LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-ID-150006

UAV Cooperation Architectures for Persistent Sensing

R. S. Roberts, C. A. Kent, E. D. Jones

March 20, 2003

UAV Cooperation Architectures for Persistent Sensing

Randy S. Roberts, Claudia A. Kent, and Erik D. Jones

Lawrence Livermore National Laboratory,
Livermore, CA 94550 USA

ABSTRACT

With the number of small, inexpensive Unmanned Air Vehicles (UAVs) increasing, it is feasible to build multi-UAV sensing networks. In particular, by using UAVs in conjuction with unattended ground sensors, a degree of persistent sensing can be achieved. With proper UAV cooperation algorithms, sensing is maintained even though exceptional events, e.g., the loss of a UAV, have occurred. In this paper a cooperation technique that allows multiple UAVs to perform coordinated, persistent sensing with unattended ground sensors over a wide area is described. The technique automatically adapts the UAV paths so that on the average, the amount of time that any sensor has to wait for a UAV revisit is minimized. We also describe the Simulation, Tactical Operations and Mission Planning (STOMP) software architecture. This architecture is designed to help simulate and operate distributed sensor networks where multiple UAVs are used to collect data.

Keywords: Unmanned Air Vehicles, Robot Cooperation, Automated Sensing, Distributed Sensing

1. INTRODUCTION

With the development of low cost, high endurance Unmanned Air Vehicles (UAVs), it is now practical to perform autonomous sensing and data collection over broad areas. These vehicles offer the ability to sense the environment directly through on-board sensors, or vicariously through sensors placed within the region. Indeed, by coupling a network of UAVs with unattended ground sensors, a degree of persistent sensing, i.e., continual collection and analysis of sensor data, can be achieved. In such networks, UAVs act as intelligent agents using the ground sensors to collect data in critical areas such as choke points and areas of ingress. As the UAVs fly overhead, data is uplinked from the ground sensors, analyzed and fused with data from other UAVs, and acted upon. Particular actions that a UAV might take depend on the sensing task. For example, the ground sensors might serve as tripwires to alert overhead UAVs to the presence of an intruder. In turn the UAV might activate an onboard imaging sensor, commence transmitting imagery to an operator, and search the vicinity of the ground sensor to obtain imagery of the intruder.

A critical aspect of the sensing scenario described above is the continual presence of a UAV to interact with ground sensors. Although a single UAV is useful for deploying new sensors and can provide communication services to isolated sensors, its failure can be catastrophic to the sensor network. Alternatively, the advantages of using several autonomous UAVs in concert to collect sensor data are manifold. Multiple vehicles allow sensing to be performed in parallel, thereby reducing the amount of time required to gather data. Moreover, if a vehicle becomes disabled, the remaining vehicles can continue sensing, albeit at a reduced collection rate.

There are two fundamental problems associated with using a fleet of UAVs to autonomously collect sensor data. The first problem is that of constructing efficient routes that allow the UAVs to collect data without duplicating effort or interfering with one another. This particular type of routing problem is fundamentally one of combinatorial optimization. A heuristic solution to this problem is reported in [1], with important advancements described in [2]. The second problem is

that of devising control schemes for individual UAVs that collectively allow a fleet of UAVs to adapt to exceptional events in a globally optimal manner. Such events include the loss or addition of UAVs to the network or changes in the sensing requirements such as increased priority of particular subregions.

In this paper we review a path planning algorithm designed for use with multiple UAVs, and describe some recent improvements to the algorithm. Next, we describe an automation architecture that allows the UAVs to cooperatively collect data from unattended ground sensors. This architecture is designed to react to several different types of events, thereby maintaining an optimal data collection configuration. Finally, we describe a software environment designed to facilitate the development of data collection networks employing multiple UAVs. The software environment, called STOMP (Simulation, Tactical Operations and Mission Planning), implements the path planning and automation architecture, and provides hardware-in-the-loop simulation capabilities.

2. UAV ROUTE PLANNING

An issue of fundamental importance in using UAVs to collect sensor data is route planning. The route planning algorithm proposed in [1] is well suited for multiple UAV data collections. Essentially, the unattended ground sensors are considered as sets of waypoints that the UAVs must travel between and dwell over to collect data. Given a set of N ground sensors (or clusters of ground sensors) and K UAVs, we desire a path planning algorithm that generates K non-overlapping, non-branching, closed paths to every sensor in the network. The route network can be modelled as a family of graphs $\{(S_k, P_k)\}$, $k = 1, \ldots, K$ where $S_k = \{s_i\}_k, i = 1, \ldots, N_k$ is the set of sensors formed into the k^{th} subnet, and $P_k = \{l_{ij}\}$ is the set of weighted, directed links that indicates a path between sensor s_i and sensor s_j. Weight c_{ij} associated with link l_{ij} is the cost to the UAV of travelling between sensors s_i and s_j and collecting data at sensor s_j. The cost of path P_k, denoted as C_k, is the sum of the weights of the links in P_k.

This particular route planning algorithm can be formulated as a variant of an M-ary Traveling Salesman problem [4]. Given the N sensors and K UAVs, we seek as assignment array x_{ij}^k, $x \in \{0, 1\}$ that will minimize cost function Z where

$$Z = \sum_i \sum_j \sum_k c_{ij} x_{ij}^k \tag{1}$$

subject to the following constraints:

$$\sum_{i=1}^{N} \sum_{k=1}^{K} x_{ij}^k = 1, \quad j = 1, \ldots, N \tag{2}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K} x_{ij}^k = 1, \quad i = 1, \ldots, N \tag{3}$$

and

$$\sum_{j=1}^{N} x_{ip}^k - \sum_{j=1}^{N} x_{pj}^k = 0, \quad k = 1, \ldots, K; \quad p = 1, \ldots, N \tag{4}$$

The first two constraints assure that every sensor is visited by only one UAV, and the third constraint provides for the continuity of routes. One final constraint is required for a formal definition of the problem. This constraint eliminates subtours, and is described in [4].

Equation (1) with the constraints (2)—(4) is a formidable combinatorial optimization problem. A heuristic solution to the UAV route planning algorithm is reported in [1] (also cf. [2]). The basis

for the path planning algorithm is the cost function

$$C = \sum_{k=1}^{K} (C_k)^a \tag{5}$$

where the individual terms C_k in the summation are the costs that each path contributes to the total cost of the network. The exponent a provides a means to balance the desires for minimum total path cost and roughly equivalent individual path costs. Values in the range of $3 \leq a \leq 6$ have been found to work well in practice. The heuristic solution is generated in three steps: 1) an initialization step which clusters waypoints into groups, 2) an initial path plan for the clusters, and 3) a balancing operation that tries to minimize the global cost of the routing plan. An overview of the path planning algorithm is described next.

As reported in [1], the first step of the initialization process is to construct K subnets of sensors S_k, from the N sensors in the network. The subnet construction procedure is based on a modified K-means algorithm. The modification to the standard K-means algorithm occurs in the initialization of the algorithm. Typically, a K-means algorithm initializes by randomly selecting K samples from a set (sensor positions in our case), and using these samples as the initial centroids of K clusters [5]. This type of initialization was found to produce poor subnets. Rather, the initial cluster centroids are found by randomly selecting K out of N samples, finding the pair of samples with the minimum distance between them, and replacing one of the samples with a new sample. This process continues until K widely separated samples are found. After these initial sensors have been determined, the K-means algorithm proceeds in the customary manner.

After the initial subnets have been formed, paths are constructed which connect all of the sensors in the subnet. Path P_k through the k^{th} subnet is constructed by first finding a circumferential path around S_k, and then including sensors interior to the circumferential path. The circumferential path around S_k is found from the convex hull of S_k. Sensors in the interior of the subnet are added to path P_k in positions that minimize their contribution to the global cost (5). The differential cost of adding sensor s_q to the path between sensors s_i and s_j is found by breaking link l_{ij} into links l_{iq} and l_{qj}, and is given by

$$\Delta c_{iqj} = c_{iq} + c_{qj} - c_{ij} \tag{6}$$

Interior sensors are inserted into path P_k at the position that minimizes (6). The process of adding interior sensors to the path continues in this manner until all sensors in the subnet have been assigned a position in the path.

After the initial paths to all sensors in the network have been computed, the UAV route structure of the overall network is balanced (optimized) by shifting sensors between subnets. Consider paths p and p' in two neighboring subnets. The differential cost of delinking sensor s_j from sensors s_i and s_k in path p and inserting it into the link between waypoints s_m and s_n in path p' is given by (cf. (5))

$$\Delta C = \Delta C_p + \Delta C_{p'} \tag{7}$$

where

$$\Delta C_p = (C_p - \Delta c_{ijk})^a - (C_p)^a \tag{8}$$

and

$$\Delta C_{p'} = (C_{p'} + \Delta c_{mjn})^a - (C_{p'})^a \tag{9}$$

If a particular combination of j, $\{i,k\}$, $\{m,n\}$, and $\{p,p'\}$ yields a $\Delta C < 0$, then the global path cost will decrease if the move is performed. By testing all sensors in all links of all paths in the network, and moving only those sensors that decrease the global path cost, an optimal path (and subnet) configuration is obtained.

For a fixed route cost structure, i.e., where c_{ij} is constant, K UAV paths can be computed using the algorithm described above. In order to optimize the route structure for the UAV data

collection problem, it is necessary to use an appropriate cost function. In the data collection scenarios previously described we desire to minimize the amount of time required for a UAV to collect and process all of its data. Hence, the appropriate cost function is based on the amount of time required for a UAV to collect, analyze and act upon the data in its subnet. Thus, element c_{ij} of the cost matrix is given by

$$c_{ij}(t) = \tau_{ij} + \Delta t_i \tag{10}$$

where τ_{ij} is the amount of time required for a UAV to traverse from the i^{th} sensor to the j^{th} sensor and Δt_i is the amount of time that the UAV dwells at the i^{th} sensor. Observe that this cost metric is highly applicable to the problem of collecting data with UAVs. For instance, headwinds and other disturbances can greatly increase the time it takes for a UAV to collect data in its subnet. As the amount of time that it takes to collect data increases, the cost of the data collection increases and the UAVs can react by spreading the increased cost over the network.

The quantities τ_{ij} and Δt_i are random quantities, and are thus required to be estimated by the UAVs. Prior to a UAV flying between sensors s_i and s_j and collecting data at sensor s_j, we can only guess at values for τ_{ij} and Δt_i. After a UAV has flown this route and collected data, the amount of time that has transpired can be measured. To accomodate both situations, two techniques are used to estimate the cost of the link. Denote the distance between s_i and s_j as d_{ij} and the maximum speed and acceleration of the UAV as V_{max} and A_{max}. If the sensors are spaced such that the UAV has time to accelerate, cruise and decelerate without overshooting the sensor, then the travel time can be approximated as

$$\tau_{ij} \approx 2\frac{V_m}{A_m} + \left(\frac{d_{ij}}{V_m} - \frac{V_m}{A_m}\right) \tag{11}$$

After a UAV has traveled the link, the travel times are measured and filtered to yield estimates $\hat{\tau}_{ij}$. The dwell time measurements are processed in a similar manner to yield estimates $\Delta\hat{t}_i$. The cost estimates are given by either the preliminary distance-based estimates, or the filtered measurement-based estimates:

$$\hat{c}_{ij} = \begin{cases} \hat{\tau}_{ij} + \Delta\hat{t}_i, & \hat{\tau}_{ij} \neq 0 \\ \tau_{ij} + \Delta\hat{t}_i, & \text{otherwise} \end{cases} \tag{12}$$

Finally, the cost of the subnet is given by

$$\hat{C}_k = \sum_{i=1}^{N_k - 1} \hat{c}_{i,i+1} + \hat{c}_{N,1} \tag{13}$$

An example of the routes produced by this algorithm is illustrated in Figure 1. That figure shows the route plan produced for fourteen UAV and eighty sensors. Other examples of the route planning algorithm are shown in Figure 3.

3. UAV AUTOMATION

The path planning algorithm previously described is suitable for relatively static scenarios where the number of UAVs and waypoints does not change, and the costs associated with collecting data remain relatively constant. However, in many scenarios of interest the number of UAVs and sensors change as well as the cost of collecting data. In these situations we desire an automation architecture that can adapt the network routing structure as conditions warrant. Recall that the data collection task is defined by a set of waypoints, i.e., spatial coordinates, where a UAV collects data. Data collection can be performed by operating onboard sensors, or by uplinking data from unattended ground sensors placed within a region.

Our investigation resulted in several control architectures based on hierarchical or distributed control of network adaptation, and variations in the route optimization algorithm developed in [1]. Essential to these architectures is a cost associated with the sensing task performed by each UAV (a

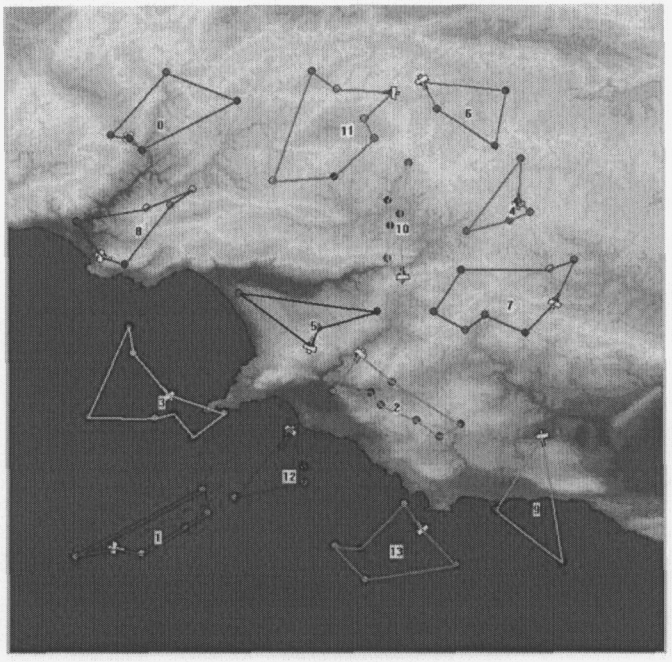

Figure 1. Illustration of route planning for 14 UAVs and 80 waypoints

generalization of the cost metric used in [1]). This cost information is continually estimated by each UAV, and shared throughout the network. In the hierarchical architectures, one UAV is selected to lead network adaptation. This leader is not unique, and can be replaced by any other UAV should the leader leave the network. This type of architecture is useful in small networks where adaptation needs to be performed quickly. In the distributed architectures, UAVs cooperatively perform local network adaptation by locally optimizing the costs of their subnets. These architectures are useful in large networks where local sensing costs can change rapidly.

With adaptability in mind, an automation architecture for collecting data with a fleet of UAVS was developed. The architecture is designed to adapt the network route structure to the following exceptional events:

1. UAVs are added or removed from the network
2. Waypoints are added or removed from the network
3. Waypoints change their positions or priorities within the network

These events are fundamental changes to the data collection architecture. More complicated scenarios can be decomposed into these fundamental events. Network adaptation can be driven either globally, locally or by several hybrid schemes [2]. The UAVs share information they gather on the state of waypoints in the network as well as their own status. Since all UAVs have similar views of the state of the network, any one UAV can direct global adaptation. Such adaptation is required if, for example, a UAV leaves the network. In this case, a single UAV recomputes the new route assignments for all remaining UAVs in the network, and instructs them to begin executing the new routes. On the other hand, for minor network adjustments, such as a ground sensor moving be-

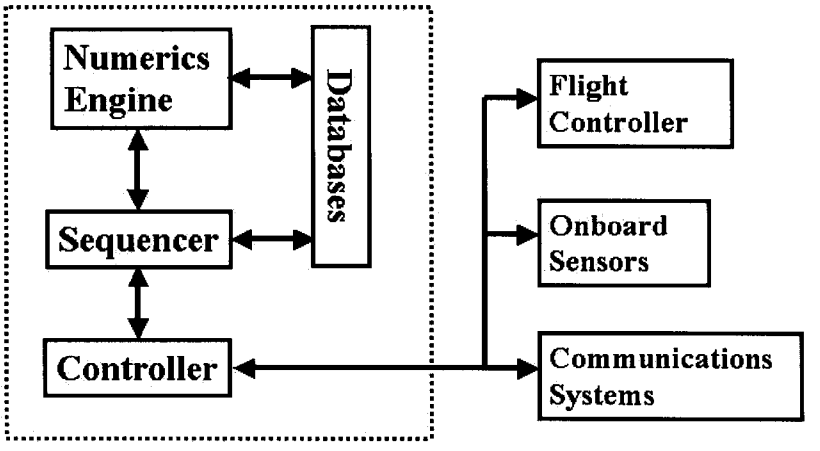

Figure 2. Block diagram of the UAV Automation Architecture

tween two subnets, the involved UAVs perform a peer-to-peer transaction where the moving sensor is transferred from one subnet to the other.

The UAV automation architecture that implements these adaptations is implemented as a set of algorithmic modules and databases that reside and execute on a control computer onboard each UAV in the network. It interfaces with the UAV flight controls and instrumentation, the UAV communications system, and any onboard sensors as illustrated in Figure 2.

The architecture is based on a three-component approach similar to those described in [7]. The first component, called the sequencer, is a looping structure that the UAV control computer continually executes. The sequencer, through the controller, takes inputs from various UAV subsystems including flight controls and communications. Depending on the state of these inputs, the sequencer branches to different routines designed to react to changing states. The second portion of the architecture is the Numerics Engine. The sequencer invokes the Numerics Engine to perform path planning and related calculations. The final portion of the architecture is the Controller, which interfaces to the UAV flight controls, communications and any onboard sensors. In the sequel, we focus on the sequencer modules and supporting databases.

The sequencer consists several algorithmic modules and two databases. The algorithmic components of the sequencer include: 1) Waypoint Service Routine, 2) Message Handler Routine, 3) Network Adaptation Routine, 4) Cost Estimator, 5) Sensor Service Routine, 6) Network Leadership Routine, 7) Local Route Optimizer, and 8) Waypoint Sequence Routine. The databases contain information relating to the operation of the network. One database contains data related to all waypoints in the network. This data includes, *inter alia*, such information as a waypoint identifier, waypoint state information, and time of last UAV visit. The second database contains information related to the UAVs in the network. This database includes, *inter alia*, unique identifiers for each UAV and the position of each UAV in leadership succession. The functions of the sequencer modules

are briefly described below.

The Waypoint Service Routine performs data collection tasks specific to each waypoint. Data collection tasks at each waypoint can vary, depending on the nature of the sensor. Such tasks might include uplinking data from a ground sensor, or dwelling over a region to observe an event with onboard video cameras. Similarly, the Sensor Service Routine operates any on-board sensors.

The Message Handler Routine (MHR) processes status messages from other UAVs in the network. In particular, two types of messages are processed by MHR, a waypoint status message (WSM) and UAV status message (USM). As the UAVs collect data at waypoints, they broadcast a WSM to all UAVs in the network. This message contains a variety of information related to the state of the waypoint. As a result, each UAV has a complete view of all waypoints in the network. Using this information, the Cost Estimator estimates the cost of each link in the network. This cost is updated each time a UAV receives a new WSM. In this manner each UAV can track the cost of every subnet in the network. The WSM is also used to broadcast the addition or deletion of waypoints from the network.

Using UAV status messages, each UAV has knowledge of all UAVs in the network. If any UAV intentionally leaves or enters the network, the other UAVs are informed via a USM. If any UAV unintentionally leaves the network, a second mechanism is used. The UAV database contains a field that indicates the health of each UAV in the network. The health of a UAV is determined by its appearance in the network routing tables. If a UAV appears in the table, it is assumed that the UAV can communicate and is capable of carrying out its tasks. The Network Leadership Routine, in conjunction with UAV status messages and UAV heartbeat, determines whether the leader is functioning. If not, leadership is transferred to another UAV. The Network Adaptation Routine on the leader UAV monitors the various status messages, and executes a global network reorganization if warranted.

The Local Route Optimizer (LRO) and Waypoint Sequence Routine (WSR) perform local optimizations. The LRO monitors costs in neighboring subnets, and initiates a local reorganization if subnet costs are not in balance. The WSR determines the starting waypoint of a UAV's traverse of its subnet. The starting point of the traverse is found as the waypoint that minimizing the maximum amount of time that any waypoint in the subnet must wait for the UAV.

The automation algorithms described here were tested using several thousand Monte Carlo simulations on random networks. The results of these simulations suggest that global reorganizations can be performed more quickly and optimally for small networks than purely local techniques. However, the global methods do not scale for larger networks where localized optimizations might have an advantage. See [2] for more details on the simulations are results.

4. SIMULATION, TACTICAL OPERATIONS AND MISSION PLANNING

In order to gain a deeper understanding of coupling cooperating UAVs with unattended ground sensors, the Simulation, Tactical Operations and Mission Planning (STOMP) architecture was developed [3]. STOMP implements the route planning and automation algorithms described in the previous sections. It contains detailed UAV and sensors objects, and performs the simulations using Digital Terrain Elevation Data (DTED) as a backdrop. Fundamental to the design of STOMP is its ability to incorporate real UAVs and sensors into its simulations. In particular, support for the Micropilot MP2000 autopilot has been implemented into STOMP [9]. Additionally, wireless Ethernet is available for receiving data from fielded sensors.

A functional block diagram of the STOMP architecture is shown in Figure 4. As illustrated in the figure, STOMP consists of the following blocks: 1) Sensor and UAV objects (where sensor objects are depicted as circles, and UAV objects are depicted as hexagons); 2) a Communication Controller; 3) an Event Controller; and 4) a Display Controller. These objects are briefly described

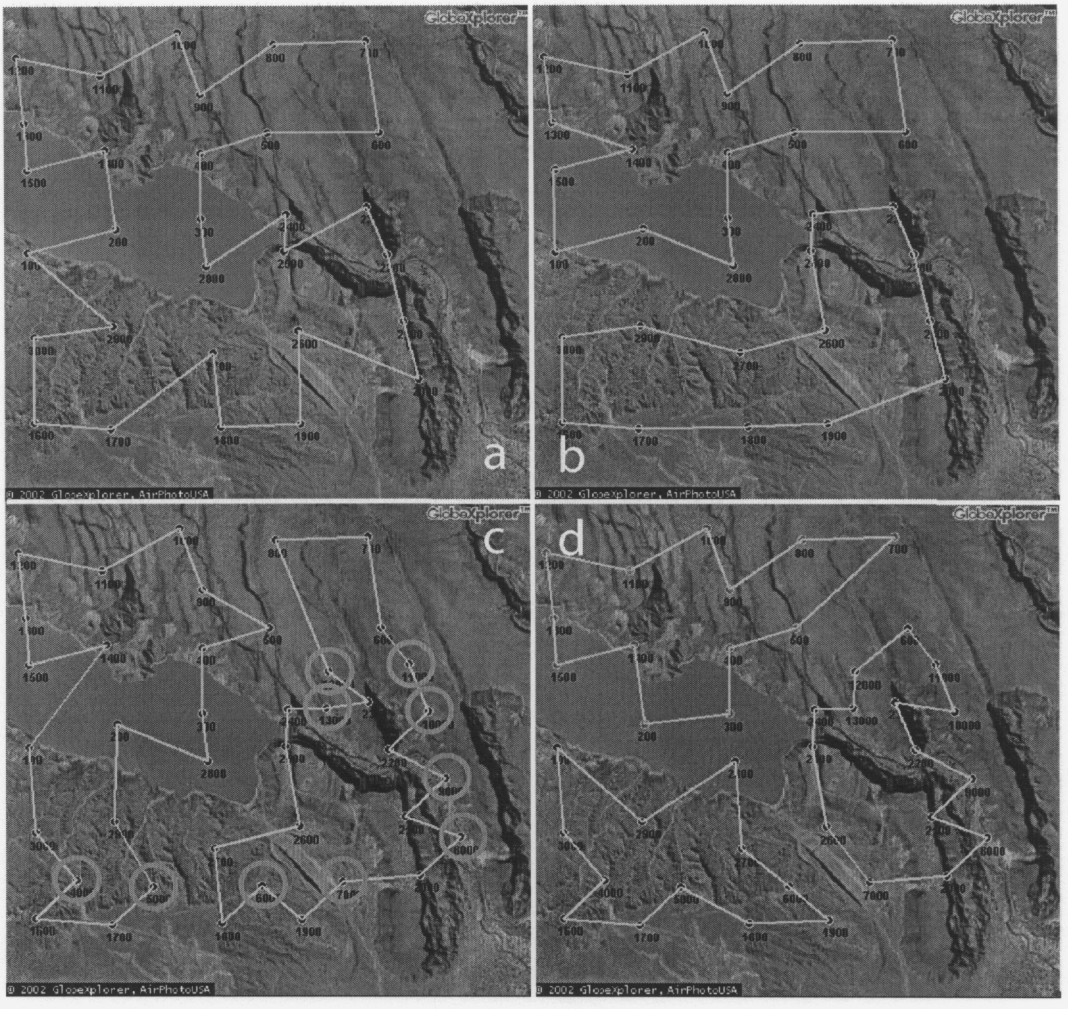

Figure 3. Four panels illustrating network adaptation. (a) Route of one UAV servicing sensors around a dam. (b) A second UAV is added to the network. (c) Several more sensors are added to the network, and the UAVs adjust their paths to accomodate the new sensors. (d) A third UAV is added to the network.

below. See [3] for a comprehensive description.

The UAV and sensor objects contain state information and algorithms necessary for the operation of both real UAVs and sensors, and simulated UAVs and sensors. The primary components of UAV objects are a flight controller, communication controller and the automation algorithms. STOMP UAV objects interact with two other objects: the MP2000 in the case of real UAVs, and the STOMP event controller in the case of simulated UAVs. For real UAVs the associated STOMP UAV object transmits waypoint coordinates to the MP2000 and asynchronously receives state updates from the controller. Virtual UAV objects behave in a similar manner, but interact with the STOMP event controller. Sensor objects contain data structures to hold sensor data, and a communications controller. In the case of real sensors, data is transported through the STOMP communications controller to a wireless Ethernet port. For virtual sensors, data is routed to the STOMP event controller.

The event controller initializes and executes simulations. It employs a graphical interface that allows users to script data collection scenarios. It also records the state history of every object in the simulation. Included in the state history is line-of-sight data for UAV-to-sensor communication links. This data is useful in analyzing network connectivity patterns.

The communications controller coordinates all communication between the event controller, and the UAVs and sensors, be they real or virtual. For real UAVs, the communications controller routes command and control data to the MP2000 autopilot via a wireless serial interface. For real sensors, the communications controller routes data through a wireless Ethernet port. Packet routing is managed using the Mobile Mesh [10] software. STOMP operates at a higher level than routing software, and thus remains independent of specific communication and routing methods.

During the course of development, several experiments were conducted with small UAVs and imaging sensors. It was found that the communication channel between the UAV and communication node was too unreliable for standard file transfer techniques. As a result of these experiments, a new protocol was devised that allows more robust file transfer. This protocol has been incorporated into the STOMP communications controller, and it is suitable for a wide variety of UAV-to-sensor applications.

STOMP has two display modes, one for designing simulations and one for executing simulations. In the design view objects are placed on a Digital Terrain Elevation Data (DTED) backdrop and their initial state set. When a simulation begins, the simulator view appears. In this view, the paths of the UAVs, the positions of the UAVs, and the positions of the sensors on the DTED backdrop are visible. The state of any object in the simulation can be accessed from this view. As a UAV acquires new data from a sensor, the sensor's icon changes color to denote the new data. The new data can be displayed by clicking on the icon. Although any kind of data can be displayed, STOMP currently has provisions for displaying imagery. An illustration of a STOMP image display is shown in Figure 4. In this example, the UAV has flown over a sensor collecting imagery from a bridge. As the UAV uploads the image, it transmits it to STOMP where the marker for the sensor changes color indicating new data. The operator clicks on the sensor indicator and the image is displayed.

5. SUMMARY

As inexpensive, high endurance UAVs become available, the use of these vehicles with unattended ground sensors will become increasingly common. Through the interaction of cooperating UAVs and ground sensors, continual sensing of an area can be achieved. In this paper, we reviewed a path planning algorithm suitable for use with cooperating UAVs, and described recent developments to the algorithm. An architecture for the coordinated operation of a fleet of UAVs collecting data from the sensors was presented. The architecture is designed to adapt to a variety of fundamental events so that data collection continues in an optimal fashion. A simulation and operations environment,

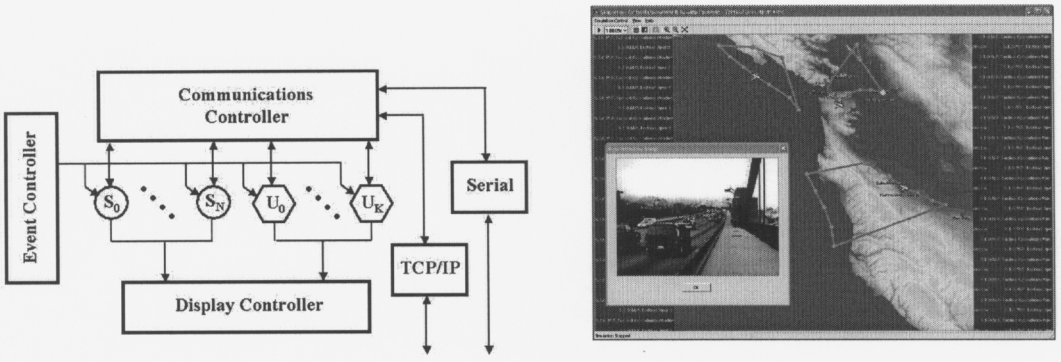

Figure 4. Functional Block Diagram of STOMP and STOMP console display

called STOMP, was developed to further the investigation into the use of UAVs and ground sensors for data collection and sensing.

6. ACKNOWLEDGEMENTS

This research was performed under a grant from the Laboratory Directed Research and Development program, Lawrence Livermore National Laboratory. The authors would like to thank Dave McCallen, Director of the Center for Complex and Distributed Systems, and Don Meeker, Associate Director of Engineering, for their support of the project. This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

REFERENCES

1. C. T. Cunningham and R. S. Roberts. A adaptive path planning algorithm for cooperating unmanned air vehicles. In *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.
2. C. A. Kent and R. S. Roberts. Cooperation and path planning for unmanned air vehicles. Technical Report UCRL-JC-149915, Lawrence Livermore National Laboratory, September 2002.
3. E. D. Jones, R. S. Roberts, and T. C. Hsia. Stomp: A software architecture for the design and simulation of uav-based sensor networks. In *IEEE International Conference on Robotics and Automation*, Taipei, Tiawan, May 12-17 2003.
4. E. L. Lawler and et. al. *The Travelling Salesman Problem.* Wiley-Interscience, New York, 1985.
5. K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.
6. J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1993.
7. E. Gat. Three-layer architectures. In D. Kortnekamp, R. Peter Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robotics.* AAAI Press / MIT Press, Menlo Park, CA, 1998.
8. D. MacKenzie, R. Arkin, and J. Cameron. Multiagent mission specification and execution. *Autonomous Robots*, 10:29—52, 1997.
9. Micropilot Corporation. *MP2000 Autopilot*, 2001. Available online at: http://www.micropilot.com.
10. MITRE Corporation. *Providing Solutions for Mobile Adhoc Networking*, 2002. Available online at: http://www.mitre.org/tech_transfer/mobilemesh.

11. K. O'Rourke. Dynamic routing of unmanned aerial vehicles using reactive tabu search. In 67^{th} *Military Operations Research Symposium*, November 26 1999.
12. G. Barbarosoglu and D. Ozgur. A tabu search fro the vehicle routing problem. *Computers and Operations Research*, 26:255—270, 1999.