StanfordAIR Technical Design Paper

Stanford Aerial Intelligence and Reconnaissance

May 5, 2018

Abstract

This report describes the autonomous UAV designed by Stanford University's Aerial Intelligence and Reconnaissance team in its first year competing in the AUVSI SUAS competition. The team's goal was to create a successful entry to the competition, minimize developmental costs, and create a novel approach to path planning and computer vision. Six undergraduates and a graduate student drew experience from different engineering disciples to create a low-cost aircraft that is capable of performing most of AUVSI's challenges. The aircraft, "Shamu", can autonomously navigate waypoints, avoid obstacles, and automatically localize and categorize alphanumeric characters on the ground. The team developed a novel variant of A* and various optimization techniques to plan and smooth an approximate globally optimal path, and used computer vision algorithms including SURF, K-Means, and convolutional neural networks to categorize and localize objects on the ground.

Contents

1 Systems Engineering

Figure 1: Weighted mission score breakdown

1.1 Mission Requirement Analysis

The StanfordAIR team began it's design process by analyzing the 2018 AUVSI competition rules in detail. The following table summarizes the results of our analysis:

Task	**Analysis Tradeoffs**
Timeline - 10%	It appears that remaining in flight past the 45 minute mark is not a significant point penalty. However, being forced to the back of the line from using a timeout may result in the team losing its opportunity to fly.
Autonomous Flight - 30%	Most of the points here are given to capturing waypoints and flying close to the waypoints, with a smaller fraction of points awarded to being in-flight autonomously for 3 minutes. We lose 2.4% of our total points each time we manually take over autonomous flight. This section accounts for 18% of total points awarded.

Obstacle Avoidance - 20%	Stationary and Moving Obstacles are equally important, and avoiding Stationary Obstacles is significantly easier so it makes more sense to focus our efforts on avoiding stationary obstacles before working on avoiding Moving Obstacles. However, these points are only awarded as long as the airplane reports telemetry information, so the first priority of our path planning/obstacle avoidance team
Object Detection - 20%	The vast majority of points (80%), are awarded for autonomously detecting, localizing, and submitting objects. A smaller fraction of points are awarded for correct characterization of objects. This part of the challenge accounts for 12% of total points awarded.
Air Delivery - 10%	Air delivery accounts for 6% of the total point structure and it will be difficult to time the opening of the water bottle precisely.
Operational Excellence - 10%	The team must perform professionally and interact with other competitors in a courteous manner.

1.2 Design Rationale

1.2.1 Environmental factors

Budget The first constraint StanfordAIR had to keep in mind was the amount of money we had to spend. Our team had a budget of around $3,000, not including several in-kind donations of equipment. We planned to spend around $1,000 on the aircraft, $1,000 on the imaging system, and save $1,000 for any miscellaneous expenditures we would encounter, such as building the launching system, power distribution board, and air drop mechanism.

Equipment and Experience StanfordAIR is based out of Stanford's premier drone interest group, Stanford Unmanned Aerial Vehicle Engineers (SUAVE). The group has access to an electronics lab with an array of equipment (soldering irons, oscilloscopes, function generators, digital multimeters). It also has access to machining equipment (band saws, laser cutters, miter saws, belt-disk sanders).

The team's most significant limiting factor in competing this year is it's small size and lack of experience with AUVSI. Firstly, we are competing against many teams with dozens of people and decades of experience with the AUVSI experience. StanfordAIR consists of only six undergraduate students and one graduate student. We have backgrounds in Electrical Engineering, Computer Science, Mathematics, and Aeronautics and Astronautics. This was our first year competing in AUVSI.

Unfortunately, this meant we spent a significant part of the beginning of the year familiarizing ourselves with the rules of AUVSI and learning about the basic challenges of the competition. We used two strategies to mitigate our team's disadvantages. First, we leveraged our small team size to minimize time spent on organizational matters, and did not have to deal with inactive team members. Both our limited human and economic capital prevented us from doing extensive physical testing and experimentation, but we made the best of this situation by quickly settling on a design and getting to software development as quickly as possible. We never felt limited by money despite only spending a small fraction of what established teams do. We also eventually learned to mitigate our team's rookie status through a thorough analysis of past AUVSI competitions, rulebooks, and other team's submissions.

Mission Requirement Prioritization A quick analysis of the percentages of point breakdowns reveals that 42% of total points are awarded for autonomy (autonomous path planning and computer vision), or 70% of the total number of points awarded for mission performance overall. The other 30% of mission performance points are given for completing the competition on time, air delivery, and operational excellence.

This meant our team needed to produce a computer vision system, a path planning system, an air delivery system, and a networking system to communicate between the other component systems. A significant management decision the team made early on was to prioritize certain parts of the competition above others. We knew that our small team size and relative inexperience with the competition would mean that we could not reasonably expect to build a drone the same way a large team does.

StanfordAIR needed to pick and choose what it wanted to do, and learn to do those tasks well. Most of our team's experience is in mathematics and computer vision, and a large fraction of points are awarded for those tasks, so we decided to prioritize the autonomy stack of the drone over air delivery. Thus, many of the design decisions we describe below were driven by autonomy, not air drops.

Design Decision Flow The first decision we made was selecting our **aircraft**. This was at the suggestion of one of the members of SUAVE. We chose the Skywalker X8, a flying wing design, because it was large enough to support the vision system, payload, computers, and the batteries required to fly for the maximum mission time. A Skywalker X8 airframe was also already in the club inventory, saving us money we were able to spend elsewhere on our UAS.

This decision led to our choice of **flight controller**: the PixRacer. As we mentioned in previous paragraphs, the composition of our team and our skill-sets and interest in mathematics led us to focus on creating our own path-planning algorithm and computer vision system. This made the choice of the PixRacer easy because it worked well with the Skywalker X8 and easily received waypoints from our path-planning algorithm running on a companion computer onboard. Additionally, numerous SUAVE members have experience working with the PX4 autopilot software it runs.

Next, we chose our **communication system**. As we needed a long range system with high bandwidth and high reliability, we decided to dedicate a significant amount of our budget here. We therefore selected an industry grade radio system built by Airborne Innovation and Microhard. It was also crucial that we were able to get the airborne half of the system, the Airborne Innovations Picoradio, at a discounted educational rate. Most importantly, this radio system has a base range of 5 miles and a bandwidth capable of transmitting the required imagery.

Our choice of **camera** was made very carefully. We chose the Z Cam E1. Since our radio is controlled by a HTTP protocol, we chose a camera that could also be easily controlled by a HTTP protocol. We determined that our computer vision algorithm needed a 4k video, with a 42 millimeter (mm) lens, flying at 150 ft, in order to accurately recognize images. Fortunately, the Z Cam E1 met all of these requirements.

To carry our camera, we designed our **gimbal**. We built multiple iterations of the design and ran several trials to ensure that our camera would have the requisite stability to provide reliable test and competition-time performance. See our gimbal section for more information on this.

Finally, we designed a **laucher** system that would allow our aircraft to take off autonomously independent of terrain. This was especially important as we do not have access to an airstrip and our aircraft is too large to hand launch.

2 Aircraft Design

2.1 Aircraft

As a first year team, Stanford AIR's airframe and propulsion system selection process was motivated heavily by cost, payload capacity and internal volume, and flight performance. Internal payload capacity was defined by the need to carry a gimballed camera system, a companion computer module, flight controls, droppable payload, and batteries. In addition to the physical volume of the components, additional room was required to accommodate the components in a configuration with an acceptable center of gravity. Additionally, in order to utilize the 45 minute competition flight window most effectively, the team desired an aircraft propulsion system that would be capable of flying the fully-loaded aircraft for a duration of more than 30 minutes, with a cruise speed of around 20 m/s. In Spring of 2017, an unused Skywalker X8 airframe was donated to the Stanford UAV Club, and was quickly co-opted by Stanford AIR for use as a competition airframe as it fit the criteria outlined above.

Upon receiving the airframe, the team began designing a propulsion system capable of flying the aircraft with the 1-2kg of payload components. *eCalc*, an online propulsion system simulation and design tool, was utilized in order to iterate through a number of propulsion systems until targets were met for projected flight endurance and cruise speed. A 520 kV motor was chosen in conjunction with a 12"x8" prop in order to boost propulsion system efficiency, since propulsive efficiency tends to be positively correlated with propeller diameter and negatively correlated with motor kV. Propeller diameter was limited to 12" by the geometry of the aircraft in order to avoid interference with the trailing edges of

the wings, and propeller pitch was selected such that sufficient thrust would be provided at the target 20 m/s cruise speed and enough static thrust could be generated during launch.

Our mission aircraft is lovingly christened " Shamu"for its resemblance to the legendary orca. This is an aircraft originally designed for long range FPV flight with large payload capacity, which facilitates the conversion to our search and rescue mission. Below is the information as well as a schematic for the internal layout of the critical components.

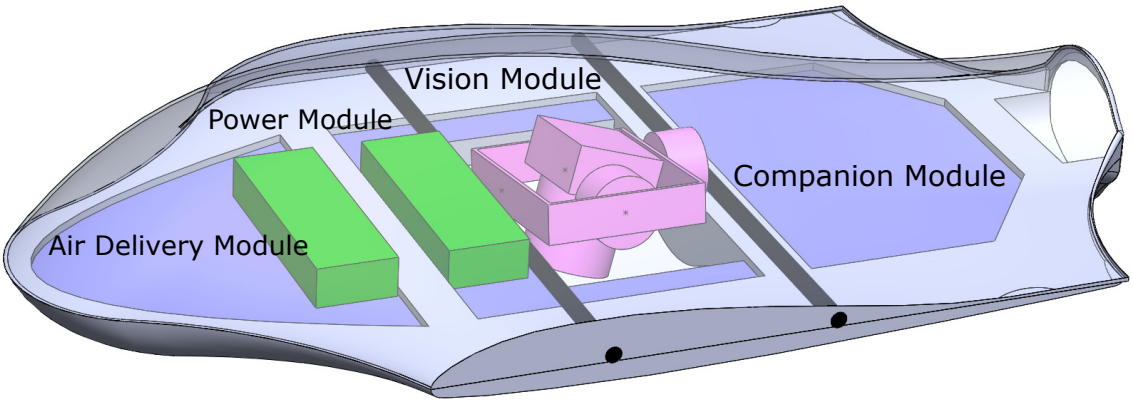


Figure 2: Component layout and specifications

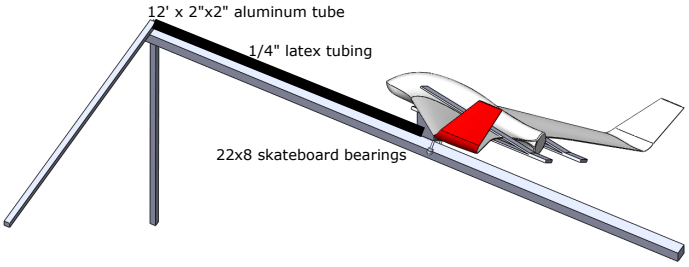
Airframe	32 x 82 x 8 (in)
Control	FrSky Taranis X9D transmitter with X6R receiver in SBUS connection with PixRacer autopilot running PX4 flight stack
Telemetry	915Hz mRobotics telemetry radio in communication with QGround-Control
Radio	2.4GHz
Battery	2x 4S 6000mAh Lithium Polymer (LiPo)
Takeoff weight	7.5 lbs
Launcher	Single rail catapult
Ground Station	QGroundControl

QGroundControl was chosen for the setting and display of real-time mission critical information for two reasons: first, it is the officially supported GCS for PX4 flight stack and also the official implementation for MAVLink; second, it offers the best reported user experience for establishing video transmission, which is also an important mission component.

To enable fully autonomous operation, an aluminum catapult launcher is being constructed that is safe, portable as well as easy to set up and use. It comprises of a carriage that holds securely onto the wing root sections of the aircraft by a bracket, which has bearings that slide along the edge of a 12 in. long square rail with minimal friction. The launch force is provided by the elasticity of U/V resistant latex tubes which are proven to be reliable for launching heavy R/C aircrafts. The tubes would

Figure 3: The UAS and launcher design.

(a) Skywalker X8 "Shamu"



(b) Catapult launcher under construction

be tightly wounded between the end of the launch rail and the carriage. A safety pin would hold the carriage with latex tubes in the stretched position, and can be disengaged at the user's convenience to initiate the launch. Once launched, the aircraft would detect the initial acceleration and initiate a stabilized climbout. After mission objectives are accomplished, or mission time is up, the aircraft would descend from a safe altitude to the specified landing spot.

2.2 Autopilot

Our Skywalker X-8 is controlled by a PixRacer v1.0 autopilot running the latest PX4 firmware, assisted by an Intel Aero Compute Board. A successful calibration of the PixRacer renders our aircraft immediately capable of autonomous waypoint capture, holding patterns, takeoff and landing. Our path planning algorithm running on the Intel Aero Compute Board receive the coordinates and velocity of each obstacle by connecting to the interop server over our radio link, and it plans a path defined by a series of geographic coordinates. The algorithm then send these coordinates as waypoints via MAVLink to the PixRacer. This functionality covers stationary obstacles, moving obstacles, and the coordinates for the air delivery.

In addition to the coordinates, the autopilot has two empty servo channels (because we are using a flying wing design), which allows the the automatic air delivery functionality to be integrated. This works by plugging the left and right servos of the delivery system to PixRacer. When the aircraft reaches a waypoint a certain distance away from the drop zone, the autopilot triggers oscillations in the passthrough channels, releasing the bottle.

2.3 Obstacle Avoidance

Obstacle avoidance is a large part of the AUVSI competition and one of the key focuses of our team. Due to possible loss of communication with the ground station, we opted to compute the flight trajectories on-board. This requires the path-planning and obstacle-avoidance algorithms to be fast enough that embedded computation of such paths would be feasible.

To accomplish this, we made heavy use of heuristics and relaxations of the path-planning optimization problem we are trying to solve. This worked well in practice. In particular, the global path-planning algorithm we used can be broken down into three major steps:

1. Construct a graph approximation of the allowed airspace and remove all vertices of the graph which pass through a static obstacle (essentially points through which the plane is not allowed to fly).
2. Construct a fast, graph-approximated path for the plane's trajectory which avoids all time-dependent moving obstacles based on the graph given in step 1.
3. Relax the fast approximation of step 2 into a smooth trajectory which satisfies all dynamical constraints (e.g., minimum turning radius, maximum velocity, etc.) and pass this path to the autopilot for execution.

The latter two steps were completed in a loop that was recomputed every 100ms, while the former was completed only once—during initialization of the plane. Since the trajectories computed were global paths (i.e., complete trajectories, taking into account all objectives, waypoints, and obstacles known at the time), this relatively slow update time appeared to be enough for the purposes of the mission.

The specific construction of each of these algorithms is defined in the following sections.

2.3.1 Construction of the fast graph approximations

The idea in this section is to construct a graph approximation of the allowed airspace such that distances in the graph are close to distances in the original space. This initial step is useful since the continuous optimization algorithm will usually take a long time to find a feasible starting trajectory given an arbitrary initial path.

This good initial starting path is found by approximating the space in question as a graph (via, say, vertices placed on a grid of points, with all neighboring vertices connected) and then removing all " invalid"vertices which exist inside of known, static obstacles, as is given in step 1. If static obstacles were the only cases of importance, we would run some simple shortest-path algorithm, and then continue immediately to step 3 and run continuous optimization on this path.

In the dynamical obstacles case, a direct shortest-path solver is not enough to construct a feasible path, since the problem of time-varying graphs with removed nodes is actually computationally-hard.[1] On the other hand, since most of the space does not include obstacles, there are several good, exact algorithms which—while computationally expensive in the worst-case—perform quite well in practice: a variant of A* with jump-point search[2] which incorporates the idea of time-varying graphs (and, conversely, the idea of time-varying obstacles) was constructed for this purpose and is fast enough to be computed on the on-board computer used in the drone.

This complete algorithm then yielded a fast, feasible starting path (i.e., a path which does not intersect any obstacle) for further refinement via continuous optimization.

2.3.2 Continuous path relaxation

Since the path given in the graph is a rough approximation, we required further refinement in order to meet the dynamical constraints of the plane since, for example, the path returned by the above relaxation does not necessarily meet the minimal turning radius constraint of the plane.

In order to accomplish this, we take the path returned in the graph approximation and map it back into continuous space in order to use it as an initial starting point for minimizing a loss function (time taken) which includes a set of hard constraints (e.g., avoiding time-varying obstacles, curvature constraints) on the final solution.[3] After applying some simple optimization algorithm (accelerated gradient descent), using the above approximation as a starting trajectory, we exited when all constraints were met and the algorithm failed to generate further improvements of the path.

The now-optimized path is a complete, globally-feasible trajectory which is locally optimal (with respect to time taken). Since the planning can be done arbitrarily far in advance, the frequency at which the trajectory needs to be recomputed is quite low, making this algorithm feasible for the embedded computing environment on the drone.

2.3.3 Obstacle prediction

The task of obstacle prediction is, in the general case, quite difficult since it usually is a problem about equilibria in games. We are given both moving and stationary obstacles to simply avoid—so the problem reduces to the slightly simpler question of path inference.

In the specific case of the AUVSI–SUAS competition, we are also provided with the functional form of the trajectories of the obstacles. Using a linear regression technique for fitting this functional form worked surprisingly well.

In this case it was simply taking the last k-points of the known trajectory of the obstacle and using a least-squares method to fit the weights of the functional forms. In other words, if we know that the

[1] I.e., this problem is NP-complete in the general case of *existence* of a path.

[2] http://users.cecs.anu.edu.au/ dharabor/data/papers/harabor-grastien-aaai11.pdf

[3] For the complete loss function, see https://guille.site/path-optimization-thoughts.html.

obstacle follows a trajectory described by a function of the form

$$f(t;\theta) = \begin{bmatrix} \theta_0^x + \theta_1^x t + \cdots + \theta_n^x t^n \\ \theta_0^y + \theta_1^y t + \cdots + \theta_n^y t^n \end{bmatrix},$$

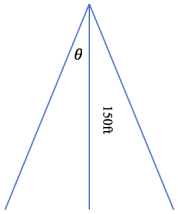
and, assigning a square loss to $f(t;\theta)$ for deviating from our observed trajectory of the obstacle $p(t_i) \in \mathbb{R}^2$ at observation times $\{t_i\}$ yields that the complete optimization objective (for a single obstacle) is

$$\ell(\theta) = \sum_i \|f(t_i;\theta) - p(t_i)\|_2^2$$

which can be minimized extremely quickly and yields good accuracy with few observations.

2.4 Imaging System

The camera was chosen to meet the constraints of our object detection and classification system, which are discussed below. After developing the architecture of our object detection algorithm, we determined that we would need at least 50px/ft to see the smallest target on the ground. We also estimated that we could safely complete the object recognition part of the competition in 20 minutes if we flew at an altitude of 150ft. A few calculations were employed first to estimate how much resolution we would roughly need.



Given an angle of view of 30° (Readily available at specialty lens stores), then each picture taken would encompass $150ft * \tan(\frac{30^\circ}{2}) \approx 80ft$, which would necessitate around $4000px$ to achieve our desired ratio of 50px/ft. This horizontal ratio can be achieved with a standard 4K resolution video stream. We considered the following cameras in our decision making process:

Camera	Features	Drawbacks
Sony QX10	This camera is extremely lightweight and offers a high 16MP resolution. It is commonly used in aerial mapping missions and has an easy to use HTTP interface and Python API.	It does not offer a high rate burst shooting mode, nor does it let the user interchange lenses.
ZCam E1	This popular camera is extremely lightweight as well and offers a well made HTTP Wi-Fi interface. It also allows for interchangeable lenses and 4K streaming.	Documentation on this camera is rather weak and it's not clear if it's been used in aerial mapping missions.
Sony R10C	This camera is designed for aerial mapping and automatically geotags images.	Extremely costly
Canon Rebel EO1	A commonly used DSLR camera. Does not offer a web interface, but many methods have been developed to allow users to take images off of the camera while it is in operation.	The camera is expensive, and very heavy (> 600grams).

Our biggest constraining factor is weight. With this mind, we chose the ZCam E1 with a 14-42mm Olympus Lens. The ZCam E1 has a 17mm x 13mm lens. Using the well known equation: $FOV = 2 * \arctan(\frac{h}{2f})$, where h is the length of the sensor along the desired dimension, we computed the field of view of the camera to be 22°, which is smaller than the back-of-the-envelope calculations we completed above. At a field of view of 22°, we will be able to capture 67px per foot flying at 150ft.

To capture still images that meet our requirements, we constructed our own 2-axis gimbal around the ZCam E1. The gimbal uses 2 brushless gimbal motors controlled by a SimpleBGC 8-bit controller board. After careful calibration and PID tuning we are able to stabilize the camera in both the roll and pitch axes. Furthermore we are looking at commanding roll angle to point the camera at off-trajectory targets.

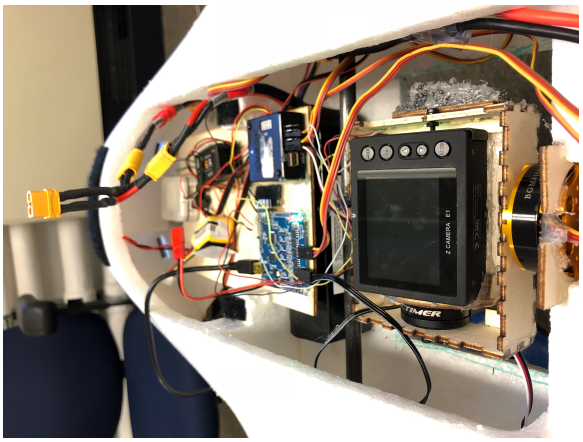


Figure 4: Test setup with ZCam E1 on prototype gimbal.

Vibration damping is still a big issue since the gimbal movements mostly compensates disturbances in angular motion. Translational vibrations induced by the propulsion and aerodynamic forces are likely to introduce oscillations in the camera frame and therefore jitter in the image, thus reducing the resolution and effective number of pixels occupied by the targets. This requires the gimbal to be attached to the airframe with dampers of the right stiffness, as illustrated by the CAD assembly.

2.5 Object Detection, Classification, Localization

2.5.1 Algorithms

Object detection is the first task of the computer vision system. After images are sent to the ground computer, we run an OpenCV implementation of the SURF (Speeded Up Robust Features) algorithm to detect shapes with likelihood of being standard objects. It serves as a blob detector, and after it's complete, we add bounding boxes around these detected blobs and crop them out of the larger image to send down to the ground computer, where they will be further processed for emergent object search and standard object classification. The centers of the blobs are geotagged using the telemetry measured at the time of image capture and the pixel locations of the bounding boxes on the image.

The next challenge is to determine the images GPS coordinates once they have been found on the screen. Given the GPS coordinates of the plane (x_p, y_p), its yaw (γ) and altitude (h), and the pixel coordinates of the object (x_o, y_o), we used the following series of equations to determine the GPS coordinates of the object. These equations assume that the gimbal is always pointing the camera vertically

Figure 5: Original image

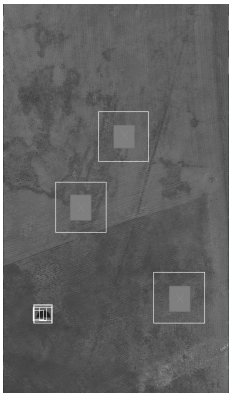


Figure 6: Image with bounding boxes after SURF

at the ground.

$$\begin{bmatrix} x'_o \\ y'_o \end{bmatrix} = \begin{bmatrix} \left(\frac{c_w}{2h \arctan(\frac{FOV_w}{2})} \right) (x_o - \frac{c_w}{2}) \\ \left(\frac{c_h}{2h \arctan(\frac{FOV_h}{2})} \right) (y_o - \frac{c_h}{2}) \end{bmatrix}$$

$$\begin{bmatrix} x_{gps} \\ y_{gps} \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \end{bmatrix} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x'_o \\ y'_o \end{bmatrix}$$

Here, the vector $\begin{bmatrix} x'_o \\ y'_o \end{bmatrix}$ are the normalized coordinates of the objects in meters. c_w and c_h are the width and height of the image in pixels. Finally $\begin{bmatrix} x_{gps} \\ y_{gps} \end{bmatrix}$ represents the GPS coordinates of the objects we localize, which is the value we return to the interoperability server.

Once blobs have been separated from the image, we apply a series of filters before attempting classification. We decided to use k-means clustering to segment the color differences in blobs in order to separate each into geometric shape and alphanumeric character. We use OpenCV to classify the color of both the geometric shape and alphanumeric character based on the range the BGR values fall into. Each of these segmented images are then binarized for easier classification analysis.

The binarized geometric shape is classified using an OpenCV implementation of contour analysis. With this, we use the number of vertices, angles of vertices, and radian length of arcs in a geometric shape to classify it into the appropriate shape. We have also developed a separated neural network for classifying shapes, in case contour analysis fails during the competition. The binarized character is cropped to 28x28 pixels and classified with a pretrained Convolutional Neural Network model. We used the EMNIST dataset to train this network. EMNIST is a dataset consisting of uppercase and lowercase handwritten alphabet characters and Arabic numerals. The Tensorflow implementation of this CNN had three hidden layers using ReLU for their activation functions. The newly-proposed Adam loss function was used to train the network, and the well-known Softmax function was used for the final output layer for estimation of character. We chose to use TensorFlow as the framework for this stage since it is one of the newest, most actively developed projects in the field of image recognition at the moment, and provides a wealth of online documentation to train new users in the platform. Over the rest of our drone development efforts, this training set of images will be built upon to improve our shape detecting neural network and improve the accuracy of the results from this stage. The full architecture is as follows: input layer ⇒ first convolution layer ⇒ first max-pooling layer ⇒ second convolution layer ⇒ second max-pooling layer ⇒ third fully-connected layer: 1024 nodes ⇒ output layer. To generate probabilities for each character, we use Tensorflow's Tf.Estimator class. This estimator uses an implementation of softmax to estimate the probabilities of a character falling into a given category out of 100%. Numbers and capital letters, with '0' and 'O' combined into one class, make up 35 distinct classes.

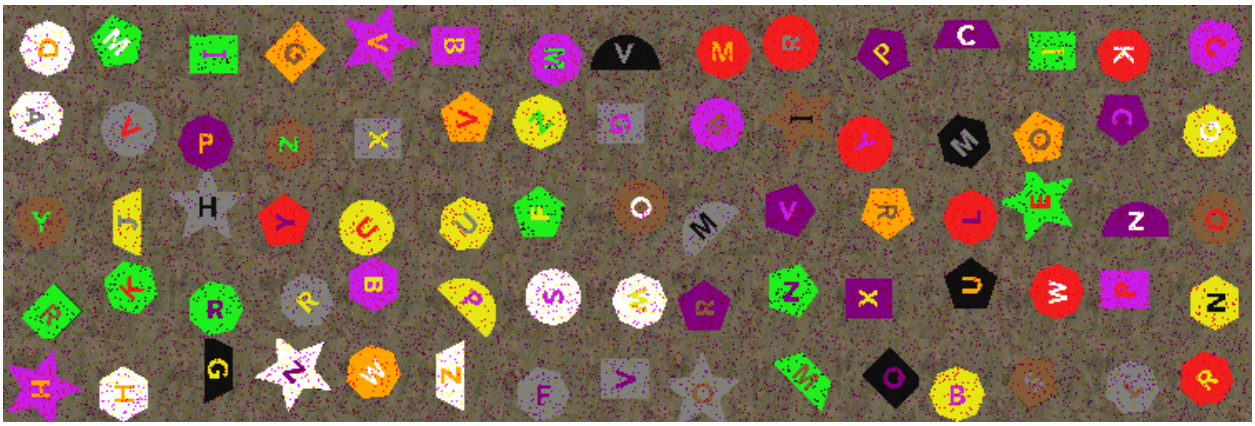


Figure 7: Sample of generated dataset

After 12 epochs our augmented 35 class dataset, using 2000 instances of each class, training accuracy was around 96% and testing accuracy was around 88%. As we test more on real data, we'll need to compare the train, validation, and test accuracies in order to to ensure that we're not over-fitting the dataset to the training data.

2.5.2 Datasets

Since there is no dataset readily available for the standard objects, we created our own datasets to train the neural network on. We've also taken advantage of the EMNIST dataset, a subset of the NIST Special Database 19, comprised of numeric and alphabetic characters.

We separated the challenge of creating objects that would be similar to standard objects we might see in the field into three tasks: background generation, shape generation and placement, and character placement. To generate backgrounds of 50x50 images that might look like the background of a field, we scrubbed Google for aerial images of fields and gathered images from our flights and wrote a script that saved 50x50 croppings of these images into a directory.

After doing this, we augmented our background image dataset using Keras' rotate, skew, and noise-adding data augmenting features. To generate shapes, we created a template of each of the shapes that we've seen from the competition - circle, half circle, quarter circle, plus sign, star, triangle, and diamond. (If we find that there are more shapes than those closer to the competition, designing shape templates isn't too difficult.) We then took 28x28 characters from the EMNIST dataset and placed them on the shapes with random rotations and very small random translations from center. With this new image, we used Keras to do random skewings in order to train for images received where the shape is not exactly parallel to the camera. Here is a sampling of the dataset we constructed:

We placed each of these new images on a 50x50 background square image with randomly translated from the center. This allowed us to place sample images on large background panes to test our SURF algorithm. After using k-means to separate the shape from the character, we collected the resultant binarized shape images and created a dataset for training a neural network to recognize shapes. We used the character output from the k-means algorithm to train a separate neural network (described above) to recognize the characters.

To recognize emergent objects, we gathered aerial images of humans and drew bounding boxes around them to train the YOLO algorithm. We trained our emergent object dataset with Google's Inception v3 dataset and used our collected images to augment that dataset.

2.6 Communications

The UAS has three wireless links with the ground. The first is the manual control link that connects the RC transmitter to the plane. It is provided by an FrSky X6R 2.4 GHz SBUS receiver on the plane and an FrSky X9D 2.4 GHz RC transmitter. This allows for direct manual control and override. The second

link is bidirectional and provided by dual mRobotics SiK Telemetry Radios operating at 915Mhz. This link was responsible for telemetry and parameter-uplink during development and remains a secondary system for these tasks, providing redundant telemetry to the pilot and groundstation. The third link is the primary connection between the UAS computers and the groundstation. This is provided by an Airborne Innovations Picoradio 2.4 GHz OEM advanced datalink module onboard the UAS and a Microhard pDDL2450 2.4 GHz OEM Ethernet & Serial Digital Data Link on the ground. This wireless connection is effectively an ethernet bridge that allows the UAS to share a network with the groundstation and directly query the interoperability network. All images are downlinked from the Intel Aero Compute Board over this link as well, and it serves as the primary telemetry feed.

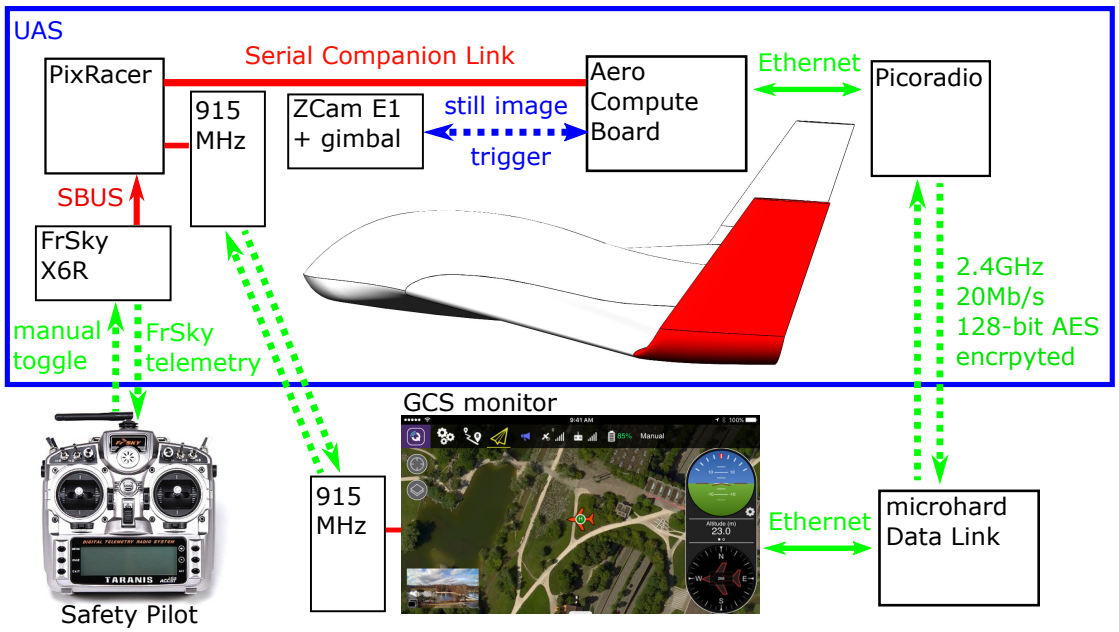


Figure 8: Communication Block Diagram

2.7 Air delivery

As described in the autopilot section, the air delivery module will be activated autonomously by the autopilot. The design is based on the fact that a standard 8oz. water bottle has a diameter of 2.25in., but naturally that could vary from company to company. Because of this, we designed the arms that grip the water bottle to be adjustable.

The adaptability of the design results from the use of opposing servos connected by a Y-splitter cable. Boomerang shaped arms made from balsa wood will sit atop of the servo arms, with their centroids sitting about 2.25in. apart. By adjusting the trim the diameter of the circle can grow or shrink based on the size of the bottle.

The module will be located in the nose of the aircraft in order to ensure that the center of gravity location is not adversely affected. An opening as been cut on the bottom of the fuselage to allow the bottle to drop without obstruction. Through significant testing, we have found that the bottle breaks upon impact on the surface from most altitudes allowed for the competition, so no modification nor attachments to the system are necessary.

2.8 Cybersecurity

UAVs used for missions involving a high level of risk require an impenetrable security system. If an attacker were to somehow hijack a UAV during a dangerous and/or time-sensitive mission, the mission could result in very undesirable consequences.

There are several different ways a cyber attack on our UAS could take place. One of the most vulnerable parts of our system is the communication between the ground station and the aircraft, where there is a potential for the signal to be intercepted by other teams and spied on or even interfered with. To combat this vulnerability, our primary communication link (between the Microhard pDDL and Airborne Innovations Picoradio) operates with industry-standard 128-bit AES encryption, which in June 2003 was approved by the US government for the protection of classified information up to the SECRET level. In order for an attacker to attempt to intercept data to or from the radio, the attacker would first need to decrypt the encryption protocol we implemented, which is an obstacle that adds a layer of impenetrability to our system.

Beyond the use of encryption, we control access to the UAS by requiring user authentication for both the ground station and the onboard computer. Furthermore, we can limit the set of authorized devices that are allowed to connect to our aircraft's network, so that only MAC addresses we know to trust are granted the ability to connect with the network.

While hardware exploits are perhaps the most likely, we also take precautions against personnel-level attacks by limiting administrator access to all of our flight-critical systems. We have internal protocols for password security and only team leads have access to all systems. These steps have been taken to avoid both intentional and accidental leaks of critical security information by team members.

If all of these security measures fail, we can switch to manual flight mode. We have maintained the maximum possible separation between the autonomous and backup manual flight controls so as to limit the ability of an attacker to compromise both systems simultaneously.

3 Safety, Risks, & Mitigations

3.1 Developmental Risks and Mitigation

Our lack of prior experience with the AUVSI competition made it especially important for us to pay attention to the dangers associated with developing a competitive aircraft. The following table summarizes the risks we anticipated during development, our assessments of their likelihood and impact, and our approaches to mitigating them.

Risk	**Likelihood**	**Impact**	**Mitigation**
Electrical	High	Low	When soldering and working with parts that reached high temperatures in routine usage (like the picoradio), we made sure to always have at least two people in the room and burn treatment nearby.
Team Morale	Medium	Medium	Since our team is small, each member had to do a significant amount of work, risking burnout. We made sure to foster a supportive team environment to prevent this issue from arising.
Machining	Low	High	Usage of the laser cutter, welding tool, and other machining equipment is extremely dangerous, especially without safety training. Through partnership with the Stanford Product Realization Lab and safety leaders in the greater SUAVE community, we ensured that we minimized the danger to all members of our team working with these tools.
Non-standard Approach	Medium	Medium	Our use of non-standard approaches to the path-planning and computer vision algorithms/systems could lead to crucial part of our aircraft not functioning properly. We mitigated this risk by starting these parts of our project at the beginning of our design process and testing each module frequently as we built the system up.

3.2 Mission Risks and Mitigations

Operational safety is a key part of the AUVSI competition. StanfordAIR decided to approach drone safety using best practices from SUAVE (StanfordAIR's parent organization). The table below describes the risks we could possibly run into, and descriptions for how we mitigated those issues:

Risk	Likelihood	Impact	Mitigation
Airplane crashes during test	Low	Medium	We ensured that the safety pilot always had a spotter trained in hobby drone flight at test flights to ensure all safety procedures were followed carefully and to double check all flight-critical systems. The safety pilot logged dozens of hours on the plane and completed a variety of disaster simulation training, including stall training and landing without power. Finally, we flew our drone in a designated UAV zone, over a mile away from the populated Stanford campus center.
Malfunction during launch	Low	High	A malfunction during launch has real potential to harm team members observing the flight. We remembered to spend significant time before flights discussing launch protocol and going over safety instructions of launches.
Loss of communications during autonomous mode%	Medium	Medium	The Pixhawk has a built in 'loiter' functionality to circle a certain spot in the air if it loses communication with the remote control. The drone is also programmed to conform to the competition's safety protocols.
Airplane servo malfunction	Low	High	The drone is inspected for mechanical integrity before every flight.
Competition rule violation	Low	High	A dedicated team member read the rules book in details and made notes of the interesting points. This team member is the point person for rules related issues and educated the rest of the team on AUVSI rules and norms.

4 Conclusion

Our main goals upon entering this competition for the first time were to successfully make it to the competition, minimize costs to show that the mission can be accomplished at a reasonable price, and take an original approach to path planning and computer vision. We are very proud that our small team was able to accomplish all of these goals for this year's competition and are excited to see how next year's Stanford team will build upon the foundation we have set. Making it to competition required extensive testing and our team learned a lot through the iterative design process, both in regards to the technical expertise that specific sub-teams developed and the process of integration and project management required to put everything together. Accomplishing all of this on a limited budged was accomplished with thoughtfulness, frugality, a well-reasoned design decision flow, and generosity of industry partners. Finally, our original approaches to computer vision and path planning were accomplished through the tireless effort and dedication of our team members. Overall, we are excited to compete and learn from our experience, and look forward to improving over many future years participating in the competition.