Presented at 46th AIAA Aerospace Sciences Meeting, Reno, NV
Jan., 2008, AIAA Paper No. 2008-0871

Intelligent Systems Software for Unmanned Air Vehicles

Gregory L. Sinsley[*], Lyle N. Long[†], Albert F. Niessner[‡], and Joseph F. Horn[§]

The Pennsylvania State University, University Park, PA 16802.

This paper describes a software architecture for mission-level control of autonomous unmanned air vehicles (UAVs). The architecture provides for sensor data fusion, worldview generation, and mission planning and execution. Details about the airborne platform and a high-level description of the control architecture are provided. As an example of the architecture's versatility a formation flight behavior is described.

I. Introduction

Unmanned Air Vehicles (UAVs) have great promise for both civilian and military applications. Civilian UAVs are used, for example, to find missing persons, fight forest fires, and for aerial photography. Military UAVs such as Global Hawk and Predator are used for both reconnaissance and combat missions. Unfortunately, most UAVs being used in the field today are primarily remotely piloted aircraft. In order to achieve their full potential UAVs need to be both intelligent and autonomous. Gottfredson[1] defines intelligence as follows:

> "Intelligence is a very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience."

According to Bekey:[2]

> "Autonomy refers to systems capable of operating in the real-world environment without any form of external control for extended periods of time."

By these definitions a system can be intelligent but not autonomous, or autonomous but not intelligent. In order to realize their full potential, however, UAVs must be both. Long et al.[3] have recently reviewed several software systems for autonomous vehicles. The conclusion of the study was that there are many possible software architectures for autonomous vehicles, each of which have their own positive and negative aspects. The ideal software system may very well be a hybrid of several systems, or a system yet to be designed. Hanford and Long[4] discuss how to evaluate cognitive architectures for unmanned vehicles and mobile robots. Also, Hanford et al[5] and Janrathitikarn and Long[6] describe the use of cognitive architectures for ground-based mobile robots.

Minsky[7] has proposed dividing intelligence into six levels (Figure 1). The lowest levels are instinctive reactions to environmental stimuli and learned reactions. These levels have been implemented in many machines. The middle levels of deliberative thinking and reflective thinking are much harder to do. At the highest levels are self-reflective thinking and self-conscious reflection. These levels are not seen anywhere, as making a self-aware machine is a very difficult task.

This paper focuses on the application of the Intelligent Controller (IC) designed at the Penn State University Applied Research Laboratory (ARL/PSU) to the control of UAVs. The controller is a behavior based architecture designed for the control of autonomous devices.[8] The architecture was initially based on

[*]Graduate Research Assistant, Electrical Engineering, AIAA Student Member.
[†]Distinguished Professor, Aerospace Engineering, AIAA Fellow.
[‡]Senior Research Associate, Emeritus, Applied Research Laboratory.
[§]Associate Professor, Aerospace Engineering, AIAA Associate Fellow.

the subsumption approach,[9,10] but actual system needs presented additional requirements. This resulted in a modified approach to intelligent control architecture.[11,12] The system has been used to control many different classes of vehicles including autonomous underwater vehicles, autonomous ground vehicles, and unmanned air vehicles.[13–15]

The first section of this paper provides an overview of the ARL/PSU IC, including details of how it has been applied to UAVs. The second section describes the UAV testbed that has been developed at Penn State to demonstrate the IC in collaborative missions. The main section of this paper describes the newest application of the Intelligent Controller, formation flight. The formation flight behavior is based on a simple leader-follower scheme. This simple scheme can be expanded to a more useful and robust mission by utilizing the intelligent capabilities of the IC.

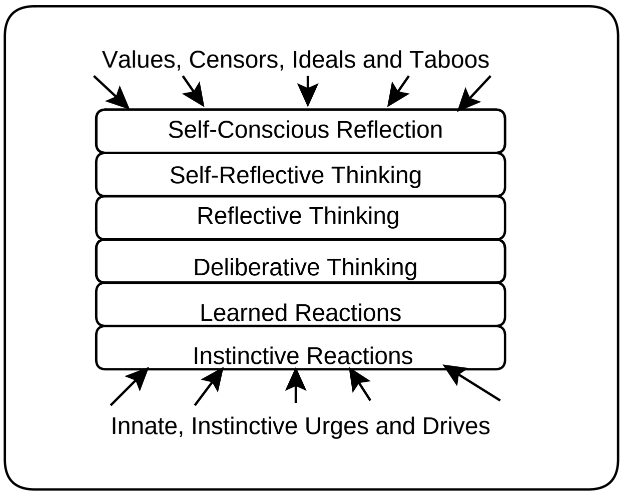

Figure 1. Minsky's Six Levels of Intelligence[7]

II. Intelligent Controller Architecture

The Intelligent Controller (IC) is responsible for mission-level control of each autonomous device participating in an independent or collaborative operation. The participating autonomous devices may be any combination of UAVs, UGVs, and UWVs. The IC has two main modules, Perception and Response. The Perception module of the IC forms an internal representation of the external world. The Response module uses this world-view to plan and carry out the IC's mission. Figure 2 shows a high-level overview of the IC architecture and its environment. The IC receives real world data from the sensors and payload inputs. The IC operates with a control loop where the Perception module processes the input data and the Response module acts on the results generated by the Perception module. Actions generated by the Response module are sent as messages to the system control components. This cycle continues until the program ends. The Perception and Response modules of the IC architecture are described in detail below.

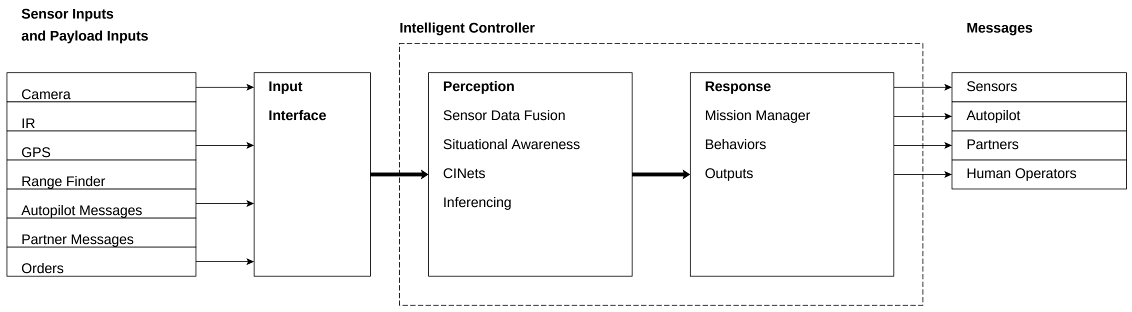

Figure 2. High level overview of Intelligent Controller

II.A. Perception

The IC Perception module is responsible for creating an internal representation of the real world. This world view is made up of many Representational Classes (RCs). An RC is simply a C++ object that represents either an object in the real world, or some property of the environment. For example, a Self RC stores information about the vehicle the IC is running in (as received from the autopilot). A Partner RC stores similar data about other ICs as received through the wireless network. A Target RC stores data about a potential target, as received from onboard sensors or from partners.

The Perception module makes inferences about real world properties through the use of Continuous Inference Networks (CINets).[11,16,17] CINets infer properties from sensor data that is often incomplete and potentially erroneous. CINets use weighted fuzzy AND and OR nodes, but can also have pre-trained neural networks as nodes. Fuzzy logic is used as opposed to binary logic because real-world date is continuous (not binary) and noisy. Figure 3 is an example CINet for obstacle avoidance. The output is a confidence factor between zero and one representing the confidence that an object (as represented by a Representational Class) is an obstacle. The simplest fuzzy AND is a MIN function, but more complex functions (which often include weights) can be used. The circles represent blend functions which normalize sensor data so that it will vary between zero and one. The CINet can be read from right to left as follows: "an object is an obstacle if it is large-enough (radius), close-enough (range), and in-my-path (angle)". CINets can be as complex as the application requires. They can include many nodes, which can be fuzzy AND, fuzzy OR, fuzzy NOT, or neural networks.

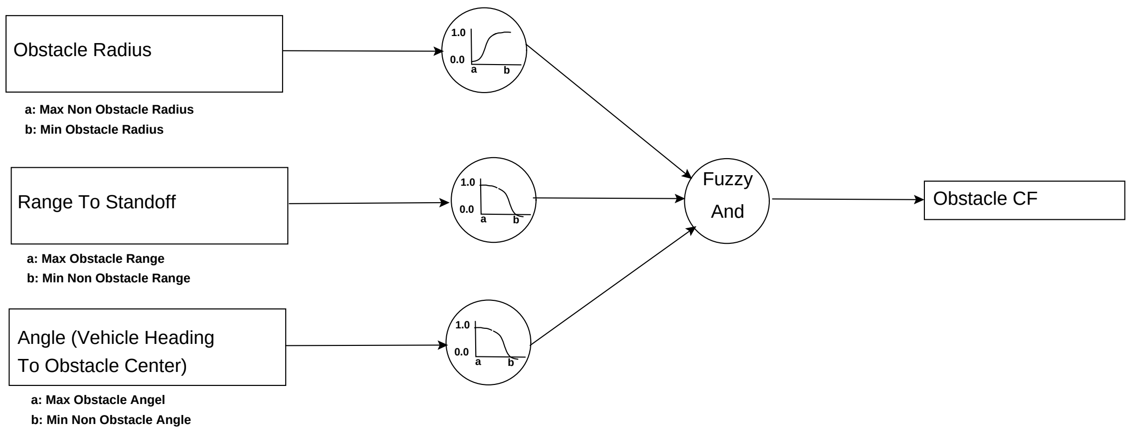

Figure 3. Obstacle Avoidance CINet

Figure 4 shows another example CINet. This CINet is used for data fusion purposes. It correlates an existing track (an RC that represents a real world object that has been detected by one or more sensors) with a new sensor report. It outputs a confidence factor representing the correlation between the two RCs. If there is a high correlation, the two RCs are assumed to represent the same object and are merged (i.e. their data is combined). If the correlation is low, they represent different objects and a new track is created to represent the new object. The CINet is read as follows: "the tracks correlate if the change in size is small AND (the change in x position is small AND the change in y position is small)".

An Input Interface module receives all data streams external to the IC and converts it into forms required by the Perception module. Data streams can come over the wireless network via TCP/IP packets, or from the serial port of the onboard Ampro ReadyBoard computer. It should be noted that the IC expects signal data to be preprocessed by sensors. Preprocessing includes noise filtering, Fourier transforms, corner detection, etc. The Input Interface accumulates data in a buffer and releases it to the Perception Module at discrete intervals called *processing cycles.* A processing cycle can be defined by the amount of time it takes an effector to complete a command and return data to the IC, or by a timer. The processing cycle is typically on the order of one second, the human control frequency.

II.B. Response

The Response module is responsible for real-time planning and execution of a given mission. The Response module has a Mission Manager, which can either load a mission at startup or receive a new mission on the

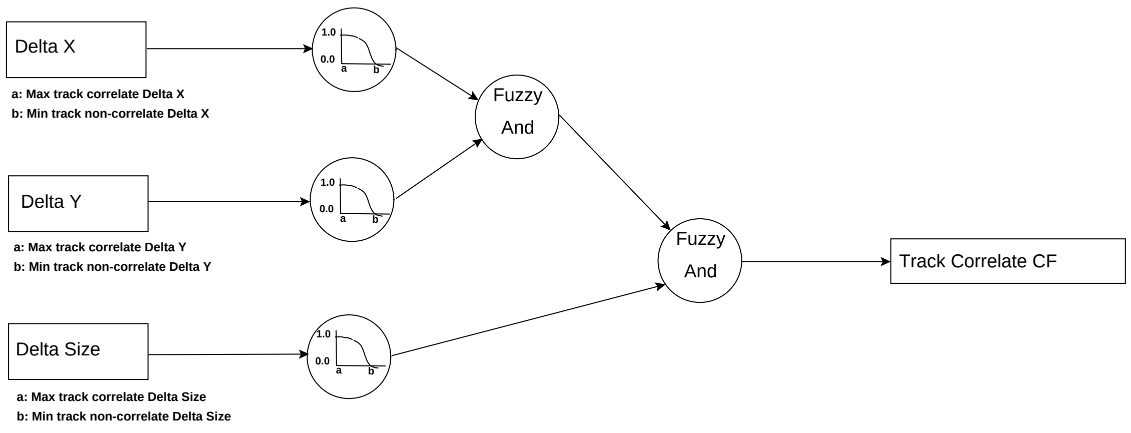
Delta X
a: Max track correlate Delta X
b: Min track non-correlate Delta X
Delta Y
a: Max track correlate Delta Y
b: Min track non-correlate Delta Y
Delta Size
a: Max track correlate Delta Size
b: Min track non-correlate Delta Size
1.0
0.0
a
b
Fuzzy And
Fuzzy And
Track Correlate CF

Figure 4. Track Correlation CINet

fly. The Mission Manager's job is to arbitrate between various Behaviors which may be requesting control based on the current mission. For instance, in a reconnaissance mission an "attack" behavior would have a relatively low priority, whereas in a combat mission it would have a higher priority. A "collision avoidance" behavior would always have a very high priority.

The Response module is composed of a number of independent Behaviors. Behaviors do not need to know the existence of other Behaviors and make their plans completely independent of other Behaviors. The reason such a modular structure was chosen was to make adding and removing Behaviors very simple. Utilizing object oriented programming also adds to this modularity. Each Behavior monitors the Perception module for objects that are of interest to it. For instance, if there is a high confidence (as inferred by CINets) that an object is a "target" the "attack" behavior will request to be enabled. The Mission Manager will choose whether or not to activate the Behavior based on the mission. The Mission Manager may also order a Behavior to take control. (Such as when a human sends an order to the mission manager or when the mission plan details that a certain Behavior should take control at a certain point in time.) An active Behavior generates a plan of action, then monitors and adapts the plan as it progresses. Multiple Behaviors may be active if they do not conflict with one another.

Actions represent the lowest level of control in the Response module. Each Behavior activates a specific set of Actions that should be executed in order to carry out its plan. An Action can send commands to the autopilot, commands to sensors for configuration or activation, or messages to other ICs or humans in the team. Example Actions include "Send Waypoints", "Send Message", and "Control Camera".

The IC is certainly autonomous by Bekey's definition;[2] it facilitates operation without the need for any human input. Its level of intelligence is more difficult to quantify. It definitely meets Minsky's first level (instinctive reactions),[7] because it can perform preprogrammed responses to various stimuli. The second level (learned reactions) occurs when the controller learns more about its environment (through Perception), and makes appropriate responses. It also occurs when Behaviors use adaptive controllers. Minsky's third level (deliberative thinking) requires the machine to evaluate several possible alternatives and choose the best one. At the Behavior level, this means a Behavior can choose one possible plan among several. At the Mission Manager level this means the Mission Manager would choose the Behavior that can best aid in accomplishing the current mission. This is a difficult problem because, when there are many choices to be made (especially decisions that will influence other decisions) the problems can quickly become intractable. For this reason deliberative thinking is only implemented for small well-defined problems. The higher levels (reflective thinking, self-reflective thinking, and self-conscious reflection) would be extremely difficult to implemented.

III. UAV Testbed

This section describes the testbed used in our airborne experiments. It is has two major components, the ground station and the airborne component. The ground station allows humans to interact with the aircraft. The airborne component has a single UAV or multiple UAVs and their subcomponents. Important subcomponents include the airframe, an autopilot for low level stabilization and control, an onboard computer that houses the IC and other onboard processes, and a number of different sensors. Each of these components is described in detail below.

III.A. Ground Station

Humans on the ground interact with the aircraft using the ground station. The ground station displays data on each of the aircraft. The ground station can also be used to send commands to the Intelligent Controller or the autopilot or to take manual control of the aircraft, if necessary.

The main component of the ground station is a laptop computer. It runs the Intelligent Controller ground program. The ground program displays relevant data from each Intelligent Controller in the team. The ground program lets users on the ground start a UAV's IC, or command the IC to go into standby mode. The IC is capable of fully autonomous operation, and once a mission file is loaded, no human interactions are necessary. The ground program does, however, allow humans to enter commands that will temporarily override an Intelligent Controller's current mission, or command the IC to upload a completely new mission. The ground program can also be used to simulate items such as sensor data or communications from other ICs for testing purposes. The ground program communicates with the ICs through an 802.11b ad-hoc network.

The laptop also runs an operator interface program from Cloud Cap Technology (CCT). The operator interface shows users on the ground the state of each aircraft (current flightpath waypoint, GPS location, velocity, altitude, attitude, and sensor health). It also allows users to send commands to each aircraft's autopilot (turn rate, velocity, altitude, or waypoint). The operator interface communicates with aircraft through the CCT Base Station, which is described below.

The CCT Base Station, along with the Operator Interface, and a Pilot Console are part of Cloud Cap Technology's Evaluation Kit. The Base Station connects to the laptop via a serial port. It communicates with each aircraft's autopilot through a 900 MHz radio link. It also has a GPS antenna so that the GPS location of the Ground Station is displayed on the operator interface along with the locations of each UAV. The Base Station also connects to the Pilot Console.

The Pilot Console is a standard radio control (R/C) transmitter modified to interface with the CCT Base Station. The Pilot Console can manually take control of one of the aircraft. It is primarily used for taking off and landing the aircraft, but it can also be used in emergency situations.

III.B. Aircraft

The airborne platform utilizes a heavily modified R/C trainer aircraft, the SIG Kadet Senior (Figure 5). The Kadet was chosen due to its stable flight characteristics, slow flying speed, and good payload capacity. The specifications of our aircraft are shown in Table 1.

The flight control surfaces are driven by standard Futaba servos. Electric power for the flight controls is provided by a Nickel Metal Hydride (NiMH) battery pack. The aircraft required several modifications in order to accommodate all the necessary sensors, and the additional weight that it has to carry. These include the following:

1. Increased fuel capacity for extended flight times (up to one hour)
2. Installation of heavy duty main and nose gear to support the heavy payload
3. Movement of the servos out of the central area of the fuselage to create an open fuel and payload space
4. Installation of the autopilot and control processors
5. Installation of pitot static tube mount
6. Installation of the GPS and communications antennas
7. Installation of a larger engine to accommodate higher payloads

In addition to the SIG Kadet Senior, there is a larger UAV based on a Bruce Tharpe Engineering Super Flyin' King under development. The Super Flyin' King is much larger than the Kadet, which provides for greater payload capacity. The additional payload will be used for new sensors including an infrared camera and a laser range finder. The Super Flyin' King will utilize electric propulsion to reduce the vibration compared to the equivalent gas engines. The reduced vibration will enhance the resolution of the onboard cameras. The motors are also mounted on the wing to provide the maximum forward fuselage space for the onboard sensors. In particular, this design facilitates using a forward looking camera in the nose. Figure 6 shows a picture of the Kadet and the Super Flyin' King side by side. Table 2 list specifications of the Super Flyin' King.

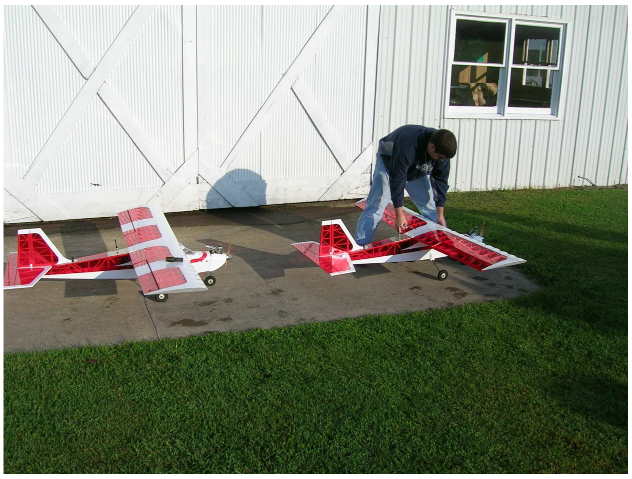

Figure 5. SIG Kadet UAVs

Table 1. SIG Kadet Specifications

Wingspan	80 inches
Wing Area	1180 sq. inches
Length	64 3/4 inches
Empty Weight	6 1/2 pounds
Gross Weight	14 pounds
Wing Loading	1.7 pounds/ft^2
Engine	0.91 cubic inch 4-stroke
Total Cost	\$11,700

III.C. Autopilot

Our UAVs utilize a Piccolo Plus autopilot that is commercially available from Cloud Cap Technologies. The autopilot is used for flight stability and low level control. It is housed in a mount that provides shock and vibration isolation. The mount is located near the center of gravity of the aircraft. The Autopilot connects to a GPS antenna in order to receive position information. It also communicates to the ground station via a 900 MHz radio link. It houses both inertial and air data sensors, and provides pulse width modulated servo outputs.

III.D. Airborne Processor

The airborne processor is an Ampro ReadyBoard 800 Single Board Computer. The unit contains a 1.4 GHz Intel Pentium Processor running Microsoft Windows XP. The processor runs the Intelligent Controller software, which is responsible for high level mission control. It also runs software for image processing and path planning.[18] It interfaces with the autopilot by an RS232 serial port and with other Intelligent Controllers or the Ground program via an 802.11b wireless card. It also interfaces with a Logitech Webcam through a USB port.

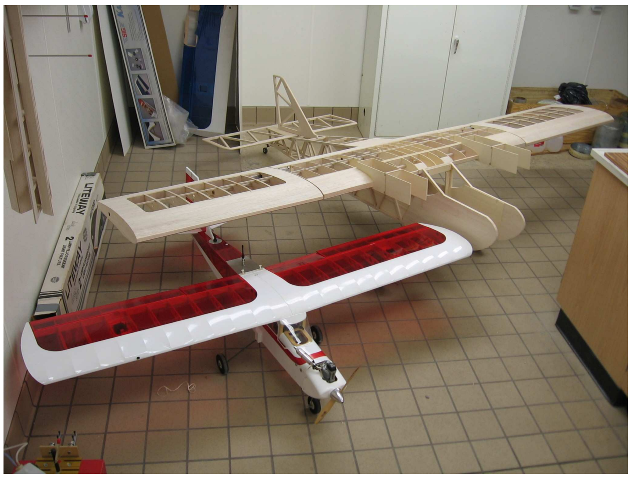

Figure 6. SIG Kadet Sr. and BTE Super Flyin' King

Table 2. BTE Super Flyin' King Specifications

Wingspan	132 inches
Wing Area	3380 sq. inches
Length	95 inches
Empty Weight	36 pounds
Payload	20 pounds (est.)
Total Cost	$16,400

III.E. Sensors and Other Payload

As described above, the autopilot contains inertial and air data sensor required for stabilization and control as well as a GPS antenna for navigation. The airborne processor runs a program that performs image processing from a USB webcam. Instead of a USB webcam, a Cannon Digital Camera can be used. It provides higher fidelity images, but they can not be analyzed in real time. The digital camera is controlled via a servo. Other payload items include an extra large fuel tank (32 oz.) and an 11.1V 4Ahr Lithium Polymer battery pack, which provides power to the airborne processor and the Piccolo Plus Autopilot.

III.F. Current Capabilities

Many autonomous capabilities of the UAV system have already been demonstrated. Sinsley et al[15] have demonstrated collaborative capabilities, including an investigation involving two UAVs. Geiger et al have demonstrated real time path planning[18] for photographing static and moving targets. Figure 7 shows the path a UAV followed in order to keep a static target in its view for a maximum amount of time. Notice that due to the downward looking camera the optimal path is a cloverleaf shape. Figure 8 shows the UAV tracking a moving target following a figure-eight path. In this case the moving target is another UAV, but it could just as easily be a person or a car. The path planner varies its path depending on how the target's speed compares to its own speed, and can take road information into account when planning it's path. The UAVs also have onboard image processing capabilities, including edge and corner detection, color filtering, and target geo-referencing. Sinsley et al[19] discuses image processing capabilities, and how they relate to problems in Earth Observation.

IV. Formation Flight

There are several different approaches to formation flight. Leader-follower[20] is an approach where follower vehicles reference their position to a leader vehicle. There can also be a hierarchy, where a number of group

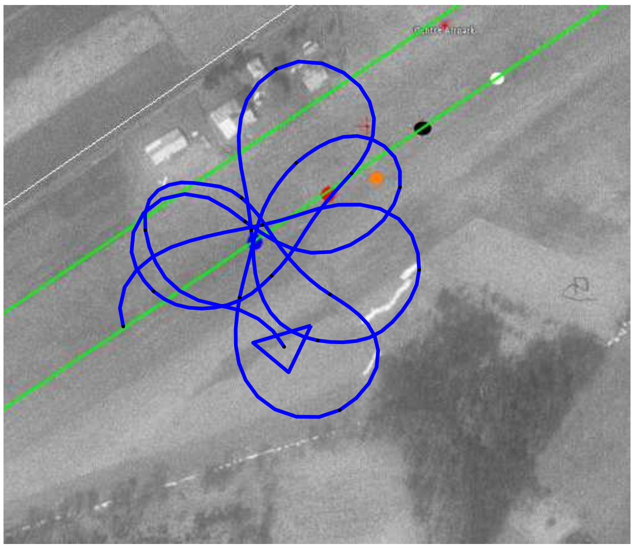
Figure 7. Path for photographing a still target

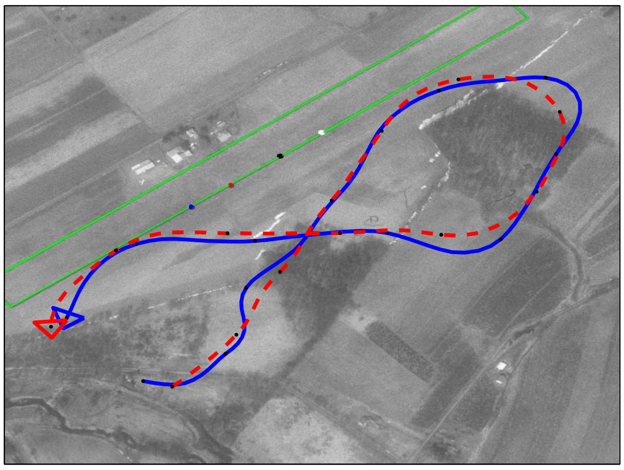
Figure 8. Path for photographing a moving target

leaders follow a master leader, and all other vehicles follow a group leader. Ren[21] proposes a consensus based approach, where vehicles use a consensus algorithm based upon graph theory in order to arrive at the correct position.

Balch and Arkin[22] describe a behavior-based approach to formation control. Although their approach uses similar terminology to ours, it is very different. Their behaviors are primarily reactive, meaning they make immediate decisions based on incoming data. Our behaviors have deliberative capabilities, meaning that they generate plans that are carried out over time. Also, in Balch and Arkin a formation behavior is made up of several motor schema such as "move-to-goal", "avoid-static-obstacle", "avoid-robot" and "maintain-formation". In our approach, a Behavior does not depend on other Behaviors.

The approach that was chosen in this paper was a leader-follower approach. The Follower Behavior is implemented as a single Behavior that will control the vehicle when it is following a leader. Leader-follower approaches suffer from some drawbacks, the worst of which is the fact that if a leader is disabled, the formation will break up (at least until a new leader is appointed). Nevertheless, the leader-follower approach does realistically represent several real-world missions. One is a mission where a manned aircraft will fly with several UAV "wingmen". The UAV's can then do dangerous missions such as taking out air defenses in order to protect the human pilot. Also in the mid air refueling problem a refueling aircraft will fly in reference to a tanker aircraft.

The formation flight approach to following a leader is in contrast to an optimal path planning approach to following a leader, as in Geiger et al.[18] In the path planning approach there is some objective function that must be minimized or maximized. In the case of Geiger et al the objective is to maximize the time that the leader is in the follower's camera field of view. This often leads to a zig-zag type of path. In a formation flight approach it is desirable for the follower to follow the same path as the leader and stay a constant distance behind the leader. The great advantage of using the IC, however is that both Behaviors can be implemented and the Mission Manager can choose which one to use depending on the situation. Also, it makes no difference to the Behaviors whether the leader's position is found using GPS, a camera, a range finder, or some combination of these and other sensors. All this work goes on in the Perception module, and the Behavior just sees the final world view. The current leader-follower development, where only the leader GPS location is used for tracking, is described in the next section.

IV.A. Basic Follower Operation

The essential elements of the leader-follower program are shown in Figure 9, a flow chart of the Follower Behavior's operation

IV.A.1. Perception

The Follower Behavior utilizes GPS data from the Perception module. A Self RC stores data relevant to the current aircraft, as received from the autopilot. Of interest to the Follower Behavior are the current GPS position, velocity, and altitude. Also in the Perception module there will be a Partner RC for each partner UAV in a team. These will contain GPS position, velocity, and altitude for each partner, as received from the wireless network.

IV.A.2. Response

In the Response module, there is a Behavior called "Follower". The basic Follower Behavior is activated when it is ordered to do so by the Mission Manager. (This is usually as a result of an order from a human in the team, although the mission may also dictate that the Follower Behavior be activated.) A Follower Order will contain several parameters: the appropriate leader, and the desired location in respect to the leader's GPS antenna. The location will be a distance behind the leader to fly, an offset to the right or left of the leader's path, and an altitude separation.

When the Follower Behavior is active it will generate and monitor a plan to appropriately follow the leader. The first aspect of the plan is the waypoints to fly. Waypoints consist of a latitude, longitude, and altitude. The altitude is based on adding an offset (as found in the order) to the leader's altitude. In the simple case where the follower's path is not to be offset from the leader's path, the latitude and longitude are based on the leader's previous positions. So the follower may have 5 waypoints, 3 based on the leader's past 3 locations, 1 based on the leader's current location, and 1 based on the leader's future location. In the more complex problem where the follower's path is to be offset to the left or right of the leader's, the waypoints will be based on the leader's past locations plus an offset in the direction normal to the leader's velocity. When the Follower Behavior has generated a list of waypoints to fly to, it will use its Send Waypoints Action to send the waypoints to the autopilot. The autopilot will then proceed to visit each of these waypoints in order.

The follower aircraft must also maintain a constant distance behind the leader. In order to accomplish this, the aircraft must fly faster until it catches the leader, and then slow down to match the leader's speed. This is implemented through proportional, integral, derivative (PID) feedback control. The Follower Order contains a set distance as a parameter. The actual distance behind the leader is calculated by finding the distance the follower aircraft must fly in order to visit each waypoint. This is used instead of just finding the linear distance to the leader, because the path may be curved. The difference between the desired distance and the actual distance is the error. The error, the integral in error, and the derivative of the error are all multiplied by constants and added together in order to find how much the speed should be varied in order to drive the error to zero. Once this new speed is determined, the Follower Behavior uses the Send Message Action to send a velocity command to the autopilot. The autopilot then matches the aircraft's speed to the set speed.

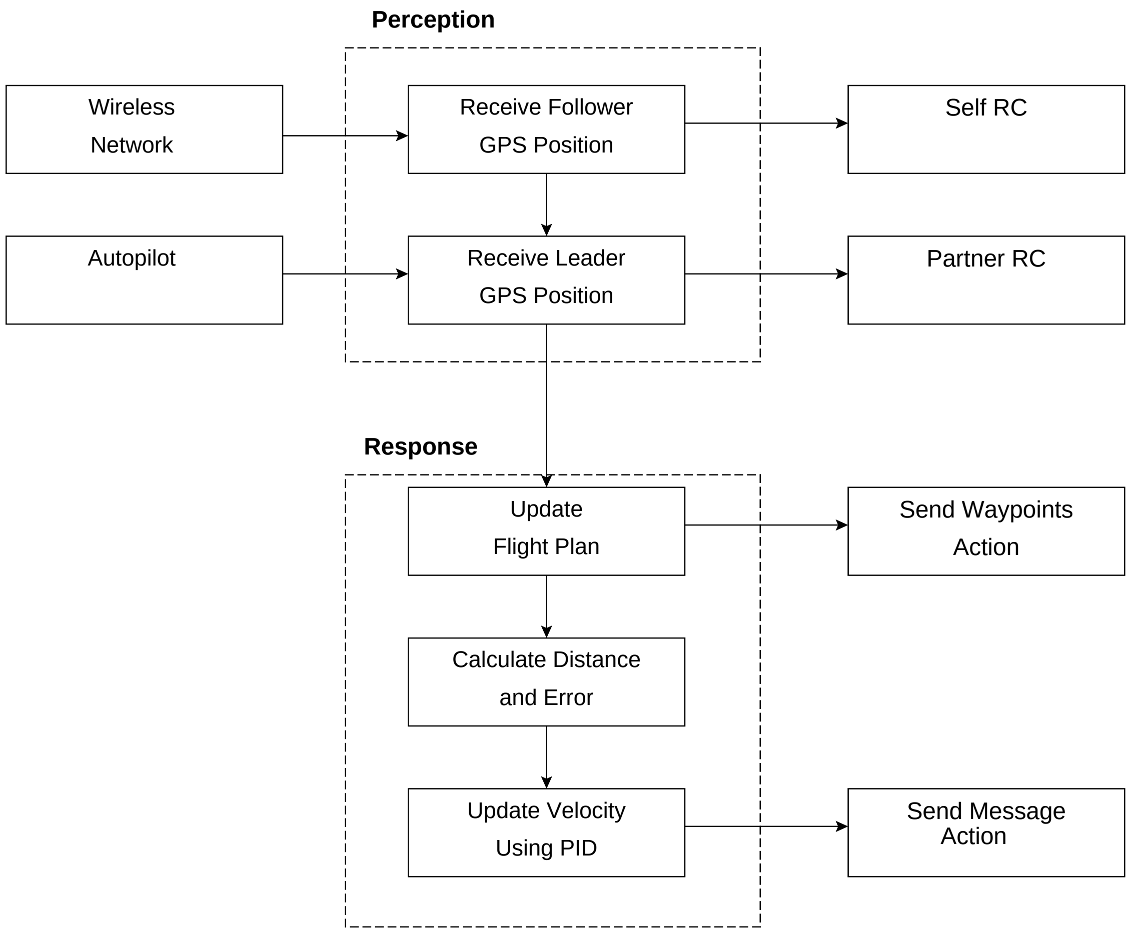

Figure 9. Flow chart of the Follower Behavior's operation

IV.B. Extensions to the Basic Follower

There are many possible extensions to the basic Follower Behavior described above. The basic Follower Behavior is primarily for demonstration purposes and does not utilize the IC's full potential. The extensions described below utilize more of the IC's power, and demonstrate some more realistic missions.

IV.B.1. Incorporating Additional Sensor Data

As mentioned above, one of the Perception module's purposes is to fuse input data. Because of this, adding additional sensors is transparent to the Response module. Therefore range sensors (radar, laser, sonar, infrared, etc.) and visual data can be used to augment GPS data on the leader. Each sensor's data will make a weighted contribution to the leader's position (with weight based on the quality of the sensor). Additional quality data will increase the confidence that the leader is at a given position. If some data is lost (due to jamming or physical damage) the confidence will go down, but the Follower Behavior may still be able to function. The Follower Behavior can use its confidence in the leader's position in order to know how closely it can follow the leader. There are also applications where radio silence may be desired (such as a battlefield) where the Follower Behavior may choose to only use only local data to stay in formation.

IV.B.2. Formation Search

Another application of the Follower Behavior could be using a formation of aircraft to search a large area. The aircraft would be spaced sufficiently far enough apart that their sensor coverage did not overlap. When one aircraft detected an object that required further investigation, it would initiate a series of negotiations to determine the best partner to investigate the object.[15] In this case the best partner would probably one on the outside of the formation or in the back of the formation that happened to have the required sensors. This UAV would then break off the formation to investigate the target. Upon completion, the UAV would return to its proper place in the formation.

IV.B.3. Mid-Air Refueling

The capability to refuel in mid air is crucial for large UAVs. So far most strategies involve remotely piloting the UAV. The control implemented by the Follower Behavior is not tight enough to guide a UAV to a tanker's boom, but the IC can be used to decide when refueling is necessary, and to fly into a loose formation with the tanker so that a human operator may take over.

In the Self RC there could be a fuel remaining parameter. This parameter could be updated every time step based on the time between time steps and the throttle value for that time step. There could also be special Partner RCs for tankers. A "Refuel" CINet would simply say, "if fuel is low and a tanker is near then refuel". Both "low" and "near" are fuzzy in meaning, so they are ideally suited for CINets. A "Go Home" CINet would say, "if fuel is low and a tanker is not near then go home". Now the Mid-Air Refuel Behavior will monitor Perception to see if there is a high confidence that it should refuel. If there is, it will request control. If it is granted control, it will proceed to join in formation with the tanker. Likewise, if the Go Home Behavior detects a high confidence that it should go home it will request control. If the request is granted, the Behavior will plan a path to the home base.

V. Results

Prior to implementing the controller for the Follower Behavior, the UAV's performance had to be characterized. The specifications require that the follower follow the lead aircraft's path at a set distance above (or below), behind, and with a lateral offset. The altitude remains constant during the flight, so it is not necessary to characterize altitude changes, just how well the autopilot holds a constant altitude. Figure 10 shows a plot of how well the autopilot holds a command altitude. Notice that the altitude error is less than twenty meters.

The correct path (with a possible offset) is followed by setting GPS waypoints in the autopilot. Because users have little control over how waypoints are followed, this command loop has not been characterized.

The most important loop for the leader-follower application is the airspeed loop. The airspeed is the part that is varied by the PID controller in order to make sure that the follower remains a constant distance behind the leader. Because of this the ability to change speeds must be characterized. Figure 11 shows an example acceleration maneuver and Figure 12 shows an example deceleration maneuver. As can be seen from these two plots, the autopilot has a slow response, and a great deal of oscillation. Because of this, using speed to control following distance is a difficult task. This limitation must be kept in mind when evaluating the performance of the follower controller.

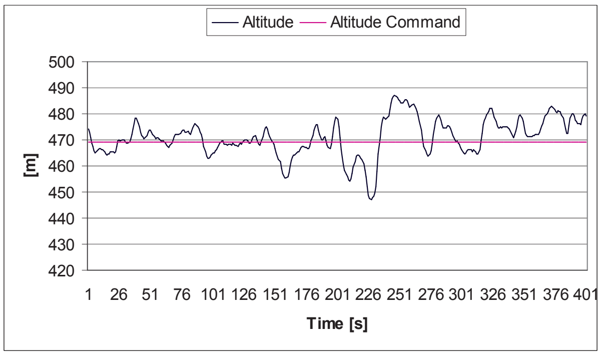

Figure 10. Command altitude and actual altitude

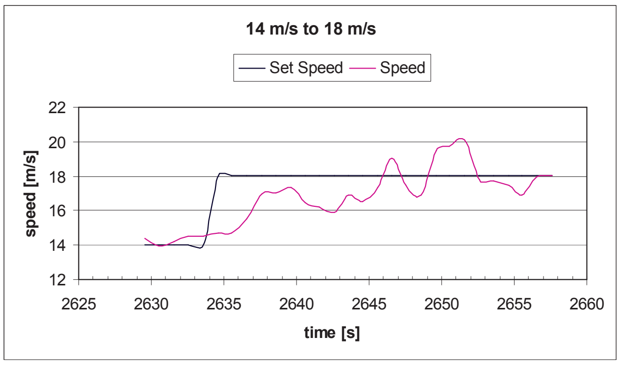

Figure 11. Acceleration maneuver

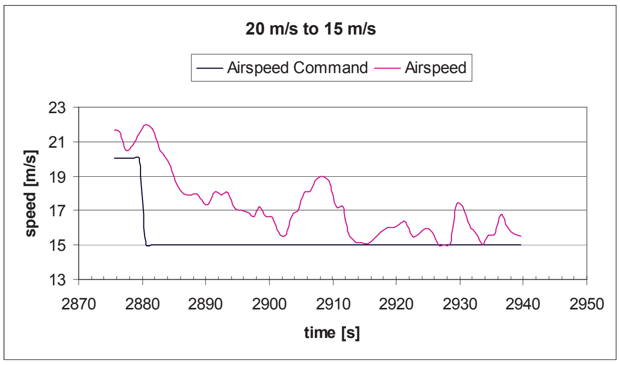

Figure 12. Deceleration maneuver

The Follower controller has been tested using Cloud Cap Technology's hardware in the loop (HIL) simulation. The HIL simulation uses the actual base station and autopilots from the airborne platform, but the autopilots are connected to a simulator running on a computer. This way we can simulate what will happen in an actual flight, without putting any expensive equipment at risk. Because the data streams coming from the autopilot would look exactly like they do in flight, the operation of the HIL simulation is transparent to the IC.

A plot of a HIL test of the Follower controller is shown in Figure 13. The top plot shows the distance between the lead aircraft and the follower aircraft. The set distance is 100 m. The closest the follower gets to the leader is 50 m, so 100 m would be a safe following distance for an actual flight. The bottom plot shows speed set by the follower controller. The speed varies between 15 m/s and 24 m/s, and the leader's speed is 18 m/s. The controller never reaches a steady state speed. This is due to several issues. First is the controller tuning. Second is the lag in the follower receiving GPS estimates from the lead aircraft. Third is due to the slow noisy response of the speed controller.

Future HIL tests of the Follower controller will focus on continuing PID controller tuning. The goal is to get the the speed and distance to either converge to a steady state value or oscillate in a predictable manner. Once the HIL phase is completed, flight testing will commence in the spring of 2008. Flight testing should hopefully confirm the HIL testing. Some tuning may be necessary, however due to differences between the actual aircraft and the HIL model.

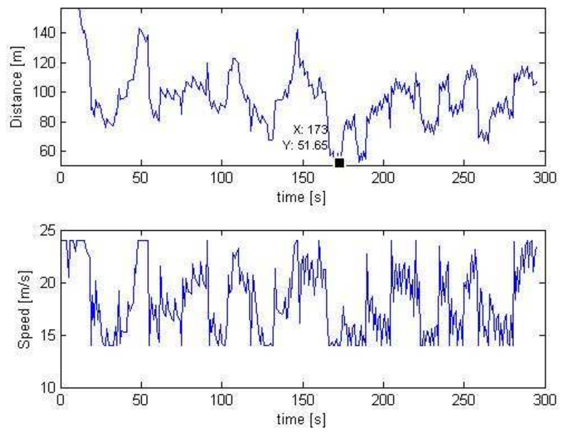

Figure 13. Follower HIL test

VI. Conclusions

This paper presented a software architecture for mission-control of autonomous UAVs. The architecture provides for sensor data fusion, worldview modeling, inference drawing, mission planning, and mission execution. Formation flight has been presented as one of the architecture's many possible capabilities. When combined with an autopilot onboard our UAV testbed, the resulting system exhibits a degree of both intelligence and autonomy.

Acknowledgment

This work was partially supported by the Intelligent Systems Lab at the Penn State Applied Research Laboratory (ARL/PSU). It was also supported by the ARL/PSU Exploratory and Foundational Research Program.

References

[1]Gottfredson, L., "Mainstream Science on Intelligence: An Editorial With 52 Signatories, History, and Bibliography," *Intelligence*, Vol. 24, No. 1, 1997, pp. 13–23, also appeared in Wall Street Jorunal, Dec. 13, 1994.

[2]Bekey, G. A., *Autonomous Robots: From Biological Inspiration to Implementation and Control*, The MIT Press, 2005.

[3]Long, L. N., Hanford, S. D., Janrathitikarn, O., Sinsley, G. L., and Miller, J. A., "A Review of Intelligent Systems Software for Autonomous Vehicles," *IEEE Computational Intelligence for Security and Defense Applications Conference*, Honolulu, Hawaii, 2007.

[4]Hanford, S. D. and Long, L. N., "Evaluating Cognitive Architectures for Unmanned Autonomous Vehicles," *22nd Conference on AI, Association for the Advancement of Artificial Intelligence*, Vancouver, Canada, July 2007.

[5]Hanford, S. D., Janrathitikarn, O., and Long, L. N., "Control of a Six-Legged Mobile Robot Using the Soar Cognitive Architecture," *to be presented at the 46th AIAA Aerospace Sciences Meeting*, No. AIAA-2008-0878, Reno, NV, Jan. 7-10 2008.

[6]Janrathitikarn, O. and Long, L. N., "Gait Control of a Six-Legged Robot on Unlevel Terrain using a Cognitive Architecture," *to be presented at the IEEE Aerospace Conference*, Big Sky, Montana, Mar. 1-8 2008.

[7]Minsky, M., *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*, chap. V. Levels of Mental Activities, Simon and Schuster, 2006.

[8]Stover, J. A. and Kumar, R., "A Behavior-based Architecture for the Design of Intelligent Controllers for Autonomous Systems," *IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, Cambridge, MA, 1999, pp. 308–313.

[9]Brooks, R. A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, March 1986, pp. 14–23.

[10]Brooks, R. A., "Intelligence Without Representation," *Artificial Intelligence Journal*, Vol. 47, 1991, pp. 139–159.

[11]Stover, J. A. and Gibson, R. E., "Modeling Confusion for Autonomous Systems," *SPIE, Science Artificial Neural Networks*, Vol. 1710, 1992, pp. 547–555.

[12]Stover, J. A. and Gibson, R. E., "Controller for Autonomous Device," U.S. Patent 5,642,467, June 1997.

[13]Weiss, L., "Intelligent Collaborative Control for UAVs," *AIAA InfoTech@Aerospace Conference*, No. AIAA-2005-6900, Washington D.C., 2005.

[14]Miller, J., Minear, P., Niessner, Jr., A., DeLullo, A., Geiger, B., Long, L., and Horn, J., "Intelligent Unmanned Air Vehicle Flight Systems," *Journal of Aerospace Computing, Information, and Communication (JACIC)*, Vol. 4, No. 1, May 2007.

[15]Sinsley, G. L., Miller, J. A., Long, L. N., Geiger, B. R., Niessner, A. F., and Horn, J. F., "An Intelligent Controller for Collaborative Unmanned Air Vehicles," *IEEE Symposium Series in Computatioinal Intelligence*, Honolulu, Hawaii, April 2007.

[16]Stover, J. A. et al., "Continuous Inference Networks for Autonomous Systems," *IEEE Conference on Neural Networks for Ocean Engineering*, 1991, pp. 177–183.

[17]Stover, J. A., Hall, D. L., and Gibson, R. E., "A Fuzzy-Logic Architecture for Autonomous Multisensor Data Fusion," *IEEE Trans. Ind. Electron.*, Vol. 43, 1996, pp. 403–410.

[18]Geiger, B. R., Horn, J. F., Sinsley, G. L., Ross, J. A., and Long, L. N., "Flight Testing a Real Time Implementation of a UAV Path Planner Using Direct Collocation," *AIAA Guidance, Navigation, and Control Conference*, No. AIAA-2007-6654, Hilton Head, SC, 2007.

[19]Sinsley, G. L., Ross, J. A., Long, L. N., Geiger, B. R., Niessner, A. F., and Horn, J. F., "A Low-Cost Autonomous UAV System for Earth Observation," *Submitted to IEEE Transactions on Geoscience and Remote Sensing*, 2008.

[20]Wang, P. K. C. and Hadaegh, F. Y., "Coordination and control of multiple microspacecraft moving in formation," *The Journal of the Astronautical Sciences*, Vol. 44, No. 3, 1996, pp. 315–355.

[21]Ren, W., "Consensus Based Formation Control Strategies for Multi-vehicle Systems," *Proceedings of the 2006 American Control Conference*, Minneapolis, Minnesota, USA, June 14-16 2006.

[22]Balch, T. and Arkin, R. C., "Behavior-based formation control for multirobot teams," *IEEE J. Robot. Automat.*, Vol. 14, No. 6, December 1998, pp. 926–939.