2016 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 7-10, 2016. Arlington, VA USA

ResQuad: Toward a Semi-Autonomous Wilderness Search and Rescue Unmanned Aerial System

M. Talha Agcayazi, Eric Cawi, Arsenie Jurgenson, Parham Ghassemi and Gerald Cook, *IEEE Life Fellow* *

***Abstract*— In a search and rescue scenario, an expeditious response is critical since the radius of the search area increases by roughly 3 km per hour while the life expectancy of the lost individual decreases. To find the object of the search as quickly as possible, the incident commander creates an overlay probability map identifying high priority search areas and develops a search plan that allocates search assets to each area to maximize the probability of success. Conventional aerial searches have the ability to cover a large area quickly, but are costly and require highly trained personnel to operate. In this work we present results of a proof of concept Search and Rescue Unmanned Aerial System, ResQuad, that capitalizes on the advances in drone industry while allowing for reduction in cost and retaining the advantages of aerial perspectives. Given the probability map from the search commander and the system constraints, a near optimal path is determined by the Path Planner in our system ensuring the coverage of high probability areas. While the Unmanned Aerial Vehicle (UAV) follows the path, an on-board computer with a camera processes and collates the data using an anomaly detection algorithm, saving the results to its memory for the base to verify in real-time.**

I. Introduction

From 1992 to 2007, a combined $59 million dollars had been spent for 65,439 Wilderness Search and Rescue (WiSAR) incidents in the various United States National Parks [1]. The average cost of each WiSAR operation was approximately $900. Although money is always a factor in the quality of a WiSAR mission, the most important resource is actually time. To save time, at the start of a land search scenario, the incident commander uses statistically based factors such as distance from the last known point, roughness of terrain or the elevation of the search area to create a map of high priority search areas. The resulting problem is to effectively allocate search resources to different areas to maximize the probability of finding the target. Currently, most field commanders use simple models to quickly allocate their available search teams [2]. Each team must then plan a path to maximize the coverage of its assigned search area.

Although conventional human controlled helicopter and fixed wing aircraft can aid the search process, their disadvantage is in their inherent inability to maneuver closer to the target and the high cost in both fuel and man-hours. On the other hand, a Micro Unmanned Aerial Vehicle (UAV) such as a quadcopter gives the ability to maneuver closer to the target and gives the possibility of delivering objects at a fraction of the cost.

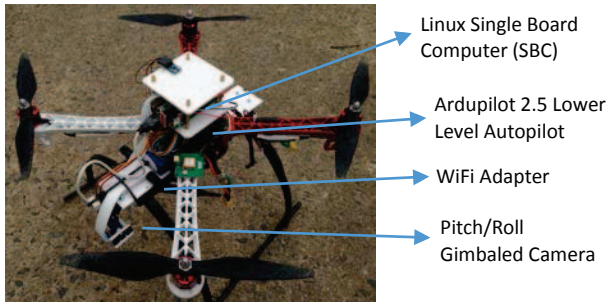

Figure 1. The UAV developed to use in testing our algorithms.

Brigham Young University (BYU) is at the forefront of developing UAV Search and Rescue systems. Their systems have efficient path planning algorithms [3] and can transmit a video stream to the base station where dedicated personnel search the footage for objects/persons of interest [4], [5]. The use of thermal cameras [6] and even the fusing of imagery from a thermal camera and a normal camera [7] have been proposed for search operations. Although these approaches are novel and aids WiSAR missions at a great scale, there must be steps taken to remove the need for a person who constantly keeps looking at the incoming imagery. The process of sending imagery regardless of spectrum does not only introduce latency and noise to the system [8] but it introduces increased human error at higher altitudes and speeds.

There exists a need for a UAV system which exploits the use of computer vision algorithms on-board the craft when searching for the individual or the traces of the lost individual in WiSAR missions to take full advantage of autonomy. Our work builds on the existing path planning algorithms with computer vision to make a proof of concept semi-autonomous search and rescue unit: the ResQuad. The on-board computer runs a real-time computer vision algorithm to detect and save anomalous patches of pictures in each waypoint to its memory for the base station to verify. Once the base station personnel verify a detected anomaly as being the lost object of the search, they can call the UAV back to where it was launched using manual commands and send in rescue crews to where the object of the search was discovered. The system is implemented using a multi-copter UAV rather than a fixed wing to take advantage of the agility and hovering capability provided by multi-copters. Since our UAV does not make final decisions on the detected anomalies, it is a semi-autonomous solution which combines the pattern recognition capabilities of the human operators and the fast monitoring and processing capabilities of the UAV and the on-board computer. The UAV developed and used in this project is shown in Figure 1.

*All authors were with George Mason University, VA at the time of research. E-mail: {magcayaz, ecawi, ajurgen1, pghassem, gcook}@gmu.edu

978-1-4673-9333-1/16/$31.00 ©2016 IEEE

In this work we have focused on two important sub-areas that are critical to a semi-autonomous UAV search system, Path Planning and Anomaly Detection. In each of the sections describing the sub-areas we present their respective results which help us evaluate the performance of the complete system. The paper continues with a general overview of the hardware in Section II. The Path Planning algorithm is described in Section III and the Anomaly Detection system in Section IV. The paper finishes with a discussion of the first results of the final integrated system in Section V. The state diagram showing the use of our system is shown in Figure 2.

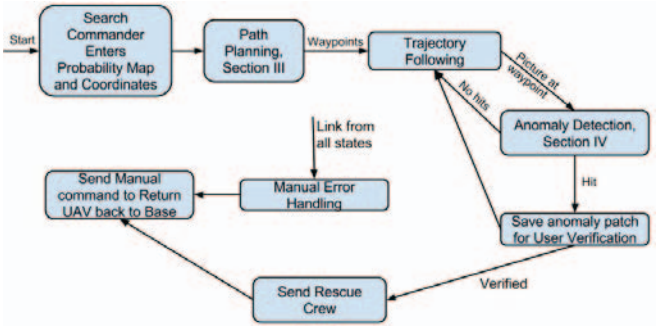

Figure 2. State Diagram for the ResQuad system.

II. Hardware Overview

We developed a custom quadcopter for this project which can be seen in Figure 1. The hardware connections of our system could be seen in Figure 3. The frame and the components were specially selected to meet our autonomy requirements while keeping the cost down. We divided the computation on-board the UAV into two pieces to guarantee flight stability with the off the shelf lower level Arducopter 2.5 autopilot [9] while running our anomaly detection algorithm with the higher level Raspberry Pi 2 [10]. The Arducopter 2.5 flight controller uses its on-board IMU and barometer sensors to stabilize the vehicle in flight. It also provides a waypoint navigation mode which uses the GPS to take the quadcopter to given waypoints. The Raspberry Pi 2 is a credit card sized Single Board Computer (SBC) with a quad-core ARM based CPU and a 1GB of RAM. To communicate with the quadcopter we set up an access point on the Raspberry Pi 2 using a WiFi adapter. As the camera, we used the Raspberry Pi camera which is connected to the quadcopter using a gimbal to compensate for aggressive maneuvers of the vehicle while pointing the camera to the search area at all times. The anomaly detection algorithm runs on the Raspberry Pi 2 and saves the data to its on-board memory. Finally, during the search, the un-edited pictures and flight data are collated for post-flight verification. The main components like the lower level and the higher level controllers in our UAV search system are modular making the upgrade process easier to handle in the future. All in all the cost of one UAV search system is about $300 dollars.

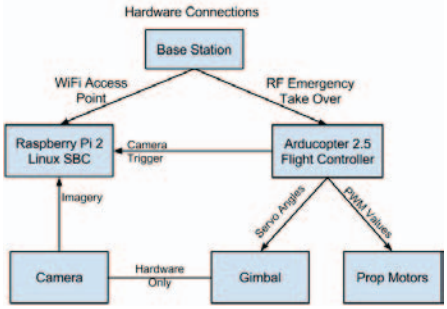

Figure 3. Hardware connections in our UAV system

III. Path Planning

A. Introduction

The literature in autonomous path planning to maximize some objective function provides many choices for algorithms. Lin and Goodrich [3] give an overview and comparison of local hill climbing, complete coverage, and evolutionary algorithms for a variety of cases. Search algorithms like A* and Markov decision processes are also commonly used algorithms for robotic path planning. In this paper, we chose to implement local hill climbing algorithms due to its simplicity. Hill climbing algorithms are easy to implement on the UAV, and paths can be generated at the camera resolution without huge increases in computation time. This simplicity also lends itself well towards future work incorporating real time objective updates.

The path planning algorithm performs three steps for every level of the search planning. First, the search area is divided into rectangular cells, and with each increasing level the previous level's cells are divided into smaller cells. Then local hill climbing algorithms are used to generate a path through the level 0 search region. Finally search effort is assigned to each cell in the path by optimizing a function presented in [11] to minimize the probability of missing the target subject to flight time constraints. This determines the allocation of optimal flight time to each cell in the path. In the next step, hill climbing algorithms are used with the flight time generated in the previous step to generate a path through the level 1 search region. This process repeats itself until the algorithm has reached the resolution matching the vehicle's camera footprint. The final probability of success P_s is defined as $P_s = \sum_{cells} P_d * P_i$, where P_d is the anomaly detection algorithm's probability of detection given that the subject is in the cell and P_i is the probability that the target is in cell i.

Mathematically, the path problem is extremely complex because there are a huge number of free parameters (e.g. imperfect parameters, possibly moving subject, changing priority search areas, etc.). Because of this complexity most search teams do not have the time to computationally optimize their search paths. This project focuses on a scenario with a static object, a static set of priority search areas modeled by a probability distribution, a fixed search area, and one search resource. With these simplifications, paths are efficiently created to optimize the coverage of the area subject to the limited flight time of a search vehicle. To make this process user-friendly, the algorithms used are handled by the system, while the only required user inputs are the coordinates and probability distribution of their search area.

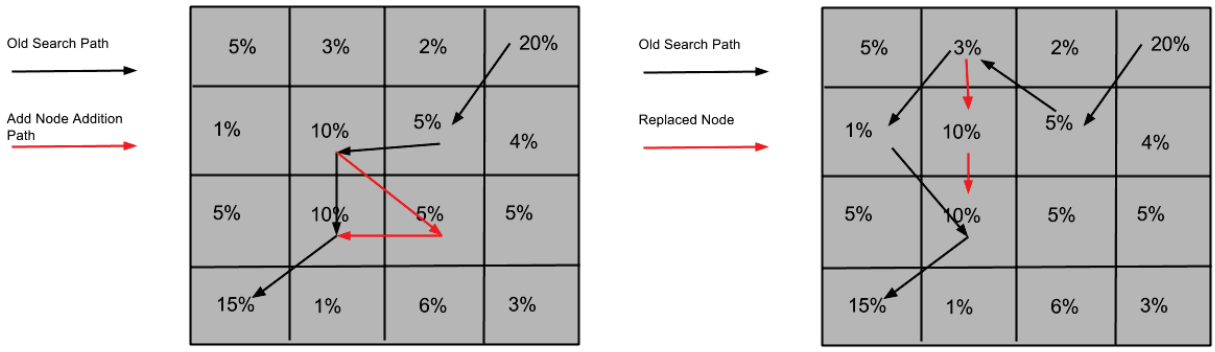

Figure 4. Algorithm choses to add a node or to replace the node to increase the overall probability of success.

B. Hill Climbing Algorithm

In a search scenario, the number of cells in the high resolution areas can be very large, which makes optimal path planning nearly impossible for humans. Computers, however, can use search algorithms and optimization techniques to find paths that are very close to optimal. One option is a local hill climbing algorithm with no set destination. At a cell, the algorithm compares the probability at each of the nearest neighbors, then moves to the cell with the greatest probability. To prevent oscillation between two high probability cells, the probability once a cell has been visited is given by a Bayesian update: $P_{inew} = P_i - P_d*P_i$. This decreases the probability substantially but accounts for the fact that the unit may have missed the subject. In the event that the algorithm is caught in a spiral, it will revisit the cell with the highest remaining probability and hopefully escape the spiral [3].

There are two main disadvantages to this algorithm. First, it can become trapped in local maxima and miss a higher probability area. Second, it has no fixed ending point, which makes planning an efficient flight path between regions harder. To best utilize flight time, it is important to start a path on one edge of a region and end at the beginning of the next. Therefore, we use an algorithm to incorporate a forced starting point, ending point, and a random restart to avoid local maxima. The algorithm starts with a randomly generated path between the starting and ending point, and then either adds points to the path or replaces existing points. These rules for changing the path are described in Figure 4. At each step, the algorithm cycles through all possible changes and chooses the change that causes the best increase in probability of success [15]. Figure 5 shows an example of a search path generated across four large search areas using these hill climbing algorithms. In this case equal flight time was allocated to each search region. In the top and bottom left search regions, the algorithm is used with no set destination because there is only one fixed endpoint at the starting and ending regions. The path is laid on top of the original probability distribution with red/warmer colors representing higher probability, and blue/cooler colors representing lower probability.

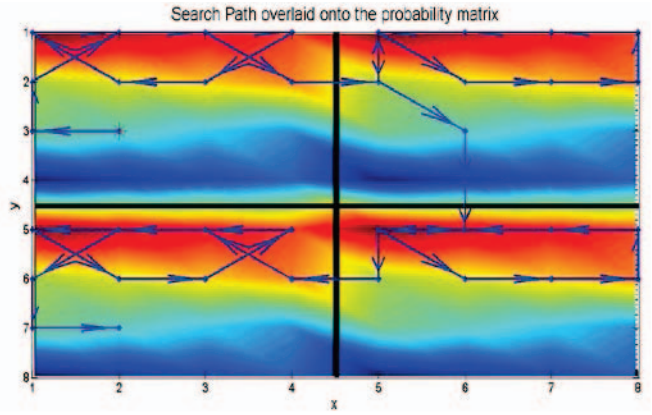

Figure 5. Example Search Path

C. Optimally Allocating Flight Time

Kadane[4] provides a search algorithm that minimizes the probability of failure for a set of *i* locations and *j* repeated searches in each location. Using P_i as the probability of the target being in location i, α_i as the probability that the subject is overlooked given it is in the cell i, and *nij* as the effort apportioned by the system at cell *i* for the *jth* search, the function to minimize is given as:

$$S = \sum_{i=1}^{\#cells} P_i e^{\sum_{t,j} n_{ij} \log(a_{ij})}$$

With c_l,t,j as the cost of search area *l* with technology *t,* for the *jth* time and C as the total allowed cost, this function is subject to the constraint as follows:

$$\sum_{i\ j} n_{ij} c_{ij} = C$$

Using the simplifying assumptions of one search technology, the same overlook probability every time a cell is searched, and the simplified flight time model, this equation is reduced to minimizing:

$$S = \sum_{i=1}^{\#cells} P_i e^{n_i \log(a_i)}$$

Which is subject to:

$$\sum_i n_i c_i = C$$

This algorithm gives the best possible effort allocation. It does not generate a path through the search area, but is used to guide the amount of flight time spent in each cell.

D. Results

Putting these algorithms together, Figure 6(a) shows an example of the module run using a circular normal distribution. This distribution was chosen because it is easy to calculate the optimal path. For this lognormal probability distribution and an arbitrary flight cost of .75*num_cells = 300 cells, the probability of success is 0.8572. Compared to the exact solution of concentric rings around the center point, which yields a probability of 0.8715, the heuristic algorithm captures nearly all of the optimal path. Figure 6(b) shows a cross section of the original probability distribution and the updated distribution after the search. After a path has been generated, it is transformed to latitude/longitude coordinates that can be uploaded into the flight controller. Figure 7 shows an example search path plotted on Google Maps. The example is generated with a similar probability distribution to Figure 5, but to account for the rectangular shape of the soccer field only two large search regions are used. Algorithms such as Lin's work in [12] or a Markov Decision Process offer more sophisticated alternatives and a better score, but using the greedy algorithm in conjunction with Kadane's optimal allocation provides a simpler alternative that comes very close to an optimal solution, and as such is easier to implement on the system.

These algorithms are used to generate an increasingly complex path through a search region, stopping at the camera footprint of the vehicle. It is important to note that the only goal of this was to generate the best possible plan of success, and many sudden turns will put stress on the vehicle. Further work will relax the flight time assumptions into a more real scenario factoring in battery power, wind, and other flight factors.

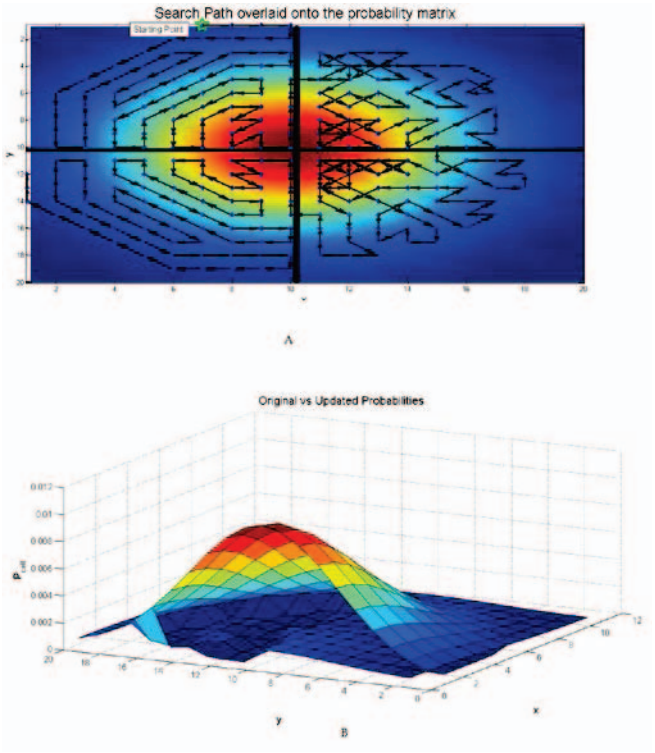

Figure 6(A). Example using a circular normal distribution. Figure 6(B) cross section of the original probability distribution and the updated distribution after the search.

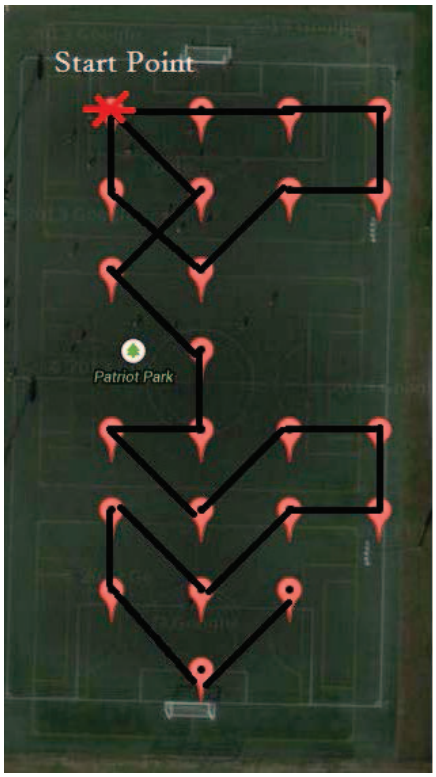

Figure 7. Example Search Path plotted on Google Maps. Patriot Park, Fairfax, VA

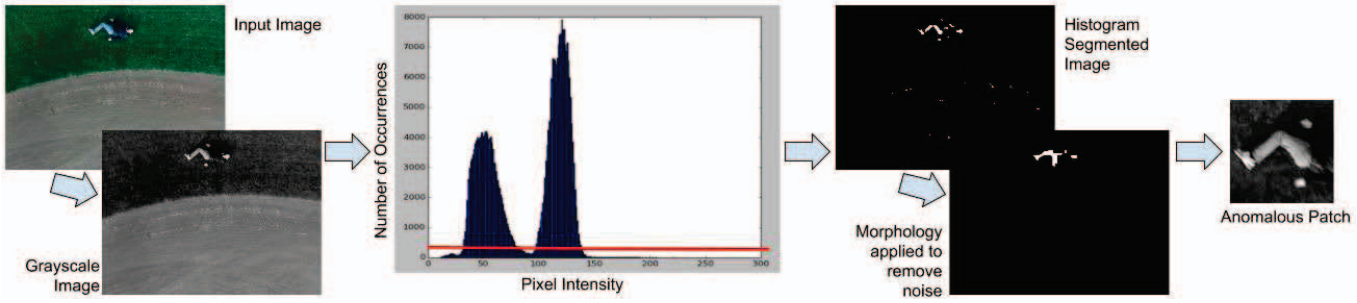

Figure 8. Process diagram showing how the anomaly detection algorithm works. The images are first converted into single band gray scale images then they are segmented using their histogram pixels with a threshold on the number of occurrences for each pixel intensity. After applying morphology operations, the biggest object is taken as the anomaly.

IV. Anomaly Detection

A. Introduction

The main goal of the anomaly detection module is to take pictures at waypoints, segment the images and suggest patches of the images as anomalies to the user in real-time.

While taking aerial pictures for our dataset, our group tested multiple algorithms to detect the humans in the pictures. Our first attempts involved developing a haar based classifier and testing pedestrian detection classifiers from other datasets to find the lost individual in the images. These efforts were soon ceased because we realized that expecting the lost individual to lay in a certain position was impractical. We then experimented with different color anomaly detection algorithms which would allow us to detect victims in different orientations as long as they had on them anomalous colors. Anomaly detection algorithms work well in wilderness since nature usually has similar colors and any object not in the same color or brightness could help find the traces or the lost individual him/herself.

Borghys et al. [16] and Morse et al. [17] describe different methods to detect anomalies in hyperspectral imagery. Although these algorithms perform well on pictures with complicated environments, they are complex and are infeasible to run on-board the UAV in time sensitive search and rescue operations since they are computationally expensive and thus require video transmission to a fast computer. The most similar work to our anomaly detection algorithm is by Brown et al. [18] where the authors propose an algorithm to find a mean color intensity in a color band (Red, Green and Blue) and filter out the band to separate the unusual color intensities. Although this algorithm works similarly to our algorithm, it has no way of selecting a threshold for what is 'unusual' enough not to get segmented as the background and the speed performance is not tested since the algorithm was not fully implemented. Our anomaly detection algorithm runs in real-time on the on-board computer to filter the incoming frames from the camera to pick only the image patches that require further observation and verification from the user at the base station. Since we provide a method to train our threshold for determining what is 'unusual' enough, it is appropriate to call our algorithm *a supervised histogram based color anomaly detector*.

B. Data Collection

Unlike many other methods to detect prone humans in aerial perspective, our method uses its own database for training. This makes our system both accurate and robust for our test conditions, and matches the way contemporary WiSAR teams train their personnel. To keep data consistent between training and prediction cases we used the same camera, the Raspberry Pi camera, when taking training pictures. The pictures were taken with a resolution of 640x480 pixels while flying with a time-lapse script running on the on-board SBC. Initial pictures from the dataset were collected with a uniform background, however, to increase diversity we took pictures in different areas with non-uniform backgrounds. Our final dataset has 597 positive and 266 negative images. Positive pictures were each analyzed and the ground truth positions were recorded for fast algorithm testing. Some example pictures from our dataset could be seen in Figure 9.

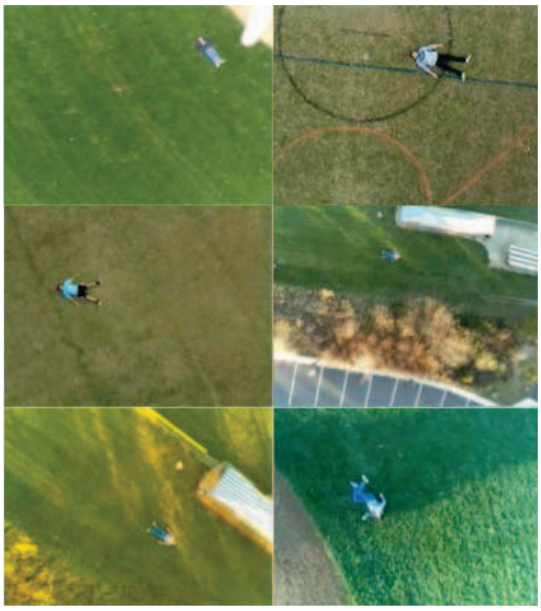

Figure 9. Example pictures from the dataset. Only patches of these pictures with human bodies are labeled as ground truth regions.

C. Methodology

Figure 8 shows the flow of our algorithm from the start where it captures an image to its suggestion of the patch image. To meet the requirements of a real-time system running on an on-board computer, our algorithm starts by converting the RGB image into a grayscale single band image. Our algorithm then computes the histogram of the image based on pixel intensities and goes through each intensity value to find the intensity values that occur more times than the threshold value. A method to train the optimal threshold value based on our data set is given in the next sub-section. After finding the

intensities that we want to predict as normal pixel intensities, our algorithm segments the original image by setting normal pixels as dark values and anomalous pixels as bright values in a new image of the same size. After the image is segmented, using OpenCV functions [19], the algorithm applies morphology to remove noise and outputs the <x,y> pixel coordinates of the biggest anomaly along with a patch of the image where the anomaly appears. By applying morphology and picking the biggest anomaly, our algorithm essentially picks the region that has the most anomalous pixels together. Aside from the simplicity of our algorithm, using well vectorized libraries we were able to achieve our real-time requirement. The waypoint picture taking command is generated in an interrupt scheme and puts each new picture into a queue to enable faster traversal of the waypoints. In between the waypoints, the anomaly detection algorithm continues to find the anomalies and save the suggestions. In a future anomaly detection algorithm, searching all of the available color bands together with searching for anomalous textures might give better results at the cost of computation power, which might be compensated with better processors of the future. This extension is mentioned for completeness.

D. Training the Threshold value and Results

To find the previously mentioned intensity threshold value that would maximize the True Positive (TP) rate of pointing the correct anomalies based on our positive dataset, a script was written to run our anomaly detection algorithm for pictures in both the positive dataset and the negative dataset while changing the threshold value at every run. These tests proved both the advantage and the disadvantage of our anomaly detection system. While the maximum TP rate of our anomaly detector came out to be 96% for a threshold of 54 through 56, the False Positive (FP) rate at the same threshold levels are at %95. This suggests that although our algorithm will detect almost all of the objects of interest, the user in the base station will need to sort them since he will also see many FPs. Figure 10 contains the plot of TP rate vs threshold and Figure 11 contains the plot of FP rate vs threshold. Even though our systems has a high FP rate, it performs its duty of filtering the incoming frames and suggesting only the anomalies to introduce a much needed semi-autonomy which takes advantage of the computing power onboard while decreasing user error when analyzing aerial imagery. For completeness, in a future version of our anomaly detection algorithm, the threshold could be changed depending on the elevation of the craft and the expected size of the anomaly since the size of the anomaly visible to the craft will change based on how high the picture was taken. This will also decrease the FP rate and pick anomalies for an expected size of the lost item, or individual. The frame rate of the anomaly detection algorithm was tested on two platforms and recorded in Table 1.

Table 1. Computational Performance of our Anomaly Detection Algorithm.

Anomaly Detection Algorithm Computation Performance		
Computer	Speed	% of Processor used
Raspberry Pi 2 – A7 @ 0.9GHz	4 Hz	25%
Laptop – I5 @ 1.7GHz	110 Hz	25%

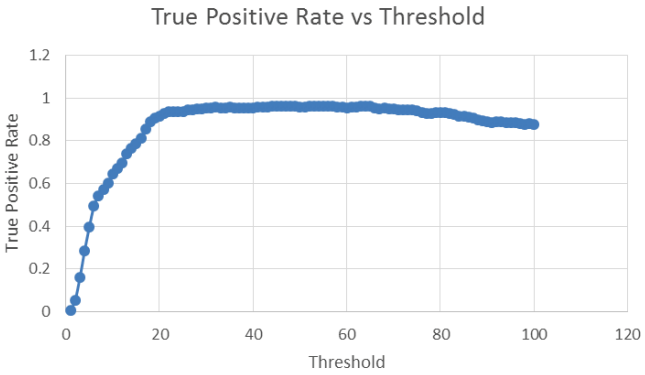

Figure 10. True Positive rate vs Threshold

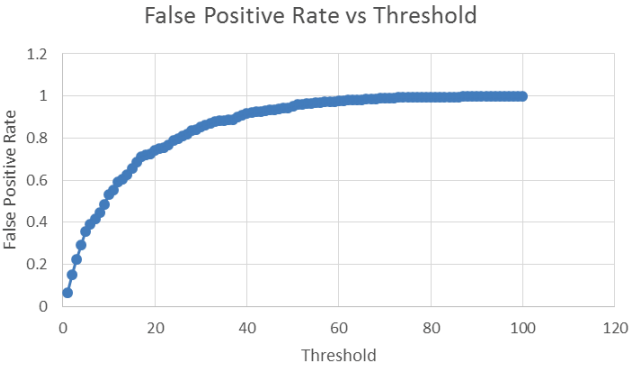

Figure 11. False Positive rate vs Threshold. Notice the speed at which the FP rate increases.

V. First Results Of the Final Integrated System

After developing the required submodules, Path Planning and Anomaly Detection, explained above, they were put together to collect the first results of the integrated system. Figure 12 shows the trajectory of our UAV system as it traversed through a set of 12 waypoints covering a rectangular area in a lawnmower pattern. The autonomous traversal is shown as green lines in the figure. After the UAV finished going through the set of waypoints, it issued the return to launch command which made it autonomously return to the launch coordinates. The landing and the take-off was performed by the pilot. The video showing the autonomous waypoint navigation of the system is available online:

https://goo.gl/bliMDC

We tested the anomaly detection module on the on-board SBC both when the UAV was flying through the waypoints autonomously and when it was flown manually. In these runs we used the trained value of 54 for the anomaly detection threshold. Some of the anomaly detections in these runs could be found in Figure 13 along with the full images.

Although the first results verified that our sub-modules worked together they also proved that more rigorous testing must be performed to accurately verify the system performance. For the time being it serves its purpose of being a proof of concept system which brings together the Path Planning and Anomaly Detection for a semi-autonomous WiSAR UAV system.

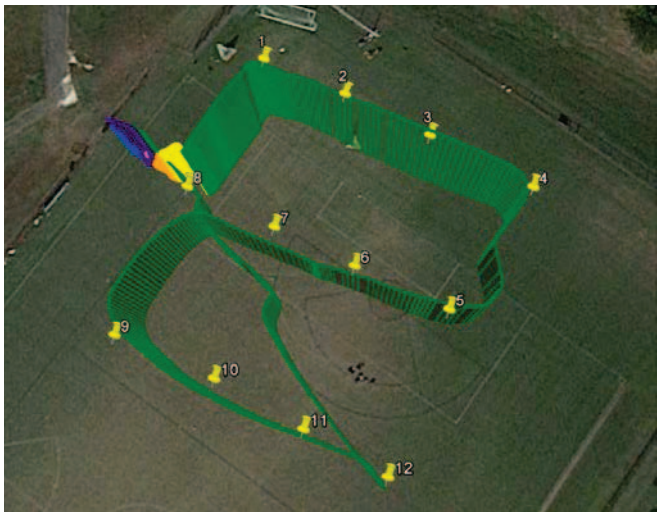

Figure 12. Trajectory of the UAV, shown on google earth, recorded with GPS coordinates as it traversed through a set of 12 waypoints. Green Lines represent autonomously flown locations.

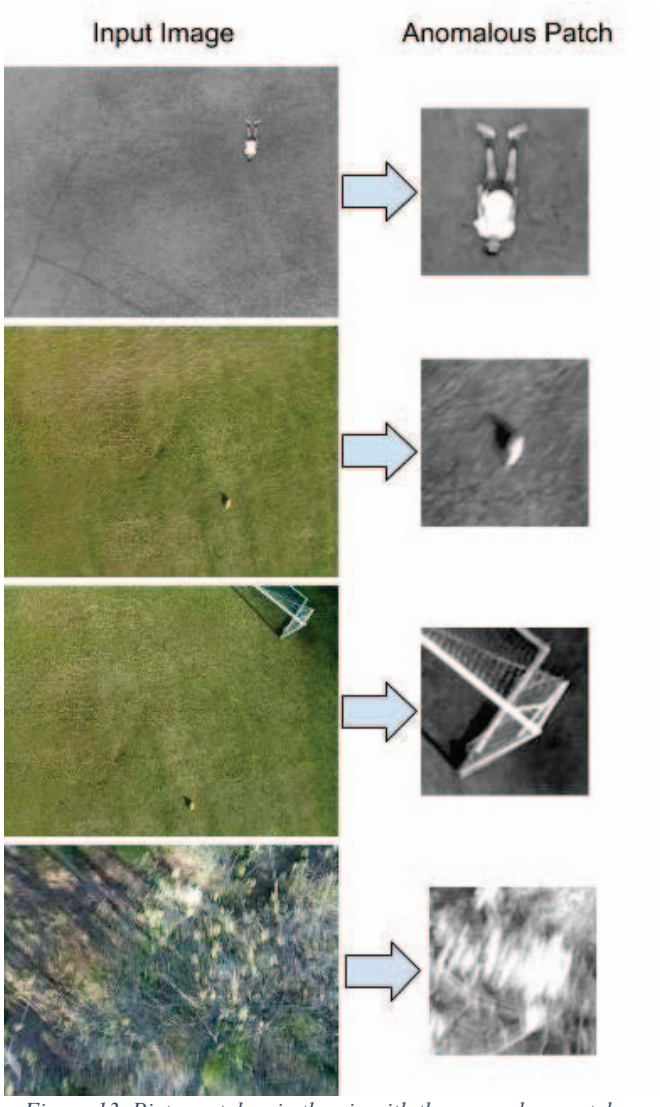

Figure 13. Pictures taken in the air with the anomalous patches computed using the trained threshold value.

VI. Conclusion and Future Work

There is a clear opportunity for UAVs and especially for Micro UAVs in Search and Rescue operations. Towards exploiting this opportunity, in this work we have presented our proof of concept semi-autonomous UAV Search and Rescue system, the ResQuad. Our UAV Search and Rescue system is unique in that, like its predecessor UAV systems, it can plan its own paths using our specially designed Path Planner and can additionally suggest anomalies in the images that it takes.

As Future works, the system could be rigorously tested and the modules could each be extended. The Path Planning Algorithm could be extended by removing the assumption of having a static object and a static search area. The Anomaly Detection algorithm could be extended by incorporating the size information and the elevation of the picture to better determine the anomalies.

VII. Acknowledgements

The authors would like to thank the Electrical and Computer Engineering Department of George Mason University for monetary funds. Our thanks are extended to Dr. Dan Lofaro, Dr. Charles Twardy Tom Kratzke and Dr. Chris Vo for their help and advice.

VIII. References

[1] Heggie, Travis W. Dead Men Walking: Search and Rescue in US National Parks. Wilderness & environmental medicine. (09/2009) , 20 (3), p. 244 - 249.

[2] Lost person behavior: A search and rescue guide on where to look for land, air, and water, 1st, DbS Productions, Charlottesville, VA, 2008.

[3] Lin, L., & Goodrich, M. "UAV Intelligent Path Planning for Wilderness Search and Rescue(October 2009)" in The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MO, 2009, 709-714.

[4] Lin, Lanny, Roscheck, Michael, Goodrich, Michael, AND Morse, Bryan. "Supporting Wilderness Search and Rescue with Integrated Intelligence: Autonomy and Information at the Right Time and the Right Place" AAAI Conference on Artificial Intelligence (2010): n. pag. Web. 22 Jan. 2016

[5] Goodrich, M. A., Morse, B. S., Gerhardt, D., Cooper, J. L., Quigley, M., Adams, J. A. and Humphrey, C. (2008), Supporting wilderness search and rescue using a camera-equipped mini UAV. J. Field Robotics, 25: 89–110.

[6] Hammerseth, Vegard B. "Autonomous Unmanned Aerial Vehicle In Search And Rescue." (n.d.): n. pag. Diva-portal.org. Norwegian University of Science and Technology, June 2013. Web. Oct. 2014.

[7] Eggett, Dennis; Goodrich, Michael A.; Morse, Bryan S.; and Rasmussen, Nathan, "Fused Visible and Infrared Video for use in Wilderness Search and Rescue" (2009). All Faculty Publications. Paper 865.

[8] Ehsan, S.; McDonald-Maier, K.D., "On-Board Vision Processing for Small UAVs: Time to Rethink Strategy," in Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conf. on , vol., no., pp.75-81, July 29 2009-Aug. 1 2009

[9] [Online]. Available: http://dev.ardupilot.com Accessed May. 06, 2015

[10] [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ Accessed May. 06, 2015

[11] Joseph Kadane, "Optimal Discrete Search with Technological Choice," in press.

[12] L. Lin and M. A. Goodrich. Hierarchical Heuristic Search Using a Gaussian Mixture Model for UAV Coverage Planning. IEEE Transactions on Cybernetics. 44(12), pp. 2532-2544, March 2014

[13] Sava, E., Twardy, C., Koester, R. and Sonwalkar, M. (2016), Evaluating Lost Person Behavior Models. Transactions in GIS, 20: 38–53. doi:10.1111/tgis.1214

[14] MapSAR: GIS for Wildland Search and Rescue. [Online] Available: http://www.mapsar.net/

[15] S. H.Jacobson, E Yucesan, "Analyzing the Performance of Generalized Hill Climbing Algorithms," J. of Heuristics, volume 10, pp 387-405, 2004.

[16] Borghys D., Kåsen I., Achard V., and Perneel C., "Hyperspectral Anomaly Detection: Comparative Evaluation in Scenes with Diverse Complexity," Journal of Electrical and Computer Engineering, vol. 2012, Article ID 162106, 16 pages, 2012.

[17] Morse, B.S.; Thornton, D.; Goodrich, M.A., "Color anomaly detection and suggestion for wilderness search and rescue," Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on , vol., no., pp.455,462, 5-8 March 2012

[18] Brown A., Estrabrook J. and Franklin "Sensor Processing and Path Planning Framework for a Search and Rescue UAV Network" Worcester Polytechnic Institute, Worcester, MA, 2011.

[19] Bradski, G. (2000). Dr. Dobb's Journal of Software Tools