J Intell Robot Syst (2010) 57:393–406
DOI 10.1007/s10846-009-9372-4

Multi-UAV Simulator Utilizing X-Plane

Richard Garcia · Laura Barnes

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 6 October 2009

Abstract This paper describes the development of a simulator for multiple Unmanned Aerial Vehicles (UAVs) utilizing the commercially available simulator X-Plane and Matlab. Coordinated control of unmanned systems is currently being researched for a wide range of applications, including search and rescue, convoy protection, and building clearing to name a few. Although coordination and control of Unmanned Ground Vehicles (UGVs) has been a heavily researched area, the extension towards controlling multiple UAVs has seen minimal attention. This lack of development is due to numerous issues including the difficulty in realistically modeling and simulating multiple UAVs. This work attempts to overcome these limitations by creating an environment that can simultaneously simulate multiple air vehicles as well as provide state data and control input for the individual vehicles using a heavily developed and commercially available flight simulator (X-Plane). This framework will allow researchers to study multi-UAV control algorithms using realistic unmanned and manned aircraft models in real-world modeled environments. Validation of the system's ability is shown through the demonstration of formation control algorithms implemented on four UAV helicopters with formation and navigation controllers built in Matlab/Simulink.

Keywords Unmanned air vehicles · Simulator · X-Plane

R. Garcia (✉)
Army Research Laboratory, Aberdeen Proving Ground, Aberdeen, MD 21005, USA
e-mail: richard.d.garcia@arl.army.mil

L. Barnes
Automation and Robotics Research Institute, University of Texas at Arlington,
Arlington, TX, USA

1 Introduction

The significant increase of UAVs can be attributed to many factors including their ability to extend situational awareness and remove the human from dangerous situations. In fact, there are currently over 950 UAVs operating in Iraq alone logging more than 14,000 h of operation per month [1]. Unfortunately, these vehicles act as a force divider where a job that usually takes one or two soldiers to complete alternatively takes ten soldiers to accomplish using a single UAV. This fact can be attributed to lack of autonomy, inefficient human–computer interfaces, and inabilities to coordinate efforts. This research attempts to help alleviate the latter of these issues by allowing the group of vehicles to act as one entity creating a force multiplier.

In practice UAVs are largely disconnected and act as distinct entities sharing little if any information between systems. This lack of cooperation and coordination creates a highly inefficient and possibly dangerous environment. By coordinating the control of multiple UAVs they can easily be tasked to create safe and optimal flight paths, work as groups to solve large complex task, and act as backup systems for failed vehicles which is estimated to occur every 80–110 h of operation [2].

The first step following the development of new and creative algorithms is to thoroughly test them. Initial testing for most types of vehicles today is handled through vigorous simulations. Unfortunately, very few simulators exist that are capable of accurately simulating multiple air vehicles in real-time. To date, software that is capable of simulating multiple air vehicles is typically developed and utilized in-house. These simulation systems typically provide either a small number of highly accurate vehicle models operating in a highly confined environment model [3–5] or provide simplified models that decrease the computational load on the simulator [6, 7]. Both types of simulation systems require the end user to develop vehicle models and very rarely include realistic world models capable of simulating outside effects such as wind, rain, thermals and microburst.

Developing vehicle and world models is a very intensive and time consuming task and many times replicates models that have already been designed for commercial off-the-shelf (COTS) simulators. In general, COTS simulators provide a database of both world and vehicle models that many times have decades of dedicated development and refinement. COTS simulators also have the advantage of highly refined graphical outputs, special optimizations for RAM and video adapters, and many times contain Federal Aviation Administration (FAA) certified versions.

Beyond being able to supply heavily developed and certified models, COTS simulators can provide a public median for allowing researchers to easily replicate work performed by other facilities. For example, this work focuses on the COTS simulator X-Plane. X-Plane has also been utilized for hardware in the loop testing using [8], UAS failure control in [9], and for heads up displays in [10]. By adapting a COTS simulator to support multi-vehicle simulations the integration of diverse research areas becomes more viable.

2 Simulator Hardware

At a high level, the simulation environment is a cluster of individual machines capable of supplying sufficient computational and graphical power to collectively simulate, control, and visualize multiple aerial vehicles. This is accomplished by allowing each individual machine to act as a vehicle. Each vehicle, or machine, is responsible for simulating its own dynamics and kinematics, collecting internal state date, communicating its state data with outside machines and implementing externally received control inputs, all of which are performed by X-Plane. This configuration directly lends itself to hardware-in-the-loop simulation where physical hardware, be it sensors or computational devices, are simply attached to or replace the specific machine.

Although infinite variations of hardware exist to configure a functional system, special attention must be given towards hardware to provide the desired results. More specifically, the hardware must meet the minimal criteria for operating the COTS simulator as well as communicating data in a rapid effective manner. For the purposes of this research, the minimal criterion was defined as the ability to graphically operate the simulator at a rate greater than or equal to 50 Hz. This criteria was selected because the utilized simulator's graphical rate is directly connected to state and control data I/O (Input/Outputs).

The hardware configuration for the simulation environment, as seen in Fig. 1, has the following characteristics:

- Simulation Machines:
  - Quantity: 7
  - Motherboard: Mini-ITX
  - CPU: 2.8 GHz Duo
  - Graphics: NVidia GeForce 7100
  - RAM: 4 Gb
- Control Machine:
  - Quantity: 1
  - Motherboard: Mini-ITX
  - CPU: 2.4 GHz Duo
  - Graphics: Intel Integrated
  - RAM: 2 Gb

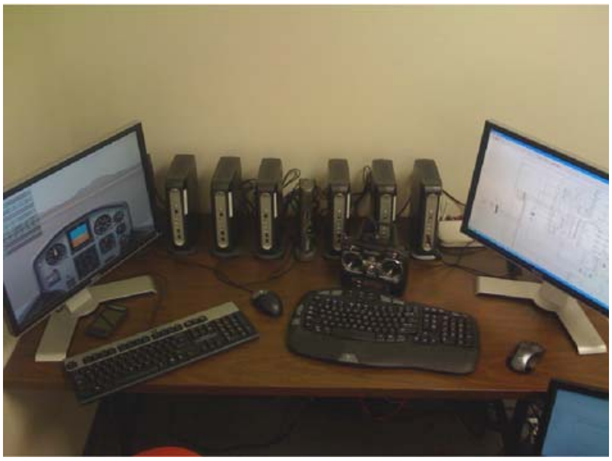
Fig. 1 Picture of the simulation environment

- Network:
  - Switch: 1,000 MHz
  - Router: 1,000 MHz

The rational for choosing this hardware configuration is multi-fold. First, the use of the mini-ITX motherboard provides the ability to utilize state of the art processors and maximize RAM size while providing a small footprint. Second, by utilizing a cluster of processing systems decentralized control can be directly implemented into the simulator. Third, the NVidia graphics adapter provides the required capabilities to sufficiently operate the simulator. It should be noted that the integrated Intel video adapter on the control machine was incapable of providing this requirement. Fourth, the high speed switch allows the system to maximize the number of simulated vehicles that can be controlled at once. This allows researchers to test the scalability of their algorithms, discussed later. Last, the low cost of the assembled environment, approximately $3,500 USD.

3 Simulator Software

3.1 X-Plane Simulator

The simulation environment is built around the COTS simulator X-Plane. The benefits of using X-Plane are:

- FAA Certifications: X-Plane can provide Federal Aviation Administration (FAA) certified simulation and vehicle models. This allows researchers to achieve high levels of confidence in simulation results.
- Operating Systems (OS) Flexibility: The X-Plane simulator is capable of running on Linux, Windows, or Macintosh OS.
- Database of Vehicle Models: Thousands of manned and unmanned vehicle models are freely available for download at multiple internet sites [11]. Although not necessarily certified, these models provide quick starting points for testing and can be easily modified using tools supplied with X-Plane. In addition, researchers can develop their own vehicle models for X-Plane.
- Highly Developed World Models: This simulator has been receiving constant refinements for over a decade. The world model used in X-Plane is able to simulate, cloud cover, rain, wind, thermals, microburst, and fog to name a few.
- Precedence of Success: X-Plane has been used as the basis for designing and testing control algorithms that have ultimately been proved viable on actual hardware in field experimentation [12].
- Visualization: X-Plane is designed to provide a visual collaboration of up to ten vehicles. This means that up to ten vehicles can be visualized simultaneously from a single display allowing for visual debugging and data collection to be performed without the need for extra software.

The main limitation of X-Plane, as with most COTS simulators, is the inability to run multiple instances of itself on a single machine. This limitation goes beyond simply not being able to start two identical processes simultaneously. This work originally attempted to bypass this limitation in an attempt to utilize as few machines as possible. Experiments were performed utilizing Linux and Windows OSs as well as OS emulators, VMWare, VirtualBox and Microsoft Virtual PC. All attempts to bypass this limitation failed for various reasons most of which were related directly to the graphics adapter or emulated graphics adapter.

To account for this limitation, the simulation environment is built around a cluster of small low cost machines. These machines are interconnected through a switch which allows them to pass state and control data to any or all machines on the network. This design gives rise to a possible bottleneck where data transmissions flood the network preventing adequate data transfer. This essentially becomes a scalability issue and is approached later in this section.

Due to the design of the simulation environment vehicle navigation and control can be distributed to multiple machines or performed on a single machine. For distributed type control, each individual machine could be responsible for its own control input or some subset of machines could collectively be calculating and distributing control to all simulators. In this work a single control machine is dedicated for calculating and distributing control inputs for all simulation machines. Specific details of the navigation and formation algorithms used to validate the simulation environment are presented in the following sections.

X-Plane's standard method for communicating with external processes and machines is User Datagram Protocol (UDP). UDP provides a minimal overhead communication method for passing data between nodes. Unlike Transmission Control Protocol (TCP), UDP is a non-guaranteed protocol and gives no assurances that data packets will arrive in order or at all. UDP is designed to minimize bandwidth usage but presents a possible problem resulting from corrupt data. Although this problem does exist and may need to be addressed given specific system designs, degradation of control and simulation has been unseen in X-Plane experimentations.

One issue that has yet to be mentioned is the scalability of the simulation environment. The two obvious bottlenecks for the described simulation environment are flooding of the network bandwidth and computational limitations of a centralized control machine. The computational ability of the centralized control machine is directly related to the method of control being tested. If the control machine is unable to perform the necessary calculations then the utilized method, or algorithms, may not scale well. This can be verified by using hardware-in-the-loop and should be considered experimentation all in itself. Ultimately, if the researcher is unconcerned with scalability on the current processor, the controlling machine can simply be upgraded to a machine capable of the desired computations.

The network bottleneck is ultimately a matter of understanding the hardware limitations as compared to the size and rate of messages passing through the network. Simply stated, as the number of simulated vehicles increases the available bandwidth on the network will decrease. By utilizing a network switch this simulation environment avoids packet collision and is available to take advantage of the hardware's full duplex communication median. Thus, the network scalability of the system is limited by the maximum inflow or outflow from any single node on the network. In the case of centralized control, the control machine will be the obvious bottleneck for communication as all simulation nodes will communicate directly with this one machine. Since both state data and control packets are identical in size, the number of vehicles that can be simulated without exceeding the network's bandwidth can be calculated using:

$$N_{sim} = \frac{R_{net}}{\left(P_{head} + N_{msg} * D_{size}\right) * F} \tag{1}$$

where R_{net} is the speed of the switch, F is the frequency of data packets and the UDP packet size is made up of the packet header, P_{head}, the size of an individually selected state data item, D_{size}, and the number of state data items selected, N_{msg}. Note that this calculation assumes optimal functionality of both the network and cluster. For the hardware described in the previous section using our method for controlling multiple helicopters, described in the next section, these values are: $R_{net} =$ 1,000 Mb, $P_{head} = 40$ b, $N_{msg} = 7$, $D_{size} = 288$ b, and $F = 50$. Using this data the maximum number of vehicles that can be simulated without exceeding bandwidth limitations is 9,727 vehicles. Realistically, control calculation time and minor delays in network traffic would reduce this number but will leave ample room for growth for large scale simulation.

It should be noted that X-Plane communication packets as well as data rates are user selectable and can vary from position and velocity data to carburetor temperatures and fuel pressure. Also, data rates cannot exceed the rending speed of the simulator. Thus, if the simulator is only rendering frames at 25 Hz then state data will be limited to 25 Hz. Rendering rates can be improved by limiting rending options in the simulator, decreasing visibility, and removing weather items such as cloud cover.

3.2 Control Software

Control software for the simulation environment is fundamentally unlimited. Any software capable of directly or indirectly receiving and transmitting UDP data can be seen as a viable option for implementing control algorithms. As such, the remainder of this section details the Matlab/Simulink model used to validate the simulation environment.

The Simulink model used to validate the simulation environment can be broken down into three major groups:

- Mission Controller: This module is designed to control the overall position of the group by selecting a setpoint in space from which the vehicles formation will originate. Advancement from setpoint to setpoint is determined by comparing each vehicle's position to its individual desired position.
- Formation Controller: This module is responsible for calculating the individual setpoints for each vehicle based on the desired formation parameters and setpoint provided by the mission controller. Individual setpoints include a desired altitude and heading as well as a 2-D vector representing a magnitude of error and a direction of travel.
- Vehicle Controller: The vehicle control module is responsible for the stabilization and navigation of each individual vehicle. As such, there will be a distinct vehicle control module for every simulated vehicle in X-Plane. This module includes the control algorithms, state data computation algorithms, and the X-Plane communication function.

These three major modules collectively make up the upper level of the Simulink model. A top-level view of a Simulink model for four vehicles can be seen in Fig. 2. It should be noted that the only identifier used to distinguish between individual vehicles is the port number, which is passed to each Vehicle Controller module. This identifier is only used by the X-Plane communication function and simply identifies the port number on which a specific simulator will be sending data.

4 Controller

4.1 Formation Controller

In order to describe the formation controller, suppose that the UAVs need to maintain an elliptical ring configuration at a fixed altitude, defined loosely by the ring's dimensions and the center of mass. Using a properly generated field, members of the swarm can be attracted to the loose elliptical ring. This formation can be described using a sequence of three concentric ellipses with center (x_c, y_c). Figure 3 depicts three elliptical rings with center (x_c, y_c). The ultimate goal is to attract swarm members to the center elliptical ring R^* described as the set of points (x, y) $\in \Re^2$ satisfying:

$$R^{*2} = (x - x_c)^2 + \gamma\,(y - y_c)^2 \tag{2}$$

where (x_c, y_c) is the center and γ is the axis ratio.

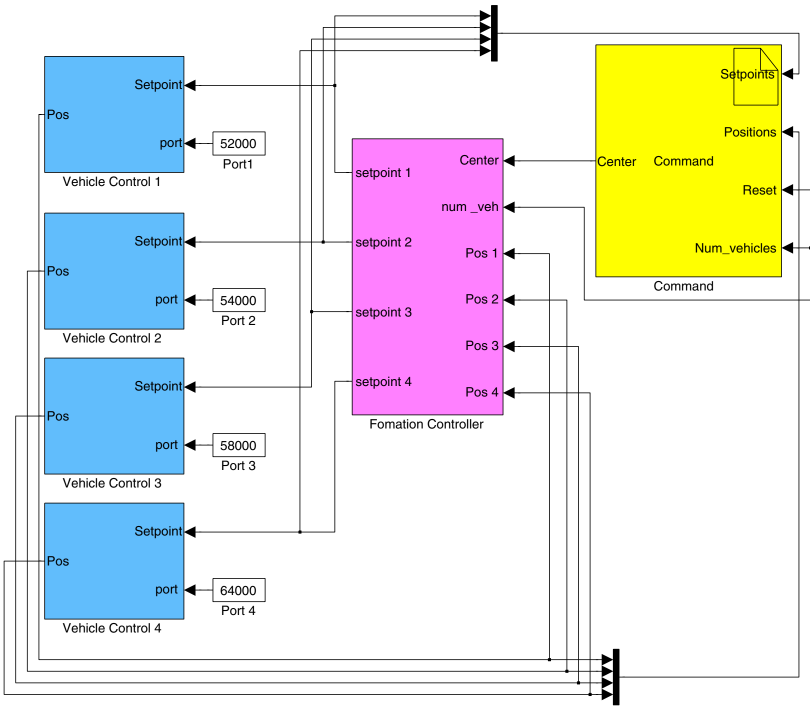

Fig. 2 Top-level of the simulink model

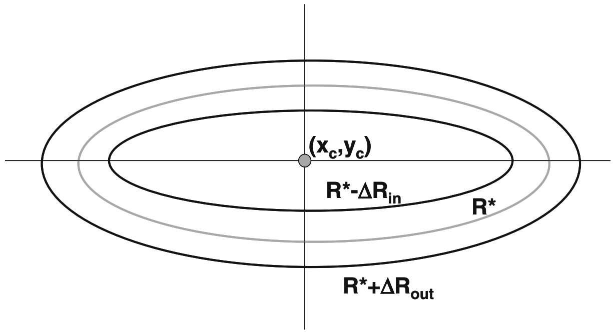

Fig. 3 Elliptical attraction band for the swarm

The general formation controller is described by:

$$V(x,y,t) = \sum_{1}^{N} w_i(x,y,t)\, V_i(x,y,t) \tag{3}$$

where $V(x,y,t)$ gives the velocity of the swarm at a particular time and place. Each of the vectors $V_i(x,y,t)$ is associated with different fields and $w_i(x,y,t)$ are weights of the overall contribution of the ith vector. In general, the field $V_i(x,y,t)$ is the weighted sum of N different vectors, each of which is acting on the swarm. In the case of this work, three different vector fields are utilized: one attracts UAVs to the elliptical band from points outside the elliptical region; one pushes UAVs away from the center towards the desired band; and one controls the movements of the UAVs within the band.

The potential field based controller uses a small number of physically relevant weights, w_i, and vectors v_i that attract UAVs to a neighborhood of the R^* ellipse. This neighborhood is shown in Fig. 3. The parameters R_{in} and R_{out} denote the inside and outside boundaries of the R^* neighborhood, respectively, as also depicted in Fig. 3. The desired vector fields will 'trap' the UAVs in these bands. Typically, this is a very narrow band of allowable space for the UAVs with a controllable width of $\Delta R_{in} + \Delta R_{out}$ where:

$$R_{in} = R^* - \Delta R_{in} \tag{4}$$

$$R_{out} = R^* + \Delta R_{out} \tag{5}$$

The vector field is constructed utilizing the normalized gradient. For every (x, y), the gradient field vector has the form:

$$V_i(x,y) = \begin{cases} W_i(x,y)\frac{1}{L(x,y)}\begin{pmatrix}(x-x_c)\\ \gamma(y-y_c)\end{pmatrix} & for\ (x,y) \neq (x_c,y_c) \\ \begin{pmatrix}0\\0\end{pmatrix} & for\ (x,y) = (x_c,y_c)\end{cases} \tag{6}$$

where:

$$L(x,y) = \sqrt{(x-x_c)^2 + \gamma^2 (y-y_c)^2} \tag{7}$$

The vector $\frac{1}{L(x,y)}\begin{pmatrix}(x-x_c)\\ \gamma\,(y-y_c)\end{pmatrix}$ is a unit vector that provides the direction of the vector at (x, y). The function $w(x, y)$ provides the magnitude of the vector at that point. Notice that for any (x, y), this vector points *away* from the center of the ellipse.

In the defined vector field, particles starting within the $R^* - \Delta R_{in}$ ellipse with:

$$R^* = \sqrt{(x-x_c)^2 + \gamma^2\,(y-y_c)^2} \tag{8}$$

move out from the center until they reach the R^* neighborhood. UAVs starting outside the $R^* + \Delta R_{out}$ ellipse move towards the center until they reach the R^* neighborhood. Eventually all UAVs will be trapped within the neighborhood given by:

$$\left(R^* - \ R_{in}\right) \leq r \leq \left(R^* + R_{out}\right) \tag{9}$$

Avoidance of individual swarm members including their dispersion about the formation is controlled by another weight, w_i, which limits how close UAVs are allowed to be to one another. This is a user-defined parameter, ΔR_{avoid}. The formation controller is described in further detail in [13–15].

4.2 Vehicle Controller

Stabilization and navigation of the individual vehicles is performed using a previously developed fuzzy logic controller. This was done to simplify the simulation environment's verification process by utilizing controllers that were already known to function properly and had thousands of hours of testing using the simulator X-Plane. These controllers are thoroughly detailed in [12] and therefore will only be summarized here.

Fuzzy logic was chosen to implement the helicopter controllers due to its inherent ability to handle minor errors, sensor noise, and contradictory inputs. Four distinct and uncoupled fuzzy controllers are utilized to both stabilize and navigate the helicopters. These individually control the roll, pitch, heading and collective of the vehicle. Throttle is assumed to be controlled by the simulator's throttle governor. The four fuzzy controllers utilize Sugeno constant fuzzy logic and a weighted average defuzzification method where the weight for all rules is equal to one. All rules for the controllers are based on the 'and' method and use membership products to determine the strength of each rule.

Although the fuzzy rule base was not modified from [12], there was minor editing performed on two input variables to allow the controllers to conform to desired multi-vehicle control. More specifically, the formation control algorithms were designed to provide vehicles with potential field vectors guiding them into formation. The fuzzy controllers described in [12] were not designed with this type of input in mind. Although this presented an initial issue, it was determined that the potential field vector could be seen as representing a unitless position error. Assuming that this unitless position error could be approximated in units of length provides the expected input into the fuzzy controllers.

This conversion is done by separating the X and Y vector components of the potential field vector, multiplying them by a constant and rotating them to the local coordinate system of the vehicle,

$$P_E = \left(-V_x{}^{*}\cos(-\psi) + V_y{}^{*}\sin(-\psi)\right){}^{*}c \tag{10}$$

and

$$R_E = (-V_x{}^{*} - \sin(-\psi) + V_y{}^{*}\cos(-\psi)){}^{*}c \tag{11}$$

where V_x and V_y are the magnitude of the potential field vector on the x and y axis respectively, ψ is the Euler yaw angle of the vehicle, and c is a constant used to adjust the magnitude of the conversion. The magnitude constant, c, is simply a way of linearly adjusting how much error a vector magnitude has. For the purposes of the work the magnitude constant was set to seven. This value allowed the small range of the potential vector to be converted to a positional error with seven times the range. It should be noted that increasing the magnitude constant increases the navigation control of the vehicle and decreases the vehicle's time to acquire a desired position. Due to the linear nature of the magnitude constant, an increase in its value also lead to a steady state sinusoidal error when the vehicle was near its desired position. This was deemed a limitation of the conversion equation and can be reduced by utilizing a non-linear equation to increase the range of the conversion. Although easily accomplished it was deemed unnecessary for this work.

5 Simulations

In order to demonstrate and validate the multi-UAV simulator, simulations were run with four simulated UAVs. The model RC UAV utilized for simulation was freely downloaded from the web and slightly tuned to mimic the Maxi Joker II electric helicopter.

5.1 Hovering Formation

In the first simulation, all UAVs initially start at ground level in random positions and are given a static center (x_c, y_c). The desired altitude is fixed as we are only concerned with formation at constant altitude for these simulations. Once the system is initialized the UAVs takeoff and immediately begin to surround the provided center point in an ellipse formation with the parameters given in Table 1.

Figure 4 depicts the X-Plane simulation environment with the four UAVs hovering in formation. In Fig. 5, the final positions of the UAVs in the formation are shown. Figure 6 plots the distance from the center of the formation over time. The UAVs remain at a consistent distance from (x_c, y_c) with minor perturbations due to

Table 1 Formation parameters

Formation	R^*	γ	ΔR_{in}	ΔR_{out}	ΔR_{avoid}
Ellipse	100	1.0	20	20	100
Flocking	50	1.0	–	–	20

Fig. 4 Screenshot of X-Plane during formation control experiment

untrimmed mechanical offsets and the low resolution of single floating point GPS coordinates, approximately 0.8 m [16]. Figure 7 shows the distance between UAVs. From Figs. 5 and 8, it can be seen that the UAVs are evenly distributed about the formation.

5.2 Multi-UAV Flocking

The flocking behavior was demonstrated by turning off the weights, w_i. Figure 8 demonstrates the swarm following a trajectory at different times. The UAVs follow

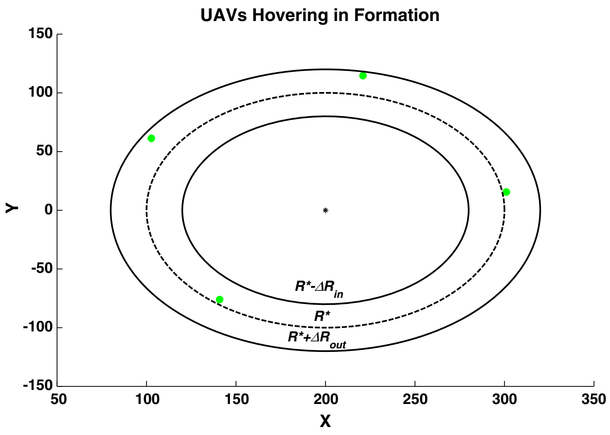

Fig. 5 Four UAVs hovering in formation

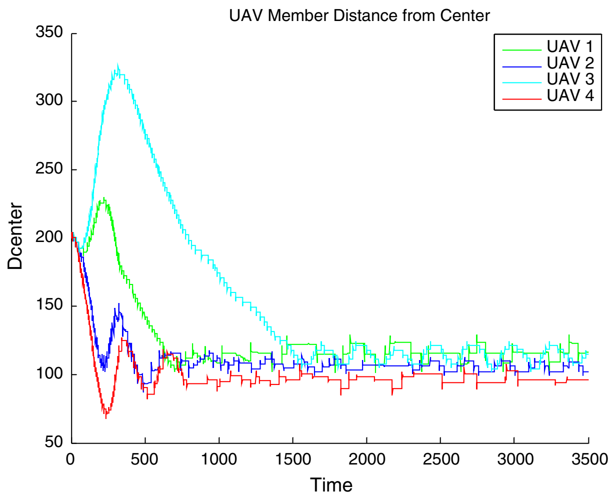

Fig. 6 Distance from formation center over time

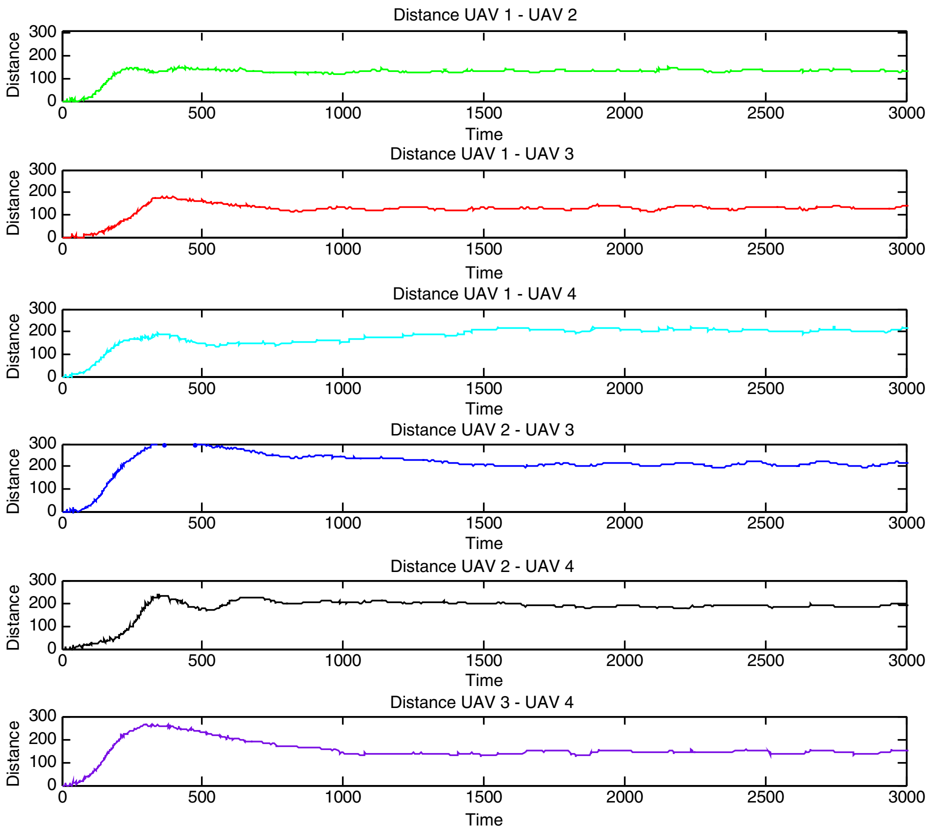

Fig. 7 Distance between UAVs

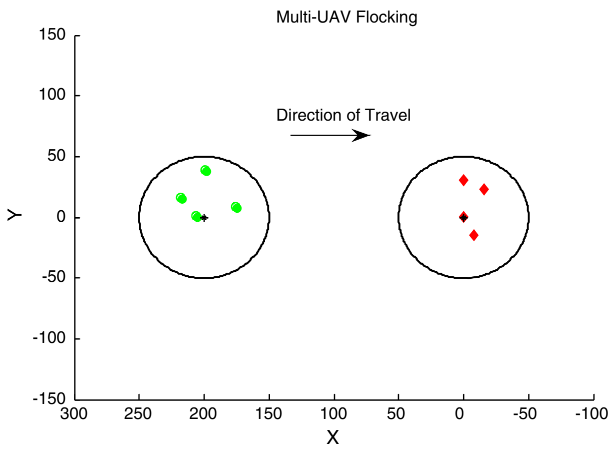

Fig. 8 Four UAVs demonstrating flocking behavior

the trajectory flocking inside the boundaries of the ellipse. The parameters utilized are shown in Table 1. Since we are not concerned with defining elliptical bands, only the R* denoting the ellipse of interest is defined. Figure 9 shows that UAVs stayed dispersed while traversing the trajectory.

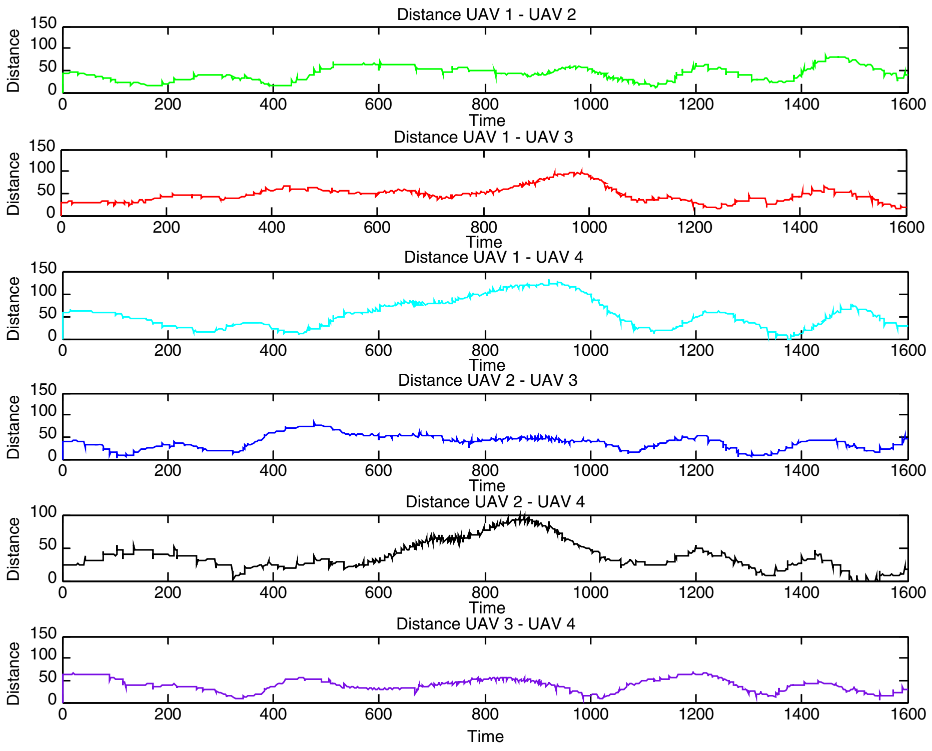

Fig. 9 Distance between UAVs with flocking behavior

6 Conclusions and Future Work

This research provides the insight and justification for utilizing COTS simulation products for multi-vehicle control and coordination. It also shows how this type of system design is fundamentally applicable to distributed control and hardware in the loop simulation. This idea is proven viable with currently available software and hardware. Overall, the main advantages of this type of simulation environment are: access to a database of thousands of vehicle models, access to highly developed world models, ease of development for new vehicle models, highly detailed real-time graphical representations, and simplified research replication/validation.

One item of immediate future work includes deriving a nonlinear vector conversion for the potential field vector into the fuzzy controllers. This update will allow navigation control to be increased without creating noticeable sinusoidal errors when the vehicle is very close to its desired position. Long term future work includes distributed control, 3-D adaptable formation control, failure tolerance and urban path identification for ground vehicle support.

References

1. Baldor, L.C.: U.S. use of UAVs in Iraq surges. In: Prompting Turf War, vol. 2009 (2007)
2. Cook, C.: Perspectives on Acquisition, Test, and Early Fielding of UAV Systems. Department of Operational Test and Evaluation (2008)
3. Kim, D.-M., Kim, D., Kim, J., Kim, N., Suk, J.: Development of near-real-time simulation environment for multiple UAVs. In: International Conference on Control, Automation and Systems, Seoul, Korea (2007)
4. Rasmussen, S.J., Chandler, P.R., Veridian, W.: MultiUAV: a multiple UAV simulation for investigation of cooperative control (2002)
5. Goktogan, A.H., Nettleton, E., Ridley, M., Sukkarieh, S.: Real time multi-UAV simulator. In: IEEE International Conference on Robotics and Automation (2003)
6. Beer, B.D., Lewis, M.: Lightweight UAV simulation for use in multi-agent human-in-the-loop experiments. In: Proceedings of the European Concurrent Engineering Conference, EUROSIS, pp. 51–56 (2007)
7. Xu, D., Borse, P., Grigsby, K., Nygard, K.E.: A petri net based software architecture for UAV simulation. In: Proceedings of Software Engineering Research and Practice (SERP04), pp. 227–232 (2004)
8. Ali, K., Carter, L.: Miniature-autopilot evaluation system. J. Comput. Sci. **4**, 30–35 (2008)
9. Garcia, R.D., Valavanis, K.P., Kandel, A.: Fuzzy logic based autonomous unmanned helicopter navigation with a tail rotor failure. In: 15th Mediterranean Conference on Control and Automation, Athens, Greece (2007)
10. Ertem, M.C.: An airborne synthetic vision system with HITS symbology using X-Plane for a head up display. In: Digital Avionics Systems Conference, DASC (2005)
11. "Downloads." vol. 2009: X-Plane.Org (2009)
12. Garcia, R.D.: Designing an autonomous helicopter testbed: from conception through implementation. In: Computer Science Engineering. vol. Ph.D., p. 305. University of South Florida, Tampa (2008)
13. Barnes, L., Fields, M.A., Valavanis, K.: Unmanned ground vehicle swarm formation control using potential fields. In: 15th Mediterranean Conference on Control and Automation, pp. 1–8 (2007)
14. Barnes, L., Fields, M.A., Valavanis, K.: Heterogeneous swarm formation control using bivariate normal functions to generate potential fields. International Transactions on Systems Science and Applications **2**, 346–359 (2007)
15. Barnes, L., Garcia, R., Fields, M.A., Valavanis, K.: Adaptable Formations utilizing heterogeneous unmanned systems. In: SPIE Defense and Security Conference (2009)
16. Reddy, M., Iverson, L.: GeoVRML 1.1 - Concepts. Available from: http://www.ai.sri.com/geovrml/1.1/doc/concepts.html (2002)