Multiple UAVs formation flight experiments using Virtual Structure and Motion Synchronization

Norman H. M. Li* and Hugh H. T. Liu†

In this paper, the synchronized position tracking controller is incorporated in formation flight control for multiple flying wings. With this technology, the performance and effectiveness of the formation controller are improved when the virtual structure approach is utilized to maintain formation geometry in simulations. It is desirable to run flight tests with the controller to study its performance in real flight scenarios. Thus, the formation controller is implemented on wireless enabled GumStix computers integrated onboard the UAVs to perform the coordinated formation flight. Flight experiments are conducted on two flying wing UAVs and the results verify the performance improvement from incorporating the synchronized position tracking strategy in the formation controller.

I. Introduction

Unmanned Aerial Vehicles (UAVs) have seen rapid development in a wide variety of areas,1 from surveillance2 to search and rescue missions3 to geographic studies4-6 and military and security applications7-9 (Fig. 1 illustrates an autonomous UAV performing a flight mission). In particular, formation flight control of multiple Unmanned Aircraft Vehicles (UAVs) has been an active research topic since it promises many practical applications.10-13

There are basically three approaches to the formation control problem for multiple vehicles, namely leader following, behavioral, and virtual structure.14-16 In the virtual structure
approach, the entire formation is treated as a single entity. It can evolve as a rigid body in a given direction with some given orientation and maintain the geometric relationship among multiple vehicles base on a reference point in the virtual structure.

To solve the more specific formation flight problem for aircraft, a consensus is that the vehicles should be able to achieve tracking for given velocity, heading, and altitude commands\textsuperscript{10,11,17,18} in order for a formation controller to be developed for the aircraft. The method of trajectory command modifications based on relative position errors has been used as part of the control algorithm to achieve formation flights of multiple aircraft.\textsuperscript{10} In,\textsuperscript{19} synchronization technology developed at UTIAS\textsuperscript{13,20} is incorporated in the formation controller base on trajectory modifications and simulation results show there is performance improvement when the synchronization control technique is employed.

In this paper, we present that the flight test results of the formation controller developed in\textsuperscript{19} on two flying wing UAVs. The controller is implemented on GumStix computer integrated onboard each vehicle for autonomous mission control. Vehicle-to-vehicle communication is enabled through ad-hoc network to achieve coordinated flights with the vehicles. Flight test results demonstrate the effectiveness and performance improvement with the proposed formation control method in actual flight scenarios.

The remainder of this paper is organized as follows. In Section II, the control algorithm and the synchronization strategy used to maintain vehicles in formation is defined,
followed by the description of hardware system to be integrated onboard the vehicles with the controlled implemented in Section III. In Section IV, the architecture for implementing the formation controller in code is provided. Actual flight tests set up and test results on formation control of multiple flying wings are given in Section V. Finally, Section VI offers conclusions and future research possibilities.

II. Formation Control Algorithm and Synchronization Strategy

In the virtual structure approach, the entire formation is treated as a rigid body. The inertial frame is defined as Frame $O$. The positions of the vehicles in the structure are usually defined in a formation frame $F$ with respect to a reference point $P_r$ in the structure. As a reference trajectory $T_r$ is given for the reference point, the desired position for each vehicle $P_i^d$ can be calculated as the virtual structure evolves in time, see Fig. 2. In our formation controller, we define the centroid of the desired formation as the reference point in the virtual structure.

![Figure 2. Coordinate frame of virtual structure](image)

The same reference trajectory $T_r$ for the virtual structure is given to each vehicle in the formation. As a result, the actual positions $P_i$ of the vehicles will deviate from the desired positions $P_i^d$. This results in relative position errors in the vehicles’ position from their desired positions, meaning that the formation of the vehicles are broken.
To eliminate the relative position errors and to keep the vehicles in formation geometry during flight, a formation controller is needed. It is typically implemented by a two loop scheme where the inner control allows tracking of commanded velocity ($V$), heading ($\psi$) and altitude ($H$). In the outer loop, the formation controller generates the reference commands for the inner controller. An autopilot that is capable of tracking velocity, altitude and heading is used in our paper, so we focus the development on the outer loop formation controller.

Base on the reference trajectory commands $T_r = [V_r, \psi_r, H_r]^T$ for the reference point $P_r = [x_r, y_r, z_r]^T$ and the defined relative distances in the virtual structure, the desired position for each vehicle during the flight $P_i^d = [x_i^d, y_i^d, z_i^d]^T$ can be calculated, whereas the actual position of each aircraft $P_i = [x_i, y_i, z_i]^T$ can be obtained from the GPS system located on the vehicle. Using the reference trajectory and the actual/desired positions for the vehicles as input, the formation controller generates the modified trajectories for each UAV to maintain the geometry of the formation during the flight. We define the relative position errors for the $i^{th}$ vehicle during the flight in the inertial frame as

$$ \begin{bmatrix} e_{xi} \\ e_{yi} \\ e_{zi} \end{bmatrix} = \begin{bmatrix} x_i^d - x_i \\ y_i^d - y_i \\ z_i^d - z_i \end{bmatrix} \quad (1) $$

To utilize these relative errors in (1), they need to be converted into errors in the formation frame using the rotation matrix $C_{FO}(t) = C_{OF}(t)^{-1}$. Therefore,

$$ \begin{bmatrix} e_{xiF} \\ e_{yiF} \\ e_{ziF} \end{bmatrix} = C_{FO}(t) \begin{bmatrix} e_{xi} \\ e_{yi} \\ e_{zi} \end{bmatrix} \quad (2) $$

The modified trajectory command for the inner loop controller is $T_{ci} = T_r + \Delta T_i$, where $\Delta T_i$ is calculated through a PI controller based on the relative position errors in (2).
\[ \Delta V_i(t) = K_{px} e_{xiF}(t) + K_{ix} \int_{0}^{t} e_{xiF}(t) \, dt \]
\[ \Delta \psi_i(t) = K_{py} e_{yiF}(t) + K_{iy} \int_{0}^{t} e_{yiF}(t) \, dt \]
\[ \Delta H_i(t) = K_{pz} e_{ziF}(t) + K_{iz} \int_{0}^{t} e_{ziF}(t) \, dt \]

(3)

\[ \Delta T_i = \begin{bmatrix} \Delta V_i(t) \\ \Delta \psi_i(t) \\ \Delta H_i(t) \end{bmatrix} \]

(4)

The formation controller applies the corrected trajectory commands \( T_{ci} \) based on \( T_i \) and \( \Delta T_i \) in (4) to the aircraft’s autopilots. These corrections account for the changes required to maintain the geometry of the formation.

As for the synchronization strategy\(^{13,20}\) developed, it uses the cross coupling concept to synchronize the relative position tracking motion of the aircraft. It utilizes synchronization errors \( \varepsilon_i \), which incorporate error information from different agents in the system, to identify the performance of the synchronization. The cross coupled error \( e_i^* \) then couples the error \( e_i \) and synchronization error \( \varepsilon_i \) through a positive synchronization gain \( \beta_i \).

\[ e_i^* = e_i + \beta_i \varepsilon_i \]

(5)

The objective of the synchronization strategy is to drive \( e_i^* \) of each agent in (5) to 0 by choosing the proper gain values, implying that both \( e_i \) and \( \varepsilon_i \) are driven to 0 as well. In other words, the vehicles are using information from each other to eliminate the errors synchronously.

In our case of two vehicles, the position synchronization errors are defined as
Then the coupled position errors are formed to include both the position tracking errors $e_i$ and the position synchronization errors $\varepsilon_i$ from (6)

\begin{align*}
\varepsilon_{x1F} &= e_{x1F} - e_{x2F}, \quad \varepsilon_{x2F} = e_{x2F} - e_{x1F} \\
\varepsilon_{y1F} &= e_{y1F} - e_{y2F}, \quad \varepsilon_{y2F} = e_{y2F} - e_{y1F} \\
\varepsilon_{z1F} &= e_{z1F} - e_{z2F}, \quad \varepsilon_{z2F} = e_{z2F} - e_{z1F}
\end{align*}

(6)

where $\beta_{xi}, \beta_{yi},$ and $\beta_{zi}$ in (7) are positive synchronization gains for the $x, y,$ and $z$ channels of the $i^{th}$ aircraft. The method of coupling the errors in (6) can vary when more than two vehicles are involved to couple errors from more than one aircraft, as introduced in.\textsuperscript{20}

The coupled relative position errors from (7) are used to calculate the trajectory modification $\Delta T_i^*$ and the new modified trajectory command that will be passed to the inner controller of the vehicles is $T_{ci}^* = T_r + \Delta T_i^*$ where

\begin{align*}
\Delta V_i^*(t) &= K_{pz}e_{x1F}^*(t) + K_{ix} \int_0^t e_{x1F}^*(t) dt \\
\Delta \psi_i^*(t) &= K_{py}e_{y1F}^*(t) + K_{iy} \int_0^t e_{y1F}^*(t) dt \\
\Delta Z_i^*(t) &= K_{pz}e_{z1F}^*(t) + K_{iz} \int_0^t e_{z1F}^*(t) dt
\end{align*}

(8)

\begin{align*}
\Delta T_i^* &= \begin{bmatrix} \Delta V_i^*(t) \\ \Delta \psi_i^*(t) \\ \Delta Z_i^*(t) \end{bmatrix}
\end{align*}

(9)

With the formation controller incorporated with the synchronization technique, simulations has shown there are performance improvements in formation flights with multiple
UAVs. Thus, it is desirable to conduct actual flight tests to study how the controller behaves in real scenarios.

III. GumStix Hardware

The GumStix computer used in this paper to develop the formation controller is a small form factor miniature computer. It is integrated to each UAV for controlling the onboard autopilot.

The GumStix used in this study consists the verdex XL6P motherboard that provides a Marvell XScale PXA270 processor running at 600MHz with 128MB of SDRAM. The GumStix embedded computer runs a full Linux environment with C/C++ library and a 802.11(b)/802.11(g) wireless expansion board (netwifimicroSD-vx) can be connected to the verdex motherboard through the 120 pin connector to enable wireless capabilities. The wireless board is included to enable communication with optional ground control station if necessary, and also to allow the possibility of incorporating additional vehicles for coordinated flights and missions when wireless vehicle-to-vehicle communication is required.

The code for the formation controller and drivers for sending commands to the autopilot is written in C++ on the GumStix. Fig. 3 shows the additional hardware that is to be added to the vehicle: the GumStix verdex motherboard and the wireless expansion board.

![verdex XL6P](image1.png) ![netwifimicroSD FCC](image2.png)

(a) GumStix verdex motherboard  (b) Wireless expansion board

Figure 3. GumStix hardware to control UAV autopilot

There are several advantages in using the GumStix as the high level control. The Gum-
Stix board has a small form factor that has minimal effect to the dynamics of the vehicle when it is integrated onboard. Also, the controller is implemented on the GumStix and since the GumStix can interface with the autopilot directly, information from the autopilot and commands from the controller can be parsed back and forth natively through hardware. The implementation of the controller on the GumStix onboard the vehicle itself eliminates the overhead and delays that are present when the controller is implemented on a ground station, where vehicle information is required to be transmitted to ground station and have the commands sent back to the vehicle in the same manner. Finally, the wireless board connected to the GumStix on each vehicle allows communication between the agents directly. This eliminates delays and noise by using a central gateway for communication between the vehicles, such as a ground control station.

The system architecture of the GumStix hardware implementation for the formation controller on the vehicles is illustrated in Fig. 4.

Figure 4. System architecture of implementation with GumStix
IV. Implementation of Formation Controller

To use the GumStix for implementing the formation controller to control the UAVs, the methods to interface with the UAV’s autopilot in both hardware and software mechanisms must be studied. First of all, for the hardware interface, the Procerus Kestrel autopilot onboard the vehicles allows serial connections from external onboard computer or microcontroller. Thus a wired connection is established from the autopilot programmable serial port to the serial pinout available on the GumStix hardware.

With the serial connection, data can then be sent between the GumStix and autopilot. The other requirement is to determine how the autopilot understands commands and what protocol is being used, which is the software interface to the autopilot. However, this information is often proprietary to the manufacturers of the autopilot system. Therefore, an agreement is reached in order to obtain and use the communication protocol from Procerus. Driver functions are written as a part of the development according to the specifications of the communication protocol. Using these low driver functions, high level commands generated from the formation controller are interpreted and communicate to the vehicle autopilot.

To achieve coordinated formation flight with multiple vehicles, vehicle-to-vehicle communication is desirable to eliminate overhead from communicating through a ground station. This communication link will be established by the additional GumStix hardware with wireless ethernet capability onboard each vehicle through the ad-hoc TCP/IP protocol. In this network mode, there is no centralized access point required and each mobile agent itself can be considered as an individual access point. It is a self-configuring network and the agents in the network are free to move randomly and organize themselves arbitrarily. This network scheme is especially suitable for cooperative missions of multiple vehicles due to its capability of connecting agents automatically and dynamically when they are in range.

With the interface to the autopilot and the communication scheme for coordinated flights set up, the software architecture to implement the formation controller is defined. Since the GumStix computer being integrated onto the vehicle has a linux environment with full C/C++ library, the flight control is then developed in C++.

The overall flight controller to control the vehicles autonomously is implemented using a multi-threaded architecture. The formation controller developed is considered a part in this flight controller. The essential parts of the controller are developed in separate threads for them to execute simultaneously with each other. There are three threads in the program.
The overall architecture of the implementation is shown in Fig. 5.

Figure 5. Multi threaded architecture of flight controller
The main thread executes the overall flight controller and perform high level decisions. It also initializes the serial and wireless communication threads. At the start of this main thread, the flight plan of the mission is uploaded into a linked list structure. This linked list is parsed and the appropriate commands are then sent to the autopilot at certain times in order for the vehicle to complete the mission. After the vehicles are launched, a rendezvous command is invoked by both vehicles to have the vehicles head to the starting point of the formation flight. When this point is reached by one of the UAVs, the formation controller starts its execution to keep the vehicles in formation.

V. Flight Test Setup and Results

The flight plan for the experiments can be divided into stages. A brief overview of each stage is outlined below.

1. Takeoff - The two UAVs are launched and commanded to head to a certain takeoff location, specified by longitude, latitude and altitude. The vehicles are launched one after the other when it is safe to do so.

2. Loiter - The UAVs are commanded to loiter at the position but at different altitudes after takeoff. This is to set up the rendezvous task.

3. Establish communication - At this stage, the UAVs establish communication through the ad-hoc network mode to share information.

4. Rendezvous - The rendezvous algorithm is run once communication is established. One vehicle will speed up until both UAVs have approximately the same heading during loiter. Both UAVs are then commanded to a common waypoint to begin formation flight.

5. Formation mission - Formation controller is enabled and the vehicles execute the algorithm to stay in formation, given the reference trajectory commands.

6. Landing - After the formation has run for a certain time period, both UAVs are commanded to land.

With the above stages of the flight plan, the mission profile for the two UAVs during the formation experiment can be illustrated in Fig. 6. The UAV mode for each vehicle is obtained from its autopilot and the value can be used to indicate what phase of the mission the vehicle is executing during the experiments.
The three scenarios (no formation, trajectory modifications and trajectory modifications with synchronization) are run in real flights to obtain data for analysis. Since there is no collision avoidance algorithm implemented in the controller, the UAVs are commanded to execute the formation flight with a separation to ensure they will not collide with each other. The desired relative distances from UAV1 and UAV2 to the reference point in the virtual structure are $[-4, -4, 10]^T m$ and $[4, 4, -10]^T m$ respectively. A velocity of $14 m/s$ and a heading rate of 0.05 radians/s (2.86°/s) are applied to the reference point with an initial South heading.

Fig. 7(a) and Fig. 7(b) illustrate the relative position errors and the trajectories of the aircraft and the reference point with no formation control. It can be seen that the formation flight is not achieved with relatively large relative position errors.

Formation control is then enabled for the UAVs based on trajectory modifications. Fig. 8(a) and Fig. 8(b) show the results with formation controller enabled (The error spikes observed in Fig. 8(a) are caused by incorrect position readings from the GPS unit, these errors are discarded during the formation control and are removed in the trajectory path calculations). It can be seen that the vehicles are now attempting to maintain formation during flight and the relative position errors are stabilizing around zero slowly, although
there are obvious differences between the errors from the two aircraft.

Finally, the synchronization technology is incorporated in the formation controller. In this case, results in Fig. 9(a) and Fig. 9(b) show that formation flight is again achieved between the two vehicles and the relative position errors in the virtual structure are now synchronized with each other. In comparison to Fig. 8(b), it is evident that the formation of the UAVs is being kept better.

A quantitative index that can be used to measure the performance of the controller is the root mean square (rms) value of the errors. The errors measured are the relative position errors of the vehicles in the $x$ and $y$ directions from their desired positions in the desired formation geometry. Using the relative position error data from the two vehicles, their error root mean square values ($E_{rms}$) in the $x$ and $y$ directions for the three simulation scenarios (No. 1, 2 and 3) are summarized into Table I and Table II respectively.

<table>
<thead>
<tr>
<th>No.</th>
<th>Formation control method</th>
<th>$E_{rms}$ in $x$</th>
<th>$E_{rms}$ in $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None (using reference commands)</td>
<td>123.5824m</td>
<td>90.8185m</td>
</tr>
<tr>
<td>2</td>
<td>Trajectory commands modification</td>
<td>35.3700m</td>
<td>18.8240m</td>
</tr>
<tr>
<td>3</td>
<td>No.2 with synchronization</td>
<td>39.2638m</td>
<td>14.3671m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>Formation control method</th>
<th>$E_{rms}$ in $x$</th>
<th>$E_{rms}$ in $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None (using reference commands)</td>
<td>29.4881m</td>
<td>27.9332m</td>
</tr>
<tr>
<td>2</td>
<td>Trajectory commands modification</td>
<td>53.2103m</td>
<td>28.2203m</td>
</tr>
<tr>
<td>3</td>
<td>No.2 with synchronization</td>
<td>47.2378m</td>
<td>18.9847m</td>
</tr>
</tbody>
</table>

The large $E_{rms}$ values for the two vehicles in the $x$ and $y$ directions with no formation algorithm corresponds to the formation being broken. When the formation controller is used, the $E_{rms}$ values in the $x$ and $y$ directions decrease by a significant amount. The decrease in the error rms values indicates that the vehicles are trying to maintain formation. The values further decrease when the synchronization strategy is incorporated. In general, there is approximately a 15% decrease in the $E_{rms}$ values. This demonstrates that the strategy
improves the performance of the formation controller in the demonstrated experiment. In other words, the vehicles maintain the virtual structure formation in a more “synchronized” pattern. However, one can see that the error values are still quite large with the formation controller enabled. This can be caused by a number of factors in the real flight scenarios. Flight conditions such as wind gusts will cause the under-actuated flying wings to respond poorly to tracking commands. Also, an average GPS error of 10m is present in the GPS unit onboard and the limiting bandwidth of 2Hz from GPS position update are also possible factors in causing the larger than expected errors. Nevertheless, the overall performance of the formation controller is improved when the synchronization technique is used.

VI. Conclusions and Future Work

In conclusion, by incorporating the motion synchronization control technique in the formation control algorithm that utilize the virtual structure approach, the formation is being kept better by the vehicles. Implementation of the controller on GumStix allows quick deployment and execution of the formation flights, with the vehicles capable of communicating with each other. Actual flight tests show performance improvements in the formation of two flying wing UAVs with the synchronization strategy in the particular experiment scenario. Addition experiments will be conducted to fine tune the formation controller to gather more actual flight data on the developed formation controller for investigation and comparisons.

Future work includes reconfiguration of the formation and also additional vehicles in the formation. Different coordinated missions can also be implemented on the GumStix to be executed with different types of aircraft, using the same implementation technique shown in this paper.

VII. Acknowledgments

This work is done in junction with Quanser Consulting Inc. The authors gratefully acknowledge Ernest Earon, Cameron Fulford and Rajibul Huq for their contributions.

References


Maslankik, J., “Polar remote sensing using an unpiloted aerial vehicle (UAV),” Seminar, ATOC7500, November 2002.


Kumar, R., Kabamba, P. T., and Hyland, D. C., “Controller design using adaptive random search for


Figure 7. Responses of 2 flying wings in actual flight experiments - No formation algorithm
Figure 8. Responses of 2 flying wings in actual flight experiments - Trajectory modification
Figure 9. Responses of 2 flying wings in actual flight experiments - Synchronization incorporated