Technical Journal Paper

UCLA Unmanned Aerial Systems

2015

Abstract

This report outlines the design process and steps taken by the University of California, Los Angeles Unmanned Aerial Systems (UAS) team in developing an autonomous aircraft system. The aircraft was designed to meet the requirements set forth by the 2015 AUVSI Seafarer Chapter for the 13th Annual Student UAS Competition. Dubbed the *Ucla Awesome System*, it is an electric-powered plane capable of both radio-controlled and autonomous flight. The aircraft will perform on-board image processing on a Raspberry Pi microcomputer, taking in images via body-fixed DSLR camera. The processed data and sensor information will be transmitted to a ground station, collecting data for objective completion and verifying safe aircraft performance.

8.3.1.2. Description of the Systems Engineering Approach

8.3.1.2.1 -- Mission Requirements Analysis

The Autonomous Flight task and the Search Area task are the primary mission objectives. For the Autonomous Flight task, the unmanned aerial vehicle must follow a given set of waypoints in a specific order. All of the parameter thresholds and objectives are included in the functionality of ArduPilot, an Arduino based autopilot module. The search area task entails flying over a given area and surveying the area for vision targets similar to the one in the below figure.

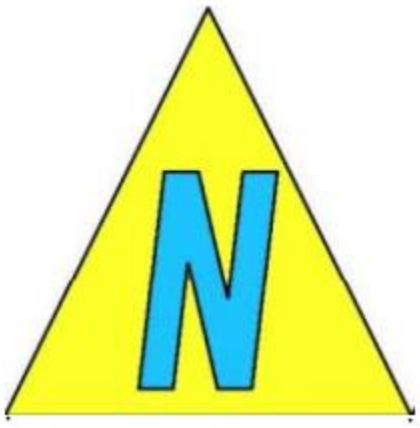

Figure 1: Example vision target. (Source: AUVSI SUAS 2015 Rules, p250)

For the Search Area task the image recognition will be accomplished through the use of an image recognition program written in Python, through the use of the OpenCV computer vision library, on an onboard Raspberry Pi computer. The Raspberry Pi also interfaces with a Canon T3i DSLR using the gphoto2 library in order to take pictures of the search area at regular intervals. UAV flight is still maintained by the ArduPilot module. The secondary tasks being attempted are the Automatic Detection, Localization, and Classification (ADLC) task, Actionable Intelligence task, and Emergent Target task. All three of these tasks are handled in the same manner as the Search Area task, using the Python based image processing software on the onboard computer.

8.3.1.2.2 -- Design Rationale

This year's UCLA UAS team was a reboot from a previously defunct extracurricular organization. To acknowledge and compensate for our team's relative lack of experience in designing and constructing aircraft, we chose to begin with an Almost-Ready-to-Fly (ARF) plane.

A number of important configuration choices were made in selecting a particular ARF model. The first decision made was concerning wing configuration. The options available were low-wing, mid-wing, and high-wing. High-wing configurations offered a number of advantages: greater stability, lighter flying weight, greater internal volume for transporting payload (as there is no support beam running through the fuselage), better ground clearance for airfoils, and a higher wing efficiency. Stability was the primary benefit: when rolling, an aircraft with a higher center of gravity (CG) with respect to wings will roll less extremely than an aircraft with a lower CG with respect to the wings. This is due to the moment from the force of gravity acting at the CG.

The next wing configuration considered was a low-wing configuration. This configuration shared the same advantage of larger carrying volume due to the wing support not running through the middle of the airframe, as well as having an easier landing gear design as the landing gear can be attached to the wing itself. The negatives of a low wing configuration are ground clearance issues, and the reduced stability when compared to a high wing configuration.

The final wing configuration discussed was the mid-wing configuration. The main advantages of a mid-wing design was greater aerobatic freedom, a higher wing efficiency than both the high- and low-wing configurations, and less buffeting. Lesser advantages included the aircraft CG being located at the vertical center of the plane -- this allows greater control when rolling, and minimizes stall at high angles of attack. The disadvantages are a higher power cost, and reduced cargo space, as on a mid-wing design the support will run through the airframe, as well as a lower aspect ratio. We decided to use a mid-wing configuration, due to the cargo space not being important (the only major use of space is the camera, which is partly externally located), as well as the aspect ratio not being a major concern due to the only cargo being the camera (approximately 3 lbf).

Another decision that would have to be made regardless of wing configuration is the wing tip type. The pressure differential is the source of lift for an aerofoil, but around the tips of the aerofoil air from the higher pressure region will attempt to escape to the lower pressure region (top of the aerofoil). This causes drag and reduces max air speed and increases power requirements. Cut off and winglets were considered, however, cut off wing tips represented too much drag while winglets were structurally weak on such a small scale. To this end, we chose to use end plate wing tips, because it is a simple design to implement while is very efficient at preventing drag.

Following the wing tip, we briefly discussed implementing canards. Canards were unanimously agreed to look cool and saves on trim drag, but are harder to analyze, potentially impacted camera vision when rolling, and structurally harder to implement to an ARF model.

The material of the plane was also under consideration. We chose to use a balsa wood-based plane instead of a foam plane because it is more difficult to implement changes to the airframe on a foam body or aerofoil (on which simple screws are superior), which are necessary for the later implementations of servos. While foam is easier to initially mount servos and other hardware, in the case of a servo burnout it is harder to replace the servo on foam, as opposed to a balsa wood design, where we attached nuts on the bottom face of the wood after drilling through such that the servo could be replaced without having to refill the foam to prevent the loosening of screws. Balsa wood is also easier to perform repairs on.

Another priority was payload, which placed implicit requirements on the wing span and aerofoil type. We estimated that the camera would weigh 3 lbf. We also had to account for the weight of the electronics, such as the motor, battery, and the controller: we estimated these to weigh about 14 lbf. This meant that the plane would need a carrying capacity of at least 17 lbf. The plane we chose came in the form of a 3D plane which fulfilled the listed requirements.

On the software and electronics side, our budget restrictions led to us reusing an ArduPilot board. While not the most complex autopilot module, the ArduPilot board has the capacity necessary to carry out the navigation for the flight mission. As this board would not be powerful enough to do both autopilot and image processing, we needed to choose a secondary computer. When deciding on a second computer, our main points of consideration were price, size, processing capability, and compatibility. After some preliminary searching, our decision came down to the Pandaboard ES, and the Raspberry Pi 2 Model B. Both computers run Linux operating systems, which are compatible with the gPhoto2 library we decided to use to interface with a camera. While the Pandaboard was more expensive, it was specifically designed for multimedia processing. However, the Raspberry Pi was cheaper and smaller, meaning that it would be easier to fit into the plane regardless of how we arrange the internal components. As the Raspberry Pi board is capable, even if not the best, of the image processing we need, we chose to run our image processing on a Raspberry Pi.

When choosing a camera for the image tasks, the first topic for consideration was which of the various target tasks we were going to attempt. A standard DSLR would be capable of accomplishing all of the image processing tasks except for the IR target. As buying a camera capable of identifying the IR target would cost more than we were prepared to spend, we chose to ignore this task and focus on the standard vision targets similar to the one shown in the Mission Requirement Analysis section. We decided on a Canon EOS Rebel T3i, as it was compatible with the software we would be using and capable of capturing high enough resolution images while in flight.

8.3.1.2.3 -- Expected Task Performance

From a structural perspective, the flight tests have shown that the drone is capable of controlled take-off, landings, and general flight, as well as specifics like non-major rolling (to allow the camera to cover more area).

8.3.1.2.4 -- Programmatic risks and mitigation methods

Early on in the structural development, we had to make the choice on whether to use a foam inset or nuts to mount the servos to the wings and airframe. The advantages of using a foam inset were that mounting the servos would be extremely simple: simply drilling through the foam. The balsa sections of the airframe would also not be weakened from being drilled though. However, this was a semi-permanent method, and in the case of a servo burnout would be problematic to replace. Furthermore, the weakening of the balsa structure was of minimal concern due to our plane both carrying very minimal weight, and the fact that our plane would be performing only non-stressful maneuvers.

Another potential risk was the motor spinning unintentionally. To stop this from occurring, a kill switch was added to the side of the plane. The motor is now only able to draw power when the kill switch is engaged. The next risk was running out of power. We set the voltage cut off at 3.5 V per cell, in order to avoid losing power mid-flight, as well as avoid discharging too much of the power.

8.3.1.3. Descriptions of the UAS design

8.3.1.3.1 -- Design Descriptions

I. Design of the aircraft

The design of the aircraft is based on the principles of making the plane adapted to the autopilot flight and providing enough space and payload for the implementation of autopilot system and visual recognition equipment. Since we bought an ARF plane this year, the main design focuses on the connection design and the ability to repair the frame.

II. Connection

We choose to connect the aileron and the wing by pin hinges. We use epoxy to fix the both tails of the hinges inside the holes on the aileron and the wing to keep the joints of hinges flexible. Epoxy is firm on the balsa material of the plane.

The two wings of the elevator at the tail of the plane are connected by an aluminum stick and controlled by one servo. At first we considered to control the two wings of the elevator by two servo to achieve better control, but decided against doing so for a number of reasons: our servos are oriented in the same direction because we do not have the transmitters to reverse their orientations. This fact means that the servos cannot control the opposite sides of the elevator to move in the same direction unless we put one servo inside out. This method is not feasible. Moreover, we did not have the resources to purchase new, proper servos. One servo is easier for replacing under emergent situations during the competition. Such a precise control of the elevator is not necessary for us. Since the wings are connected together, it is not necessary to have two servo control.

We want to efficiently use all the servos we have based on the fact that our team was rebooted last year.
The reason for choosing an aluminum stick is that aluminum is lighter compared to other metals like steel and more durable than wood.

III. Fixation

In terms of the fixation of the servos, we decided to use epoxy to set nuts under the surface of the plane and fix the servos with the screws, instead of directly fixing the servos with epoxy. The reason for this decision is that in case of the malfunction of the servos during the competition, it is much easier to replace the servos fixed by screws than that fixed by epoxy directly.

In terms of the fixation of the battery, we used the Velcro with one side of tapes. We use the tape side to fix some pieces of the Velcro on the battery and on a platform within the fuselage. Then we use the Velcro sides to stick the battery to the board. In this way, we can make sure the fixation of the battery is firm enough. It is also easy to adjust the position of the battery and remove it since the Velcro is removable and reusable.

IV. Autopilot

The plane is controlled using an APM ArduPilot board. The board has an integrated IMU, augmented with a wing mounted Pitot tube and a GPS Module. Sensor data and other telemetry are transmitted in real time down to the ground station via a 915MHz radio. The autopilot is connected to the plane through a multiplexer that allows transfer to manual control at any time. This feature is useful for both safety and tuning purposes.

V. Image Processing

The image processing system is based in a Raspberry Pi Model B computer housed in the plane's fuselage. The computer connects to the Canon EOS Rebel T3i DSLR through a USB 2.0 cable and controls it through a program written in C++ using the gphoto2 library. The program takes pictures and regular intervals and saves them to the Raspberry Pi's memory. The time stamps along with flight data from the autopilot allow for geotagging of images.

The actual image processing is handled through a Python program running on the Raspberry Pi. The program uses the OpenCV computer vision library to search for and detect shapes in the images and classify them as targets or not. If a target is determined some characteristics are also identified. This program is further discussed later in this paper.

The image recognition system is capable of autonomously identifying the basic vision target used in the primary search area task, ADLC task, Actionable Intelligence task, and Off Axis Target task.

8.3.1.3.3 -- Photo of UAS Aircraft.

8.3.1.4. Test and evaluation results

8.3.1.4.1 -- Mission Task Performance

I. First test flight

The plane exhibited a slight yaw and roll tendency to the left side. In order to account for this, we changed the aileron differential base settings. We also found that the battery life was much less than expected -- we had originally estimated that with a 5000mAh 6-Cell 22.2V (55 Wh x 2) battery, i.e. we would have approximately 15-20 minutes of flight time. However, upon our initial flight test we found that we only had approximately 3-5 minutes of stable flight time -- far less than anticipated. This was confirmed through an on-ground test -- though this was a high bound test in which the plane was run at full power (the battery life was found to be exactly 3 minutes under these conditions).Due to the fact that the plane will not be flying at full power, we decided that it was viable to use a new battery of the same specs, and charge the battery between flights (the battery charge time being approximately 30 minutes).

Furthermore, we noted that our controller was only 72 MHz -- this meant that even simple radio transmissions risked sending inputs to the drone (this fact was determined to be the cause of last year's accident: the controller accepted too low frequencies that included background radio interference).To fix the interference problem, we upgraded to a 2.4 GHz controller.

II. Second test flight

We performed 2 flights in this test. During the assembly, the test pilot found out that the screw used to fix the rod that connects the servo and the elevator was loose. The reason for this situation is that the straight rod leads to the angle between the servo and the rod easy to cause the screw loose during the movement of the elevator. To solve this problem, we bent the rod a little to change the angle.

Furthermore, at first we attached 5 pieces of the lead weight at the head of the plane to balance the center of the gravitation. During the first flight, the plane experienced strong fluctuation, as it was not nose heavy enough. So after the first flight, we add 4 more pieces of the lead weight at the head of the plane, and the plane performed well during the second test flight.

III. Target Recognition Task

The main bulk of the image recognition program was created in Python 2 using the OpenCV module. OpenCV, the Open Source Computer Vision Library, is an open source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. While starting up, it first finds the contours of prototype images we have created in order to check the ground targets against for shapes. Upon receiving the image of a target, we first call a canny edge detection algorithm on the image. The first step of the algorithm applies a Gaussian filter, used in order to filter out white noise from image compression:

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This gives the impulse response in both the x and y directions, and uses the standard distribution of the Gaussian distribution in order to calculate it. The canny edge detect works off of this by calculating the intensity gradient for the partial in each direction, and also finds the direction of these gradients (measured in θ):

$$G = \sqrt{{G_x}^2 + {G_y}^2} \qquad \theta = tan^{-1}(G_y, G_x)$$

After the canny edge detection, the next step of the algorithm is to draw the contour map of the edges. We will compare these contours to those of the prototype images which were calculated earlier. To find the contours of an image, the function creates an array of the moments in the image, by this equation (because the canny edge produces a raster image):

$$m_{ji} = \sum_{x,y} (array[x, y] \cdot x^j \cdot y^i)$$

These spatial moments are saved into a NumPy array, so that we can compare these contours to those of the prototypes. The benefit to this method is that contours are direction independent, so regardless of the orientation of the targets, the contours can be similarly compared. The third step of the algorithm is where the processor runs the matchShapes function against each prototype. The function returns the amount of difference between the contours, so we return the shape which the function returns the least amount of difference:

$$I_1(A, B) = \sum_{i=1}^{n} \left| \frac{1}{{m_i}^A} - \frac{1}{{m_i}^B} \right|$$

This implementations uses the method of solving for Hu invariants. Then, after finding which shape matches the captured image's contours best, we want to find the color. In order to do that, we again use color prototypes. We use ranges for different colors (and biases away from the floor / ground colors), and create an image using the inRange function in the OpenCV. We return from that function the color range containing the highest number of pixels in the image, and use that for the color. These processes are done similarly for the letters.

8.3.1.4.2 -- Payload System Performance

The main payload is separated into following sections:

1. The structure payload, including the weight of the servos, and wires connecting said servos
2. The camera payload
3. The computer payload
4. Miscellaneous payload, including dedicated weights which we use to keep the center of mass in axis, the structures we use to fix the camera and the computer inside the plane

Component	**Total weight**
Structure	14 lbf
Camera	1 lbf
Computer	0.25 lbf
Miscellaneous	3 lbf
Net weight	18.25 lbf

8.3.1.4.3 -- Autopilot system performance

Our autopilot system of choice is ArduPlane 3.2.3, run by an ArduPilot Mega 2.5, controlled by Mission Planner. Our initial test was to run the system in a hardware in the loop simulation. FlightGear was chosen for our simulation, as the group had familiarity with this software. A hardware in the loop version of ArduPlane was loaded onto the board, a protocol was established for data transfer between FlightGear and Mission Planner. FlightGear was then initiated with a WW2 Zero model. From there, the plane was initially controlled by a game pad in manual mode. Once it was clear that the hardware in loop was set up properly, the plane's control loop routine parameters were tuned by switching into fly by wire mode. After the plane could fly steadily, it was switched into full autonomous mode and given waypoints. It was able to successfully follow a given set of waypoints.

8.3.1.4.4 -- Evaluation results supporting evidence of likely mission accomplishment.

User controlled test flights were used to confirm the ability of the airframe itself to fly in a controlled manner. The ArduPilot was initially tested with a hardware in the loop simulation using FlightGear, a flight simulator. This program was used to get a baseline tuning of the autopilot. Upon the initial unmanned test flight, the ArduPilot was given control while the official rc pilot held the controller in case it was necessary to override the ArduPilot's commands. This test flight was used to fine tune the settings of the ArduPilot. Testing of the image processing program was accomplished by capturing images using the DSLR camera and feeding them into

8.3.1.5. Safety considerations/approach

8.3.1.5.1 -- Safety criteria for both operations and design

I. Launching / preparing for take-off

Priority of ground and air space should be given to launching planes. When starting or running the engine, be mindful of plot blast and residue from rotor blades, and alert people in the immediate area before starting an engine. Members and spectators should be conscious of the prop arc and area in front of the rotor. Engine running time in pits / start up area / runway should be kept to a minimum. Idling the engine for an extended period of time (greater than one minute) should be done in an isolated location. Furthermore, only engage the plug switch when preparing to launch the plane. Only plug in kill switch when preparing to launch plane. When working on / moving plane, always keep kill switch disengaged.

II. Flight period

Avoid occupying the area directly under the drone flight path. Avoid allowing the drone to fly at low altitudes near spectators. Avoid low flying near hazards such as tall buildings and power lines.

III. Landing

Verify the runway is clear for landing. After the plane has touched down and come to a stop, wait until the propeller has stopped moving before approaching the plane. In receiving the plane, first disconnect the kill switch before operating on the plane.

8.3.1.5.2 -- Safety risks and mitigation methods

1. In the 2013-2014 year, the controller was at a low enough frequency to be affected by outside radio interference, representing a safety hazard. To mitigate this we switched from 72 MHz to 2.4 GHz.
2. A kill switch would have delayed if not prevented the crash accident last year, so we added a kill switch that could be accessed from the exterior of the airframe.
3. The prop also represents a risk when preparing for launch once the kill switch has been engaged. To mitigate risk of injury from the prop, team members should avoid being in front of plane when kill switch is engaged.
4. The landing should take place where no people are occupying the runway. Also when the battery charge is lower than 21 Volts, the emergent landing mode will be turned on.
5. The loose screws may lead to serious disintegration during the flight. So all of the screws, especially those on the servos, should be double-checked before the flight in case of loose screws.
6. Large fluctuations during the flight may result in disintegration and impair the visual recognition function. So before the test, it is necessary to perform 1-2 test flights to adjust the center of the gravitation with lead weights to make sure the smooth flight.