University of California, San Diego
Autonomous Airplane Team

2016 Student UAS Competition

Technical Paper

Project Manager: Ryan Nakutin
Airframe Leads: Zhenhua Li/Leonardo Chan
Embedded Lead: Changtong Qiu
Software Lead: Marco Flowers

Competition Team:
Otto Jursch, Matthew Ho, Winson Quan, Vachan Vadmal, Michael Park

Faculty Advisor: Dr. Ryan Kastner

Abstract

The UC San Diego Autonomous Airplane Team (UCSD AUVSI) is proud to present the Sig Rascal 110, our flight platform for the 2016 AUVSI Student UAS Competition. This system is a robust and versatile platform for conducting autonomous aerial missions, including intelligent path planning, ground target identification and classification, and virtual obstacle avoidance. This system is completely student-designed and built, and houses our onboard computing system, autonomous flight hardware, and image capture platform, with an emphasis on high performance and payload adaptability. This paper demonstrates a detailed analysis of our design methodology, an evaluation of our testing procedures and results, and the measures taken to ensure safety in order to demonstrate our system's competition readiness.

May 2016

Contents

Figure 1: The UCSD AUVSI Team with the Sig Rascal Airframe (Photo Credit: Eric Lo)

1 Mission Requirements Analysis

The 2016 UCSD AUVSI competition system is designed to be fully capable of accurately and efficiently performing the majority of competition tasks. This system represents the culmination of 11 years of competing, and with this experience we have developed a robust and reliable platform that is able to fulfill the mission requirements.

1.1 Airframe

The airframe must have the ability to perform its mission objectives under the duration of the mission time length, as well as being able to house the necessary flight and payload systems. This includes an endurance of 25 minutes at nominal takeoff weight of 20 lbs. The payload must also be easily accessible in order to minimize setup time, which is achieved in under 10 minutes. Furthermore, it must also be portable enough for convenient transportation to competition and to the field.

1.2 Flight Systems

The flight hardware contained in the system must be capable of autonomously directing the vehicle throughout the entirety of the mission, from takeoff to landing. This hardware payload must be capable of guiding the plane through a prescribed set of waypoints, immediate re-tasking for updated mission objectives, and enacting safety measures in the event of a system failure, including manual takeover. In addition, our communications equipment must be able to maintain constant data uplink and downlink throughout the entirety of the mission in order to prevent data loss and retain control over the system.

1.3 Payload Systems

The payload systems in place on the vehicle are primarily tasked with gathering imagery to be processed for ground targets. This is performed by a DSLR mounted in the airframe which is triggered to take images at greater than 1 frame per second, so as to cover enough of the search area to adequately search for targets.

1.4 Software System

The primary task of the software system is to direct the mission of the autonomous system with minimal human input. Mainly, the software system must be capable of processing the gathered imagery for ground targets and identifying the characteristics per competition specifications, in real-time, during the course of the mission. This also includes prescribing waypoints and developing an optimized search path in the competition area, and accomplishing SRIC tasks.

1.5 Expected Mission Performance

Will accomplish: Autonomous flight, search area, actionable intelligence, emergent target, interoperability
Will attempt: ADLC, SRIC, SDA, QR
Will not attempt: Off-axis target, airdrop

Based on flight testing and previous results we believe that we will accomplish all the tasks listed under will accomplish. We have tested ADLC, SRIC, SDA, and QR, and feel confident in our ability to attempt these tasks at competition.

1.6 Programmatic Risks

Throughout the project, we identified several risks to our project, and took our best steps to eliminate these risks. The primary risk we anticipated was the integration of our new path planning software. To minimize this risk, we extensively tested our path planning algorithms with Software in the Loop (simulated flight) testing, as well as by developing visualization tools to better understand and debug our algorithms. We also anticipated potential problems in putting together the separate parts of our system. To mitigate this risk, we made sure we tested our whole system several times before flying the plane.

2 Airframe

The Sig Rascal 110 airframe is a lightweight, high performance wood frame-mylar skin aircraft designed for simplicity in set up, modification and flight. As a result of its size and lifting capacity, it provides an optimal set up for carrying a high resolution camera along with associated processing hardware.

2.1 Design

The airframe consists of a plywood fuselage, using balsa for shear paneling, in order to provide a stiff and light frame. The wings consist of a plywood spar-rib-skin structure, with plywood reinforcements around areas where point loads or other major loads are expected, and a combination of balsa and mylar skin in order to take all lifting pressure loads during flight. An aluminum spar connects the two wing halves, and this assembly is slotted and bolted into the fuselage. The wing halves are additionally reinforced by aluminum struts which help to alleviate much of the bending loads during flight. The landing gear set up is a conventional landing gear system with a two wheel main gear and a single pivoting rear wheel for taxi-ing.

Aerodynamically, the airframe was designed for a combination of efficiency and cost-effectiveness. It was built with elliptical wings in order to minimize the effects of induced drag, along with other drag-reduction measures, such as adding an airfoil profile to the struts. The airframe has dihedral wing design spanning 110 inches. The chord of the wing is 16 inches at the root. The flight characteristics and overall size of the airframe almost identically mimic that of last year's plane used at competition.

The fuselage houses an internal payload with a capacity of 1600 in^3. The payload is divided into two major sections: the flight box, which houses the ESC, the flight batteries that power the motor, and the payload batteries that power all remaining onboard electronic systems; the payload section, which houses the fixed Canon SL-1 DSLR camera, the onboard computer, other equipment required for telemetry, imaging and onboard computing platform, and control surface servos. Additionally, all of the communications equipment are located in this section. The internal payload can be accessed from the top after removing the wings and allows for quick repairs and adjustments to the payload configuration. The extra payload space and unused weight allows for the SIG Rascal 110 to house additional sensors and antenna to accommodate for a wide range of mission objectives. Despite its size, this airframe is relatively portable, as it can be easily fit into the back of a standard sedan. Setup and disassembly typically takes ten minutes for the system to go from stowed to mission-ready. Installation of the wings, tail, struts, and landing gear can be easily done with a Phillips screwdriver and Allen key set. The batteries and propeller are easily accessible from the nose of the plane.

The airframe is propelled using a Neutronics 1515-2Y geared in-runner motor, in conjunction with a standard 19 x 8 APC propeller, which allows the airframe to fly at speeds anywhere between 25 and 40 mph at nominal takeoff weight. The motor is controlled by a Castle Creations Phoenix HV-Edge-60 electronic speed controller. The propulsion system is powered by a set of two 6-cell 5200 mAh lithium-polymer batteries connected in series and provides 44.4 volts, with a maximum current draw of 60 amps. This power configuration results in an average flight time of 25 minutes.

2.2 Autonomous Control

The Sig Rascal autonomous flight system utilizes a Pixhawk autopilot, designed by PX4 Open-Hardware Project and manufactured by 3D Robotics. The Pixhawk is an advanced autopilot system capable of autonomous waypoint navigation and waypoint re-tasking during a mission. Compared to the previously used

ArduPilot Mega 2.6, the Pixhawk features a 32-bit co-processor with an integrated failsafe processor in case of system failure, a throttle safety switch that provides additional protection from unexpected arming, a backup accelerometer and gyro to ensure the overall stability of the system, and a faster CPU with more RAM that results in the ability to add features and better performance during flight

Powered by the APM Plane firmware, the failsafe function on Pixhawk is a robust solution that ensures the safety of the entire system. It allows for quick enough reactions to throttle, telemetry and GPS failure to prevent catastrophic failure. For the detailed behaviors of this system, please refer to the Safety section.

3 Embedded Systems

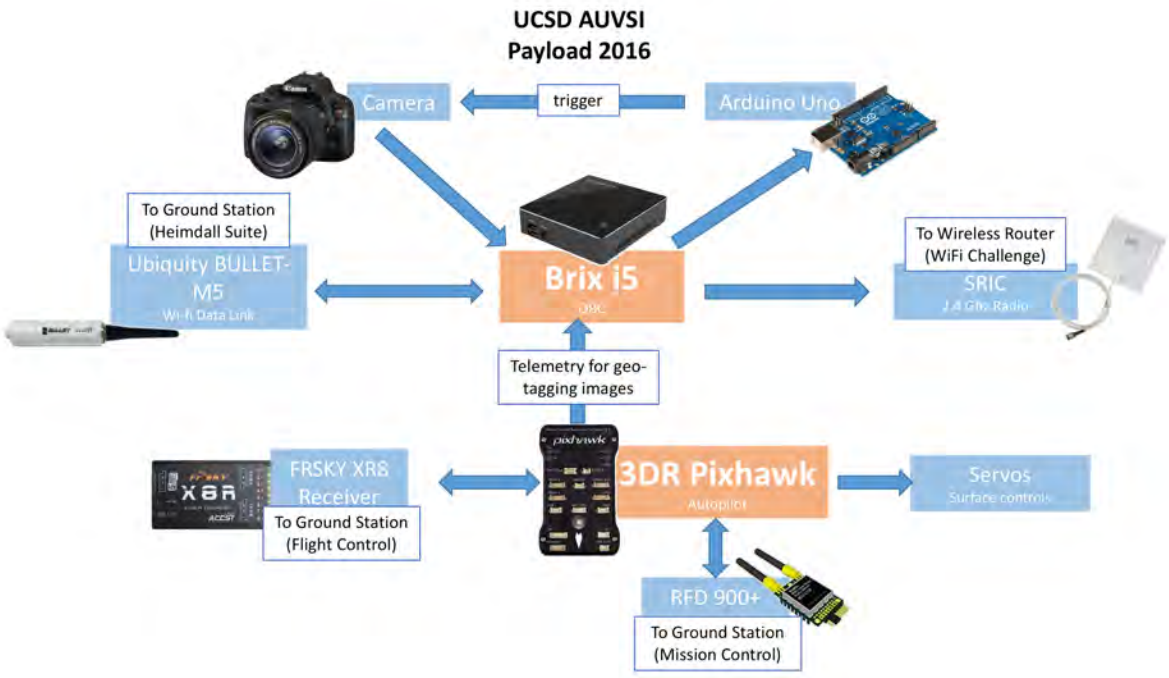


Figure 2: 2016 Payload Systems.

3.1 Communications

There are three communication links between the ground station and the aircraft: manual RC control link, an autopilot telemetry link, and a high bandwidth Wi-Fi data downlink.

The RC control system provides the ability for the Safety Pilot to override the autopilot in case of autopilot malfunction. It is critical that this link never fail so there is always a human in the loop capable of manual override. To provide this robust link, the FrSky X8R, which operates in the 2.4 GHz band, has been selected for its demonstrated performance by long range first person view RC hobbyists. In addition, it employs frequency hopping technology to prevent jamming and minimize the impact of interference.

The autopilot telemetry link provides real-time feedback of the autopilot state, as well as the ability for the Autopilot Operator to send commands and waypoints to the aircraft. The RFD900+, which operates in the 900 MHz band, has been selected for this link due to its demonstrated performance at 60 km without requiring high gain antennas.

The high bandwidth data link enables the transfer of images from the onboard computer to the ground station as well as a means to interface with the payload systems. To serve this requirement, carrier grade wireless equipment from Ubiquiti Networks is used to provide the data link, with a demonstrated range of several miles while maintaining high data bandwidth. The Ubiquiti Bullet M5 used in Sig Rascal utilizes

the 5.8 GHz band, and is capable of sustained, reliable data transfer of over 10 MB per second at the ranges required.

3.2 Visible Spectrum Imaging

The camera is the heart of the imaging platform. It provides a means of Electro-Optical (EO) reconnaissance and target detection. As the primary airborne payload, it is given priority in the design process to maximize the overall system effectiveness. A feasibility study was conducted to select the appropriate imager. This study considered the altitude, ground speed, and camera parameters to ensure that the chosen imager possessed the following properties:

- Optical density of at least 1 pixel per square inch at 400 ft and 45 kts
- An image overlap factor of at least $^1/_3$ when travelling at 150 ft and 35 kts
- Motion blur not exceeding 1 inch at 400 ft and 35 kts
- Angular sensitivity not exceeding 12 inches of motion blur at 400ft and 45 kts
- High speed image download support over USB

Additional considerations included cost, size, and Linux interface capability.

The Canon EOS Rebel SL1 has been chosen as the preferred camera. It features an 18 MP sensor, which provides a large number of pixels on target, resulting in a clearer view of each target. At the desired flight altitude of 400 ft, this provides 40 pixels for the minimum target dimension of 2 ft. In addition, it is capable of shooting at up to 4 frames per second, which enables the airframe to travel at higher speeds while maintaining the same coverage and the ability to capture redundant imagery to provide further verification of a target.

In addition to its imaging capabilities, the SL1 is useful due to its smaller size and reduced weight (the SL1 is 30% smaller and 30% lighter).

3.3 Power Management

The power system has two major components: 12S LiPo flight pack and 4s LiPo payload pack. The 12S LiPo flight pack powers:

- Electronic Speed Controller
- FrSky X8R Receiver (through a 5V BEC)
- Servos (through a 5V BEC)

The 4S LiPo payload pack along with a voltage regulator powers:

- Pixhawk
- RFD900+ (through a 5V BEC)
- Brix i5 (through a 12V voltage regulator)
- Bullet-M5 (through a 12V voltage regulator)

By using a power module on the 4S LiPo Battery, we are able to monitor the voltage and current of the battery. These data is transmitted through the telemetry link and our GCS then takes over and decides if any fail-safe action should be taken.

Critical components that relate to flight controls such as the FrSky X8R and servos are mainly powered by the 12S LiPo battery. In case of power failure from the 12S LiPo, the backup power from the 4S LiPo comes in. The autopilot can also use the 12S LiPo as a backup power in case of a power failure from the 4S LiPo.

This makes sure that the plane is controllable as long as there is at least one pack of battery still working.

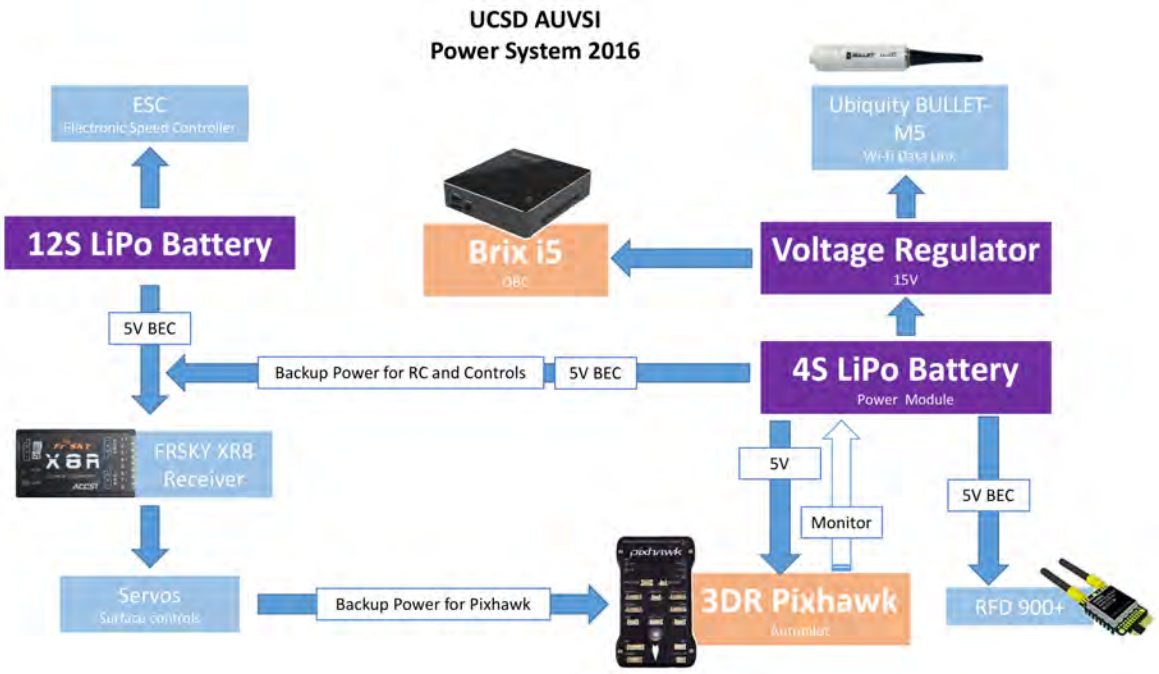


Figure 3: 2016 Power System.

4 Software Systems

Building on our software systems from last year UCSD AUVSI expanded our computer vision system, improved mission planning and path planning for SDA, and updated our overall system for robustness and fault tolerance. Nearly all systems have been developed in Python to minimize development costs in terms of human labor. We also use C++ for the computer vision tasks which require intensive data processing.

To improve the ease of use of the system and simplify software engineering, we use an overarching mission control system to organize mission tasks and distributed software components. This provides a framework for increasing component-wise self-management of the system; decreasing the amount of human intervention needed at the front end; and improving software robustness with distributed fault-tolerance. These system components allow for a flexible ground station design. Components can be run on different computers as needed, and as the software improves, components can be run on fewer computers with fewer human operators. The importance of this grows with the increasing complexity of the SUAS competition, with new tasks being added every few years. Rather than have a task specialist at the ground station for each unique mission task, we have a single Mission Control Operator to coordinate most of these tasks, with only two additional operators for autopilot, telemetry and failsafe systems monitoring; and an operator for emergent target search and saving the output of the automated target information aggregator. Whenever possible, software and payload component anomalies are automatically detected and addressed, or a human operator is notified so that corrective action can be taken.

4.1 Mission Control

Mission Control allows the Mission Control operator to view what is going on on the plane and with other systems and coordinate tasks. All other components in the system send their status to Mission Control through UDP because, for status updates, the operator only cares about the latest update. The status updates allow the operator to view and debug if necessary any errors that are occurring. The other components in the system such as the plane on board computer and the heimdall computer vision system expose HTTP API's that allow the mission control operator and the different components to communicate with each other. The Mission Planner node runs the popular ground-station UAS software MissionPlanner, which we use in

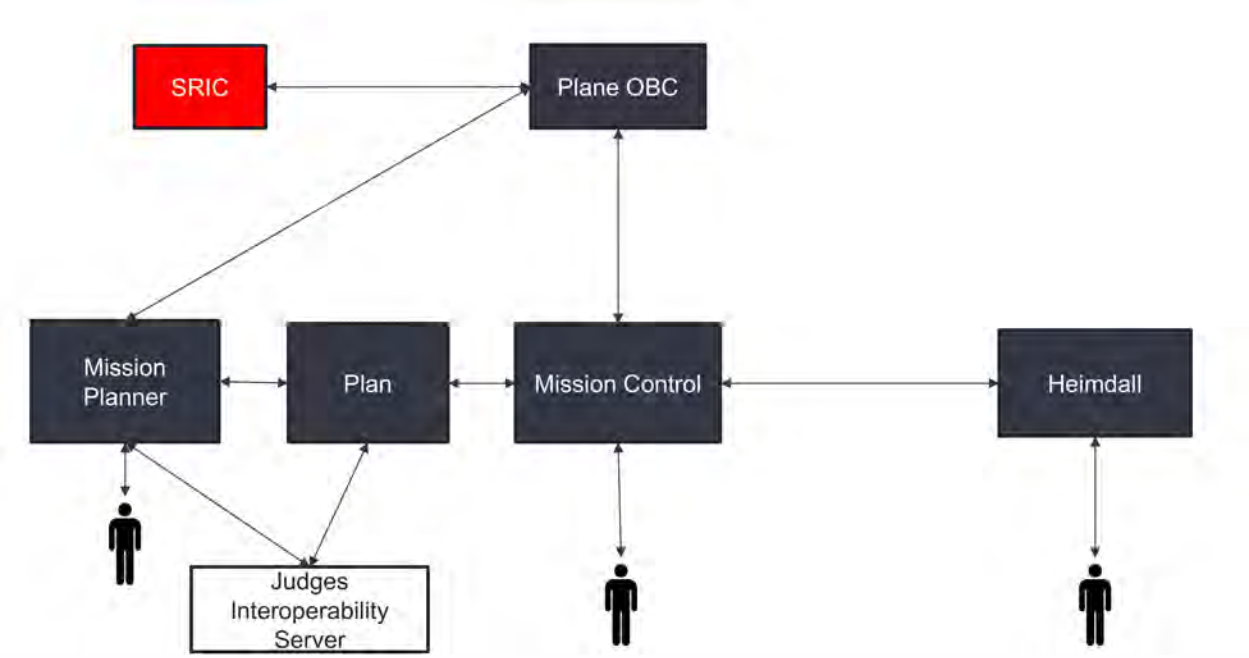


Figure 4: Overall architecture of 2016 software systems.

order to route real-time telemetry data to the interoperability server.

Over the Ubiquity Wi-Fi data link, we use two separate communications protocols with the onboard computer. From the Heimdall computer vision ground server, we retrieve georeferenced objects-of-interest cropped out by the onboard saliency preprocessor. These images are further processed by the rest of the ground-based computer vision suite to automatically identify and characterize targets; a human operator also checks these objects-of-interest for the emergent target.

4.2 Plane Onboard Computer

The plane onboard computer retrieves, geo tags, and crops images from the camera, and handles the communication for SRIC. The images are retrieved from the plane using gphoto2 a library for interfacing with cameras. When connecting to the camera we time sync the camera with the computer. At the same time we save time stamped GPS data from the Pixhawk. This allows us to match the time stamps from exactly when the shot was taken that are embedded in the exif data of the image with GPS data we are receiving from the Pixhawk giving us accurate geo tagging from when the shot was taken.

4.3 Plan

Mission Planning is conducted autonomously based upon a set mission, and waypoints. The planning module outputs a path that avoids obstacles, covers the entire search area, and optimizes the timing for secondary tasks.

4.3.1 Mission Planner

Mission are run through Mission Planner, an open-source software capable of controlling flight mode adjustment, waypoint generation, and control system tuning, both before and during flight. Mission Planner provides the Autopilot Operator with real-time control and mission status of the vehicle, as well as standard positioning reference outputs for use by the judges.

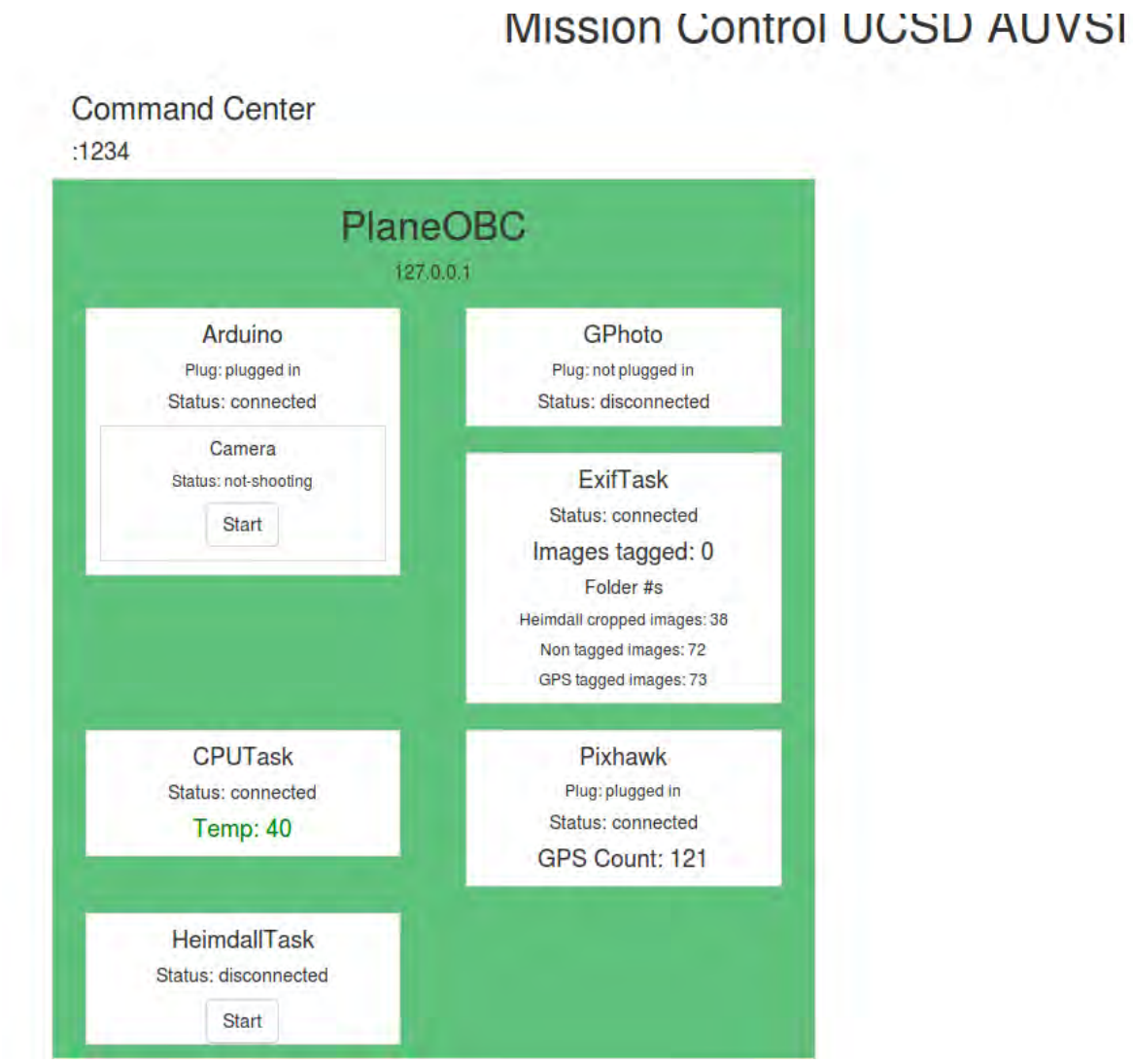


Figure 5: User Interface for Mission Control Operator

4.3.2 Automated Search Path Generation

Instead of manually determining a search path of the flight area, UCSD AUVSI has developed and applied a search path generation algorithm that finds an efficient path through an arbitrary search area within the flight boundaries.

The algorithm analyzes wind speed and direction to create a search path that maximizes flight parallel to the wind. This prevents rolling moments due to crosswinds and provides increased stability during flight, which in turn results in higher quality images. The algorithm then attempts to minimize the turns in order to maximize steady level straight flight.

The flight path generated by the algorithm allows the plane to efficiently cover the entire mission area, minimizing necessary flight time. The search path is capable of covering the entire mission area in under 10 minutes.

We found in a feasibility study that we will need at least 5 image pixels per QR-code-pixel; with our camera resolution, this indicates we need to be flying below 300 feet, whereas our usual cruise altitude is about 400 feet. In order to read these targets, we decided to fly a second pass at a lower altitude after observing target locations at the higher altitude. We accomplish this by linking the computer vision systems target aggregator to the flight planning software. After the plane does an initial higher-altitude target search,

the target aggregator sends the locations of detected objects-of-interest and targets to the flight planner, so that the plane can fly at a lower altitude and re-image them with a very high number of pixels on target. With this setup, we can ensure that we will successfully be able to decode any target. This vision-based feedback process is fully automated, so that the flight planner and mission control operator simply watch as the first-pass targets are located and the second-pass flight path is generated. The software systems that enable this process are described in the Software Systems Overview section.

4.3.3 Sense Detect and Avoid

To avoid static obstacles, we implemented our own version of the RRT* algorithm. RRT stands for rapidly expanding random tree, and the general idea of the algorithm is to add random points to a tree, expanding towards the goal region. We most closely followed the algorithm presented in the paper "Anytime Motion Planning using the RRT*" [1][1]. The benefits of the RRT* algorithm are that it improves on the standard RRT algorithm by converging to a more optimal path by continually rewiring and adjusting the tree while building it.

Our algorithm works by picking a biased point (heavily biased towards the goal point), and then finding that nearest node on the tree. We check if the line between the near node and the random node (scaled a specified distance) is free of obstacles. If it is free of obstacles, we then add the new scaled point to our tree, and find the lowest cost parent and check if the new node is a better parent to any of the nearby nodes. We continue this process until we get a node that is within the scaling distance of the goal node.

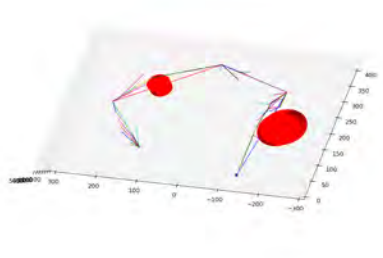


Figure 6: The tree generated by our RRT for one of our test flights

We use the RRT* algorithm once we have generated our initial mission path, and utilize it to both avoid obstacles. Given the mission, our path planning module checks every path between two obstacles. If it detects an obstacle, it runs the RRT* algorithm, and generates a new path between the two points.

Our RRT* takes into account several constraints to ensure that we generate a flyable path. One of the most important constraints is turn angle; before we add a point to our mission, we ensure that the turn is as slight, or as close to a straight line, as possible. In addition, our RRT* takes into account other mission objectives, such as staying in the flight area, or staying within 100 feet of the waypoint path.

4.3.4 Task Waypoint Traversal

In creating our automated waypoint generation system, we made sure that it was extensible for not only our primary tasks, but our secondary tasks as well. Since there are many different secondary tasks that we need to accomplish, each located in different parts of the whole mission area, we looked at the problem as a traveling salesman problem (TSP) that we needed to solve, to find the optimal path among the target waypoints of the secondary tasks. Our approach to this problem was to use a hillclimbing algorithm that incrementally searched for the optimal solution path to these points by an iterative depth first search through all the possible variations of an Eulerian tour through the task waypoints.

[1] http://personalrobotics.ri.cmu.edu/files/courses/papers/Karaman11-anytimerrtstar.pdf

Figure 7: Path between two waypoints that avoids an obstacle. The mission is last year's competition.

4.4 Computer Vision System

Computer vision plays a large role in the many imaging tasks in the competition. We approach these tasks by expanding our in-house computer vision software suite, named Heimdall, developed since 2013. Heimdall is a flexible distributed system that allows the computer vision portion of our ground station to consist of one or more laptop computers. This design fits well into the rest of our flexible module-based mission control software, and means that our entire software suite can be run on any number of computers.

The system was developed from the ground up to be highly adaptable to both module replacements and system expansions in the competition mission; this has been greatly helpful this year as QR code targets have been added for the first time. The C++ backbone lets our software team design a complete vision system as a simple flowchart, whose links are TCP sockets passing data between the image processing modules. The complete sequence encompasses both the onboard computer and the ground station, since the first step, salient object detection, begins on the plane.

The backbone has Python embedded into its C++ framework. This allows rapid prototyping and easily maintained computer vision algorithms; many of the algorithms are based on the popular OpenCV library, and behind every image processing call is a fast compiled C++ algorithm so that the use of an interpreted language does not affect system performance.

4.4.1 Salient Object Detection

Salient object detection is the first step in the computer vision system. It runs on the onboard computer, starting with the 18 megapixel images from the Canon SL1. First, the image is converted to the CIELAB color space, and a saliency map is produced using the fast algorithm presented in the paper Saliency Detection: A Spectral Residual Approach [1][2]. The normalized grayscale map produced is shown in the upper-left corner of the figure below. Next we apply blob detection to this map by thresholding a difference of gaussians pass, which is used for further blob enhancement. The result of this pass is shown in the lower left corner of the figure below; note that the algorithm successfully suppresses noise from the tree. We wrap bounding boxes around these detected blobs and crop them out of the 18 megapixel RGB image.

The centers of the blobs are georeferenced using the telemetry measured at the time of image capture. These georeferenced objects of interest are then ready to be passed to the ground station, where they will be further processed for automatic target detection and the emergent target search. Not passing the full-sized 18 megapixel images significantly reduces the bandwidth required to run a mission. In a test run with 50 images, the algorithm reduced 410 megabytes of image data to 3 megabytes of cropped objects in 24 seconds. This means our system has the ability to run in real-time at a rate of up to 2 Hz on 18 megapixel imagery, with a bandwidth requirement of merely 0.7% of the original data, or roughly 100 kilobytes per second depending on imaging frequency.

[2]http://www.klab.caltech.edu/ xhou/papers/cvpr07.pdf

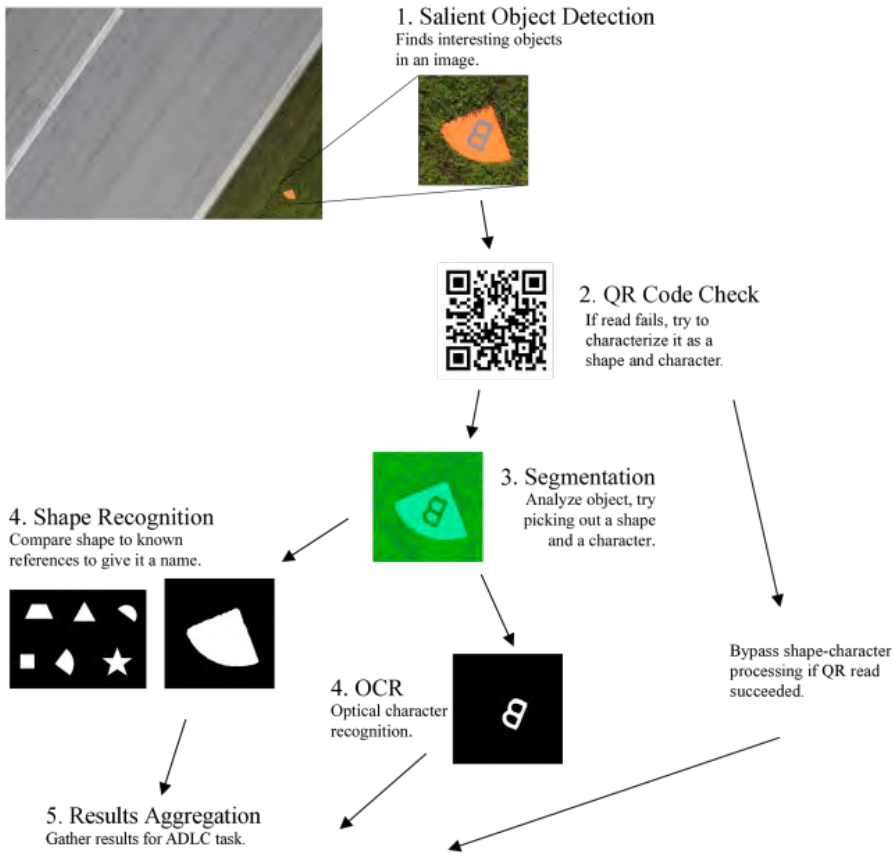


Figure 8: Computer vision system flowchart.

Figure 9: Salient object detection finds emergent target "Eric" and a painted plywood target during an imaging test flight.

4.4.2 QR Code Detection

When cropped objects of interest arrive on the ground, the first step in the automatic target detection is to check for a QR code. This is done using off-the-shelf libraries in Python. Since this has an extremely low false positive rate, we can be sure that if it reads a message, we can confidently pass the read code straight to the target aggregator, bypassing the otherwise-standard pipeline through segmentation, shape recognition,

and character recognition.

The QR code targets can be read from any orientation, but require a high number of pixels on target in order for the library to decode the message. This has been accounted for by the new two-pass flight search, described before in the Automated Search Path Generation section.

4.4.3 Segmentation

The cropped images from Saliency are then passed to the Segmentation algorithm. The algorithm begins by converting the image from the RGB color space to the more useful CIELab color space, in which numerical distances between colors correspond more closely to the human eye. The image is then filtered using an edge-preserving bilateral filter; the result is shown in the second step of the figure below in its raw CIELab color space.

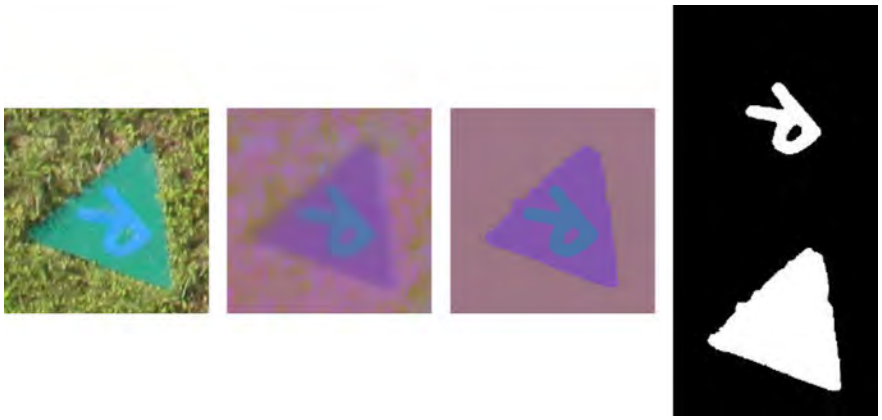


Figure 10: Processing steps in segmentation from left to right.

The next step is color quantization using k-means++ clustering, in which the centroids of clusters converge to the dominant colors in the histogram of the image. This allows us to extract binary masks of the shape and character, shown in the last two images on the right in the figure above. After this step the shape and character masks are ready to be passed off to the OCR and Shape Recognition modules. The colors found by the histogram cluster centroids are automatically converted to their English name equivalents by nearest-neighbor classification.

4.4.4 Shape Recognition and Optical Character Recognition

There are two primary components to the characterization of the shape and letter based targets. In the system flowchart these algorithms are run in parallel, both immediately after segmentation provides the binary masks.

Shape recognition matches the target outline to the closest reference shape based on a comparison metric that is robust to noise, rotation, and scale invariance; the implementation is described in the paper [2][3]. Each comparison uses the area integrated difference in height between two turn functions that trace the angle of the edges of the shape, and minimizes amongst possible orientations by offsetting one against the other. The functions are size-normalized by fixing the integration between 0 and 1, and integral approach makes the algorithm robust to noise in the case of small random fluctuations along the edges.

Optical Character Recognition, or OCR, is the algorithm that simultaneously classifies the alphanumeric character and finds its orientation angle. To find the orientation, we rotate the images in increments and report the orientation and character with the highest match.

We found that the task of simultaneously finding the character and orientation was a difficult task for most OCR libraries. The classifier we use is based on recent advancements in machine learning using deep convolutional neural networks, which was introduced in the seminal 1998 paper by Yann LeCun et. al. for

[3] http://www.dtic.mil/dtic/tr/fulltext/u2/a210105.pdf

optical character recognition[4]. The training process takes as long as 30 hours on a desktop graphics card when training with tens of thousands of characters per iteration. At competition time, we only need it to classify characters one-by-one, so computational expense is much less of a concern, and we have found in testing that it works efficiently on our laptop computers. The library we use is Theano, based on Python.

4.4.5 Target Aggregator

Once image processing is finished on the target images, they are sent to the Target Aggregator which clusters and confirms targets in a database. The module uses the DBSCAN clustering algorithm to group multiple looks on a target based on both GPS coordinates and target attributes. The database is then sorted based on consistency of attributes. This aggregation allows us to minimize potential inaccuracies in the computer vision system by taking advantage of multiple looks of the same target to confirm its attributes. The automated ranking of the database ensures the reported targets found for the computer vision mission tasks have been characterized with high accuracy. After the first-pass search at the higher altitude, at which targets and potential targets are found, the aggregator sends coordinates to revisit from its database to the flight planning software to be flown in the second-pass flight. The target locations are also displayed on Google Earth, allowing the computer vision team to observe targets being automatically located.

5 Testing and Evaluation

5.1 Flight Testing

Mission objectives are simulated using large cloth targets with various alphanumeric characters painted on of similar size, shape, and coloring as those used for competition. The test targets are also typically more difficult to autonomously identify than competition targets, as they use characters that could be mistaken for others a different shades of the same color for the lettering and the background. During flight testing, the imaging system surveys the test area along a generated search path, collects images of the targets for onboard processing, and transmits these to the ground station for further processing. The full system test runs through every aspect of the system at the same level as what would be done during competition, and ensures that every component of the system is fully integrated and operational.

The vehicle has undergone a thorough tuning process to ensure a consistent flight behavior. Out of 6.8 hours of the total flight time, tuning constitutes 3.2 hours. To achieve best results, tuning procedure was applied on simulated aircraft model in FlightGear flight simulator. With the values obtained from simulation, the autopilot demonstrated a reasonable performance on the physical vehicle, which served as a baseline for subsequent fine tuning.

[4]http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

5.1.1 Stall Speed Calculation

Prior to flight testing, the stall speed of the vehicle was calculated using the following equation:

$$V = \sqrt{\frac{2Wg}{\rho S C l_{max}}}, \text{ where}$$

$$V = \text{Stall speed of the vehicle}$$

$$W = \text{Weight of the vehicle} = 15\text{lbs} = 6.804\text{kg}$$

$$g = \text{Acceleration due to gravity} = 9.81\text{m/s}^2$$

$$\rho = \text{Density of air} = 1.225\text{kg/m}^3$$

$$S = \text{Wing area} = 1522in^2 = 0.9819\text{m}^2$$

$$Cl_{max} = \text{Coefficient of lift at stall} = 1.1$$

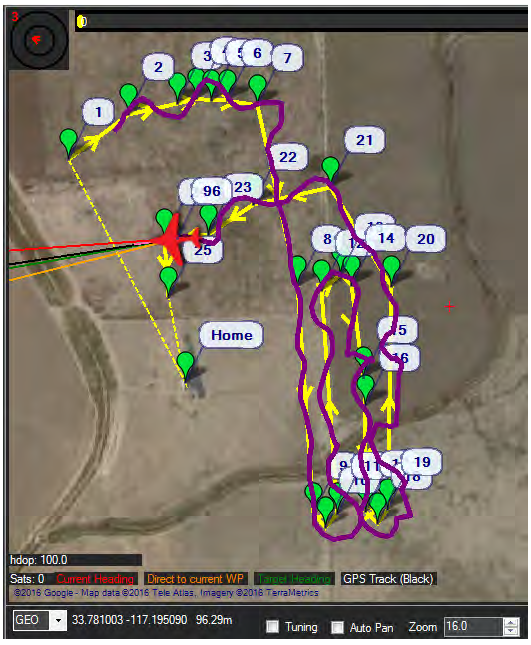
$$V = \sqrt{\frac{2 \times 6.804 \times 9.81}{(1.225 \times 0.9819 \times 1.1)}}$$

$$= 10.0446\text{m/s}$$

Based on the stall speed, the autopilot determines the pitch and throttle output necessary to maintain its speed and altitude as a part of its total energy control system (TECS).

5.1.2 Tuning Procedure

For each flight, we followed the standard procedure for tuning the Pixhawk autopilot to set the vehicle-specific parameters, such as maximum bank angle and climb rate.



(a) Flight path before tuning

(b) Flight path after tuning

5.2 Software and Payload Systems Testing

Improving the robustness and reliability of the software and payload systems was a significant goal of development this year. From the ground up, our distributed software system enables compartmentalization of software bugs, so that most of the system and other ground station computers are unaffected. For basic

quality assurance, every software component was subjected to regression testing, tracked with our revision control system git. In addition, considerations were made to ensure robustness by monitoring system faults and providing fallback functionality in case of contingencies. Tests were designed for each payload component to allow for fallback functionality that in most cases still provide for mission success.

Refer to the Appendix, where these tests are itemized. Each case considers the impact of an incidental payload or ground station component malfunction, and how we have developed our system to address these cases and minimize risk to mission operations.

5.2.1 Evaluation of Salient Object Detection

A rigorous evaluation of the saliency module was conducted using a total of 800 15-Megapixel test images collected from past competitions and test flights. The following table summarizes the result. On a subset of

Table 1: Evaluation Result

Data Set	#Images	#Targets	True Positive	False Positive	False Negative
All	1479	-	188	1392	-
Target Images	121	193	131	85	62

test images that contain targets, the saliency module achieved an accuracy of 47.1%. This relatively low rate is largely due to the high false positive rate. False positives include not only the resulting cropped images with no salient objects but also those with salient objects that are not targets. Current design relies on the result of later image processing algorithms to compensate for the high false positive rate.

5.2.2 ADLC supported

Out ADLC system supports the following targets.

Letter orientation: N, NE, E, SE, S, SW, W, NW

Shape: Circle, Semicircle, Quarter Circle, Triangle, Square, Rectangle, Trapezoid, Pentagon, Hexagon, Heptagon, Octagon, Star, or Cross.

Alphanumeric: 0123456789ABCDEFGHIJKLMNOPQRSTU VWXYZ

Alphanumeric color: White, Black, Gray, Red, Blue, Green, Yellow, Purple, Brown, or Orange.

Background color: White, Black, Gray, Red, Blue, Green, Yellow, Purple, Brown, or Orange.

6 Mission Operations

Flight operations, both at competition and during test flight, are organized such that each member is uniquely tasked with a specific operation to conduct during flights in order to maximize efficiency and safety. This chain of command is in place in order to ensure clear lines of communication between parties and to maintain coordination during flight procedures and monitoring.

The above diagram illustrates our communication channels during fights. The main hub of communication is the Mission Control position, where communications between the tarmac, the ground station, and the judges are centralized and responded to, so that these members can focus on their mission priority and be promptly alerted when necessary. The Mission Control position provides the judges with a direct link to communication with the team that also minimizes distraction so that flight operation do not become unorganized. Flight Line, Mission Control, and the Safety Pilot are equipped with radios to allow for clear communication, rather than risk unclear shouting across the runway.

7 Safety

UCSD AUVSI prioritizes safety as the number one issue when designing, manufacturing, and operating the system, and maintains policies to minimize the dangers inherent to autonomous flight.

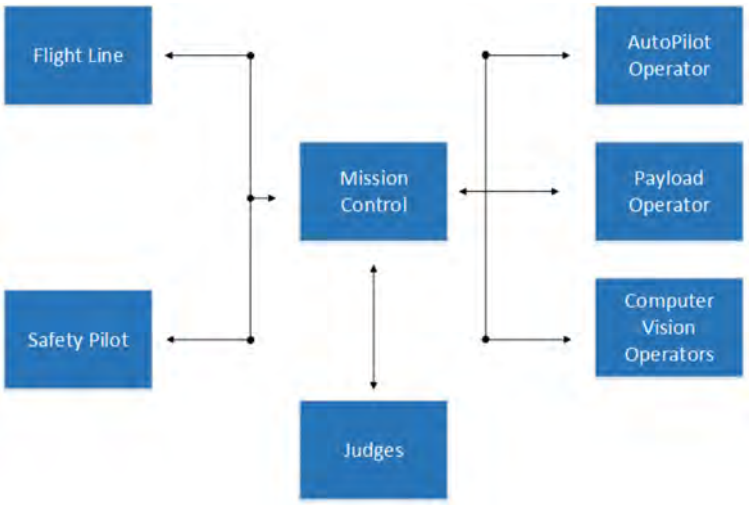


Figure 12: Chain of command during mission

7.1 Mission Operations Plan

The Mission Operations plan dictates that each task of the ground station is assigned to a specific person, to maximize efficiency and establish a clear line of communication. This prevents inaccurate exchanges of information.

7.2 Checklist

We refine the pre-flight checklist and integrate it into the individual flight log. To further eliminate human errors during the process, we employ scripts in the GCS to check certain items in the pre-flight checklist automatically. Items marked with (E) means they can be check automatically by the GCS. This includes flight modes, GPS stats, telemetry link health, airspeed reading, etc. Items that do not meet the requirement will be translated into a warning message on the GCS display.

This checklist can also be performed online using Google Form for easy archiving.

7.3 Airframe Visibility

The airframe is painted in distinct blue and white, with bright yellow highlights to maximize visibility in flight, as well as in the case of a crash. The left wing has a bright yellow stripe to indicate the direction that the plane is facing. The battery packs are colored blue and red to similarly increase visibility in that case that they are separated from the airframe or are left exposed.

7.4 Safety Shunt

Being in close proximity of the propulsion system for the airframe is highly dangerous. To prevent any arming hazards, the plane employs a safety shunt that requires a key to connect the motor to the power system. Without the key, the propulsion system is entirely disconnected with the rest of the electronics. Before the key is inserted, all members of the team aside from the flight line operator moves a safe distance behind the plane, and the insertion of the shunt key is verbally announced to minimize dangers that the propellor can pose.

7.5 Power Management

We identify the RC receiver, servos and the autopilot as components that are critical for a controllable flight. Thus, these components are powered and backup powered by all possible power supplies on board.

As we mentioned in section 3.3, there are two power sources: 12S LiPo flight pack (12S) and 4S LiPo payload pack (4S). The RC receiver and servos are mainly powered by 12S. Pixhawk the autopilot is mainly powered by 4S.

We route two 5V BECs from two battery packs to the servo rail on Pixhawk. This makes sure the receiver and servos can be powered by either one of the power sources. Also, Pixhawk is designed such that the power that goes into the servo rail can be used as a backup power. Thus the autopilot can also be powered by the 12S thanks to this design.

7.6 Telemetry

Telemetry link is one of the most important links on board. We improve the link quality by fine tuning RFD900+ and increasing the number of sources the link can be accessed.

Telemetry data can be transmitted through the 900MHz radio for normal mission planning and monitoring. It can also be transmitted from the OBC, through the Bullet M5 (5GHz) to the ground. It makes use of the second telemetry output on Pixhawk for both geo-tagging images and transmitting data back to software team's GCS.

The RC link can also transmit telemetry data in theory, and that's what we are currently working on. Our goal is to use all possible radio resources (2 * 900MHz, 2.4GHz) in order to make the telemetry data link reliable and accessible.

7.7 Hardware/Autopilot Failsafe

We choose hardware for the payload system with safety in mind. The 3DR Pixhawk equips with a backup side-processor and backup sensors (gyro, etc.). For sensors that do not have a backup version in Pixhawk, such as the compass, we choose external GPS module that has a built-in compass as a external backup.

The Pixhawk autopilot comes preprogrammed with the following failsafe modes and procedures. In the case of extended loss of communication with the transmitter, the Pixhawk will autonomously guide the plane to the launch site.

Failure	Trigger Duration	Action
RC Link	0 sec	No Change
	1.5 sec	Circle if not in Auto Mode
	20 sec	Return to Home
	3 min	Flight Termination
Autopilot Link	20 sec	Return to Home

The autopilot also has the Extended Kalman Filter built in, which enables the autopilot to predict and estimate critical values using currently working sensors' data in case of one or two sensors malfunction.

8 Conclusion

This technical paper has demonstrated the flight-readiness of UCSD AUVSI's Sig Rascal system for the 2016 AUVSI Student UAS Competition. The system was first defined from mission requirements, and its components were chosen from these specifications as well as our past experience with previous configurations. The system was then evaluated during full scale system tests. Over the course of nine months, this team has successfully developed and produced a complete unmanned aerial system which is capable of executing the desired mission objectives.

9 Acknowledgements

UCSD AUVSI would like to acknowledge the generous support of our corporate sponsors:

The team would also like to thank the following individuals and groups for their continued help and guidance:

- Dr. Ryan Kastner, Faculty Advisor, Computer Science & Engineering
- Ben Martins, Ph.D Candidate, Structural Engineering
- Eddie Jimenez, Northrop Grumman
- Lewis Anderson, Microsoft
- Yen and Vinh Hoang
- UC San Diego AIAA Student Chapter
- UC San Diego IEEE Student Chapter
- SAMPE San Diego Chapter

10 Appendix: Software and Payload Systems Testing

These tests were developed to rigorously evaluate our payload and software systems to ensure a high probability of mission success.

Component	Failure Mode	Test Performed	System Response	Impact on Mission
Onboard Computer	Hard reboot, or server program crash.	Forced reboot during simulated missions; server intentionally crashed.	Server program auto-restarts; ground station notified.	Payload operations resume undeterred within seconds.
Payload to Ground Link	Poor signal quality or communications blackouts.	Receiver antennas intentionally obscured or disconnected.	Necessary bandwidth minimized; faster network recovery upon reconnect.	Payload operations continue whenever connection is available; image processing can be delayed to ground-based postprocessing.
Arduino Uno	Crash or reboot.	Forced reboot during simulated missions.	OBC automatically reconnects; ground station notified.	Payload operations resume undeterred within seconds.
DSLR Camera	Crash or connection failure.	Disconnected or reset during simulated missions.	OBC automatically reconnects; ground station notified.	May automatically produce waypoints for re-imaging lost search area.
Onboard telemetry, imaging or other component affecting target accuracy	Poor GPS or compass accuracy; poor telemetry synchronization; poor image quality.	Bad telemetry or image data given to computer vision system.	Targets seen more than once have their georeferenced positions and visual characteristics averaged by the target aggregator.	The system chooses its most accurately known targets for the ADLC task; minimal affect on other vision tasks.
Any payload component	Component improperly connected.	Improper setup before simulated missions.	Status queried by Mission Control before takeoff.	Flight line notified to resolve issue; minor mission delay.
Any ground station software component	Crash.	Intentional crashes or shutdowns during simulated missions.	Component restarted. All distributed software components can resume work upon restart.	A few seconds without the component in question; minimal impact on mission.