Exploring Network-Centric Information Architectures for Unmanned Systems Control and Data Dissemination

Michael R. Clement,* Eugene Bourakov,† Kevin D. Jones,‡ Vladimir Dobrokhodov§
Naval Postgraduate School, Monterey, CA 93943

The exploration and development of an information architecture for networked unmanned systems is described. The unmanned systems discussed utilize standard components for guidance and navigation, coupled with additional computing devices for interfacing with a network. These platforms in turn communicate with a broader network of devices, applications, and users via a variety of wireless network links. Networking a platform that is traditionally operated via serial control links and analog sensor downlinks provides two distinct advantages: (i) high-level control, or "tasking," of the platform is easily extended from the single operator to any authorized user on the network; and (ii) sensor data and status information may be disseminated rapidly across the network to all interested recipients. The architecture developed through this exploration is applied in a prototype UAV which is utilized as both a high-resolution imaging platform and a wireless network relay. Testing and evaluation of the architecture occurs on an ongoing, quarterly basis through a cooperative field experiment program run by U.S. Special Operations Command and the Naval Postgraduate School.

I. Introduction

NETWORKS AND NETWORKING TECHNOLOGIES have penetrated diverse markets and arenas over the past several years, becoming a pervasive way of communicating information between humans and machines. In everything from banking to telephony, and from factory automation to control of public utilities, organizations have opted to switch from the model of dedicated, two-way control and data links to networks where many systems can interact, sharing, aggregating, and synthesizing information in complex ways to support advanced monitoring, analysis, and control of those systems. However, in the arena of unmanned systems, the traditional operational model of a single control station controlling a single unmanned system remains nearly ubiquitous. In the current climate, particularly in the defense domain, there is a strong push toward network-centric operations and distributed operations. This provides an impetus for exploring alternative methods of providing access to unmanned systems as information resources. These resources, coupled with an appropriate network-centric information architecture, could provide users with immediate access to both real-time and archived information associated with a resource.

The concept of network control of unmanned systems has been considered by others. However, the concept of network-enabling the platform itself and developing a comprehensive information architecture to allow access, both to task the platform and to retrieve data from it, is relatively unexplored. Several papers address incorporating a wireless network capability with an unmanned system [1-3]. Most are focused on either the dissemination of data from the platform or implementing distributed control algorithms. The DARPA HART program [4] does offer mission tasking over a network, but implements this through additional equipment collocated with the ground control station, maintaining the traditional ground control to platform communications link. One noteworthy example of a networked information architecture that directly integrates with the platform comes out of Idaho National Laboratory [5], describing an end-to-end workflow from mission planning through downloading of processed data from the unmanned platform. However, their approach assumes all mission planning and platform tasking is done pre-mission. Another good example comes from a research group at Berkeley [2] who developed a networked architecture for mission tasking and UAV collaboration. Their design goals are the nearest to those in this paper, though they place a greater emphasis on collaborative platform tasking.

* Research Associate, Dept. of Information Sciences, Center for Network Innovation and Experimentation.
† Research Associate, Dept. of Information Sciences, Center for Network Innovation and Experimentation.
‡ Research Associate Professor, Dept. of Mech. & Astronautical Eng., Associate Fellow AIAA.
§ Research Assistant Professor, Dept. of Mech. & Astronautical Eng., Member AIAA.

This approach is also novel from a user's point of view in that the replacement of platform control with higher-level tasking alters the requirements for human-machine interaction. There has been work done in identifying and modeling situational awareness for unmanned systems [6,7]; most of this has been aimed at understanding and improving the current mode of operations, though some has extended to multiple operators using multiple platforms. Other approaches, including [8], propose opening up low-level control of a platform to the networked user: for instance, translating mouse movements over a map into turn-rates provided to an aircraft.

During the past several years, in support of the quarterly, joint Cooperative Field Experiments conducted by U.S. Special Operations Command and the Naval Postgraduate School, developments have been made to provide an information architecture where the unmanned system becomes a node in the tactical network, allowing for remote users anywhere on the network to submit tasks and retrieve sensor data. This idea of integrating sensors and systems into the tactical network is a central focus of the experimentation program [9]. The advantages of such a scheme are many. This approach puts control and information flow into the hands of the end-user (not necessarily the Ground Control Station operator); perhaps someone sitting in an office in Langley, or an operator working alone from a forward location. As a network node, the unmanned system could provide connectivity between forward users and the Tactical Operation Center (TOC). Multiple unmanned systems could share information amongst themselves, and work out safe, cooperative behaviors to share the same operating space and engage in complex joint tasks.

In this paper we present our work to date developing an information architecture which encompasses both tasking an unmanned system and disseminating data from that system while it is deployed. Our approach considers the unmanned system as a networked node running software services that may be accessed over the network by applications residing on a user terminal (e.g. a laptop or ruggedized PDA). We discuss both the tasking and high-level control model, and the various methods by which data from the platform is made available to the user during the platform's deployment. The network architecture enabling communication between the platform and end-users is also discussed. We then present specific applications of this architecture utilized in our field experiments, using a prototype network-enabled UAV equipped with a high-resolution imaging system. This includes the case of the forward-deployed mission, for which the platform is tasked to navigate to a network-austere location and connect with a single user carrying a network device; this user may then task the platform for a period of time, and finally send it back to its home location.

The remainder of this paper is as follows. Section II discusses the design considerations and general implementation of the network-centric information architecture. In Section III, the application of this architecture to a prototype networked UAV is described. Section IV provides results from field experiments along with a description of tested integration with other analysis and end-user applications, and Section V completes the discussion with several conclusions and an outline of future directions.

II. Network-Centric Information Architecture

As mentioned above, the current modus operandi dictates that an unmanned system consists of one platform and one ground control station (GCS), with perhaps one or more viewers for the sensor data. This setup establishes a very specific path for command and control (C2) and data dissemination. All C2 is communicated directly between the GCS and the platform via some form of serial-based RF data link. By design, this enables exactly one operator to command the platform at any given time. Live sensor data such as video is generally provided across an analog or serial link, and is effectively disjoint from all C2 aspects of the aircraft. Unlike the C2 stream which is point-to-point by design, sensor data is generally broadcast – anyone with a compatible receiver may receive the sensor feed. However, this still requires all recipients to be within radio proximity of the platform.

Our network-centric information architecture considers each unmanned system as *just another node in the network*, capable of exchanging information and both performing services for and accessing services from other applications distributed across the network. This design is highlighted in Fig. 1. Each unmanned system – represented here by a collection of U*x*Vs – connects directly to the same network as both servers and end-user applications, and has equal connectivity to the larger Internet (or Global Information Grid in the defense domain) and with it any commercial networking apparatus (satellite or cellular handhelds, et cetera).

As shown in Fig. 1, there are several distinct types of entities that interact with the networked unmanned system. One is the local operator or observer, accessing the platform from a laptop or handheld general-purpose computer, or potentially a cellular or other miniature handheld device. These individuals most likely interact with the platform directly via the wireless network connection, which in our implementation is most often an ad-hoc wireless mesh network. Whether the individual is adjacent to the platform in the network, or one or more hops away connecting via mesh relays, nearly all of the data exchange with the platform occurs without the aid of an intermediate server.

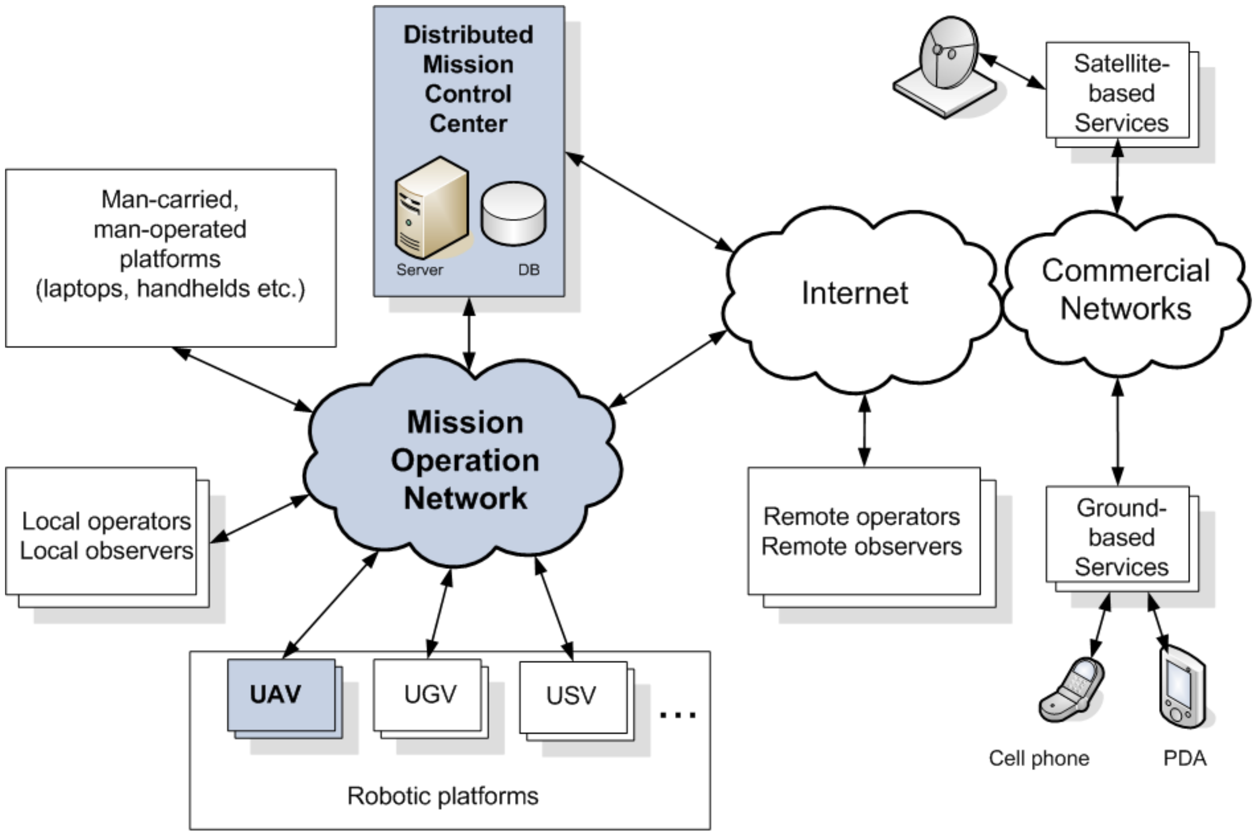
Distributed Mission Control Center
Server
DB
Satellite-based Services
Man-carried, man-operated platforms (laptops, handhelds etc.)
Internet
Commercial Networks
Mission Operation Network
Local operators Local observers
Remote operators Remote observers
Ground-based Services
UAV
UGV
USV
...
Robotic platforms
Cell phone
PDA

Figure 1. Network-Centric architecture overview.

Another type of entity is the remote operator or observer, who may interact directly with the unmanned system, but may also interact with an intermediate server, which acts as an authenticator, prioritizer, and translator for incoming requests. Sometimes the remote user exchanges data via a different protocol which requires translation into the lingua franca. Other times, several users want the resource at nearly the same time, requiring mediation through placing priorities on each request and queuing requests until the platform (or *a* platform in the case of cooperative behavior) is available. An intermediate server may also download stored information from the platform, acting as a distributor to decrease the load on a constrained network link. One more case would be one U*x*V tasking another U*x*V to support or complement its mission. A final type of entity is a data-processing system which interacts with the platform, gathering data from it in order to generate a new information product; these will be discussed later in the paper, and include geo-rectification and image-stitching applications that produce actionable imagery from a platform with a video or still image sensor.

There are four essential facets to this architecture. The first is network-enabling the unmanned system; a moving, sensing, acting platform must be capable of accepting commands from other network nodes, and providing information to them. This in turn implies a software architecture that allows authorized users to request services from the platform, not only in the form of information, but also activity such as movement and sensing. Third, the protocols used for communicating between the unmanned system and the rest of the network must be designed in a way that is both sufficient to articulate a task and standard such that the platform may integrate easily with a wide variety of devices and applications. A final facet is the application interface that allows users to utilize the platform. Although many such interfaces may exist, there are several important design considerations not only pertaining to ease of use, but also regarding sensitivity to the networked nature of the platform; certain constructs that may have worked well in a dedicated control link may no longer be effective. Each of these facets is explored further in the following subsections.

A. Network-Enabling the Unmanned Platform

In the current architecture, both the C2 and sensor components are designed as part of the network from the beginning. Network-enabled controllers encapsulate navigation and sensor functionality, exposing those functions as interfaces which are accessible from anywhere on the network. This encapsulation is shown in Fig. 2 for a prototype UAV platform. The orange blocks outside the dashed box in the figure depict the network-enabled extensions, both onboard the platform and on user terminals, which enable this new network-taskable and -accessible interface. A singular connection between platform and GCS is augmented with connections between a higher-level navigation controller and the network, enabling bidirectional communication with any authorized user on the network. Likewise, the sensor becomes network-enabled, making it possible to view sensor feeds from the platform, not only for operators within analog range, but for commanders watching from hundreds or thousands of miles away across a global reach-back network. Moreover, C2 and sensor components are now directly connected, facilitating tighter coupling between the activity of the platform and the produced data.

While there exist a number of candidate network technologies for communication between the platform and other systems, this research has predominantly emphasized mobile ad-hoc, or mesh, networks. These allow systems to join and leave the network "on the fly," and enable dynamic re-routing of information based on the environmental and link conditions at the time. Once communications reach a well-established hub such as a TOC, other mediums such as long-haul point to point or satellite links may be most appropriate to extend the reach of the platform to remote sites.

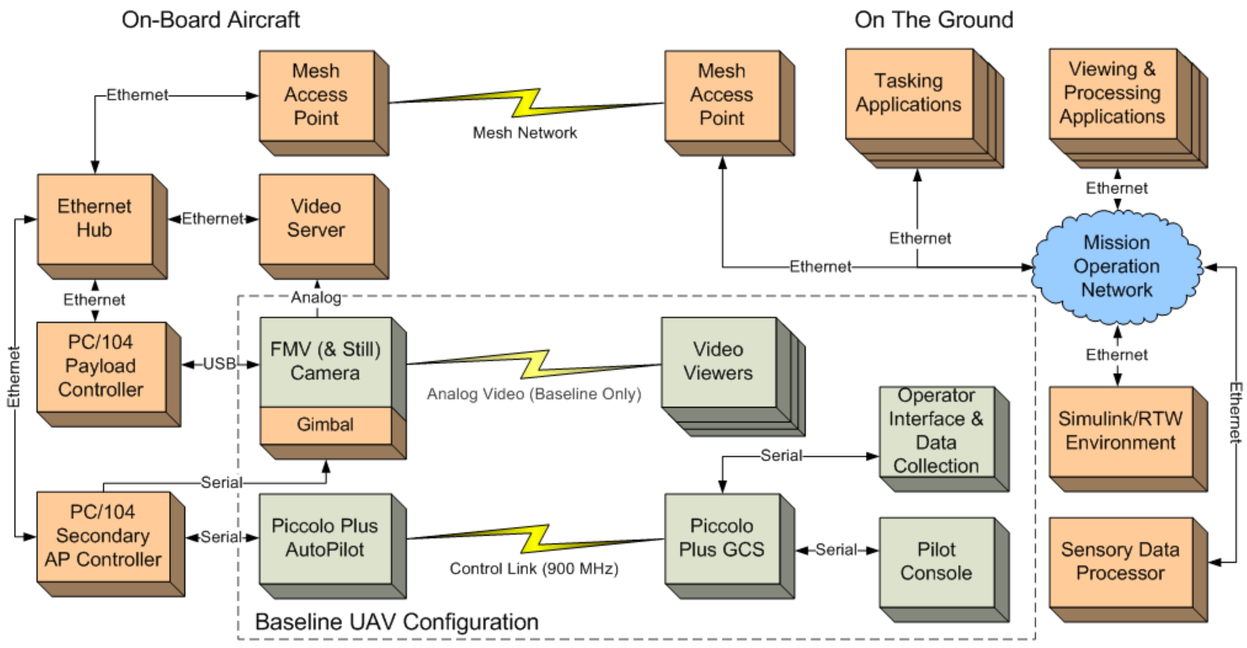

Figure 2. Example of base-platform architecture (in gray) and integrated network-centric architecture (in orange) for a prototype UAV.

B. Developing the Onboard Information Architecture

Onboard the unmanned system, there are four major functional groups to our architecture that are above and beyond the standard subsystems; these are shown in Fig. 3. The first is a Command Parser and Queue: this component is the primary interface between the platform and all tasking applications. Any request to navigate a path or use a sensor must be processed by this component. Its role is to mediate all requests, authenticating and validating all requests against provided mission parameters, and giving priority to certain tasks based on the user, application, or mission context.

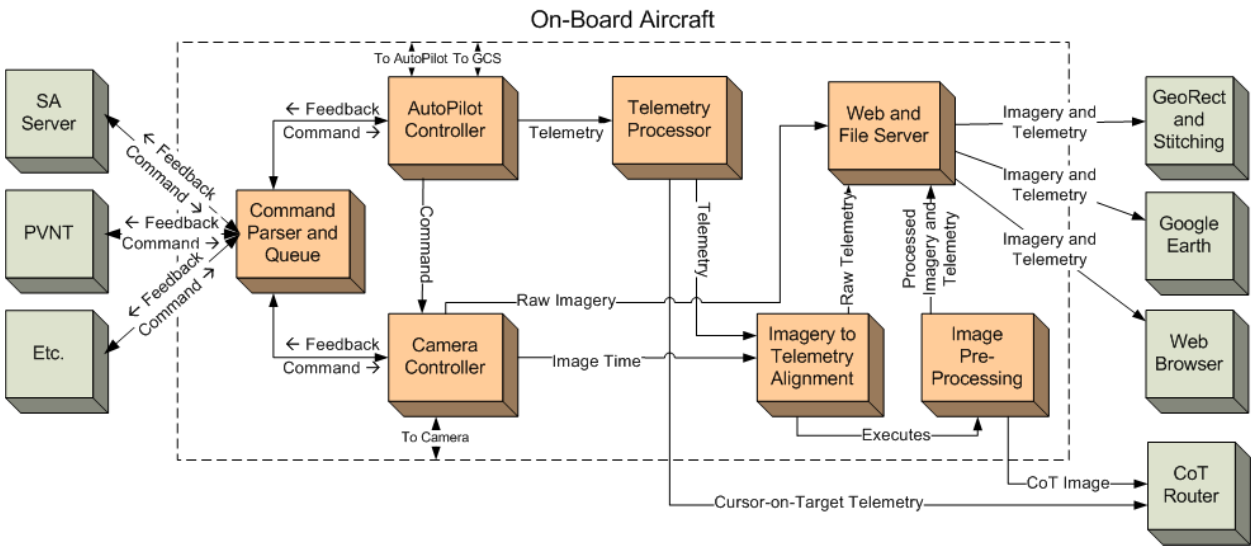

Figure 3. Example of information architecture for an unmanned system, from tasking through information processing and retrieval.

The second functional group consists of one or many Controller components, each of which interacts with a specific device onboard the platform. In our example architecture, there are two separate controllers: one interacts with the autopilot to plan and execute waypoint and path-following navigation, the other interacts with the sensor payload to manage data capture and collection. A controller accepts commands from the Command Parser and possibly from other Controllers, and produces command feedback and possibly raw data.

The third group is the collection of Data Processors. These components accept raw output from controllers and generate useful data products ready for the consumer. In our example architecture, there are three processor components: a telemetry handler, a timestamp aligner, and an image processor. The telemetry handler pre-processes raw telemetry from the autopilot controller, inserting additional metadata and converting to necessary intermediate formats for the other processors. It also outputs processed telemetry messages in formats that are usable by external applications. The second component is the aligner, which performs the crucial function of aligning the telemetry feed with sensor data. This allows the initial geo-rectification of sensor data without the addition of a dedicated INS. Additional sensor data processing functions are combined into a third component, which generates all processed data products ready for consumption.

Finally, an output group handles access to processed data. This is comprised of both services for pulling data from the platform, and the ability for the platform to push data to consumers. The former is exemplified by the web and file services available onboard, which provide access to anyone on the network equipped with a web browser or an operating system with the ability to share files across the network. However, other domain-specific applications, including Falconview and Google Earth, are also supported utilizing the appropriate protocols. Examples of the push method include the Cursor-on-Target (CoT) formatted messages generated by the telemetry handler and also by the sensor data processor. Under certain network conditions, the wireless network may be inadequate to support many users directly accessing services on-board the platform; for this reason, a separate archiving and data relay server has also been explored, such as the distributed mission control center shown in Fig. 1.

C. Unifying Network Communication Protocols

Another major facet of the system integration is the development of a uniform means of tasking the unmanned system and accessing its data products. During the early phases of development, several interfacing protocols were developed ad-hoc in order to allow different researchers to utilize a given platform; however, over time it became clear that a standard set of commands should be made available in one or more syntactical formats. These formats should be both stable and pervasive.

To accommodate both sets of needs, we adopted a framework for creating interpreters between the external (network) protocols and the internal tasking protocol. Each interpreter is fully customizable, conforming to a simple set of functions; this allows any researcher to develop a new language for tasking while having minimal knowledge of the internal workings of the platform. A simplified internal tasking language forms the basis for all commands;

each external request that arrives at the platform is decoded appropriately, and the corresponding internal command is evaluated based on user privileges and priority. The privilege and priority scheme is also fully configurable, allowing the interpretation of commands to be decided based on both the source and priority of the command and the current tasking status of the platform itself.

For initial implementation of an external language, we chose Cursor-on-Target, an XML-based protocol developed by MITRE for tactical data exchange [10]; CoT is a standardized and documented format, and it is in use by a sufficiently large body of users common to our experiments that it made sense as an initial interface. Other protocols under consideration for future development are JAUS [11] and STANAG 4586 [12]. Likewise, the means of accessing data products must be open and standard so that integration with any external platform or application is straightforward. For this we chose a combination of CoT output, a web server, and an SMB/CIFS file server. The latter two utilize protocols available by default on nearly every major operating system.

D. Designing User Tasking and Sensor Visualization Interfaces

The current architecture is the product of a long-term, iterative study, utilizing several communications technologies, network topologies, and protocols in the course of finding suitable mechanisms for effective tasking and control of each unmanned system and accompanying sensors. One of the most important parts of the architecture is a situational awareness tool providing two-way network communication with the onboard computer to allow a platform to be controlled by local or remote operators over a simple and meaningful user interface.

One of the challenges in designing an interface that interacts with the platform across a network is that it will under some conditions behave in ways that differ from a dedicated, real-time communications link. Some tasks are well-suited to IP-based communications, while tasks that are not latency-tolerant should be avoided. In earlier versions of both unmanned systems and user interfaces, the emphasis was on mimicking the capabilities of traditional unmanned systems control. For instance, directly controlling camera movement, such as with a joystick. However, the latency, jitter, and packet loss introduced by a packet-switched network made this approach untenable. Many operators would respond to latent camera movements by continuously actuating the same control until they perceived a change in the video. However, the combination of control and video latency caused a situation where, by the time the correct camera location was seen in video, the operator had already initiated the command to move further, causing overcompensation and overall decreasing the effectiveness of this control model.

Such effects, relatively unseen in existing platforms with dedicated control and video links, drove an impetus to change the model of user “control” to one of user “tasking,” and this in turn informed the user interface design. We adopted the mantra “fly the sensor, not the platform” and began redefining all parameters of sensor command and control in terms of the final data product. In this paradigm, the user specifies the location of the desired target along with parameters relevant to the data product (e.g. image resolution in units such as centimeters per pixel). The platform is responsible for determining how best to put the sensor on that target within its own operating constraints. For instance, a UAV with confined airspace may determine an appropriate side-look angle to track a target outside its flight boundaries.

This mode of operation has a few interesting implications. First, tasking operations are largely asynchronous; that is, the end-user may specify a task, shift to an unrelated activity while the platform carries out the task, and retrieve the final data product later at the user’s convenience. Alternatively, the user may desire to make small changes as the task is carried out, ensuring that the supplied target coordinates are accurate and that the data product is of sufficient quality. With the movement of the platform de-coupled from sensor pointing, attempts to use joystick-style sensor steering would end in frustration. Instead, by using a joystick-like interface to modify the given location of the target, the user may continue to adjust the task without having any knowledge of the platform’s movement. A prototype interface for high-level tasking is shown in Fig. 4.

Such an application allows any operator on the network with appropriate authorization to request an unmanned system to carry out a task. This means that an individual does not necessarily have to be in proximity of the platform in order to task it. Likewise, an operator deployed away from the GCS, but with a direct or relayed connection to the platform, may also task it.

This concept is taken even further with the development of tasking over handheld devices, such as the Blackberry Curve 8310 shown in Fig. 5. Such a mechanism adds a significant benefit: it leverages a widely available, reliable network commercially provided by domestic or overseas cellular phone services. In the current implementation, messages traverse either the public cellular network or a private, local cellular network to a central server. The server in turn translates and forwards the task message to the platform across the existing wireless network. In future implementations, the platform itself may be equipped with a cellular device to communicate with users directly via the cellular infrastructure.

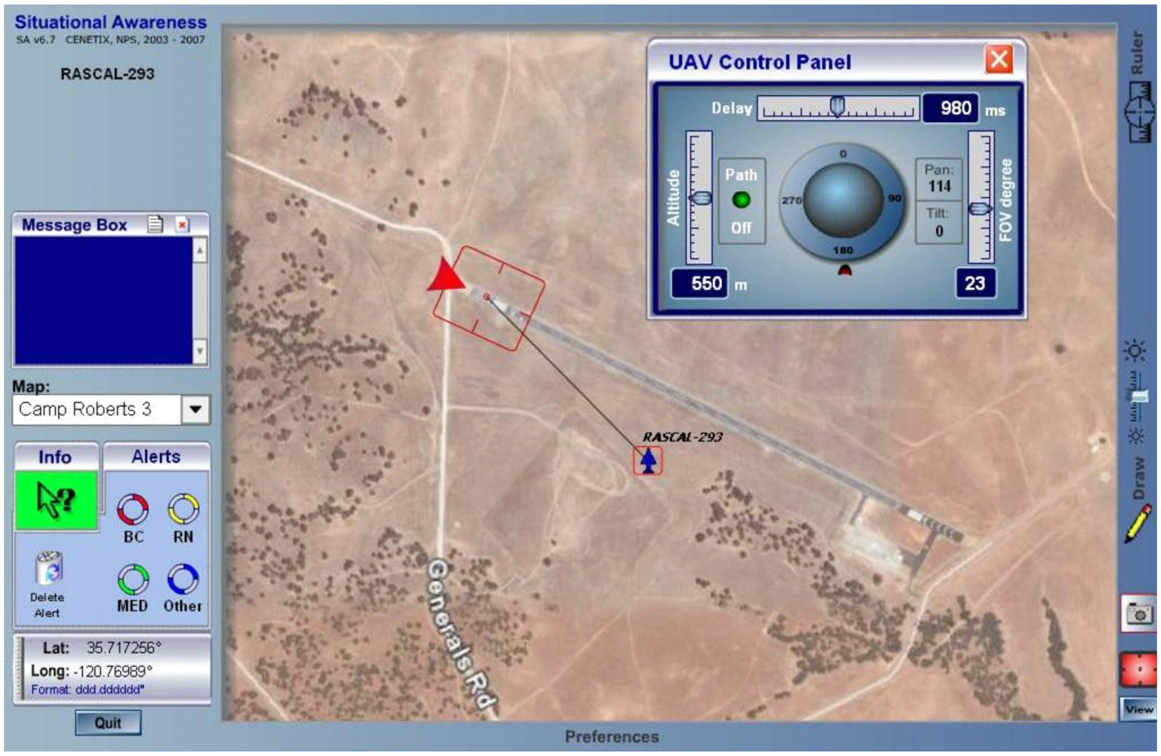

Figure 4. Situational Awareness user interface for mission tasking. The inset UAV Control Panel provides controls for the sensor, including UAV altitude, camera FOV, heading, and a variable delay used to compensate for shutter lag.

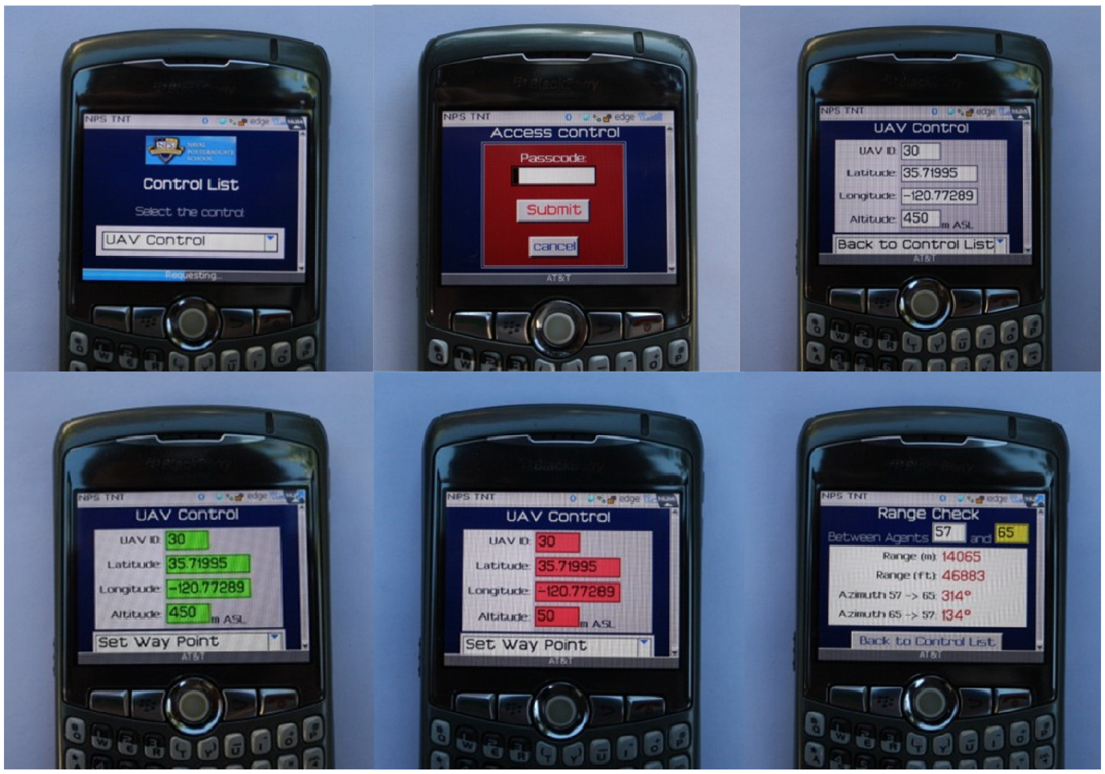

Figure 5. Control of a UAV from a handheld cellular device.

Implementation of cellular network infrastructure naturally enables another valuable feature: voice control. In many cases, even the most experienced operators have little to no opportunity to look away from the task at hand in order to concentrate on a computer screen. Unlike a visually-rich common operational picture tool, voice communication interferes substantially less with an operator's ability to carry out a concurrent task. This approach may be one of the few feasible solutions for tasking unmanned systems and getting response messages while keeping hands and eyes free for more immediate tasks.

The last decade's advances in VoiceXML, CCXML, CallXML, and other voice-based computer control techniques provide a unique background for this new research dimension - voice control of computer infrastructure, platforms, and sensors. Voice commands may be delivered to the platform for execution either over a wide area network connection, like the Internet, or from within the local cellular network infrastructure. The use of a combination of both may be very beneficial to extending this capability to the edge, providing any network- or cellular-equipped forward operator a means to directly interact with unmanned systems, not only by clicking and viewing, but by speaking and hearing.

The voice control solution represents a good example of an alternative approach to sharing situational awareness information between human and machine across a network. In an operating environment augmented by unmanned systems, this takes us significantly closer to the long-anticipated goal of seamlessly using natural language with robots. We call this approach Voice-on-Target (VoT), which also provides the platform with the ability to exchange data through synthesized voice. In result, operators can literally hear unmanned systems "talking" back to them, providing the current status of a task via voice report. This in turn enables operators to begin "sensing" more intuitively through unmanned systems, thus improving their cognition and situational understanding.

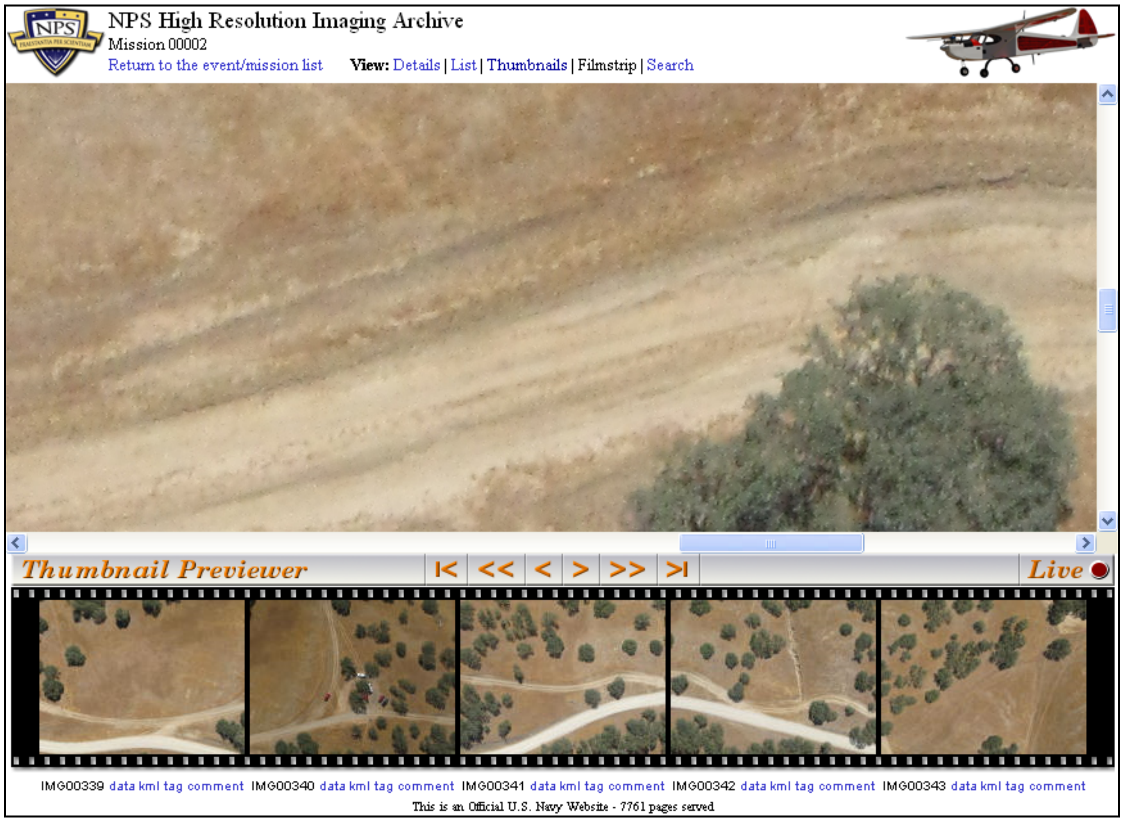
NPS High Resolution Imaging Archive
Mission 00002
Return to the event/mission list
View: Details | List | Thumbnails | Filmstrip | Search
Thumbnail Previewer
|< << < > >> >|
Live
IMG00339 data kml tag comment IMG00340 data kml tag comment IMG00341 data kml tag comment IMG00342 data kml tag comment IMG00343 data kml tag comment
This is an Official U.S. Navy Website - 7761 pages served

Figure 6. Web-based interface for accessing live and archived imagery.

Effective presentation of sensor data products is also essential. Sensor data should be displayed in a way that incorporates geospatial data and that is meaningful and intuitive to the user. For instance, sensor imagery may be overlaid on top of background imagery in applications like Falconview and Google Earth. The user may also want to view an archive of all data obtained so far during a specific tasking. A web interface providing access to live and archived data is one example of the latter. This interface could run onboard the platform itself, and would be immediately accessible to any authorized user on the network. As new data becomes available, the interface is automatically updated, enabling a near-real time view of data as it is obtained. The web interface is shown in Fig. 6; examples of imagery overlays are shown in section IV.

III. Integrating with Prototype Networked UAV

Taking the network-centric architecture described above, we are then able to leverage any unmanned system as an integrated, networked sensor platform. One example implementation is detailed throughout the remainder of this section; in this case, a high-resolution imaging system onboard a tactical-level UAV with the intended application of providing satellite-like imagery in near-real time.

It is important to note that many previous solutions have been proposed to taking digital aerial imagery from low-cost aircraft. There is a relatively long history of the use of low-cost imaging sensors in the hobby industry. There are even hobby companies that have addressed this capability with specialized hardware to interface with popular cameras [13]. We became interested a few years ago when participants at our field experiments literally taped a Commercial Off-the-Shelf (COTS) point-and-shoot camera to the bottom of a hand launched tactical UAV, put the camera in time-lapse mode, and shot a sequence of very nice photos through the entire flight [14]. While inspirational, this scheme had several distinct disadvantages: the photos could not be retrieved until the UAV returned; the full flight was recorded, including launch, providing the finders of a downed aircraft a means to locate the launch site; and there was no way to couple camera operation to the autopilot or to record telemetry for accurate image placement. Others have coupled the camera with the flight system closely enough to link imagery with telemetry [15,16], but again, could not retrieve imagery until the vehicle was on the ground. In yet another study, a cell phone with an embedded 5MP camera and GPS was programmed to shoot in time-lapse mode while recording the GPS location of the aircraft [17]. While elegant in its simplicity and extremely compact, there was no way to record pointing direction for the camera, and no reliable way to retrieve the photos in flight.

Our implementation utilizes an off-the-shelf lightweight research airframe with a custom payload including sensor and networking devices. The Sig Manufacturing Rascal 110, as shown in Fig. 7, is the airframe used. The plane has a 2.8m span, uses a 26cc gas engine, and has an all up weight of about 10kg, including 1500cc gas which provides a 1.5 to 2 hour flight time. The Rascal can be purchased in an Almost-Ready-to-Fly (ARF) form, making it an extremely cost-effective way to prototype new flight-control systems and sensors. There is ample cabin room for the autopilot and the desired computing hardware and sensors.

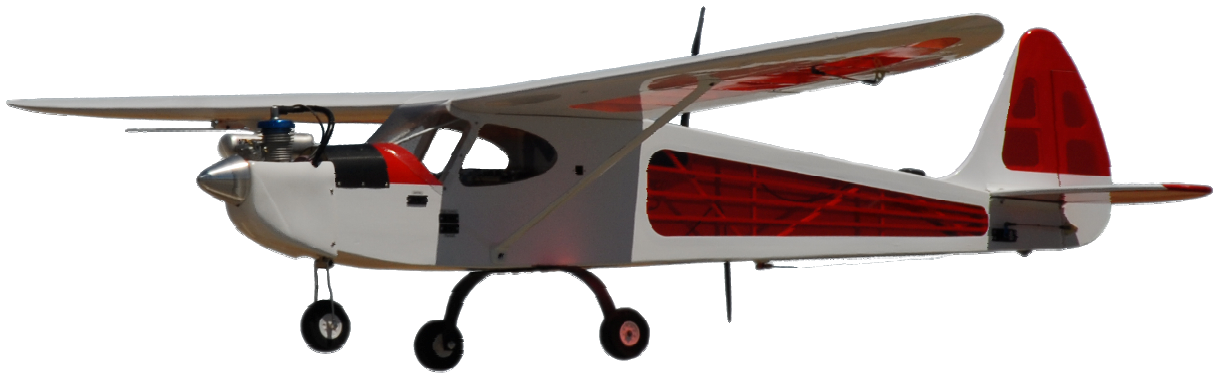

Figure 7. Sig Rascal 110 research aircraft.

The Piccolo Plus autopilot is used for primary flight control, with its dedicated 900MHz serial data link. The avionics also includes a pc/104 running a real-time OS that functions as a secondary flight controller, and a second pc/104 running either Windows XP or Linux which controls the imagery acquisition process and links all the other avionics components. In the original implementation, an ITT/Motorola MEA mesh PCMCIA card was used to supply the payload wireless link, but recently we moved to a Persistent Systems WaveRelay router. A Pelco NET300T video server is used to stream the analog video feed from the camera, and all network devices are linked through a Linksys 5-port hub. An overhead view of the avionics is shown in Fig. 8.

The camera used here is the Canon G9, with a 12MP sensor, 6:1 optical zoom lens, and optical image stabilization. In the first trials the camera was hard-mounted in the bottom of the aircraft, but in the present version it is mounted in a roll-axis gimbal to compensate for bank angle and provide a limited side-look capability. The camera and gimbal mount are shown in Fig. 9. The gimbal is driven by hobby servos, one for the roll axis, and one to operate the camera power switch. Both servos are driven via serial-to-PWM interfaces driven by the onboard computers. The gimbal is controlled by the real-time computer while the powering is controlled by the Linux computer, functionally separating sensor operation and platform movement.

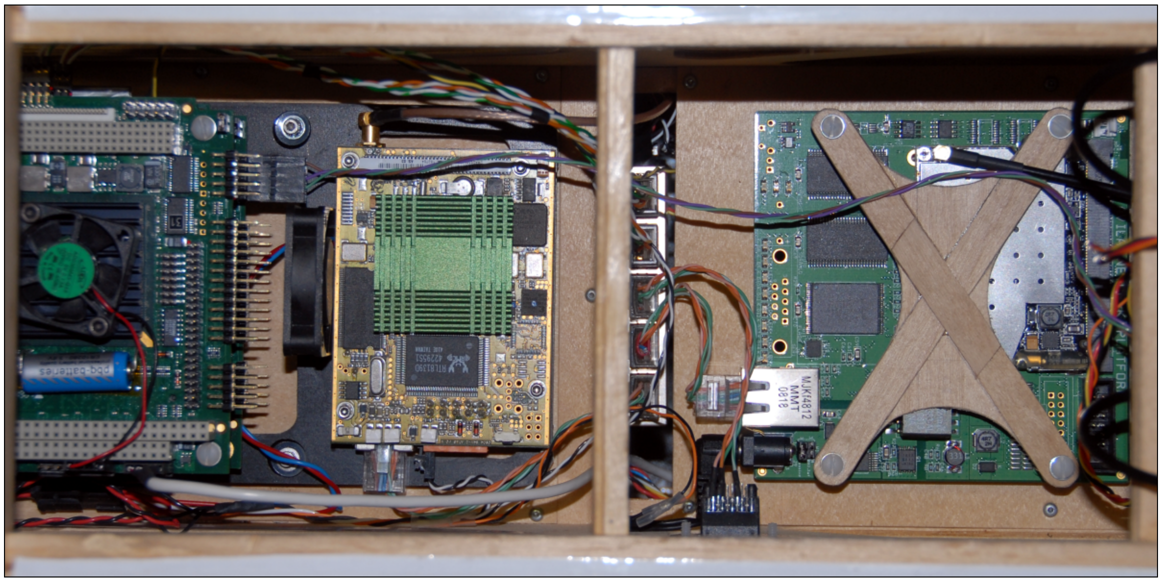

Figure 8. Avionics bay in the Rascal. From left to right: PC104 stack, cooling fans, Pelco video server, network hub, and WaveRelay radio.

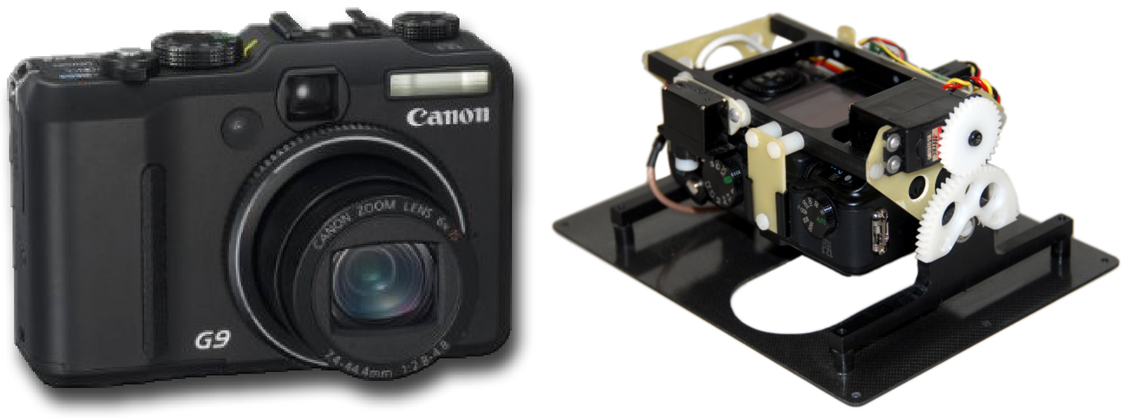

Figure 9. Canon G9 camera and roll-axis gimbal mount providing +/- 45 degree of roll compensation and side-look pointing.

While the network control and the high-resolution imaging system perform adequately with conventional waypoint navigation as used in virtually all fielded UAV platforms, we have been developing a path-following algorithm that uses a secondary controller to augment the autopilot control using an L1 adaptive controller [18-24]. Without going into detail, this system allows for much more aggressive flight performance than can be generated by waypoint navigation. In the current implementation, the user may scribble out a desired sensor path on a digital map; the path is sent to the UAV where the secondary controller generates an optimal flight path that will keep the sensor on the sensor path, as illustrated in Fig. 10. This approach enables the end-user, without any UAV-specific training, to task the platform to carry out complicated missions.

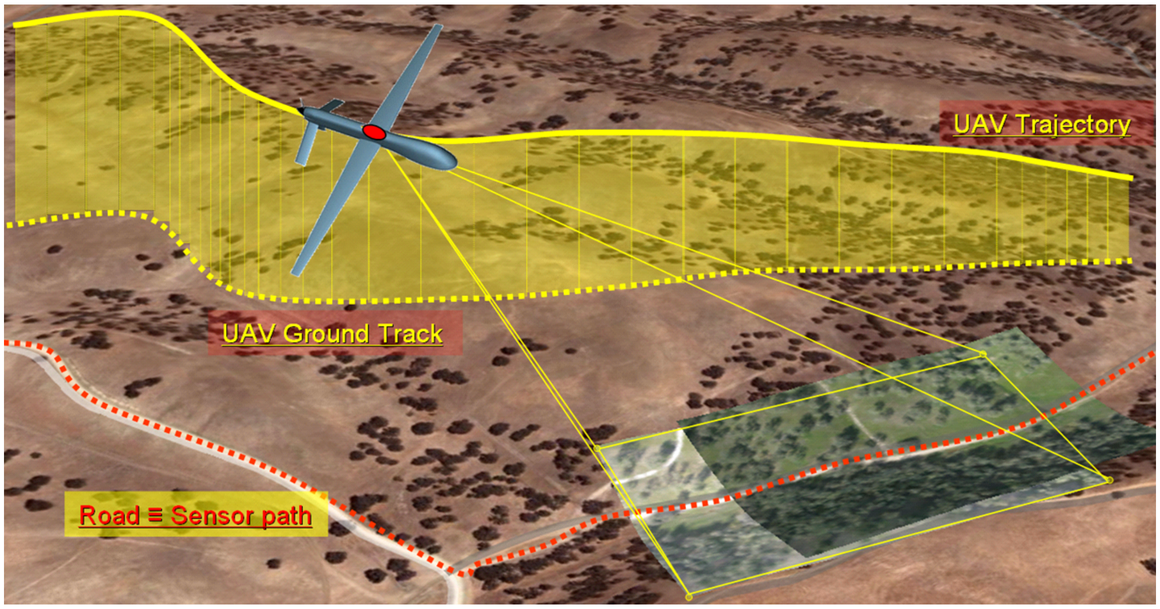

Figure 10. In the feature-following concept, the user defines the sensor path on the ground by scribbling the mouse cursor over a digital map. The sensor-path is sent to the UAV, and the secondary controller on the UAV determines an appropriate flight path and gimbal movements to keep the sensor on the path.

IV. Experimentation Results

As previously mentioned, all of this work was in support of the quarterly, joint USSOCOM/NPS Field Experiments, held at McMillan field in Camp Roberts, California. To be of practical value to the program, everything we develop needs to be demonstrated in the field under realistic operating conditions. In this section, sample results from several experiments are shown providing some insight into the capabilities of the system.

In the original implementation, with a body-fixed camera mount and the Windows driver using the Canon SDK, the most practical mission was precision imaging of specific, discrete locations. This was enabled by clicking on a point of interest on a digital map, and specifying a desired heading and footprint size on the ground. The secondary flight controller would generate a suitable flight-path (using waypoints) to fly over the target, and the camera would be activated at the appropriate time. Variations on this scheme included triggering the release of a small capsule at a precise location to simulate a medical supply drop, and the release of a micro air vehicle which would carry out a secondary ISR mission.

To improve both the inter-frame rate of imaging and the alignment of telemetry with imagery, the camera controller was rewritten using a Linux-based camera driver and new telemetry alignment code. With these improvements it was possible to produce continuous, overlapping imagery at standard flight altitudes. New task types, including continuous orbits and area sweeps, or “lawnmower” patterns, were added to the tasking interface. A roll-axis gimbal was also added to enable acquiring nadir imagery while banking and to allow limited side-look capabilities. Using the recorded telemetry from the autopilot, the current system generates KML files to produce ground-overlays in Google Earth. Sample images are shown in Fig. 11, taken from 300m AGL with a 20 degree FOV. An example of a complete path with continuous overlapping imagery is shown in Fig. 12.

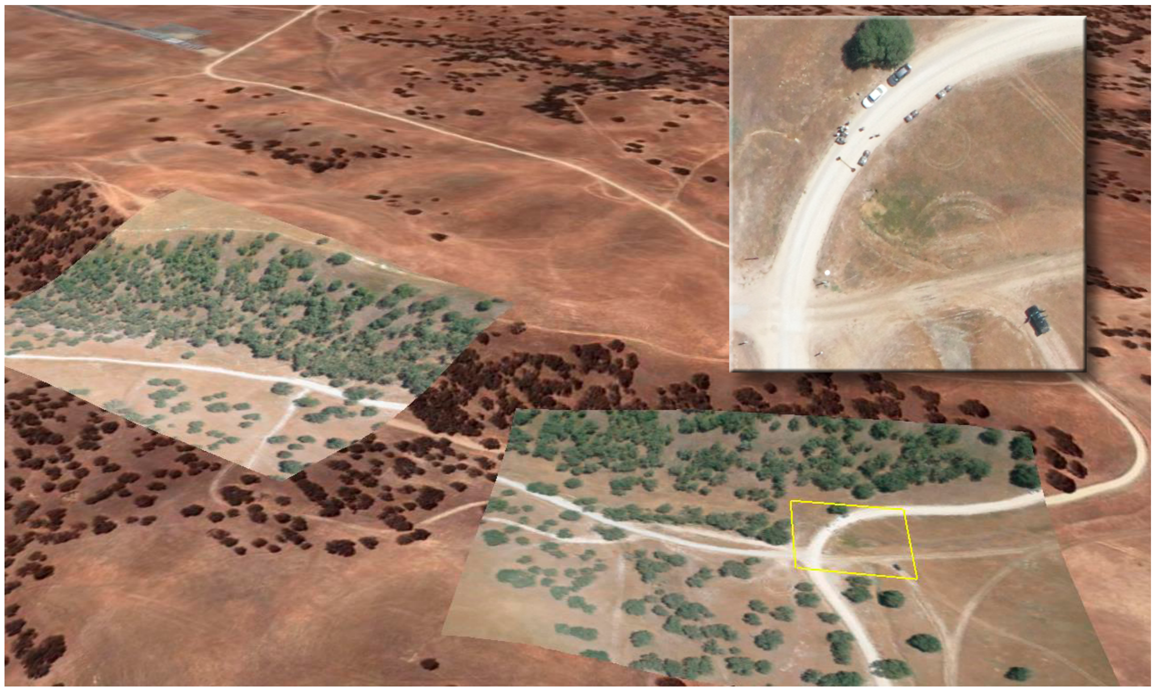

Figure 11. Sample photographs automatically placed over background imagery and terrain using only recorded telemetry for position. The inset photo illustrates the level of detail afforded by the system.

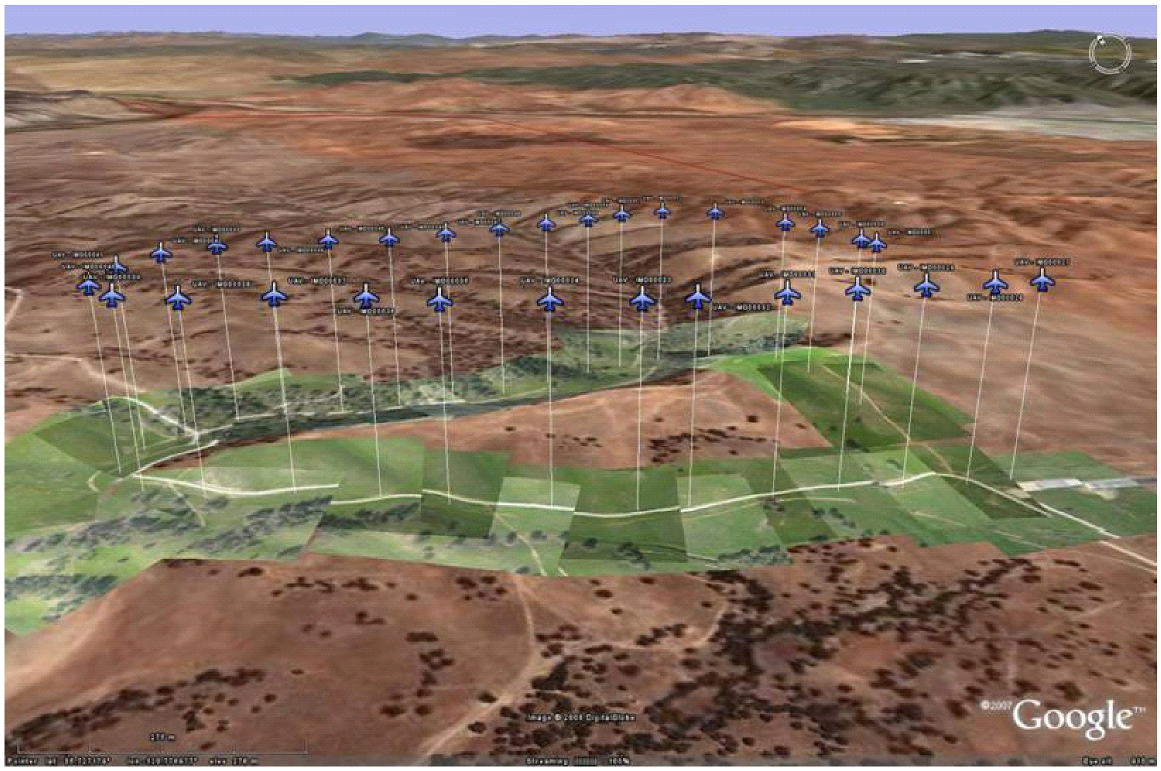

Figure 12. Example of continuous overlapping images captured during a path-following exercise, automatically placed in Google Earth. Mismatch of the image edges is due to lens distortion and telemetry errors.

The most promising applications of the high-resolution imaging system appear to involve georectification, georeferencing, and stitching of the imagery to provide a low-latency, low-cost, satellite-like imaging capability. These processes would likely be done on the ground using computationally-intensive software. A collaborator from the computer vision industry, 2d3, used their software package to consume 158 images along with the estimated aligned telemetry to produce the mosaic shown in Fig. 13, covering a 1km × 2km area on the ground. The composite shows the first step in the mosaic process, but does not include exposure correction, lens distortion removal and adjustments for terrain distortion. Interestingly, the typical alignment error of the discrete images, due to telemetry errors, was on the order of 20m, with heading errors as high as 30 degrees or more depending on the wind. In contrast, the average alignment error of the composite image was considerably smaller, due to an inherent averaging effect, and the heading error was all but gone. The photos were shot over an 18 minute period from about 600m AGL with about a 30 degree FOV, providing roughly a 7cm resolution.

Even without post-processing on the ground, direct image overlays in applications like Google Earth (Figs. 11 and 12) and Falconview prove useful for seeing an update-to-the-minute view of an area on the ground. During a recent experiment, initial attempts were made to directly import imagery taken by the prototype UAV into Falconview. Due to restrictions with resolution layers, the results were limited. However, work is ongoing to improve the speed and quality of displaying near-real time imagery into an end-user application.

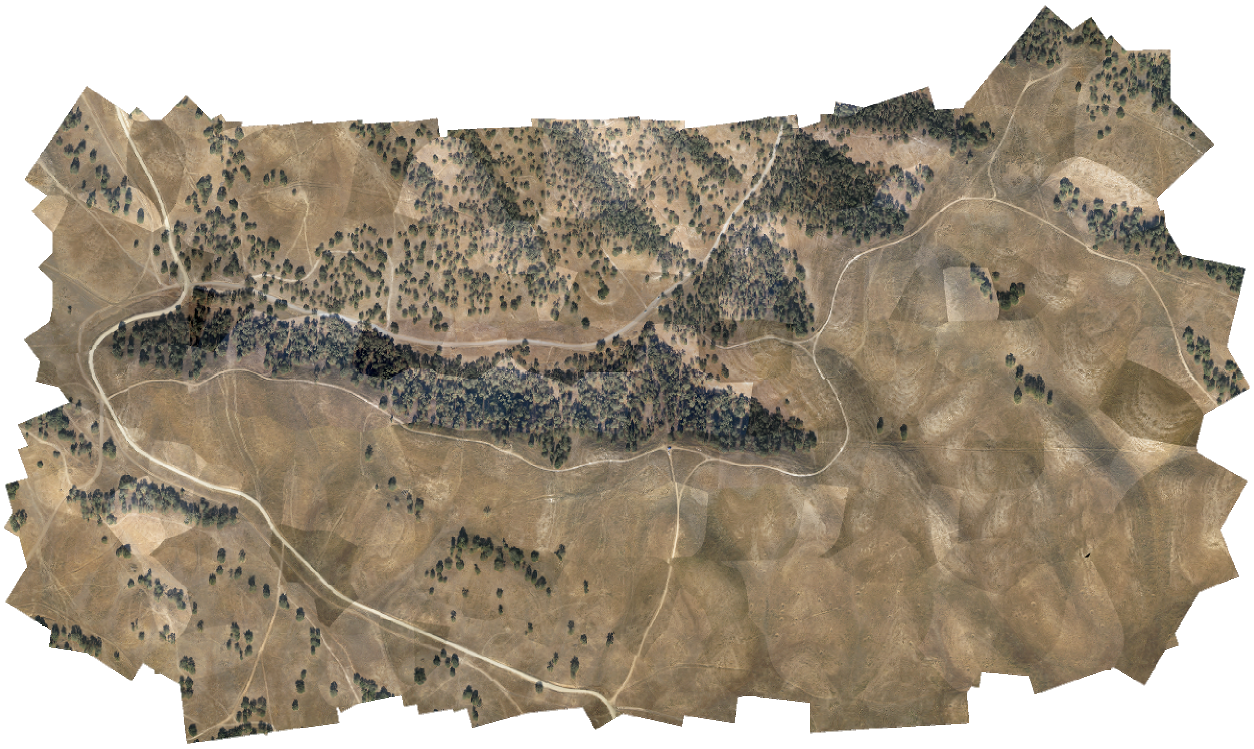

Figure 13: Mosaic of 158 images covering a 1km × 2km area, stitched together by 2d3 [25].These images were captured with the newer gimbaled G9 sensor. The near-nadir imaging makes stitching much simpler.

V. Conclusions and Outlook

In this paper, a network-centric information architecture for unmanned systems is presented, along with specific discussion relating to the implementation on a prototype networked UAV. The system enabled networked users to task the UAV to fly to specified coordinates and capture imagery of discrete targets, while the platform internal implementation ensured that the sensor was delivered to the desired target. Tight integration of tasking and control, navigation, and sensor control enabled the platform to deliver high-quality sensor data to both nearby and remote networked users while in-flight and in near-real time.

During field experiments, users have successfully tasked the UAV using several devices and applications, including cell phones, wireless mobile laptops, and workstations in a remote TOC. These tasks were processed and immediately executed by the aircraft, which in turn provided up-to-date, high-resolution imagery of discrete targets, orbited areas, and large swept areas. Image data from these tasks has been used directly for search of high-value targets, feature-matching, and image database updates.

Future work will include incremental process improvements as well as new task types. One area currently being investigated is dynamic path generation, allowing the end-user to scribble a path on a map which becomes the sensor path during flight. Cooperative queuing and assignment from a pool of assets is also being explored.

Acknowledgements

The authors would like to acknowledge the support and contributions of the researchers, staff, and collaborators from the Tactical Network Topology field experimentation program, for discussions, suggestions, and feedback during this exploration. Specifically, Dr. David Netzer, the recently-retired Program Director for the field experimentation program. Also, Dr. Alex Bordetsky, the Principle Investigator for Networks for the program, and the director of the Center for Network Innovation and Experimentation.

References

1. Christmann, H.C., Johnson, E.N., "Design and Implementation of a Self-configuring Ad-hoc Network for Unmanned Aerial Systems," AIAA Infotech@Aerospace Conference, Rohnert Park, California, May 7-10, 2007, AIAA 2007-2779.
2. Tisdale, J., Ryan, A., Zennaro, M., Xiao, X., Caveney, D., Rathinam, S., Hedrick, J.K., Sengupta, R., "The Software Architecture of the Berkeley UAV Platform," In Proc. of IEEE International Symposium on Intelligent Control (ISIC), October 2006, Munich, Germany.
3. Barton, J., Castelli, J., Chiu, C., Ferris, D., Hawthorne, C, Keiser, C., Marshall, S., Van Giesen, E. and Watkins, A., "Autonomous UAV Swarms for Detection and Classification of Chemical and Biological Agents," Presented at the AUVSI Unmanned Systems North American Conference, June 2008.
4. Pagels, M.A., "Heterogeneous Airborne Reconnaissance Team (HART)," project presentation, Defense Advanced Research Projects Agency, August 2008. http://www.darpa.mil/ipto/programs/hart/docs/HART_Overview.pdf
5. Hruska, R.C., Lancaster, G.D., Harbour, J.L., Cherry, S.J., "Small UAV-Acquired, High Resolution, Georeferenced Still Imagery," Idaho National Laboratory, September 2005.
6. Drury, J.L., Scott, S.D., "Awareness in Unmanned Aerial Vehicle Operations," International C2 Journal, Vol. 2, №. 1, 2008, pp. 1-28.
7. Taylor, R.M., "Human Automation Integration for Supervisory Control of UAVs," In Proc. of Virtual Media for Military Applications Meeting, pages 12-1 – 12-10, Neuilly-sur-Seine, France, 2006.
8. Crouse, J.D., Blue, P.A., Cobb, R.G., "Cursor-on-Target Control for Semi-autonomous UAS," AIAA Guidance, Navigation, and Control Conference, Honolulu, Hawaii, August 18-21, 2008, AIAA 2008-6798.
9. Bordetsky, A., Hayes-Roth, R., "Extending the OSI model for wireless battlefield networks: a design approach to the 8th Layer for tactical hyper-nodes," International Journal of Mobile Network Design and Innovation, Vol. 2, No. 2, 2007, 81-91.
10. Butler, M., "The Developer's Guide to Cursor on Target," The MITRE Corporation. Bedford, MA., August 2005.
11. Clark, M.N., "JAUS Compliant Systems Offers Interoperability across Multiple and Diverse Robot Platforms," AUVSI Unmanned Systems North America Conference, Baltimore, Maryland, June 28-30, 2005.
12. Platts, J.T., Cummings, M.L., Kerr, R.J., "Applicability of STANAG 4586 to Future Unmanned Aerial Vehicles," AIAA Infotech@Aerospace Conference, Rohnert Park, California, May 7-10, 2007.
13. URBI R/C Interface website, http://blip.com.au/.
14. Private communications at the USSOCOM/NPS TNT 06-3 Field Experiment, May 2006.
15. The MLB Company website, http://www.spyplanes.com/.
16. Patterson, M.C.L. and Brescia, A., "Integrated Sensor Systems for UAS," 23rd Bristol UAV Systems Conference, April 2008.
17. Lizarraga, M.I. and Illstrup, D., "Aerial Photography using a Nokia N95," project report, University of California, Santa Cruz, Dec. 2007.
    http://www.soe.ucsc.edu/classes/cmps290b/Fall07/UAVImageRegistration/NewSite/page9/files/pr_report.pdf
18. Kaminer I.I, Hovakimian N., Patel V., Cao C., Young A., Pascoal A., Dobrokhodov V.N, "Time-Critical Coordinated Path Following for Multiple UAVs via L1 Adaptive Output Feedback Controllers," European Control Conference, Kos, Greece, July 2-5, 2007, TuA08.1.
19. Kaminer I.I, Yakimenko O., Dobrokhodov V.N, Pascoal A., Hovakimian N., Cao C., Young A., Patel V., "Coordinated Path Following for Time-Critical Missions of Multiple UAVs via L1 Adaptive Output Feedback Controllers," AIAA Guidance, Navigation, and Control Conference, Hilton Head, South Carolina, August 20-23, 2007, AIAA-2007-6409.
20. Cao C., Patel V., Hovakimian N., Kaminer I.I, Dobrokhodov V.N, "Stabilization of Cascaded Systems Via L1 Adaptive Controller with Application to a UAV Path Following Problem and Flight Test Results," American Control Conference (ACC 2007), New York, July 11-13, 2007, WeC11.3.
21. Cao, C. and Hovakimyan, N., "Guaranteed Transient Performance with L1 Adaptive Controller for Systems with Unknown Time-Varying Parameters and Bounded Disturbances: Part I," In Proc. of American Control Conference, pages 3925–3930, New York, NY, July 2007.
22. Cao, C. and Hovakimyan, N., "L1 Adaptive Output Feedback Controller for Systems with Time-varying Unknown Parameters and Bounded Disturbances," In Proc. of American Control Conference, pages 486–491, New York, NY, July 2007.
23. Yakimenko, O., "Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories," AIAA Journal of Guidance, Control, and Dynamics, Vol.23, No.5, 2000, pp.865-875.
24. Dobrokhodov V.N., Yakimenko O.A., "Synthesis of Trajectory Control Algorithms at the Stage of Rendezvous of an Airplane with a Maneuvering Object," Journal of Computer and Systems Sciences International, Vol. 38, №. 2, 1999, pp. 262-277.
25. Personal communications with Jon Damush and Scott Cubbage of 2d3, www.2d3.com, 2008.