A Framework for Autonomous and Continuous Aerial Intelligence, Surveillance, and Reconnaissance Operations

Christopher Korpela[a], Philip Root[a], Jinho Kim[b], Stephen Wilkerson[c], S. Andrew Gadsden[b]

[a]United States Military Academy, West Point, NY, USA, 10996;
[b]University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA 21250;
[c]United States Army Research Laboratory (ARL), Aberdeen Proving Ground, MD, USA 21001

ABSTRACT

We propose a framework for intelligence, reconnaissance, and surveillance using an aerial vehicle with multiple sensor payloads to provide autonomous and continuous security operations at a fixed location. A control scheme and a graphical user interface between the vehicle and operator is strictly mandated for tasks requiring remote and unattended inspection. By leveraging existing navigation and path planning algorithms, the system can autonomously patrol large areas, automatically recharge when required, and relay on-demand data back to the user. This paper presents recent validation results of the system and its sensors using the proposed framework.

Keywords: Unmanned aerial systems, intelligence, surveillance, reconnaissance

1. INTRODUCTION

Unmanned aerial vehicles (UAV) continue to proliferate across all sectors (private, government, and military) with innovative use-cases emerging daily. Commonly referred to as drones, these vehicles typically include camera systems for real-time or post-processing video. The military uses UAVs for intelligence, surveillance, and reconnaissance (ISR) missions in addition to target acquisition and weapons deployment. ISR is also an integral part of fixed installation force protection and is an important component of incident response. Current best practices for conducting ISR missions require direct or indirect human monitoring through the use of checkpoints, guard towers, roving patrols, and human intelligence. Robotic systems offer the potential to remove humans from these labor-intensive situations while maintaining the reliability and accuracy of the response team in the event of a security breach. Applying robotic solutions to this domain also contribute to the Department of Defense (DoD) unmanned systems goals outlined in the Unmanned Systems Integrated Roadmap FY2011-2036.[1] Furthermore, robotic ISR solutions are a force multiplier because these systems can be deployed and allowed to collect data autonomously while recharging when necessary. Using robotic platforms keep humans away from potentially hazardous areas and these systems do not experience fatigue in the same manner as their human operators. Finally, if successful, this project has potential DoD-wide and future civilian applications.

1.1 Trends

Aging stocks of munitions and newly developed systems are creating larger quantities of dangerous materials that require constant monitoring and potentially, multiple emergency responses.[2] In the current austere fiscal environment, enlarging a highly trained, professional security response team poses a significant challenge. Thus, it is more desirable to automate IRS missions to improve efficiency and reduce manpower where possible. One way to accomplish this goal is to automate repetitive tasks using currently available technologies such as aerial robots and networked data acquisition systems. This automation can allow a smaller team of trained personnel to effectively as well as efficiently manage a larger amount of tasks that would traditionally require a large team.

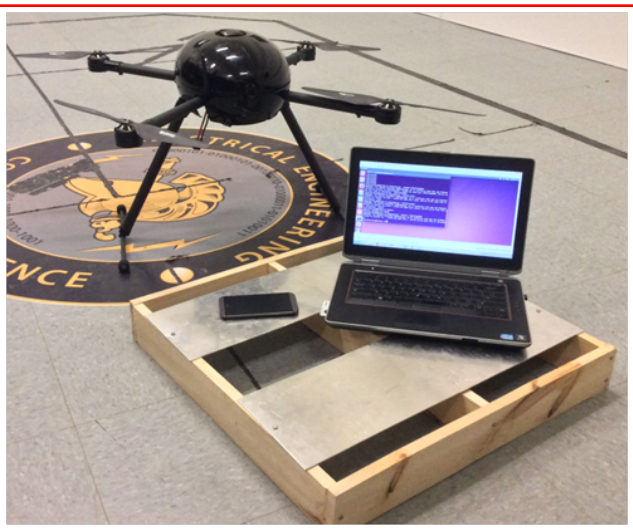


Figure 1: VTOL-ISR prototype vehicle during flight tests inside an indoor GPS-simulated environment. Picture shows quadcopter, charging station, base station (laptop), and user interface (phone or tablet).

1.2 Concept of Operations (CONOPS)

Radford Army Ammunition Plant is one of the largest ammunition propellant manufacturers for the United States Army and the sole supplier of TNT to the Department of Defense. Located in Pulaski, Virginia, the plant and surrounding post have considerable security dangers, as a public river park runs through the middle of the post placing any intruder legally in close proximity to high power explosives and chemicals. There are 1038 buildings housed on 7000 acres of land, creating a serious problem for post security and lending any manned security system ineffective. It is estimated that within ten minutes of infiltration an intruder would be able to have access to key plant facilities likely without any detection. Due to a low budget and geographical challenges, the post simply cannot higher more security guards to constantly patrol the post, and so must turn to an alternate solution to solve their security problem.

1.3 Specifications

To meet the requirements of the CONOPS described above, the proposed framework provides persistent security capable of both autonomous flight and autonomous recharge for the Radford Army Ammunition Base. A user interface allows for on-demand video observation and manual control to redirect the vehicle to higher priority missions. The UAV will cover an area of 10 km^2 to provide surveillance with minimum human interaction, under all-weather conditions to include rain, snow up to 10 cm in height, and can withstand wind speeds less than 10 m/s. Other performance criteria include:

- Autonomous take off and landing
- 10-15 minutes of flight time (estimated based on flight tests)
- Randomized flight paths to reduce predictability
- Minimum of 45 minutes of recharge time

1.4 Organization

This paper presents a framework to the aforementioned problem using a small aerial vehicle equipped with a multiple sensor payloads. The mobility platform for the system (Fig. 1) is modeled with a proposed control scheme in Sec. 2. Navigation and path planning algorithms are described in Sec. 3. To allow for unattended operation in a fixed, known environment, Sec. 4 describes the user interface to remotely monitor sensor data and toggle between teleoperation and autonomous modes. Simulation results and flight tests are found in Secs. 5 and 6.

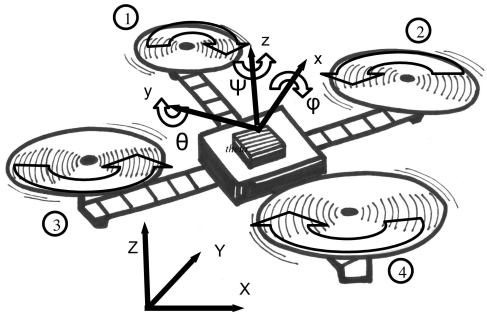


Figure 2: Coordinate frames for the VTOL-ISR system.

2. MODELING

Multi-rotor copters have become center stage for the realization of small autonomous drones. A majority of these vehicles have fixed pitch angle blades as opposed to variable pitch angle blades. In particular, we will consider the control of a quad rotor helicopter by varying the speed on each of the rotors. This does not preclude the inclusion of additional rotors but in this derivation we will focus on a 4-rotor system. Our derivation closely follows a simplified version in.[3,4]

A free body diagram for the quadrotor system is shown in Fig. 2. The rigid body dynamics and operation of quadrotors are well understood.[5,6] Each rotor produces lift and therefore a force or moment on the system. Two pairs of rotors (i.e. 2 and 4 in Fig. 2), having more speed than rotors 1 and 3 will induce a *Roll* φ moment about the *x-axis*. On the other hand, two rotors, i.e. 1 and 2 having greater speed than rotors 3 and 4, will induce a *Pitch* θ angle on the *y axis*. Finally, by varying rotors 1 and 4, versus 2 and 3 will induce a *Yaw* ψ moment about the *z-axis*.

In a typical configuration, the quadrotor is assumed to be symmetric with respect to the x and y axis so that the center of gravity is located at the geometric center of the quadrotor. Each rotor is located at the end of the support arm whose length from the center of the rotor is ℓ. Each rotor generates thrust *Force* where $F_i(i = 1, 2, 3, 4)$ which is perpendicular to the x-y plane as shown in Fig. 2. J_i is the moment of inertia with respect to each axis and we use ρ as a force to momentum scaling factor. From these definitions, equations of motion without considering air drag can be written:

$$\begin{bmatrix} \ddot{\mathrm{x}} \\ \ddot{\mathrm{y}} \\ \ddot{\mathrm{z}} \end{bmatrix} = \frac{1}{m}(\sum_{i=1}^{4} F_i) R\bar{e}_3 - g\bar{e}_3 \tag{1}$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \ell((F_1 + F_3) - (F_2 + F_4))/J_1 \\ \ell((F_3 + F_4) - (F_1 + F_2))/J_2 \\ \rho((F_1 + F_4) - (F_2 + F_3))/J_3 \end{bmatrix} \tag{2}$$

where $[x, y, z]$ is the inertia reference frame and $[\phi\ ,\ \theta\ ,\ \psi]$ are the roll, pitch, and yaw angles of the quadrotor with g representing gravity. e_3=[0 0 1] is the unit vector in the z direction. The rotation matrix can be found from:

$$R = \begin{bmatrix} \mathrm{c}\theta\mathrm{c}\psi & \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi - \mathrm{c}\phi\mathrm{s}\psi & \mathrm{c}\phi\mathrm{c}\psi\mathrm{s}\theta + \mathrm{s}\phi\mathrm{s}\psi \\ \mathrm{c}\theta\mathrm{s}\psi & \mathrm{c}\phi\mathrm{c}\psi + \mathrm{s}\theta\mathrm{s}\phi\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{3}$$

noting that 'c' stands for *cosine* and 's' stands for *sine* respectively. A more detailed model to include ground effect will be included in future work. For our configuration the input terms for normalized lift u_1, roll u_2, pitch u_3, and yaw u_4 are given by:

$$
\begin{aligned}
u_1 &= (F_1 + F_2 + F_3 + F_4) \\
u_2 &= ((F_1 + F_3) - (F_2 + F_4))/J_1 \\
u_3 &= ((F_3 + F_4) - (F_1 + F_2))/J_2 \\
u_4 &= \rho((F_1 + F_4) - (F_2 + F_3))/J_3
\end{aligned} \tag{4}
$$

Combining equations (1) - (4), the overall equations can be represented in vector form as $\ddot{\mathbf{x}} = \mathrm{f}(\mathbf{x}) + \mathrm{g}(\mathbf{x})\mathbf{u}$, with $\mathbf{x} = [\mathrm{x}, \mathrm{y}, \mathrm{z}, \phi, \theta, \psi]^T$ and $\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ or as:

$$
\begin{aligned}
\ddot{x} &= u_1(cos\phi sin\theta cos\psi + sin\phi sin\psi \\
\ddot{y} &= u_1(cos\phi sin\theta sin\psi - sin\phi cos\psi) \\
\ddot{z} &= u_1(cos\phi cos\theta) - g
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
\ddot{\phi} &= u_2\ell \\
\ddot{\theta} &= u_3\ell \\
\ddot{\psi} &= u_4
\end{aligned} \tag{6}
$$

For our case we assume symmetry about the center of gravity (CG) of the quadrotor and the distance ℓ is the distance to the center of the rotor from the CG (in other words the x and y distance from the CG).

3. NAVIGATION

We divide the system's navigational requirements into two categories: patrolling flight at altitude via GPS and autonomous landing via optical flow algorithms. The system requires no obstacle avoidance scheme given the lack of natural or manmade obstacles within the vicinity of the designated landing pads; we did not deem the addition of an obstacle detection capability to be worth the additional required mass, power draw, and software complexity. Future implementations will benefit from an enhanced obstacle avoidance schema to allow implementation in a wider variety of environments.

A standard patrolling mission would begin with the flight vehicle ascending from the landing pad at a specified ascent rate while maintaining position via GPS. We designate a flight altitude that guarantees safe separation from all ground obstacles within the patrolling region. Once at flight altitude, the vehicle begins a randomized waypoint selection scheme as in[7] that maximizes the probability of detecting an aggressor attempting to breach the perimeter. The waypoint selection algorithm has the additional benefit of deterring any potential threats given its stochastic time to return to a point along the perimeter. We contrast this approach with a deterministic waypoint selection scheme where the flight vehicle returns to each point along the perimeter at a constant rate; while such an approach minimizes the time between visits, often called "latency" within autonomous patrolling literature, we contend that the deterministic time between visits provides any would-be aggressor with a known amount of time to breach the perimeter without detection. We, therefore, randomize the waypoint selection along the perimeter according the Course Reversal policy to maximize the probability of detection while remaining unpredictable or stochastic in application.

3.1 Practical Implementation

Let T be the time required to circumnavigate the perimeter, and let S be the expected attack duration. Then $c = S/T$ is the ratio of these two times, and it is a measure of susceptibility where $c \to 0$ indicates that the perimeter is large and more easily breeched without detection. One of the key findings of[7] is the optimal rate of course reversal, called μ^*, for all $c \in [0, 1]$. Fig. 3 depicts this optimal nonlinear μ^* for all c.

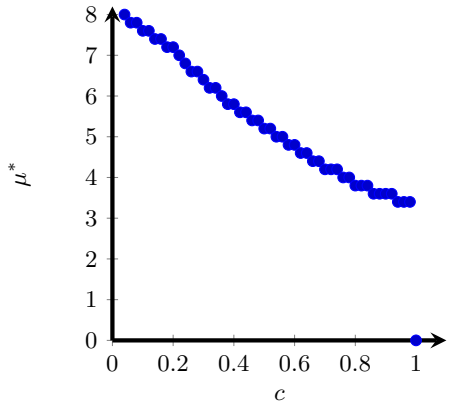


Figure 3: Optimal rate of course reversal, μ^* for all time ratios, $c = s/M$.

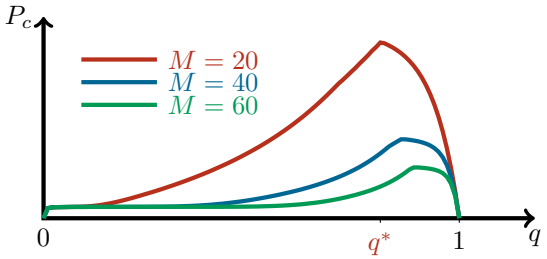


Figure 4: Monotonous decrease in capture probability with increasing M.

We note that μ^* is a rate of course reversal where our control architecture requires discrete decisions. Our navigation algorithm begins by discretizing the continuous perimeter length into M waypoints. We then calculate the optimal course reversal probability as:

$$q^* = \mu^*/M \tag{7}$$

Upon arrival at each waypoint, our navigation algorithm chooses to proceed to the following waypoint in the current direction, either clockwise or counterclockwise, with probability $1-q^*$. With probability q^* the navigation algorithm chooses to reverse direction and proceed to the previous waypoint. We contend that this scheme represents an improvement in patrolling logic for those environments where the would-be aggressor can observe the patrolling UAV and time his attack to maximize the probability of success.

We balance this theoretical finding that the optimal patrolling policy is independent of our choice of discretization M with the realization that in application the UAV cannot reverse directions instantly as assumed in.[7] Rather each course reversal requires some finite time even with the most aggressive control scheme. We can modify the Markov transition matrix to account for this finite time required for course reversal; Fig. 4 demonstrates the decrease in capture probability, P_c, for increasing discretization M. Therefore for this patrolling application we sought to balance the discretization of the perimeter such that M waypoints provide sufficient fidelity for the perimeter shape.

4. USER INTERFACE SOFTWARE SUITE

A standardized graphical user interface (GUI)* was developed to be robust, intuitively usable by anyone, and serve as an all-in-one software suite (Fig. 5) for servicing inputs and outputs via a middleware, called Robot Operating System (ROS), which acts as the backend of the GUI. The GUI frontend, as seen by the user, is responsible for relaying environmental sensor and navigation data. The GUI software backend is standardized for use on any aerial robot that is able to traverse through a 6 degree of freedom (DOF) work space using an under-actuated controller. In teleoperation mode, the user is able to control the UAV's movement with a single

*GUI is synonymous with Operator Control Unit (OCU).

virtual joystick for controlling both forward and turning motions. A wrapper is written to translate the joystick commands to actual linear velocity and yaw orientation commands. A second virtual joystick is available and dedicated to manipulating the camera viewpoint towards regions of interest in real time. This secondary joystick allows the user to control a pan-tilt unit attached to the camera base. The shared bandwidth for camera feed and joystick commands is automatically adjusted and delegated based on the bandwidth that is available at a particular instance in the wireless connection from the aerial platform to the base station. The GUI will alert the user if any bandwidth limits have been reached, and will attempt to provide the user with the most up-to-date sensor and video data, allowing the user to make the most informed decisions possible in a given situation. Bandwidth concerns have a real impact upon mobile safety applications and must be taken into account;[8] goes into further detail about how mobile constraints can alter the user interface and backend.

4.1 Navigation Modes

As previously mentioned, the GUI allows the user to utilize two joysticks with one being for UAV motion, and the other for camera pan-tilt motion; this two joystick scheme constitutes the first control mode. The GUI is designed to feature additional control modes. The control modes change the level of autonomy the aerial robot exhibits during its mission. This feature, of course, requires for the different control modes to be hard-coded into the UAV to ensure full functionality, both in terms of autonomous motion and safety of equipment in the airspace being traveled. The change in control modes helps reduce the probability of mission failure or loss of UAV if the base station loses connection to the UAV. The modes also change the level of involvement of the user in navigation duties, thus allowing the user to focus on higher level duties such as specifying areas of interest for surveillance and intelligence gathering. This reduction of mental fatigue on the user helps increase mission success rate. All navigation modes utilize the GUI's built-in features of map updating when differentials are found, battery level alerts, and on-board sensor warnings.

The second navigation mode allows for full autonomous navigation of the airspace regardless of having any *apriori* knowledge of the environment or having a map. The map data is held in the on-board PC of the UAV and is synchronized with the base station. All navigation and obstacle avoidance actions are recorded on the full 2-D map, as metadata, that is generated as the UAV progresses through the airspace. Video data can also be toggled on if the user desires to visually inspect mission progress. By default, in order to reduce bandwidth saturation, map data is not transmitted to let mission critical sensor data take transmission priority. A fail-safe feature sends a warning message to the user to indicate that the UAV has reverted back to Mode 1 for manual navigation. This only occurs if the UAV is unable to continue through the airspace or has encountered a navigational error in its on-board programming. If the user takes no action or the connection has been lost, the UAV again elevates autonomy to its original setting to reverse course and return to the mission start position.

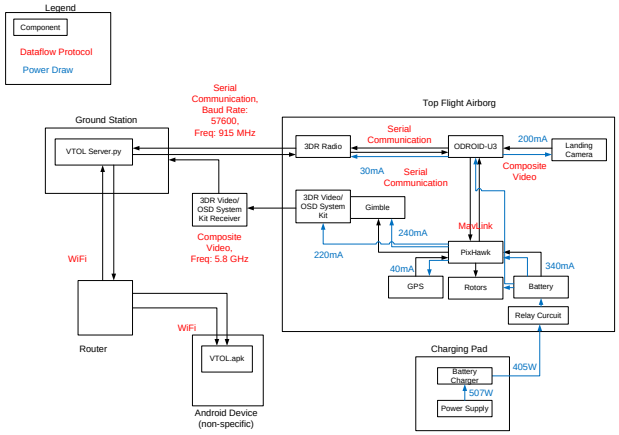


Figure 5: System flow of aerial surveillance interface.

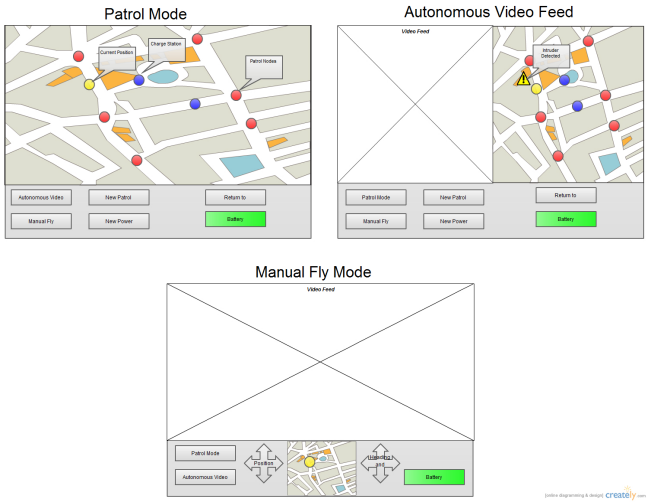


Figure 6: Screenshot of the proposed web-based GUI allowing for view of the camera and 2-D map.

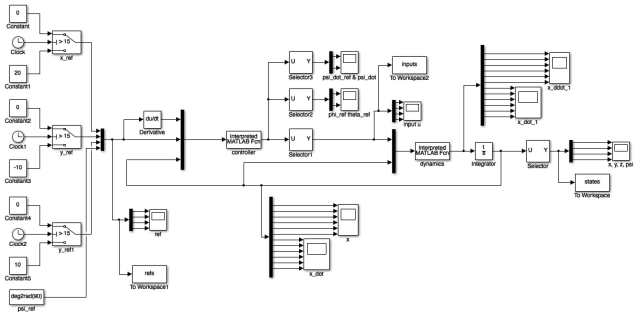


Figure 7: Simulink Model for UAV control.

4.2 HTML Interface

The desire to create a web-based control application has been explored since the inception of the Internet which is further described by use cases provided in.[9] The GUI is a web application that has all the mentioned features to be toggled on or off through a settings window which is separate from the main robot interaction window in order to reduce the clutter the user sees while performing a mission (Fig. 6). This will reduce the likelihood that the user will mistakenly click or press the wrong button during mission critical moments. The aforementioned navigation modes can be toggled on-the-fly by the user and are accessible directly from the main robot interaction window. The GUI frontend itself is based on Robot Web Tools[10,11] which is a set of modules that help create web-based robot control applications. It allows ROS software package messages and topics to be accessible through a web interface that is constructed using HTML5 and Javascript via a wrapper called *rosjs*.[12] The GUI is essentially a web accessible overlay that allows the user to send and receive data across a connection between a ROS server, the base station, and its corresponding ROS client, the UAV. The advantage of using HTML5 is its cross-platform compatibility allowing the web application to be used by PCs, tablets, and smart phones.[13] On the ROS client, the UAV computer runs a *roscore* node that utilizes a package called *rosbridge* which allows socket-based access to ROS through Javascript.[14]

5. SIMULATION

In this paper we use a Simulink model (Fig. 7) and a basic PID controller to simulate the control of the quadcopter. Simulink is a graphical programming language tool for modeling, simulating, and analyzing multi-domains that is well-suited for the modeling of a quadcopter. In our model we have allowed for the future development and incorporation of other control algorithms. For this initial effort the PID and PD controller is appropriate. For the simulation results shown in Fig. 8, we have the following parameter settings:

$$x_0 = 10m, y_0 = 10m, z_0 = 10m \tag{8}$$
$$\phi_0 = 20deg, \theta_0 = 20deg, \psi_0 = 20deg \tag{9}$$
$$K_{p,\phi}, K_{d,\phi} = [40, 20] \tag{10}$$
$$K_{p,\theta}, K_{d,\theta} = [40, 15] \tag{11}$$
$$K_{p,\psi}, K_{d,\psi} = [5, 5] \tag{12}$$
$$K_{p,z}, K_{d,z} = [45, 20] \tag{13}$$

The quadcopter reference input switches from (20, -10, 5) to (0, 0, 0) with a 90 degree heading angle during the flight. The results from the PD controller yield satisfactory initial results for simple missions. However, for more complex scenarios, more in-depth controllers and additional sensor integration will be required.

6. FLIGHT TESTS

Flight tests were performed to demonstrate and evaluate the basic functionality of the autopilot system for outdoor operations. Basic navigation and stabilization control were tested at outdoor sites in Maryland and New York state. For these experiments, the Top Flight Technologies Airborg M4 Systems Class I UAV electric quadrotor is utilized. This aerial drone features 17 inch rotors with a 16000 mAh LiPo Battery having fairly good flight time and some excess lift capability. The UAV can operate in manual, semi-autonomous, or fully autonomous modes and includes an on-board flight data recorder. The remote link has a 2-mile maximum range, and the on-board gimbaled camera has 8 Mega-pixels of resolution with a high-definition downlink.

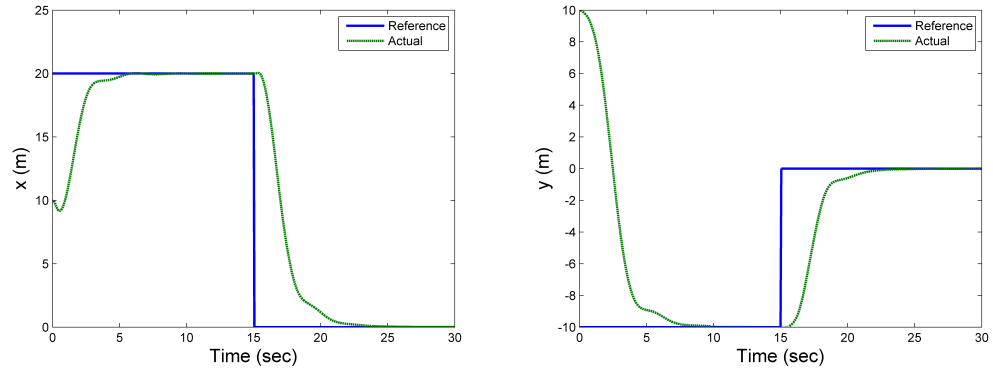
Due to the excellent payload capability, it is capable of carrying additional sensors. A Global Positioning System (GPS) was used to demonstrate go to coordinates, hold position, fix altitude, and return home operations. For our initial tests the Pixhawk autopilot was chosen. The Pixhawk is an advanced autopilot system designed by the PX4 open-hardware project and manufactured by 3D Robotics. It includes an advanced processor and sensor technology from ST Microelectronics and a NuttX real-time operating system. The Pixhawk offers the flexibility and reliability for controlling any autonomous vehicle for the applications in this work. The Pixhawk system includes integrated multithreading, a Unix/Linux-like programming environment, completely new autopilot functions such as Lua scripting of missions and flight behavior, and a custom PX4 driver layer ensuring tight timing across all processes.[15] These features allow for the flexibility needed to conduct surveillance and reconnaissance experiments in a GPS environment as well as a GPS denied environments.

7. CONCLUSIONS

Persistent surveillance and reconnaissance capabilities in near-earth environments possess the potential for improving efficiency and safety through the use of aerial robots. Robotic systems can remove humans from dull, dirty, and dangerous missions while accurately and reliably conducting tasks that are critical to operations on all Army installations. These improvements in efficiency and safety can be obtained in a fiscally responsible manner through the use of inexpensive off-the-shelf systems. Additionally, this research benefits the Army long-term by exposing future officers to this technology.

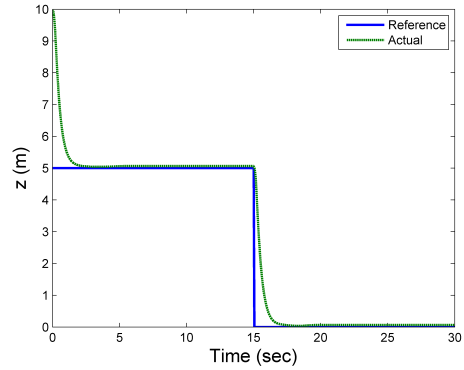
ACKNOWLEDGMENTS

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Defense or the U.S. Government.

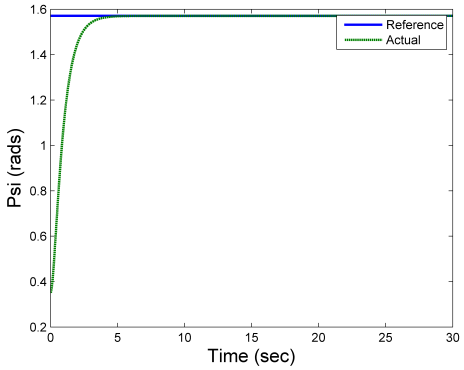


(a) $x[m]$

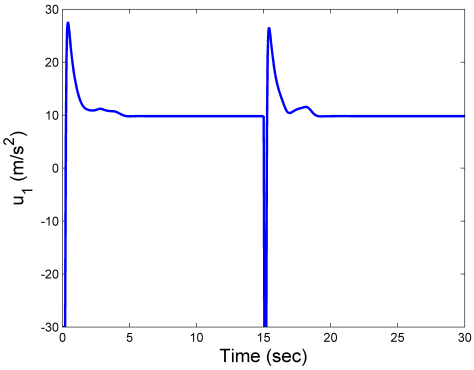
(b) $y[m]$



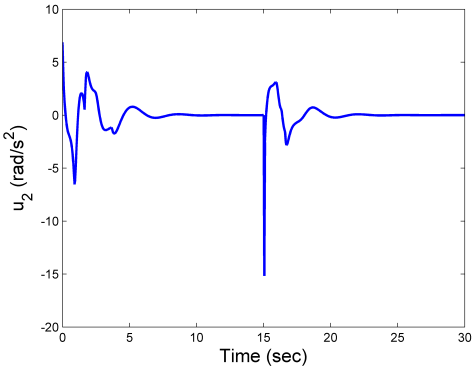
(c) $z[m]$



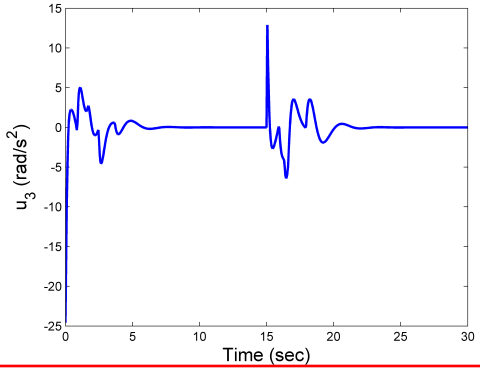
(d) $\psi[rads]$



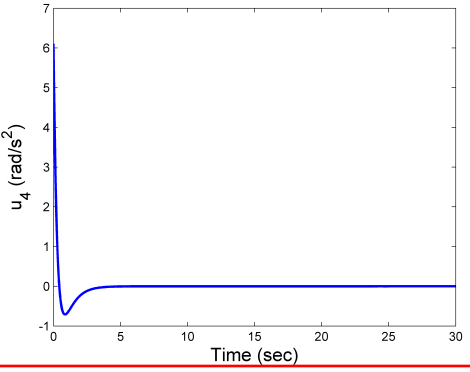
(e) $u_1[m/s^2]$



(f) $u_2[rads/s^2]$



(g) $u_3[rads/s^2]$



(h) $u_4[rads/s^2]$

Figure 8: Trajectory of UAV in 3-D axes with PID controller without uncertainty and sensor noise.

REFERENCES

[1] D. of Defense, "The unmanned systems integrated roadmap fy2011-2036," 2011.

[2] D. of the Army, "The army industrial hygiene program," *Army Publishing Directorate*, vol. Pamphlet 40, no. 503, April 2013.

[3] D. Lee, H. Jin Kim, and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter," *International Journal of Control, Automation and Systems*, vol. 7, no. 3, pp. 419–428, 2009.

[4] E. Altug, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 72–77.

[5] G. M. Hoffmann, H. Huang, S. L. Wasl, and E. C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *In Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2007.

[6] R. E. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robot. Automat. Mag.*, pp. 20–32, 2012.

[7] P. Root, "Persistent patrolling in the presence of adversarial observers," Ph.D. dissertation, Massachusetts Institute of Technology, 2014.

[8] P. Erickson, A. Weinert, P. Breimyer, M. Samperi, J. Huff, C. Parra, and S. Miller, "Designing public safety mobile applications for disconnected, interrupted, and low bandwidth communication environments," in *IEEE International Conference on Technologies for Homeland Security*. IEEE, 2013.

[9] K. Goldberg and R. Siegwart, *Beyond Webcams: an introduction to online robots*. MIT Press, 2002.

[10] http://robotwebtools.org/.

[11] J. Lee, "Web applications for robots using rosbridge," *Brown University*, 2012.

[12] S. Osentoski, G. Jay, C. Crick, B. Pitzer, C. DuHadway, and O. C. Jenkins, "Robots as web services: Reproducible experimentation and application development using rosjs," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 6078–6083.

[13] W. Hilton, D. M. Lofaro, and Y. Kim, "A lightweight, cross-platform, multiuser robot visualization using the cloud," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1570–1575.

[14] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Proceedings of the 15th international symposium on robotics research (ISRR)*, 2011.

[15] www.3drobotics.com.