An Unmanned Aerial System for Autonomous Surveillance

Michael Freed, Will Fitzgerald and Robert Harris
NASA Ames Research Center

Abstract

This paper describes a system for efficient autonomous aerial surveillance of many sites with a single unmanned aerial vehicle. The system integrates numerous hardware and software components including autonomy software for planning, execution, monitoring and flight control. The integrated system provides a unique surveillance capability and incorporates a novel surveillance planning approach involving selection among a portfolio of planning algorithms. We describe the system's components and discuss integration issues affecting system design and performance.

Surveillance Mission Requirements

The Autonomous Rotorcraft Project (ARP) is an Army/NASA effort to develop a practical, versatile and fully-autonomous airborne observation capability. Although the effort is ultimately intended to support a wide range of mission types, work to date has focused on missions to monitor for incidents at multiple sites using a single unmanned aerial vehicle (UAV). In such cases, the sensing capability provided by the UAV represents a scarce resource. The primary objective is to use that resource efficiently.

For example, in a fire detection application, there may be many structures that could potentially catch on fire but only a single UAV moving from site to site to check for an outbreak. Sites may vary greatly in importance, probability of catching on fire, remoteness from other sites, remoteness from firefighting resources and many other factors that affect the value of observing the target at a given frequency. The optimum behavior might involve patrolling only a subset of the sites, or visiting some far less often than others. Analogous applications in, e.g., disaster management, force protection, Earth science and security present the same fundamental problem.

We have formally characterized this general class of missions in order to define specific autonomy requirements (Freed et. al, 2005) and performance evaluation criteria (Freed, Harris and Shafto, 2004). In these missions, which we term *periodic surveillance missions* since targets are observed periodically rather than "persistently," events of interest tend to be rare. Mission performance cannot be evaluated effectively on the basis of events actually observed, but must instead be characterized in terms of how well the agent reasoned about the probability and costliness of events that *might* have occurred. This entails an essentially decision-theoretic approach.

Given that the vehicle can generally observe only a single target at a time and must spend time transiting to and examining each target, it will necessarily not be observing most targets most of the time – i.e. it will be "ignorant" of the state of these targets. We define Expected Cost of Ignorance (ECI) for the period between successive observations of a given target as the sum, for all time points t in the interval, of the probability of an event occurring at t multiplied by the cost if it occurs at t (and thus not detected until the next observation pass). Summing for all such intervals and for all targets, we can compute a total ECI for the mission given a specific target observation schedule. The goal of an autonomous agent is to minimize overall ECI accumulated over the time period of a mission.

$$ECI = \sum_{T}^{\text{Targets}} \sum_{\text{i}}^{\text{Intervals}} \int_{\text{t=t1}}^{\text{t2}} p(t) \cdot Cost(t2 - t)dt$$

The objective of efficiently using a UAV to surveil many sites presents numerous theoretical and practical requirements. The Autonomous Rotorcraft Project has focused on requirements common to multiple application areas including especially those seen as critical for scientific monitoring and defense. One such requirement is inherent in the mission concept: the autonomous system must be able to maximize the value of information delivered to users over the course of a mission by making good choices about which site to visit next and what observation actions to carry out. In particular, it must be able to generate mission plans of high quality based on a metric of the form above. Ideally, the system should do this job better than human UAV operators who, in current state of practice, make all mission-level decisions.

Further requirements arise from the need to execute these decisions in an unpredictable task environment. One effect of unpredictability (and consequent uncertainty) is that some decisions cannot be effectively planned in detail. For instance, an initial mission plan might define the order in which observations sites are to be visited, but not specify observation behavior at each site since the correct behavior depends on hard-to-predict factors such as wind and the locations of detectable viewing obstacles. Another effect of unpredictability is that conditions arising as the plan is executed may threaten vehicle safety, invalidate the plan or reduce its effectiveness at minimizing ECI. Coping with unpredictability requires a range of plan execution capabilities such as monitoring the environment for conditions of interest, elaborating incomplete plans as new

information becomes available and repairing a plan or invoking replanning when invalidating conditions arise.

A third set of requirements stem from the operational environment. The system must possess qualities that engender trust such as safety, reliability, predictability and conformance to standard rules and procedures. Equally important, the system must support a range of operating modes with varying human involvement. A spectrum of human roles have been defined by user communities such as DoD (OSD 2005), NASA and the Open Geospatial Consortium (OGC 2005). *R/C (remote control) pilots* actively control UAV flight. *Operators* typically pilot the system indirectly using an autopilot and payload control automation. *Supervisors* exercise infrequent control of the UAV system, relying on automation and intervening only if needed. Finally, a *user* or *consumer* defines information-acquisition goals and preferences, but is completely removed from direct control of the system. Support for the full range of human/automation roles and the ability to switch gracefully between levels of autonomy is required for such systems to find acceptance in operational communities (Musliner and Pell 1999)

This paper describes the UAS (unmanned aerial system) developed by the Autonomous Rotorcraft Project to meet requirements for a practical UAS-based surveillance capability. Following an overview of ARP system components and integration (see Whalley et al. 2005 for more detail) we discuss autonomy design and integration issues that arose in development of the surveillance planning capability and an approach based on runtime selection from within a portfolio of alternative planning algorithms.

Autonomous Rotorcraft System Overview

Hardware

The Yamaha RMAX helicopter was selected as the project UAV platform. Originally developed for remote control agricultural seeding and spraying, the RMAX is notably sturdy and reliable. With a one hour hover flight duration and 65 lb. payload capacity, the vehicle represented a sweet spot between large, highly capable aircraft that are expensive to operate and small aircraft whose limited range and payload make it difficult to support a meaningful autonomous surveillance capability.

Numerous modifications were made to the RMAX to support autonomous operation. These include the addition of an avionics payload which carries a navigation and flight control computer, experimentation computer, inertial measurement unit, GPS receiver and radio communications equipment. The payload was designed for simple maintenance and to be easily transferred between aircraft.

Separate from the avionics payload is a vibration-isolated stub wing on which various cameras can be mounted. A tilting mechanism supports a pair of monochrome 640x480 resolution IEEE-1394 cameras. These have a one-meter baseline to provide accurate passive-ranging of obstacles at distances sufficient for path replanning. The tilting mechanism provides +10 to -100 degrees of pitch travel. Cross-shafting provides sufficient stiffness to ensure the stereo cameras maintain proper alignment through their range of travel and under vehicle vibratory loads. Any vehicle subsystem can interrogate or reposition the tilting system through the messaging software that connects all vehicle subsystems to one another. A color 640x480 IEE-1394 mounted alongside the left monochrome camera provides real-time progressive-scan streaming imagery to the ground at 10 fps (30 fps stored onboard). A video server allows access to camera video imagery when needed by any process including the stereo passive ranging system, a monocular tracker and video compression and downlink system.

Mounted under the nose of the aircraft is a SICK PLS scanning laser user for obstacle detection and high-resolution mapping. It provides 75 Hz centimeter-accuracy range measurement every one degree over a 180 degree field of view to a range of 80 meters. The sensor elevation can be easily repositioned to meet different requirements; e.g. downward for high-resolution mapping or forward for obstacle detection.

User Interfaces

Several UI components are used to support several human interaction roles. Users submit Observation Requests using the Mission Planning Interface (MPI) shown in Figure 1. The first step is to load a map and specify mission parameters (e.g. maximum flight duration and vehicle start position). The user then drags targets on to the map from the target palette (upper left), specifying attributes of each needed to generate an optimal surveillance mission plan. For example, a user wanting the vehicle to monitor a site for potential fire would specify information about the likelihood of such an occurrence, the value of the site (i.e. the cost suffered if the site were to burn without intervention) and the kind of observation behavior required to check for the condition. Target palettes may be specialized for particular events types (e.g. fires; intrusions into a secure site; scientifically interesting anomalies) with distinct default parameters. Users can make new observation requests at any time, with requests made while a vehicle is in-flight typically causing autonomy software to replan the mission.

The overall ARP objective includes support for concepts of operation involving multiple users at physically distributed sites (OGC 2005). Accordingly, the MPI is implemented as an AJAX web client supporting unlimited instances. A UI for displaying sensor data products resulting from vehicle observations is also implemented as a web client, allowing users at dispersed sites to receive live or archived data from the vehicle.

Figure 1 Mission Planning Interface (left) and Operator's Quad Display (right)

Monitoring and control of the UAV is mediated by a separate UI, the quad display shown in Figure 1. Its four quadrants include (clockwise from upper left) a technical display showing vehicle adherence to the intended path, a live video feed from the vehicle, a moving map display and a live video feed from a ground tracking camera mounted on a trailer used to transport the UAV and as a mobile command center. The moving map shows many kinds of information including vehicle position, immediate path, location of targets, location of standoff positions from which targets will be viewed, obstacles, weather data and commands from autonomous control software. From the quad display, the operator can access a "nudge panel" used to suspend autonomous mission management and directly control vehicle position and attitude.

Autonomous Control Software

High-level autonomous control is provided by Apex (Freed et al. 2005), an open-source reactive planning and execution system used for a range of autonomy applications. The system provides distinctive general capabilities for multitask management, rapid detection of complex conditions and visualization of autonomy behavior. For ARP in particular, Apex handles mission-level decision-making, at-target (tactical) data collection behavior, navigation, response to vehicle health and safety contingencies and interaction with human users.

The Apex software architecture is based on a well-known approach in which autonomy functionality is divided into three modular layers (Bonasso et al. 1995; Gat, 1998). The *deliberative* layer contains computationally expensive AI planning capabilities including, in our case, algorithms for generating ECI-minimizing surveillance mission plans and an Obstacle Field Route Planner described below. A fast-acting reactive *executive* comprising the second layer composes output from planners with stored partial plans (procedure templates) to continuously decide action, outputting commands to platform- and application-specific control functions in the *skills* layer.

The three-layer approach makes two design concepts paramount. The first is to separate out deliberative functions that may be too expensive to meet response-time requirements. This allows the executive and skills layer to function as a responsive outer loop control system. Second, functionality that tends to be reusable across different applications and platforms is separated from less reusable skill-layer functions. These separations have proven invaluable for accommodating frequent design changes in project components and facilitate software reuse for substantially different platforms and missions.

Apex differs from typical 3-layer approaches in ways that pose significant integration challenges. First, deliberative processes are invoked and governed directly by the executive rather than running concurrently as a peer process. This simplifies the interaction between the 2 layers and is useful for managing use of computational resources, but requires that the executive incorporate deliberation management functions. Second, both the executive and deliberation layers consist of loosely coupled functional modules rather than a single monolithic subsystem. Deliberation modules (solvers) may perform completely distinct problem-solving tasks or may, as discussed below, address the same problem but vary in speed, expected plan quality and the conditions in which expected plan quality is higher than that of alternative solvers.

The executive is designed as a set of Reasoning and Control Services (RCSs) designed to meet demanding system response time requirements. The simplest and most fundamental service is *dispatch*. In a completely

predictable world, the executive would receive a schedule specifying every action to be performed during the mission down to the lowest level of detail – i.e. a list of command messages to be sent to subsystems and an exact timepoint at which each is to be sent. The Apex executive can be viewed as a dispatcher plus a set of RCSs that either interpret input conditions, manage internal state (such as memory usage) or dynamically control changes to the dispatch schedule. Many of these services are discussed in (Freed et al. 2005).

ARP autonomy is served by two critical software components in addition to Apex: an Obstacle Field Route Planner (OFRP) and Control Law (CLAW) software used as the skill layer for controlling the UAV platform. OFRP (Howlett, Schulein and Mansur 2004) generates 2-D route plans that avoid known obstacles. The algorithm uses a four phase approach: Voronoi graph generation from obstacle edges, graph culling using binary space partitioning, shortest path search using Eppstein's search method (Eppstein 1998), and path smoothing using binary space partitioning again. Apex calls OFRP in several conditions: during initial mission planning prior to takeoff in order to estimate flight times between all pairs of targets; to update these estimates when targets are moved or added; prior to transiting to the next target specified in the plan; and whenever a new obstacle is detected while in flight. Obstacle locations may be predefined or added dynamically, either by onboard sensors or by human operators on the ground.

Control Law (CLAW) software provides attitude stabilization and waypoint guidance control. Given vehicle state estimates and three-dimensional OFRP-modified waypoints from Apex, the Path Smoother subcomponent returns a larger list of waypoints. These define a smooth path between the original waypoints, each defining a radius used to construct a corridor for the smooth path. Next, a velocity profile for the path is calculated, taking into account user-supplied values for maximum bank angle and cruise speed.

One key feature of the flight control system is the ability to command heading independent of path. Apex uses this capability to dynamically coordinate flight behavior with payload (sensing) behavior in order to acquire sensor data about targets of interest. For example, whenever the vehicle is within 25 m of a target and below 3 m/sec total airspeed, Apex points the vehicle and camera at the target. More generally, this capability enables a wide range of observation behaviors (pirouettes, lateral sweeps, target-tracking arches,…) that produce fundamentally different sensor data products and serve different observation goals.

Surveillance Planning Algorithm Selection

Anticipating needed classes of extension is especially important for applications where requirements are either not well-understood at the outset or are likely to evolve during the operational lifetime of the autonomous system. For example, the Autonomous Rotorcraft Project has as an objective to provide a flexible capability to use a single UAV to maintain situation awareness at spatially separated sites. Both NASA and the U.S. Army, joint sponsors of the project, carry out missions and routine operational activities that fit this general description. However, there is no well-defined concept of operations for using autonomous UAVs in this capacity.

The need to adapt to evolving requirements is especially acute for the problem of generating surveillance mission plans. Missions can vary in ways that predictably have a large effect on planning algorithm effectiveness and thus on the kind of planner one would choose to design to generate a plan. For example, some missions will involve a greater number of targets and some fewer. In some, the targets will be close enough together that transit time will depend strongly on aircraft state and a model of its flying characteristics will be required to compute the value. Targets may fall into spatial patterns that can be used to simplify the planning problem (e.g. clusters, globular dispersions) or may tend to uniform spatial distributions. Targets may be of equal value or value might vary widely. Constructing a single planner that produces good surveillance plans in all conditions presents a difficult, perhaps insurmountable challenge. Anticipating which conditions will be encountered in operational use, and thus what kinds of planner(s) to build, seems equally difficult.

An alternative is to develop a variety of surveillance planners with varying strengths and weaknesses and a capability to select the best planner for the current mission. This would not only provide extensibility needed for evolving requirements, it would also allow flexibility for different missions in the same operating context and adaptability to changes in mission definition during flight. It does, however, present several challenges:

- Identifying attributes of surveillance missions that are predictive of algorithmic performance in the mission
- Creating a set (portfolio) of surveillance algorithms that collectively provide strong performance across the space of possible missions
- Calibrating each algorithm's performance with respect to identified mission attributes
- Classifying a specific (current) mission in terms of these predictive features

Surveillance mission types

To define the space of surveillance missions, we looked to the formal structure of the problem to identify candidate dimensions. As described earlier, such missions can be defined by a set of observation targets and functions for each specifying the time-varying likelihood that an important event will occur, the time-varying cost of not observing an event once it has occurred, and the time-cost

of observing a given target to detect the event of interest. Based on an analysis of variables in our fire incident models (Freed, Harris and Shafto 2004) , we chose to focus on the following five dimensions: number of mission targets, spatial scale of the operational area scaled against vehicle turn radius at cruise speed, the existence of spatial structure (e.g. clusters, enclosures) across the set of targets, variability in target value (asymptotic incident cost) and variability in the rate at which cost accumulates after the incident begins.

Surveillance mission planners

As noted, the role of a surveillance planner is to generate a sequence of transits and observations that minimizes expected cost of ignorance for the mission. This problem bears some resemblance to the Traveling Salesman Problem (TSP) in that both are concerned with finding a cost minimizing path across a set of targets. However, unlike TSP, the surveillance problem (SP) allows for the possibility that some sites should be visited more often than others due to differences in, e.g., their importance and in the rates at which observed information becomes obsolete (i.e. the rate at which ECI accumulates). It may be best to omit visits to some (possibly most) sites entirely in order to observe the most important ones at a higher rate. Surveillance scheduling thus combines task ordering with task selection, a combination notorious for increasing the computational complexity of any solution.

In an Orienteering Problem (OP) (Golden, Levy and Vohra 1987), the goal is to choose a path that visits an optimal subset of targets given a variable reward value for each target, a traversal cost for each pair of connected targets and some maximum cumulative cost for all traverses. This resembles the surveillance problem better than TSP, but SP has important distinct characteristics. First, SP typically entails multiple visits to a target during a mission, with value accrued each time, whereas OP only yields value on the first visit. Second, the value of a visit is not a constant, but a function of several factors including time since the last visit. Third, cost to traverse between a pair of targets may be a function (e.g., of vehicle state and wind) rather than a constant.

We then constructed two surveillance planning algorithms as simple examples of two distinct approaches. The first planner looks for the best repeatable cycle (patrol pattern) using an algorithm based on 2-Opt TSP (Reinelt 1994) that accounts for vehicle state when computing transit cost and allows fractional cycles. The second planner (WAM) uses best-first search with replacement, using a heuristic based on target proximity, obsolescence (time since last visit) and centrality of target in the mission space to incrementally construct a full mission plan. WAM was also able to compute state-dependent transit cost.

Calibrating Algorithm Performance

To characterize algorithm performance, we initially created a test-set containing exemplars of 243 mission types – 3 values in each of the 5 previously listed mission-classifying dimensions. One factor that constrained the size of the testbed is that we also chose to evaluate human performance at the surveillance planning task to baseline current state of practice. It took human subjects approximately 6 hours to work through one exemplar of each of the 243 mission types (Freed and Shafto 1994).

The study showed that humans did better in some conditions – i.e. found a plan that was more effective in reducing total mission ECI – but the algorithms did better in most conditions and significantly better overall. An analysis of the study results was compiled into a Performance Profile Table (PPT) indicating the best algorithm (including humans as a special case) for generating a surveillance plan for any given mission type.

Following this initial study, we created test-set generation software able to create a wider range of mission exemplars than appeared in the original set. One use for this software has been to help analyze our planners and choice of dimensions for distinguishing mission types. For example, a study to determine the potential ECI improvement from correctly selecting between our two planners showed that, in the 30 conditions (mission types) examined, the difference in performance between planners was negligible in many conditions and as high as 60% in others.

Classifying Missions and Selecting an Algorithm

Another capability of the test-set generation software is to automatically generate a PPT for a given portfolio of surveillance planning algorithms. Combined with a capability to classify algorithms so as to index into the table, this provides a basis for automatics selection of a planning algorithm. For example, given a mission consisting of 14 targets in a globular dispersion pattern, large spatial scale (compared to vehicle turn radius) and a uniform distribution of maximum-cost and cost accumulation rate values for all targets, the mission might be matched to the PPT entry for the mission-type *16:large:globular:uniform:uniform*. In this case, the WAM algorithm would be selected.

The difficulty of creating a classifier for surveillance missions can be understood by examining each classifying dimension individually. Four are metric dimensions that present no significant classification problem. For example, to classify a mission based on number of targets given a PPT with entries in that dimension for 4, 8, 12, 16, (etc…), the classifier needs only to select the nearest value. A sensitivity analysis may be needed to determine required density of table values, but classification is unproblematic.

The one non-metric dimension required determining which classification of spatial structure most accurately describes a set of targets – in our tests, these include: *2-*

cluster, n-cluster, globular, enclosure and *uniform*. Rapid classification is required to support potentially frequent replanning and use by planners to analyze target subsets. A method we found that effectively balances speed and accuracy involves generating a histogram of normalized, length-sorted distance pairs and extracting simple classifying features such as the 33rd percentile distance and mass fraction of the first quartile (see example in Figure 2). With 12+ targets, classifier accuracy ranges .from .8 to 1.0 for different spatial categories.

Flight Tests

The ARP system has been flying autonomously since March 2004 and has been flight-tested approximately weekly for 30-60 minutes per flight. Frequent testing has made it possible to incrementally add to and modify the autonomy subsystem that takes close account of lessons learned. The system has been flown in the described configuration and capabilities since November 2005. In addition to flight tests, it has been successfully demonstrated in a variety of larger exercises including events sponsored by the Disaster Area Relief Teaming center, the U.S. Forest Service and the Naval Postgraduate School/U.S. Special Operations Command.

Conclusion

The unmanned aerial system described here demonstrates a capability for fully autonomous and mixed-initiative execution of surveillance missions making efficient use of limited airborne sensing resources. The practicality of this system and approach has yet to be demonstrated in an operational context, though weekly flight tests demonstrating system reliability and successful recent demonstrations to operational organizations give positive indications.

Future steps include incorporation of general-purpose AI planning capabilities for more versatile at-target sensing actions and recoveries, construction of more sophisticated surveillance planning algorithms and extension of the approach to multiple UAVs.

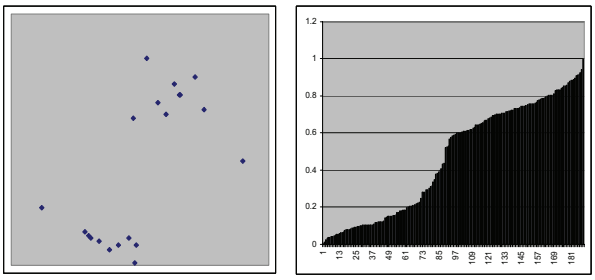

Figure 2. Multicluster pattern and distance histogram

References

Bonasso, R.P., Kortenkamp, D., Miller, D. P. and Slack, M. G., "Experiences with an Architecture for Intelligent Reactive Agents" *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

Eppstein, D., "Finding the *k* Shortest Paths," SIAM Journal on Computing, Vol. 28, (2), pp. 652–673, 1998.

Freed, M., Bonasso, P., Dalal, M., Fitzgerald, W., Frost, C. and Harris, R., 2005. An Architecture for Intelligent Management of Aerial Observation Missions. *Proceedings of the American Institute of Aeronautics and Astronautics Intelligent Systems Conference.*

Freed, M., Harris, R. and Shafto, M.G. (2004) Measuring Performance at UAV-Based Autonomous Surveillance. In *Proceedings of the 2004 Performance Metrics for Intelligent Systems Workshop*, Gaithersburg, MD.

Freed, M. and Shafto, M., "Human versus Autonomous Control of UAV Surveillance" in proceedings of the AIAA 1st Intelligent Systems Technical Conference, Chicago, IL, September 2004.

Gat, E. 1998. Three-Layer Architectures. In *Mobile Robots and Artificial Intelligence*, Kortenkamp, D., Bonasso, R. P., and Murphy, R., Eds.: AAAI Press.

Golden, B., Levy, L. and Vohra, R., 1987. The Orienteering Problem. *Naval Research Logisitics*, 34:307-318, 1987.

Howlett, J., Schulein, G., and Mansur, H., "A Practical Approach to Obstacle Field Route Planning for Unmanned Rotorcraft," in proceedings of the American Helicopter Society 60th Annual Forum, Baltimore, Maryland, June 2004.

Musliner, D. and Pell, B (eds), 1999. Working Notes of the 1999 AAAI Spring Symposium on Agents with Adjustable Autonomy. AAAI Technical Report SS-99-06.

Office of the Secretary of Defense, 2005. "Unmanned Aircraft Systems Roadmap 2005-2030."

Open Geospatial Consortium Inc., 2005. OpenGIS Sensor Planning Service. Reference number OGC 05-089r1.

Reinelt, G. (1994) "The Travelling Salesman. Computational Solutions for TSP Applications," Vol. 840 of Lecture Notes in Computer Science, Springer-Verlag, 1994.

Whalley, M., Freed, M., Harris, R., Takahashi, M., Schulein, G., and Howlett, J. "Design, Integration, and Flight Test Results for an Autonomous Surveillance Helicopter." *Proceedings of the AHS International Specialists' Meeting on Unmanned Rotorcraft*, Jan. 2005. Mesa, AZ.