AIAA SciTech Forum
9 - 13 January 2017, Grapevine, Texas
AIAA Information Systems-AIAA Infotech @ Aerospace

AIAA 2017-0671

An Integrated System for Autonomous Search and Track with a small Unmanned Aerial Vehicle

Anjan Chakrabarty *, Robert Morris †, Xavier Bouyssounouse‡
and Rusty Hunt§
NASA Ames Research Center, Moffett Field, CA 94035

This article presents a framework for searching, detecting, and tracking an object of interest with a small unmanned aerial vehicle (sUAV). The vehicle is given an area to search for an object of interest. Once the object is detected, the sUAV follows the target while maintaining a fixed distance and centered on its image plane. This paper describes an architecture for integrating autonomy capabilities for search and track for sUAVs. The system has been implemented in the Robot Operating System (ROS) framework using the Parrot ARDrone platform. In this paper we present our technical approach, the system architecture and demonstrate the principles of the approach in a simulation environment and indoor flight tests.

Keywords: Search and Track, Visual Servoing, UAV, Object Tracking.

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

I. Introduction and Motivation

Small Unmanned Aerial Vehicles (sUAVs) are increasingly being used for both military and civilian purposes. Several sophisticated systems are available of the shelf for many different applications. One of the applications that has strongly emerged for unmanned systems is surveillance and reconnaissance. UAVs are regularly being used by firefighters, security personnel, and law enforcement officers for various applications including inspection, search and rescue etc. all around the world. But most of the operations being carried out are performed by manual flying. A need for developing autonomous systems for surveillance and reconnaissance have never been greater.

Many applications of search, identify and track (henceforth SI&T) require sophisticated sensing and decision-making capabilities. For example, consider the problem of border patrol by an unmanned aerial vehicle,[1].[2] sUAVs with night and day sensors are deployed to search over a large area and locate objects and subjects of interest. These objects must be identified in a timely manner so that necessary action can be taken. Some of the technical challenges in sensing, navigation and decision-making for this application include choosing where to search, distinguishing between different objects based on their characteristics and tracking an unfriendly target trying to elude capture. The integrated SI&T system will also involve precise communication of location to operators, a ground control system to enable human supervision and situational awareness of the sUAV activities, and potentially more than one sUAV in a coordinated operation. Other examples include searching for lost hikers in an expansive landscape, or searching for survivors in a rescue mission. A similar example can be the problem of finding, locating and tracking poachers of rare white rhinos in Africa.[3] Thus designing an autonomous system to achieve this task of inspect, search and track requires a articulated complex robot behavior based on external perception and co-ordinated robot action.

Complex robot behavior has been well studied in the literature.[4] Game development programming has propelled this research, particularly designing behaviors for non-player characters. Designing an autonomous system for real world applications is more challenging. Architectures for autonomy invariably consists a hierarchical organization of capabilities that must satisfy mission goals in an uncertain environment while remaining safe. Coordinating these capabilities into a hierarchical control system is a difficult design challenge, and there have been different representations proposed to accomplish this coordination. In this paper we show how finite state machines can be used effectively to achieve high-level control.

A finite-state machine (FSM) is a mathematical model of computation used to represent and control execution flow.[5] A finite state machine works with two kinds of primitives: stimuli and states. Stimuli are inputs from the

*Research Engineer, SGT Inc, Intelligent Systems Division. AIAA Member.
†Research Scientist, Intelligent Systems Division. AIAA Member.
‡Research Scientist, Intelligent Systems Division. AIAA Member..
§Research Scientist, Intelligent Systems Division. AIAA Member..

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

sensors and states are the response to those stimuli. The control structure of an FSM is designed in terms of states transitioning to other states based on the stimuli. FSMs can be grouped into a hierarchical FSM; they have been used to design autopilots for unmanned aerial vehicles.[6]

In this paper a Hierarchical Finite State machine is implemented for a search and track mission. More specifically SMACH, a python based FSM has been integrated to facilitate search and track. The main contribution of the paper is the development and demonstration of an integrated SI&T system that can search, identify and then track a human being once it has been detected. Solving the problem of autonomous search and track requires capabilities in vision-based control, path planning, and object recognition.

The remainder of this paper describes the technical approach we employed for integrating capabilities for SI&T using hierarchical FSMs (section II) and the results of experiments conducted in simulation and in indoor field experiments (section III).

II. Technical Approach

t

	Automated	**Manual**
Search	Automated Planning and Execution	Joystick operations
Identification	Object recognition	Human sensing
Track	Autonomous Tracking	Joystick operation

Table 1. Component Capabilities for SI&T and their Possible Instantiations

An integrated SI&T system will consist of a strategic combination of human and automated control and intelligence. Table 1 lists the three main component capabilities of an SI&T system and their instantiation as manual or automated. A fully manual integrated system (right column) will consist of a console and a 'joystick' or other control device for human operation. Conversely, a fully automated solution (left column) will combine autonomous sensing, navigation and decision making on the sUAV or on a ground system that remotely commands and receives data from the vehicle. A partially autonomous system will then consist of a choice for each row of an autonomous or manual capability. Hybrid systems combining human and machine capabilities are also possible: for example, human vision for tracking might be augmented with infra-red sensing on the sUAV.

There is a great deal of active research in areas of planning and control, object recognition, and autonomous tracking for sUAVs. The current focus of this work, summarized in the remainder of this section, has three main components:

- Automated search for and detection of targets of interest;
- Autonomous tracking of found targets; and
- Image Based Visual Servoing Controller.

A. Architecture for Search Identify and Track

Searching for a target in general requires sophisticated path planning. Planning for the goal of searching for a target of interest is different from traditional path planning, which typically solves the problem of going from an initial location to a known goal location. In planning for target searching there is no specified goal location to reach; instead, the best plans to solve the problem are those that cover as much space as needed to find the target (or give up in case no target is found and resources are low).

Figure 1 shows the overall architecture of the system. The coordinating module is the Finite State Machine (FSM): it takes sensor inputs and chooses the appropriate behavior of the sUAV. There are three high-level behaviors that the sUAV might deploy: search, investigate and track. These behaviors are refined into control actions to the sUAV.

We use the Parrot AR.Drone 2.0, a COTS quad-rotor UAV as our platform. The UAV transmits live video stream of its frontal camera and flight telemetry data over wi-fi link. All the processing are done on an off-board computer. The ground computer runs the core software components. The major components of the software architecture are the finite state machine which interacts with object detection system along side the image based visual servoing controller.

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

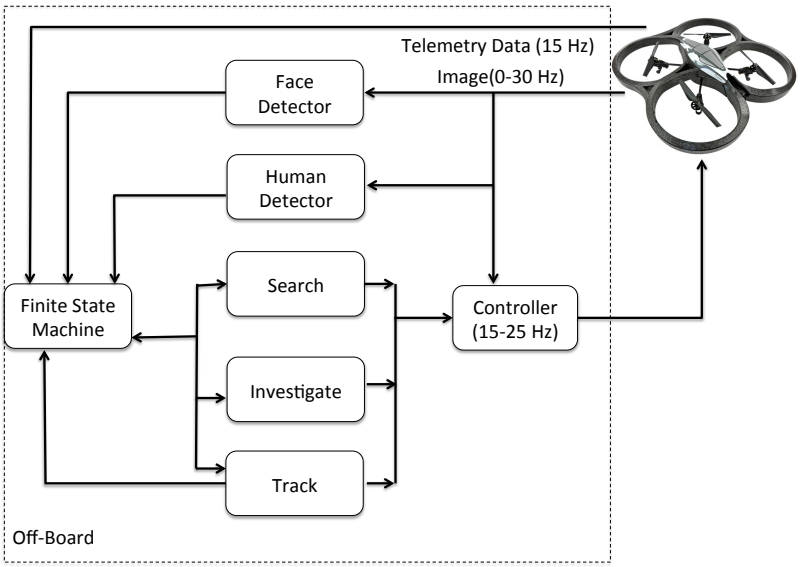

Figure 1. System Architecture

Sensor inputs are fed into an object detector. The goal of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Each detection is reported with some form of pose information; here, we assume the system returns a bounding box around the object. Object detection systems build a model from a set of training examples. Methods for inferring models are either *generative* or *discriminative*. The former create probabilistic models of pose variability and appearance; the latter create classifiers that can distinguish between images that contain an object from those that do not. Here we use a person detector based on a classification technique using grids of histograms of oriented gradients (HOG) descriptors and a linear Support Vector Machine (SVM) to classify images as person/not person; the system is described in details in.[7] A similar detector is used for face detection.[8]

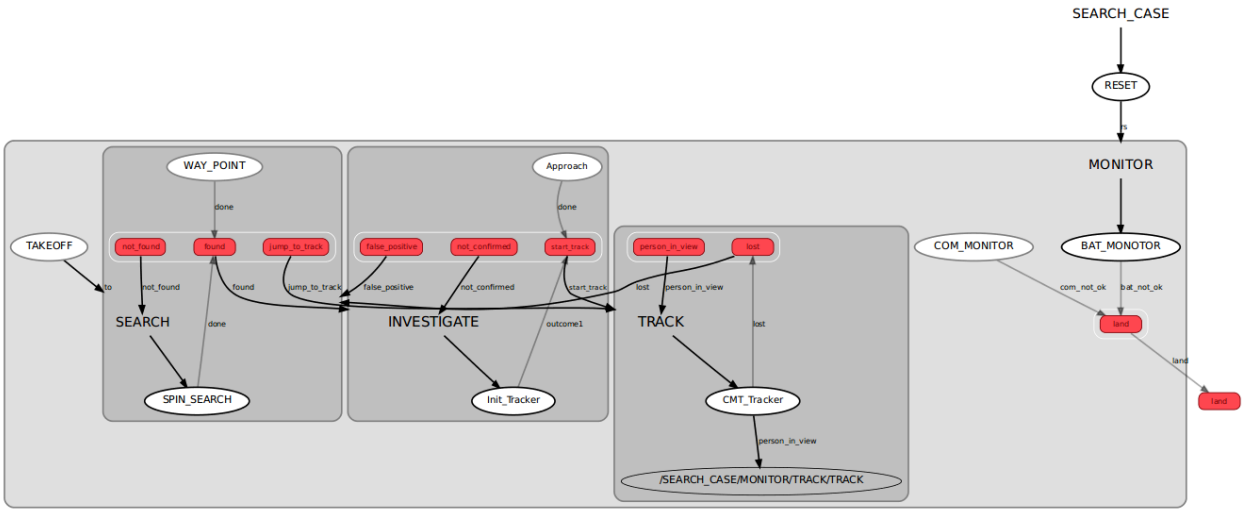

Figure 2. Multiple Levels of SMACH Plan for Search Identification and Track missions

For planning the appropriate behaviors we use the SMACH framework. SMACH[9] ,[10] is an approach to planning and execution based on composition of behaviors using a state transition model. SMACH provides structures for

procedurally generating programs. SMACH is used to build and execute hierarchical concurrent state machines. In addition to providing an interface for representing states, SMACH also provides a representation of concurrent behavior, in which one can execute more than one state simultaneously. The SMACH-ROS interface library provides a visualization interface for runtime introspection of a SMACH plan. The SMACH Viewer renders the structure of a plan, highlights the executing states at runtime, and lists the contents of the user data dictionary for a given container. See[11] for more details.

Figure 2 shows the overall architecture of the search and track system. The figure shows some output of the SMACH viewer system shows the different states and the connections between them. The three main components of the system are search, investigate and track. After an initial reset of the system where the system re-calibrates all the sensors, the system goes into $MONITOR$ mode. The main task of the $MONITOR$ mode is to keep the system safe at all operating times. The $MONITOR$ mode monitors the communication link and the battery percentage remaining. This $MONITOR$ mode runs as a concurrent state machine at all the times. If there is any communication error or the battery goes below the safe threshold, it commands the drone to abort the mission and perform a controlled land. Note that this mode can be used for collision detection and other adversarial situations.

If the $MONITOR$ state reports that the system is ok to proceed, the $TAKEOFF$ state makes the UAV airborne. Immediately after takeoff the system enters into *SEARCH* mode. In this mode the system looks for a human being in the field of view using histograms of oriented gradients (HOG) described above. Two methods of *SEARCH* have been implemented so far: *SPIN_SEARCH* and *WAY_POINT* search. In *SPIN_SEARCH* the UAV spins in its own axis in anti clockwise direction using fixed yaw rate. In *WAY_POINT* search a series of way-points are given to the UAV to fly. Both the search methods terminate if the human detector reports that it has found a human being in the image plane.

If a human target is detected in the *SEARCH* state the state machine switches to $INVESTIGATE$ mode. In this mode the drone is commanded to hold in a hover state. A face detector that works in parallel with the human detector confirms if the person detected is not a false alarm. If both the face detector and the human detector simultaneously reports positive, then human detection is confirmed. Once the human being is confirmed two things happen simultaneously: first the bounding box of the human being identified by the human detector is used to initialize the tracker. Second, The state machine switches to $TRACK$ mode. If the object detected is not confirmed to be a human being, the system reverts back to *SEARCH* mode.

In $TRACK$ mode the UAV continuously keeps the human being in its field of view. The $TRACK$ mode uses an image-based visual servoing (IBVS) system, which is described in the next section. The system remains in $TRACK$ mode while the confidence in tracking is over a predefined threshold. If the confidence of the tracker decreases below the threshold, the system reverts back to *SEARCH* mode. While in *SEARCH* mode the system resumes its previously defined search sequence of either *SPIN_SEARCH* or *WAY_POINT* search. If the target is re-acquired while searching the system jumps to $TRACK$ mode without going through the $INVESTIGATE$ mode. The criteria for re-acquiring the target is that both the human detector as well as the confidence of the tracker should be positive.

Finally, the $MONITOR$ state runs as a concurrent state machine in parallel to search and track. It only comes into play if the battery is critical or communication is lost with the drone.

B. Tracking

Tracking is the problem of detecting an object of interest in the filed of view of the camera of a sUAV over an extensive period of time. The object of interest can be either be specified by an user or can be acquired automatically. The current SI&T system is equipped to identify human beings using OpenCV's Person detection classifier. OpenCV has a pre-trained HOG coupled with Linear SVM model that can be used to perform pedestrian detection in both images and video streams. This works well both with a real person as the target and with simulation models.

The Clustering of Static-Adaptive Correspondences for Deformable Object Tracking (CMT) algorithm[12] can be viewed as the state-of-the-art in tracking algorithms. The CMT tracker[12] represents a tracked object as a set of feature correspondences of key points, relating the current position of a feature to its position in the original image. Furthermore, CMT distinguishes between static correspondences (between the current image and the original image) and adaptive correspondences (between successive images), and uses both kinds of correspondence to update its object model. Adaptive correspondence is better at handling different object appearances due to deformation, whereas the static model is better at handling the reappearance of the object after occlusion. Second, CMT introduces a tolerance parameter in order to allow the set of correspondences between frames to be robust to changes due to deformations. The CMT tracker uses heuristic estimates to generate values related to scale and rotation of the bounding box. These values, in addition to an estimate of the center of the tracked object, are the outputs of CMT.

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

The tracking problem is initiated by a bounding box b_0 and area A_0 in an initial frame. The initial bounding box is identified by OpenCV's HOG detector. The initial set of key points $P_0 = \{x_1^0, x_2^0 ..., x_n^0\}$ and the corresponding tracked points in the subsequent frames are denoted as $P_t = \{x_1^t, x_2^t ..., x_m^t\}$. (See figure 3 for the output of the CMT tracker.) The BRISK[13] detecter algorithm is used to detect the n key points. See[12] for details.

The estimate of scale of the tracked object is given by

$$s = med\left(\left\{\frac{||x_i^t - x_j^t||}{||x_i^0 - x_j^0||}, i \neq j\right\}\right) \tag{1}$$

where *med* denotes the median values of the pairwise scale between the tracked points. Thus the area of the tracked object is given by $A_t = s * A_0$. The controller compares this value to a reference size and distance to approximate the distance from the tracked object.

The CMT algorithm is prone to noise, and will find key point matches even when the object is not in the field of view. This will create a false positive with detection and led to instability of the controller. To overcome this outcome a measure of confidence c was added to the CMT tracker. The initial number of points tracked in the object of interest is noted. Then at each frame the number of tracked points is compared with the initial number of points. The ratio between the current tracked points and initial number of points gives the measure of confidence in tracking.

$$c = \frac{card(P_t)}{card(P_0)} \tag{2}$$

where *card* denotes the cardinality of the set of points. If the confidence of tracking goes below a predetermined threshold value, the tracker does not output any value. This forces the *TRACK* state to revert back to *SEARCH* state.

C. IBVS Controller and Implementation

Given the output of the CMT tracker as the cluster of tracked points and the scale of the object, the following equations provide feedback to the controller.

$$f_u = \frac{\sum_i x_i^t}{n_i^t}; \tag{3}$$

$$f_v = \frac{\sum_i y_i}{n_i^t}; \tag{4}$$

$$f_\Delta = \sqrt{\frac{A_{image}}{A_t}} \tag{5}$$

where f_u and f_v are the horizontal and vertical feedbacks to the controller and $\sum_i i = n_i^t$ is the total number of points tracked at time t. Collectively these variables track the centroid of the tracked object. A_{image} is the area of the whole image. Thus the ratio of the whole image and the tracked object gives an approximate estimate of depth. Given the size of the object being tracked this gives an estimate of the distance to the object being tracked.

These control feedbacks are decoupled into altitude and lateral movements and are then sent to a previously tuned PD controller which outputs speed commands. The final commands sent to the controller are $\{\theta_r, \phi_r, \frac{d\psi_r}{dt}, \frac{dz}{dt}\}$, which are pitch, roll, heading rate and altitude rate. See[14] for complete details of the controller.

The IBVS and tracker were implemented as modules with ROS.[15] The AR.Drone was commanded from a computer via WiFi link using the AR.Drone Autonomy ROS package.[16] The C++ CMT code was used,[17] and was built as a ROS package and integrated into the IBVS controller.

This summarizes the SI&T framework. The framework can obviously be extended in many ways and is intended here to illustrate the methodology underlying the integration of components of SI&T. For example, we have been working on high level planning to be included for search in unknown spaces and adding more sophisticated sensors for aiding search procedures. But for the work of this paper, SMACH provides the high level control mechanism which was tested in both simulation and flight tests.

III. Experiments

In this section we summarize the experiments conducted on SMACH based SI&T system described in this paper. The tests here are meant to verify the feasibility of SMACH approach to continuous planning for SI&T. The performance metrics for feasibility include correctness (SMACH responds to trigger inputs to switch between the states and

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

the interconnection between the state is valid) and timeliness (there is no significant delay in the deliberation of the states that may fail the mission.)

Both simulation and flight tests were conducted to validate the architecture. Simulation was conducted using the Gazebo[18] simulation package and flight tests were conducted in the an indoor testing facility at NASA Ames Research Center.

A. Simulation Results

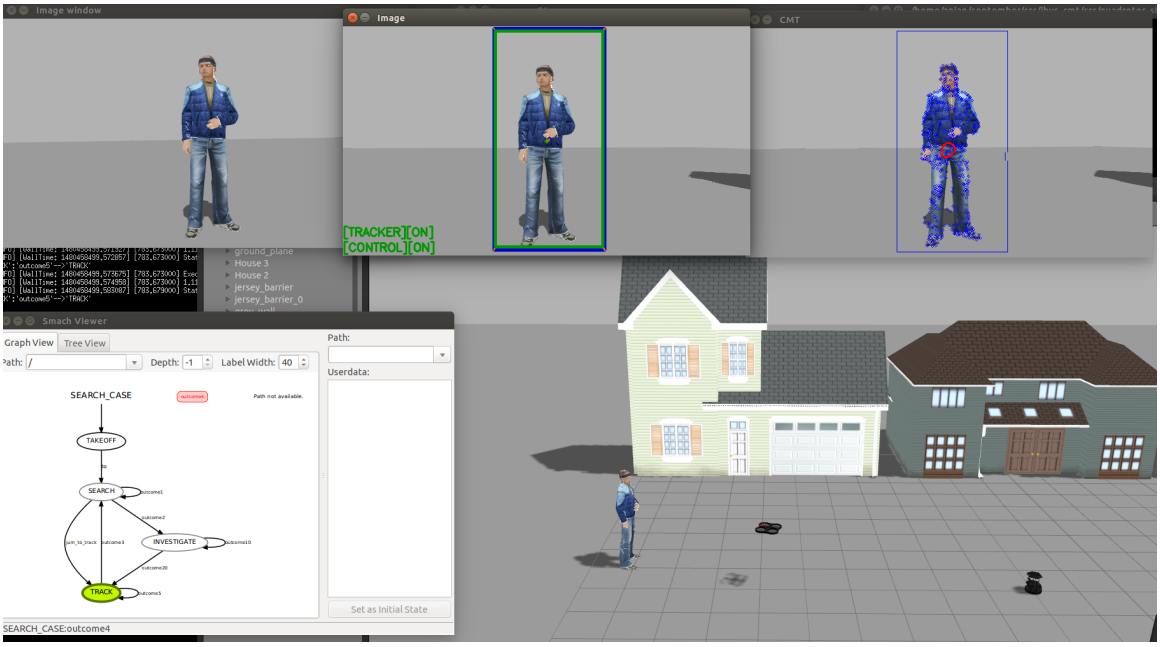

Figure 3. Simple Gazebo environment for testing SI&T in simulation.

Figure 3 shows a typical run of the system in the Gazebo simulation environment. The simulation allowed for quick debugging and adjustment of system parameters that influence the performance of the system. Some of these parameters include the expected size and distance of the target (used by IBVS controller).

The human target would begin out of sight of the sUAV. The sUAV would take of and if the *MONITOR* state okays the battery and comm link the system would go in *SEARCH* state (which was always true in simulation). In *SEARCH* mode a simple spin search was implemented. Once the human target was found by the person detector node the system swiftly transferred to INVESTIGATE node and then to TRACK node.

In figure 3 the lower bottom right panel shows the gazebo environment. The SMACH viewer output is seen on the bottom left. Note that the *TRACK* node is active(highlighted in green). The upper middle panel shows the bounding box around the person target maintained by the CMT tracker, and control input (orange arrow) to visual controller. The right panel shows the bounding box constructed by the HOG detector.

Figure 4 shows the output of the different interconnected systems. Figure 4a shows the output of the person detector node and the CMT confidence parameter. These two outputs drive the decision of the SMACH system. As stated above the system starts with the human target out of sight. Thus the system starts in the *SEARCH* mode. Figure 4b. Figure 4c,d shows the controller outputs. Note in the *SEARCH* mode all the commanded outputs are zero except for the yaw commanded. The yaw command in the spin search is a set constant where the vehicle essentially turns at a fixed rate to look for humans. Once the person is detected the system switches to *TRACK* mode (Transition from *INVESTIGATE* to *TRACK* mode is instantaneous). Until the CMT confidence level is published none of the controllers publish. After that CMT-IBVS controller follows the target.

The human in the simulation was then moved again from the view of the drone. As the confidence of the tracker decreases to almost zero the system reverts back to *SEARCH* mode. The yaw command is again constant showing constant spin search. Once both person detected and the confidence goes above the threshold (0.6) the system jumps back to *TRACK* mode. This alternate behavior was shown to work in simulation. Note that during the re-*SEARCH*

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

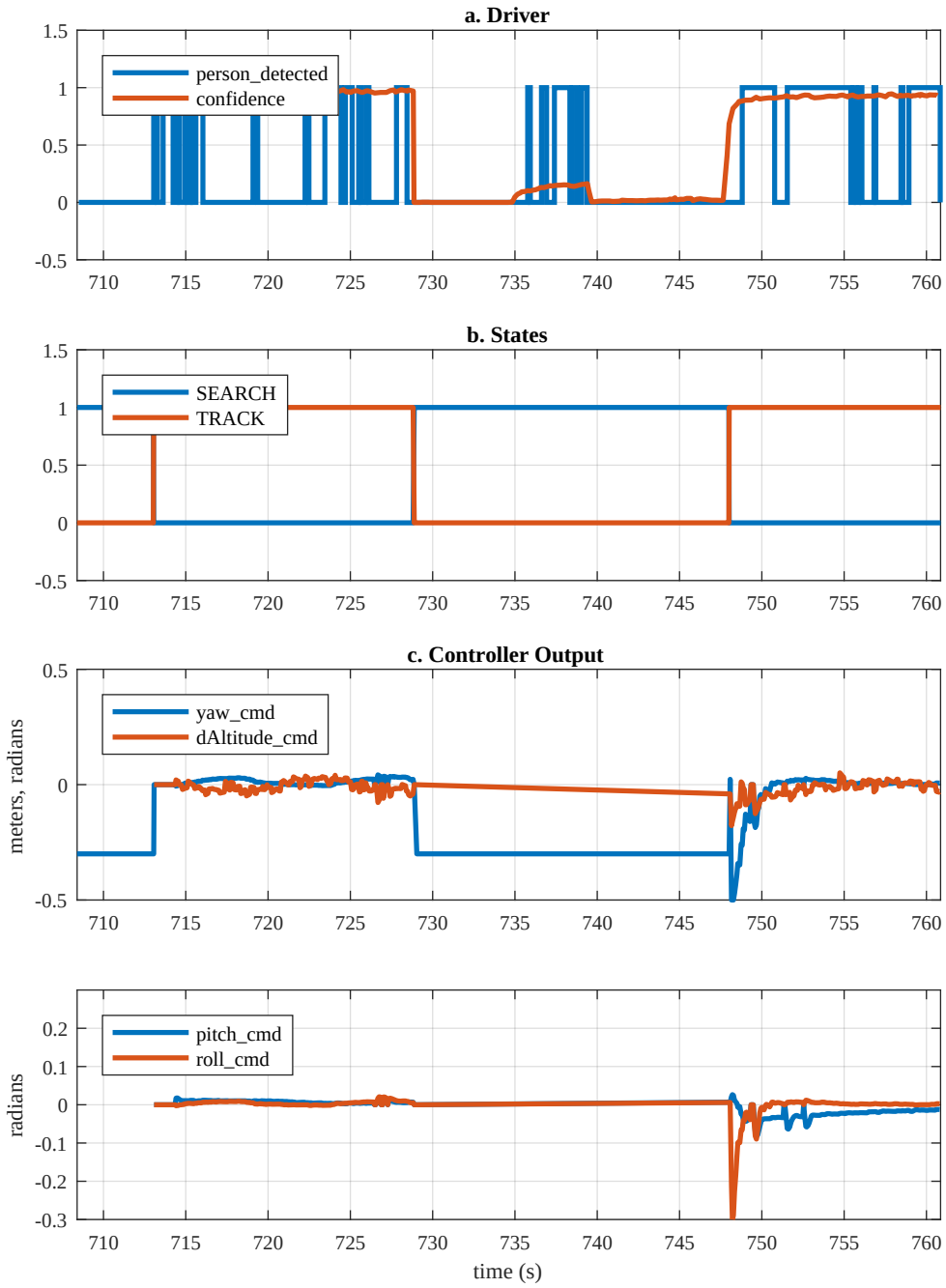

Figure 4. Simulation Results

mode (during 728-747 seconds) there was false identification of human target. But since the confidence was lower than the threshold the system did not go to *TRACK* mode.

The simulation tests helped to fine tune certain parameters which were very essential for flight test. These include introducing forced delay between search and track operations. Without an induced delay the position of the bounding box that is inherited by the CMT tracker from the person detector is shifted. Also, certain controller parameters were fine tuned like expected size and distance of the target. Once the system was verified to be working in simulation flight tests were conducted to show the system working in real flights.

B. Flight Tests

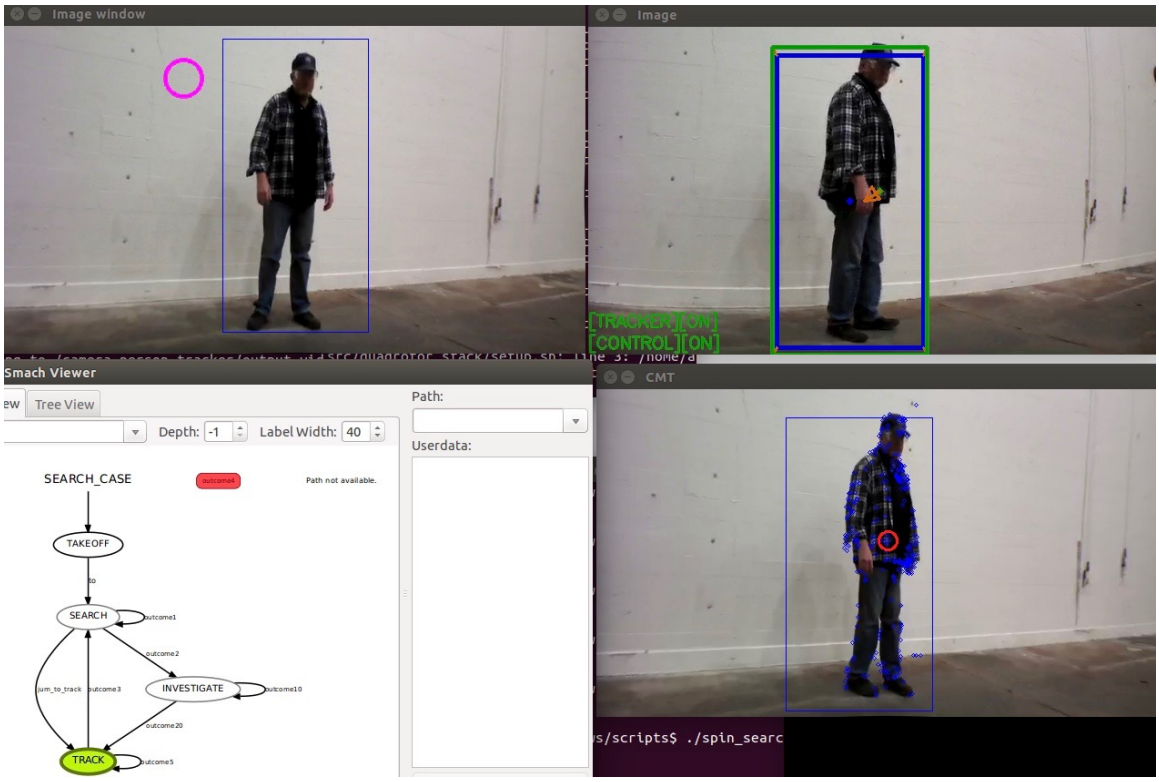

Figure 5. Flight Tests

Flight tests were conducted in an indoor test facility in the Ames Research Center. We deliberately chose a fairly controlled indoor environment in order to isolate the interactions between the continuous planning framework and the underlying sensing and control frameworks, thereby demonstrating the feasibility of a hierarchical autonomy architecture based on SMACH. Consequently, however, there is little in the way of assurance that the system is robust to realistic dynamic environments or noisy sensors. Future outdoor field tests are planned, using more powerful platforms than the AR Drone, in order to improve robustness.

In a typical run of the system, a human target would begin out of sight of the sUAV. The sUAV would take off and begin a pattern maneuver (such as a square pattern) in search mode. Once the human target is found, we studied the response time of the system to transition into track mode. Once in track mode, the target would first move in a slow linear walk to demonstrate simple tracking behavior. At some point, the target would take evasive maneuvers until the subject was out of sight of the sUAV. We then could observe the transition into (re-) search mode, which consisted of following another pattern search (typically a simple rotation). This alternating behavior of search and track was typically repeated many times in a single run.

Figure 5 shows a screen shot of the run time behavior of the SI&T system. The top left window shows the output of the human detector window with the bounding box. The bounding box is inherited by the CMT tracker as shown in the bottom right window. The top right window shows the output of the IBVS controller. The bottom left windows shows a stripped down part of the SMACH planner. At the particular instance the system was in *TRACK* mode successfully tracking the human subject.

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

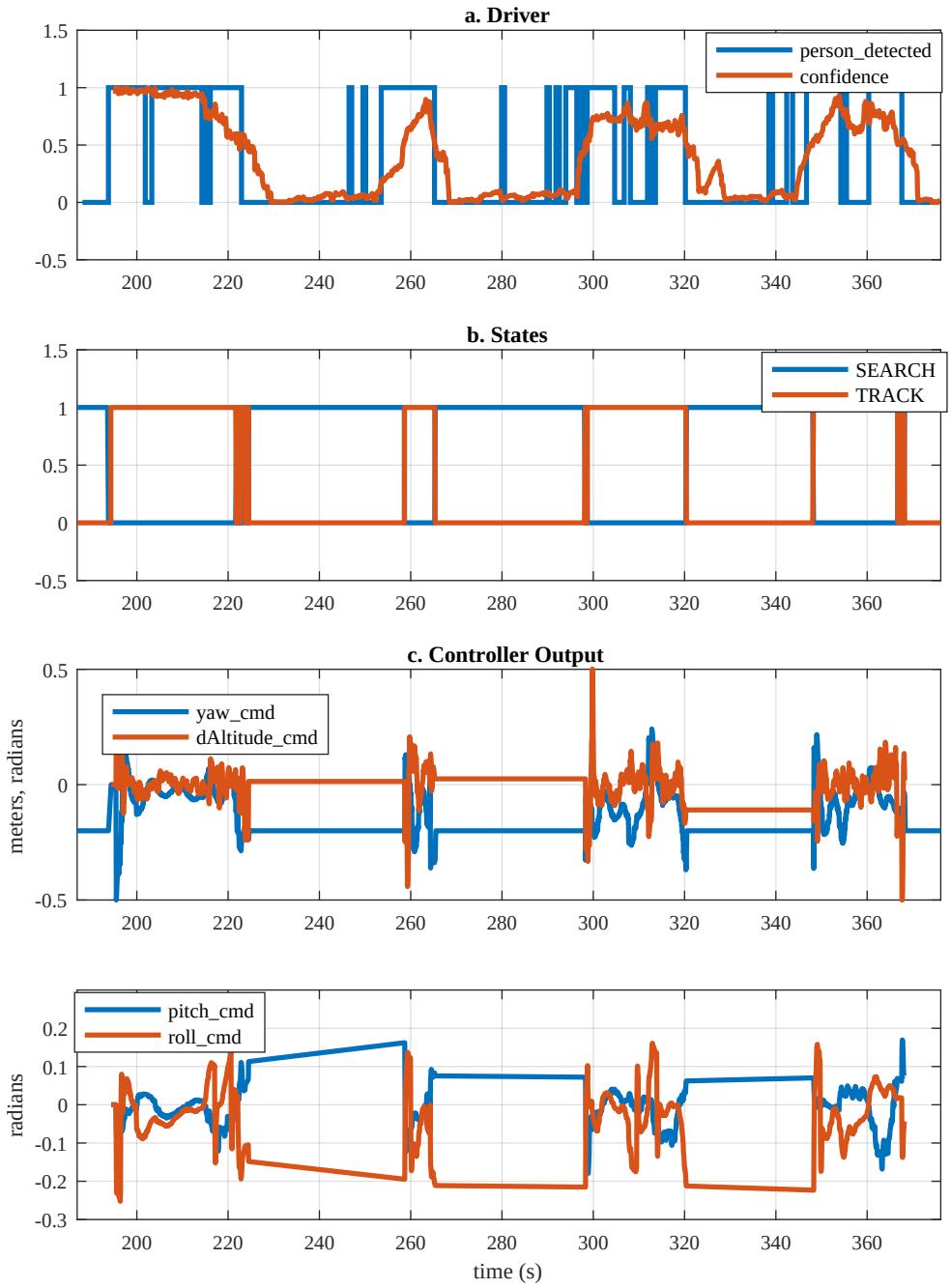

Figure 6. Flight Test Results

Figure 6 illustrates the output of the system. As discussed in the simulation results, the *person_detected* and *confidence* of the CMT tracker dictates the state of the system. It can be verified that the system switched from *SEARCH* to *TRACK* only when both the detector finds the human figure and the confidence variable exceeds the desired threshold (As can be seen during 260-265 seconds). This is important as both the variables are prone to noise as they are solely dependent on visual features. The controller outputs only in the *TRACK* mode. In the *SEARCH* mode the yaw is held constant at -0.2 rad/sec.

To summarize, the experiments conducted, both simulation and flight tests, validated the feasibility of using the hierarchical architecture of SMACH for SI&T mission. Through successive refinements of operational parameters we were able to demonstrate successful searching and tracking of unknown human targets using COTS UAVs. SI&T is a difficult challenging problem for autonomy because of the interconnection of the different subsystems and cluttered environments it operates in. We shoed a high level decision maker controlling all the subsystems for effective SI&T operations.

IV. Summary

An integrated Search Identification and Track (SI&T) approach was presented that combined sensing, motion planning and reactive control. The developed system is based on ROS and the SMACH planning framework, integrating state-of-the-art algorithms for object recognition and object tracking. The ability to coordinate complex components to enable continuous search and tracking using a sUAV illustrates the challenges and rewards of autonomy. In both simulation and flight tests the ability to communicate between the different complex components of SI&T to achieve a high level goals were demonstrated.

Future work will integrate and test more robust object identification systems using more capable UAVs. The un-instrumented COTS UAV provided both ease of use and limited payload capabilities. Using better and more sophisticated sensors including thermal cameras, lidars and object recognition tools, the system can be expanded for both day and night time operations. This paper provides the building block of integrating complex algorithms together in the decision making framework. Our research will build upon this work to make an autonomous deployable system for real life SI&T mission.

Acknowledgments

The authors would like to acknowledge the support of NASA's DELIVER project, and would also like to thank Colin Theodore and Kalmanje Krishnakumar of NASA Ames Research Center for their support of the research presented in this paper.

References

[1]A. R. Girard, A. S. Howell, and J. K. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1. IEEE, 2004, pp. 620–625.

[2]D. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of uavs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[3]"Air shepherd drones stop elephant and rhino poaching," https://www.indiegogo.com/projects/air-shepherd-drones-stop-elephant-rhino-poaching.

[4]H. Gintis, *Game theory evolving: A problem-centered introduction to modeling strategic behavior*. Princeton university press, 2000.

[5]T. S. Chow, "Testing software design modeled by finite-state machines," *IEEE transactions on software engineering*, vol. 4, no. 3, p. 178, 1978.

[6]M. Hejase, A. E. Oguz, A. Kurt, U. Ozguner, and K. Redmill, "A hierarchical hybrid state system based controller design approach for an autonomous uas mission," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3294.

[7]N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005.

[8]P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–511.

[9]"Smach wiki," http://wiki.ros.org/smach.

[10]J. Bohren and S. Cousins, "The smach high-level executive [ros news]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[11]J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the pr2," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5568–5575.

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671

[12]G. Nebehay and R. Pflugfelder, "Clustering of static-adaptive correspondences for deformable object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2784–2791.

[13]S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2548–2555.

[14]J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy, "Computer vision based general object following for gps-denied multirotor unmanned vehicles," in *American Control Conference (ACC)*, 2014.

[15]M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros : an open-source robot operating system," in *EEE International Conference on Robotics and Automation (ICRA 2009)*, 2009.

[16]M. Monajjemi, "Ardrone autonomy : A ros driver for ardrone 1.0 & 2.0," 2012. [Online]. Available: https://github.com/AutonomyLab/ardrone autonomy

[17][Online]. Available: https://github.com/gnebehay/CMT

[18]N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

Downloaded by NASA AMES RESEARCH CENTER on February 15, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-0671