent aircraft size and speed information is shown in Table 1.

<table>
<thead>
<tr>
<th>Table 1: DG 808 Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
</tr>
<tr>
<td>Wingspan</td>
</tr>
<tr>
<td>Length</td>
</tr>
<tr>
<td>Cruise Speed</td>
</tr>
</tbody>
</table>

The following equations represent numerical results of the augmented linear state space for the DG808 aircraft:

\[
A_{\text{lat}} = \begin{bmatrix}
-0.44 & 0.54 & -0.01 & -0.95 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
-10.74 & 0 & -20.97 & 2.93 & 0 & 0 \\
10.08 & 0 & -1.29 & -0.66 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad B_{\text{lat}} = \begin{bmatrix}
0 & 0.16 \\
0 & 0 \\
140.6 & 0.62 \\
-6.77 & -6.5 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]

American Institute of Aeronautics and Astronautics
\[
\begin{bmatrix}
-0.2 & -7.7 & -32.2 & 0 & 0 & 0 \\
-0.02 & -9.64 & -0.02 & 0.96 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0.2 & -121.4 & 0.01 & -5.48 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
3.04 & -0.85 \\
2 \times 10^{-5} & -0.3 \\
0 & 0 \\
-1.73 & -40.8 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]

III. Pre-Flight Software and Hardware Testing

For simulating multi-agent UASs, the GNC algorithms and aircraft dynamic model were developed on MATLAB-Simulink software platform. The simulation software consists of navigation, guidance and control blocks that excite the six degrees of freedom (6-DoF) nonlinear aircraft equations of motion. The whole setup is strictly coupled in a closed loop feedback framework, allowing a close to real flight simulation of the GNC algorithms before an actual flight test. Sensor data or navigational states are generated by integrating the 6-DoF aircraft equations of motion with respect to carefully chosen and pragmatic initial conditions for aircraft position and orientation states, as well as control surface inputs. The aircraft states generated henceforth are fed back to the sensor emulation blocks where real time noise and external disturbances are introduced into the system. This information is fed into the navigation algorithm that is used to generate a trajectory according to the user-defined waypoints. The trajectory and aircraft states are then fed in as an input to the guidance algorithm. With the given path, the guidance algorithm propagates the state commands as mentioned in the previous sections. Finally, the controller uses the guidance state commands to produce the desired control surface deflections, and then applies them to the aircraft dynamic model.

Furthermore, this simulation process is a consistently refining practice, as in preparation for a new flight test, the simulation is carried out by using initial conditions (states) of the aircraft from previously conducted flight tests’ data. These states are retrieved from the flight test data, from the exact data point when the pilot engages the autopilot switch. As a practical test for enabling the autopilot system in the simulation, a time dependent and user controlled engagement feature is implemented so that the switch can be turned on and off as in an actual flight test.

The hardware used on the aircraft primarily consists of two processor boards: a Tegra K1 (onboard computer with GNC software) and a custom-built Data Acquisition Board (DAQ). The DAQ board acts as an interface for data exchange between the control surface servo actuators, remote control pulse width modulation (PWM) signals, the dynamic pressure sensor and the Tegra. State sensors include GPS (Global Positioning System), IMU (Inertial Measurement Unit) and dynamic pressure sensor. For GPS and IMU, Vector NAV VN-200 [4] module is used and is connected directly to Tegra. AMS 5812 pressure sensor is used for measuring the dynamic pressure (air speed) via a pitot tube that is interfaced with the DAQ board. A telemetry radio device Xbee Pro 900 XBP9B-DMST-002 is used as the main communication transceiver for exchanging data between the ground control station and both of the aircraft. Communication reliability and robustness are the most important factors for maintaining formation in a cooperative aircraft flight. The Xbee supports mesh networking that enables efficient and fast data packet transfers in real time. Experiments show that the maximum round trip time taken by the 2 aircraft in exchanging data is 2ms. The Tegra runs on Linux operating system (Ubuntu 14.04 LTS) and is equipped with a meta-operating system known as “Robot Operating System” (ROS) framework that is used as the main communication server managing all the individual sensors and GNC processes.

Before conducting a flight test, a Hardware-in-The-Loop (HiTL) test is conducted on both the aircraft testing the onboard GNC software with actual servo control surface deflections and real time wireless communication between the aircraft and the ground control station. Figure 6 shows the HiTL setup. A Windows-based machine is used to run the Matlab 6-dof aircraft dynamic model that generates vehicle states. It is connected serially with the Tegra processor on each aircraft. The controller process node collects the aircraft states data, with the data from other aircraft through Xbee and computes servo commands, which are sent back to Matlab to excite the 6-DoF model and to the servo actuators. Therefore, the HiTL test bed entails real-time testing of 900 MHz communication between the aircraft. Both aircraft are connected to the Q-Ground Control Station (QGCS) running on a separate computer, that shows the real-time formation flight trajectory and aircraft states.
Simultaneous flight tests for two collaborative aircraft were conducted in order to validate the multi-agent GNC algorithms. Two different flight tests were conducted with varying and random initial conditions in terms of position and orientation states, for both the aircraft with respect to the user-defined waypoints. Four waypoints are uploaded on the onboard computer via the Q-Ground Control Station (QGCS), and they are approximately positioned as a rectangle. The waypoints forming the four sides of the rectangle are referred to as north, south, east and west waypoint legs. The north and south legs are approximately 2500 feet in length, and the east and west legs are approximately 1000 feet in length. The aircraft positions, sensor information, and guidance commands from both aircraft were updated over telemetry in real time and displayed on the ground station during flight testing, as shown in Figure 7.

In the first test scenario, the first aircraft (U1) takes off in manual RC mode and the pilot flies it to an altitude of about 300 feet before engaging the autopilot. The second aircraft (U2) is flown by another pilot and is taken to an altitude of about 400 feet before turning on the autopilot. There is a vertical distance between the two aircraft of about 100 feet for this flight test so as to conduct a safe 2D formation flight and ensure no collision in the event of any system failures. Both of the aircraft merge together laterally with a close proximity of approximately 400 feet just south of the southern waypoint leg (Figure 8, left) and with a heading pointing towards the north-east. Throughout the
flight, the aircraft tracked a pre-defined formation of 200 feet separation in the along-track direction and 100 feet separation in the cross-track direction.

Figure 8: Collaborative Flight Test, Tracking Results

Figure 9 shows one of the collaborative flight tests conducted using two DG-808 UASs. Considering FAA UAS flight rules, the UASs had to be in line of sight at all time, and operated by individual pilots. Pilots usually had problems tracking UASs beyond 0.25 miles distance, confining the flight to a 1000 ft by 2500 ft area. In addition to wind and environmental disturbances, the small flight test space had an adverse impact on the overall performance of UAS altitude hold and tracking around the corners of the waypoint box. The desired relative distance between the two DG808 UASs was set at 223.6 ft. Although the desired initial position of UASs was assumed to be 223.6 ft, in reality UASs were 612 ft apart when both autopilot systems were engaged (~388 ft. error @ t=0). Other flight test experiments showed the same randomness in the UASs’ initial conditions and inability of pilots to precisely control UASs in 3D space.

The morphing potential field collision avoidance logic was also active, where the separation distance between UASs was set to be 100 ft. As shown in Figure 10, around 70 seconds into flight when UASs got closer than 100 ft, and the collision avoidance logic kicked in and UASs were commanded to return to a safe separation. Such loose formation might be acceptable for a small number of UASs in missions where tight formation is unnecessary (e.g. search and rescue). However, when great precision is required (e.g. synthetic aperture radar) or the number of UASs is more than two, simple tracking methods (such as VARP) are not practical.

Figure 10: Agents Distance Error

Figure 11 and Figure 12 depict the flight test with simulated trajectories in a 2D northeast view, a 2D vertical height and east view, a plot of the rate of change of altitude, and a 3D view of the aircraft path. The simulations were
conducted using the exact initial conditions as observed in the flight test. It can be seen that the trajectory from the flight test and simulations are close along the northeast corner, with a maximum error of about 200 feet along the northwest corner. Overall, the trajectories match well, validating the simulation procedure and aircraft dynamic model. Flight test and simulations can be easily distinct due to environmental disturbances, such as wind gusts that are unpredictable and difficult to emulate in simulations.

![Figure 11: Trajectory Tracking Comparison with Simulation, Aircraft U1](image1)

Figure 11: Trajectory Tracking Comparison with Simulation, Aircraft U1

![Figure 12: Aircraft U1 Altitude Rate and 3D Tracking Comparison with Simulation](image2)

Figure 12: Aircraft U1 Altitude Rate and 3D Tracking Comparison with Simulation

Figure 13 shows the aircraft U1 lateral-directional states and control inputs compared with the simulation data. It can be observed that flight parameter curves for the roll angles and the yaw rates are a match well, with some sensor noise that is expected to be observed during real flight test.
Similarly, longitudinal states and control inputs are shown in Figure 14.

For aircraft U2, the trajectory and altitude tracking and pertinent state and control surface inputs are shown in Figures 15 through 18 and is compared with simulation data.
Figure 15: Trajectory Tracking of Aircraft U2

Figure 16: Aircraft U2 Altitude Rate and 3D Tracking Comparison with Simulation

Figure 17: Aircraft U2 Lateral-Directional States and Control
Validation and verification flight tests was successfully performed for two collaborative fixed-wing autonomous UASs. The autopilots for the UASs were initialized with random initial positions in the flight testing area, and the systems successfully merged into the aggregation mode and held the desired formation shape while at the same time tracking the waypoint trajectories. The confined flight test area, the large wingspan of UAS platform (4 meter), and the high cruise speed (~35 knots) had adverse impact on the overall tracking of the desired formation. The morphing potential field collision avoidance algorithms was enacting in real-time and to prevent UASs from getting closer than 100 ft of each other. The comparison of flight test tracking performance, along with aircraft states and control inputs, with that of the simulation predicted and closely matched.

Acknowledgments

This project was funded by NASA CAN project #NNX15AN04A and the Paul G. Allen Family Foundation (PGAFF). The authors would like to thank NASA and PGAFF for their support. Authors would also like to thank Hady Benyamen (KUAE Master’s Student) and Grant Godfrey (KUAE Bachelor Student) for all of their help and technical supports as well as our pilots Matthew Tener and Rudy Voldrich.

References


