Brigham Young University
AUVSI Capstone Team

Capstone Fall Camp Documents

Contents

Objective Statement

ID	Rev.	Date	Description	Author	Checked By
PC-001	0.1	09-05-18	Fall camp draft	Kameron Eves	Tyler Miller
PC-001	0.2	09-10-18	Make more applicable to competition team	Ryan Anderson	Tyler Critchfield
PC-001	0.3	09-13-18	Revisions after design review	Brandon McBride	Kameron Eves

Improve upon last year's BYU AUVSI unmanned aerial system (UAS) by improving path planning, obstacle avoidance, visual object detection, and payload delivery by April 1, 2019 with a budget of $3,500 and 2,500 man hours.

Requirements Matrix

ID	Rev.	Date	Description	Author	Checked By
RM-001	0.1	09-07-18	Fall camp draft	Brady Moon	Jacob Willis
RM-001	0.2	09-14-18	Revisions after design review	Derek Knowles	Kameron Eves

			Units	Feet	Count	Count	Percentage	Feet	Yes/No
	Product: UAV Subsystem: N/A		Performance Measures	Waypoint Accuracy	Minimize Takeovers	Minimize Obstacles Hit	Percent of Objects Classified	Accuracy of Drop	Complies with AMA Safety Code
	Market Requirements		**Importance**	9	9	6	6	3	9
1	The UAS shall be capable of autonomous flight.		9	■	■	■		■	
2	The UAS shall be capable of avoiding static obstacles.		6			■			
3	The UAS shall be capable of visual object capture.		6				■		
4	The UAS shall be capable of delivering a payload.		3	■				■	
5	The UAS shall follow safety requirements.		9						■
			Lower Acceptable	NA	NA	NA	50	0	Y
			Ideal	0	0	0	100	0	Y
			Upper Acceptable	25	5	15	NA	100	NA

Team Charter

ID	Rev.	Date	Description	Author	Checked By
PC-001	0.1	09-07-18	Fall camp draft	Ryan Anderson	Brandon McBride and Kameron Eves
PC-001	0.2	09-14-18	Revisions after design review	Jacob Willis	Kameron Eves and Tyler Critchfield

Team Member Names	Contact Information	Preferred Contact Method in Order of Preference
Andrew Torgesen	andrew.torgesen@gmail.com 661-210-5214	Slack, Email, Text
Brandon McBride	brandon.mcbride4@gmail.com 801-520-9165	Slack, Phone, Email
Derek Knowles	knowles.derek@gmail.com 405-471-4285	Slack
John Akagi	akagi94@gmail.com 858-231-4416	Slack, Email, Text
Brady Moon	bradygmoon@gmail.com 435-828-5858	Slack, Text, Call
Tyler Miller	tylerm15@gmail.com 385-399-3472	Slack, Text
Ryan Anderson	rymanderson@gmail.com 208-789-4318	Slack
Jake Johnson	jacobcjohnson13@gmail.com 801-664-7586	Slack, Text, In Person
Tyler Critchfield	trcritchfield@gmail.com 206-939-8274	Text, Email, Slack
Jacob Willis	jbwillis272@gmail.com 208-206-1780	Slack, Email, Text
Connor Olsen	connorolsen72@gmail.com 385-230-3932	Slack, Text, Email
Kameron Eves	ccackam@gmail.com 702-686-2105	Text, Email, Slack

Team Member Names	Strengths related to teamwork and the team's assigned task	Weaknesses related to teamwork and the team's assigned task
Andrew Torgesen	Organization, ROS, General Programming, Mathematics, Controls	Sometimes impatient with slow progress, Airframe design
Brandon McBride	C++, Python, MATLAB, Some ROS	Mornings
Derek Knowles	Python, Basic OpenCV, Rapid Prototyping	Late nights
John Akagi	Python, MATLAB, Asking Questions	ROS, Computer Vision, Airframe Design
Brady Moon	MATLAB, Python, Willing to ask the contentious questions	ROS
Tyler Miller	Firmware, C++, C, ROSFlight	Airframe, Computer Vision
Ryan Anderson	C++, MATLAB, Git, Bash, Structural design, Writing	Computer Vision, Python
Jake Johnson	OpenCV, Python, C++, A little bit of ROS, Linux	Airframe design, Not enough ROS
Tyler Critchfield	Advanced Dynamics, MATLAB, Command Line, Git, C++, Editing	ROS, Haven't taken controls
Jacob Willis	Embedded Linux (C), Hardware, Project Management, Git	ROS, Airframe Design
Connor Olsen	C++, OOP, Embedded Linux, Matlab, PCB Design, Taking notes	ROS
Kameron Eves	Previous Flying Experience, Writing, Matlab, Business Theory	ROS, Computer Vision

Collaboration

- We will conduct weekly group meetings on Monday where each subgroup will present on progress and milestones
- With larger artifacts (more than 1 page), two people will check the author's work: one for clarity and one for technical accuracy. Otherwise, one person will check it.

- On Slack, each subsystem team will use different channels.
- When working on documentation that is to be submitted to an outside party, we will follow our formalized protocol for collaborative LaTeX document generation (see LaTeX documentation for protocol).
- We will use Git as version control.

Roles and Responsibilities

- Andrew Torgesen: Team Lead, Editor in Chief of All Documentation
- Connor Olsen: Agenda and Meeting Minutes
- Tyler Miller: Space Scheduling
- Ryan Anderson: Task Manager
- Kameron Eves: Subsystem Integration Coordinator
- Team Lead: Ensure Everyone's Opinions are Heard
- Everyone: Devil's Advocate
- Jacob Wills and Tyler Miller: Ensure Uniformity and Clarity in Code Style
- Jacob Willis: Safety Officer

Schedule and Due Dates

Schedule:

- Meet weekly all together on **Mondays at 8:00am**
- Artifacts for the weekly reports are to be turned in by **Tuesday**

Due Dates:

- Obtain Opportunity Development stage approval by **October 15, 2018**
- Obtain Concept Development stage approval by **November 15, 2018**
- Obtain Subsystem Engineering stage approval by **February 15, 2019**
- Obtain System Refinement stage approval by **March 10, 2019**

Requirements for Attendance and Tardiness

- Each member of the team will attend and be one time to on time to all applicable meetings.

- If any member will be late (by more than 5 minutes), he must provide advance notice on Slack.

Acceptable Reasons for Missed Deadlines and Meetings

- Family/Personal Emergency
- Miscellaneous excuses as approved by the team via Slack

*Note that unless otherwise approved by the team, studying and preparing for midterms or finals is NOT an acceptable excuse.

Protocol for Deadlines and Meetings

- Assignments will be made during weekly meetings and included in meeting minutes.
- The next week's meeting agenda will include following up on the previous assignments.

As soon as team member becomes aware of the fact that they will miss a meeting or deadline the must:

- Notify the team on Slack
- If applicable, ask on Slack for help to complete the deliverable

Note that:

- The subsystem team has first responsibility to ensure missed tasks are completed.
- The team member who misses a meeting will be responsible for inquiring about what they missed.

Quality of Code and Documentation

- An agenda will be prepared and minutes kept for every team meeting.
- Monday team meetings will not require a formal presentation or special preparation.
- Code will be written according to the style guide.
- Deliverables will follow the LaTeX formatting guide.

Moral and Interpersonal Communication

- Team members will make a conscious effort to make all feel heard and comfortable sharing ideas.
- The team lead will ensure all feel heard and comfortable sharing ideas.
- Team members will make a conscious effort to persevering through challenges.

- Team members will make a conscious effort to be comfortable asking for help.
- Team members will ask for help early and often.
- Team members will make a conscious effort to expect and learn from mistakes.

Methods of Organization

- Tasks will be tracked on Trello.
- Assignments will be tracked in meeting minutes.
- Subteams will hold frequent "stand-up" meetings.

Concept Selection

ID	Rev.	Date	Description	Author	Checked By
DJ-001	0.1	09-13-18	Fall camp draft	Kameron Eves	John Akagi
DJ-001	0.2	09-13-18	Revisions after design review	John Akagi	Kameron Eves

Purpose

The purpose of this artifact is to present the reasoning behind the decisions we made to form our first prototype. In this artifact three decision matrices are presented to justify our decisions in choosing a path planner, airframe, and visual object classification method.

Path Planner

Decision Justification

Almost all aspects of the competition rely on a path planning system. Even the vision based sections can not function without a path planner leading the aircraft to the search area. As such, careful consideration was given to choosing the optimal path planning system. Last year's AUVSI team used Rapidly Exploring Random Tree (RRT) to moderate success. The largest detriment to the RRT system already in place is its inability to deal with dynamic obstacles. This limitation is primarily due to computation speed. After following the logical decision process described in the decision matrix, we decided to move forward with Fast Model Predictive Control (MPC). What follows is a short high level description of each algorithm.

Alternatives Considered

- **RRT (Reference)** uses the monte carlo method to randomly place way points around the navigation area. A straight line path is then placed between each random point and the closest point of a previously generated path. If the path passes through an obstacle then that path is considered infeasible and dropped from consideration.
- **A*** was one of the first path planning systems developed for robotics and has proven itself robust. In A* a grid system is used to map the area to be traversed. Each point on the grid (known as a node) is given a cost to obtain that node. The algorithm explores each point adjacent to a previously explored point until the path with the lowest cost is found.

- **D*** is very similar to A* but is designed for circumstances when the location of obstacles is not known. It is assumed that no obstacles exists until the robot encounters an obstacle.
- **RRT*** is a form of RRT that is biased towards a goal or waypoint. This would decrease the computational effort required as fewer unhelpful paths would be explored.
- **Fast MPC** is a feedforward form of high level control. This system uses a dynamic model of the aircraft to predict the future state of the aircraft and so allow it to make better informed control input decision at the current time step. A cost function is then used to bias the path towards a goal and away from obstacles. The predictive nature of the system would help with dynamic obstacles. The biggest downside of MPC in general is the computation speed due to the dynamic calculations. However, this is mitigated or even eliminated using the Fast MPC methods presented in a paper by Yang Wang and Stephen Boyd. (Wang and Boyd, 2008) Using these methods, MPC has been performed at speeds of up to 500 Hz, which is more than fast enough to accomplish our goals.
- **Artificial Potential Fields:** uses a dummy force modeled as a spring to push the aircraft away from obstacles and towards waypoints.
- **Probabilistic Road Map:** uses a monte carlo method similar to RRT, but rather then connecting each point to the nearest path, each point is connected to k other nearest points.

Evaluation Methods and Results

As can be seen from the decision matrix in Table 1, Fast MPC performed the most favorably in the requirements that held the highest weight. Fast MPC would perform particularly well in Real-Time path mapping (i.e. it would best help us avoid dynamic obstacles). Fast MPC also does well under kinematic feasibility and shortest path length. Admittedly, Fast MPC is not known to be a light load on the link bandwidth or to be easy to implement. However, as shown in the matrix, these minor difficulties were out weighed by the other important performance metrics.

Table 1: A decision matrix for a path planning system. A scale of 1-5 was used for weights with 5 having high importance and 1 having low importance. A 1-5 scale was also used to rate each option's performance under each requirement. In this case, a 1 was used to indicate poor performance while a 5 indicates favorable performance. As can be seen, Fast MPC was shown to be the most favorable.

Path Planning Algorithm	Weight	A*	D*	RRT*	Fast MPC	Artificial Potential Fields	Probabilistic Roadmap	RRT (Reference)
Ease of Implementation	4	4	3	2	2	4	4	3
Computation Time	5	4	4	2	4	4	4	3
Link Bandwidth	2	3	3	3	2	3	3	3
Real-Time Path Flexibility	4	1	2	4	5	4	4	3
Robustness	5	4	2	3	3	2	2	3
Kinematic Feasibility	3	2	2	3	5	4	3	3
Shortest Path Length	2	4	2	3	5	4	4	3
Totals	.	80	66	70	92	88	85	75

Airframe

Decision Justification

Several options were considered in choosing an airframe. Last year the team designed and built their own airframe from scratch. This year we also considered purchasing an "off the shelf" airframe or purchasing and modifying a commercially available aircraft. In the end we choose to modify an purchased aircraft because we can obtain the flexibility of building an aircraft with the efficiency of purchasing and aircraft.

Alternatives Considered

- **Purchasing** an aircraft would involve using only an off the shelf airframe exactly as is. While being fast and easy, this method would lack the flexibility that would be essential to our project.
- **Purchase and Modify** an aircraft would involve using an off the shelf airframe with modifications to make it more flexible. Such modifications might include but are not limited to wing extensions, improved body payload, and replacing motors and servos.
- **Design and Building** an aircraft would involve a completely custom design using the principles of aerodynamics. This method would allow us the most flexibility, but would also take the most time.

Evaluation Methods and Results

As can be seen, modifying a commercially available aircraft was the chosen option primarily because of its development time and reusability. Previous years have encountered issues with crashing at or near the competition. This worst case scenario left the team without a usable aircraft. For this reason we rated replaceability highly. We also rated development time highly in this decision matrix because we intend to focus the majority of our effort on the controls and vision aspect of the competition. A purchased and modified aircraft will allow us maximum flexibility and reliability with minimum effort, thus aiding our efforts in other areas.

Table 2: A decision matrix for a path planning system. A scale of 1-5 was used for weights with 5 having high importance and 1 having low importance. A 1-5 scale was also used to rate each options performance under each requirement. In this case, a 1 was used to indicate poor performance while a 5 indicates favorable performance. As can be seen, purchasing and modifying a commercially available aircraft was shown to be the most favorable.

Airframe	Weight	Purchase	Purchase and Modify	Design and Build Reference)
Cost	2	2	1	3
Development Time	5	5	4	3
Flight Speed	4	2	4	3
Replaceability	5	5	4	3
Durability	5	4	4	3
Flexibility	5	1	2	3
Totals	.	87	88	63

Visual Object Classification

Decision Justification

In choosing a method for visual object classification we considered for high level options. The method chosen was autonomous object classification without a gimbal. Previous teams that have won the competition have used a gimbal. We felt that this precedent made it important to consider the option. The decision to exclude the gimbal was made primarily because of simplicity. Gimbals can be finicky and would require more work than their benefit justifies. An autonomous system was chosen because we feel that using an autonomous system could differentiate us from other teams in the competition.

Alternatives Considered

- **Autonomous w/ gimbal** is end-to-end autonomy with gimbal stabilization. It includes automatic recognition and submission to judges. It also allows targeting of objects outside bounds.
- **Autonomous w/o gimbal** is end-to-end autonomy with no gimbal stabilization. It includes automatic recognition and submission to judges. It does not include targeting of objects outside bounds. As such it will require the UAS banking to view the off axis object.
- **Manual w/ gimbal** uses manual recognition and submission to judges. Gimbal

stabilization also enables easy targeting of objects outside bounds. With this option we also considered the possibility for partial autonomy.

- **Manual w/o gimbal** uses manual recognition and submission to judges. Lack of gimbal stabilization requires some type of banking algorithm for objects outside the boundary. With this option we also considered the possibility for partial autonomy.

Evaluation Methods and Results

As can be seen in Table 3, we choose an autonomous system without a gimbal. This system was shown to be most favorable primarily because of its performance in the robust and reliable category. The chosen system did not perform well in viewing the off axis object, but this requirement was weighted low because of the relatively low amount of points obtained for identifying the off axis target. On the other hand, the chosen system did perform well under the robust and reliable requirement. This requirement we weighted highly because of its effect on the overall function of the UAS.

Table 3: A decision matrix for a visual object selection system. A scale of 1-5 was used for weights with 5 having high importance and 1 having low importance. A 1-5 scale was also used to rate each options performance under each requirement. In this case, a 1 was used to indicate poor performance while a 5 indicates favorable performance. As can be seen, using an autonomous system without a gimbal was shown to be the most favorable.

Requirements	Weight	Autonomous w/ Gimbal	Autonomous w/o Gimbal	Manual w/ Gimbal	Manual w/o Gimbal
Detect Features	5	4	4	5	5
Off-Axis Object	1	3	1	4	1
Return Object Locations	5	5	5	3	3
Emergent Object	2	4	4	5	5
Autonomous	3	5	5	1	1
Robust and Reliable	5	3	4	2	4
Totals	.	86	89	67	74

References

Wang, Y. and Boyd, S. (2008). Fast Model Predictive Control Using Online Optimization. [online] Web.stanford.edu. Available at: <https://web.stanford.edu/~boyd/papers/pdf/fast_mpc_ifac.pdf> [Accessed 12 Sep. 2018].

Concept Definition

ID	Rev.	Date	Description	Author	Checked By
DJ-002	0.1	09-12-18	Fall camp draft	Ryan Anderson	Jacob Willis
DJ-002	0.2	09-14-18	Revisions after design review	Ryan Anderson	Kameron Eves

Path Planner

Concept Selected: Fast MPC

In order to meet design requirements of autonomous flight and the ability to avoid static obstacles, the Fast MPC algorithm will be implemented in the controls of the aircraft. This algorithm will be used to create a path to reach several waypoints as determined by the competition. The following description is taken from DJ-001:

> Fast MPC is a feedforward form of high level control. This system uses a dynamic model of the aircraft to predict the future state of the aircraft and so allow it to make better informed control input decision at the current time step. A cost function is then used to bias the path towards a goal and away from obstacles. The predictive nature of the system would help with dynamic obstacles. The biggest downside of MPC in general is the computation speed due to the dynamic calculations. However, this is mitigated or even eliminated using the Fast MPC methods presented in a paper by Yang Wang and Stephen Boyd. (Wang and Boyd, 2008) Using these methods, MPC has been performed at speeds of up to 500 Hz, which is more than fast enough to accomplish our goals.

As illustrated in fig. 1, the Fast MPC will begin by creating a path to reach all required waypoints while avoiding obstacles at their current state. Then, it will predict the state of the aircraft at a future time and update its path accordingly, effectively evading dynamic obstacles. The genius of Fast MPC lies in its predictive model. The state of the plane at a future time is predicted based on the velocities and positions of dynamic and static obstacles, which is fed back into the optimizer, which adjusts its route accordingly. Obstacle velocities change continuously, so the predictive model must be continually updated to compensate.

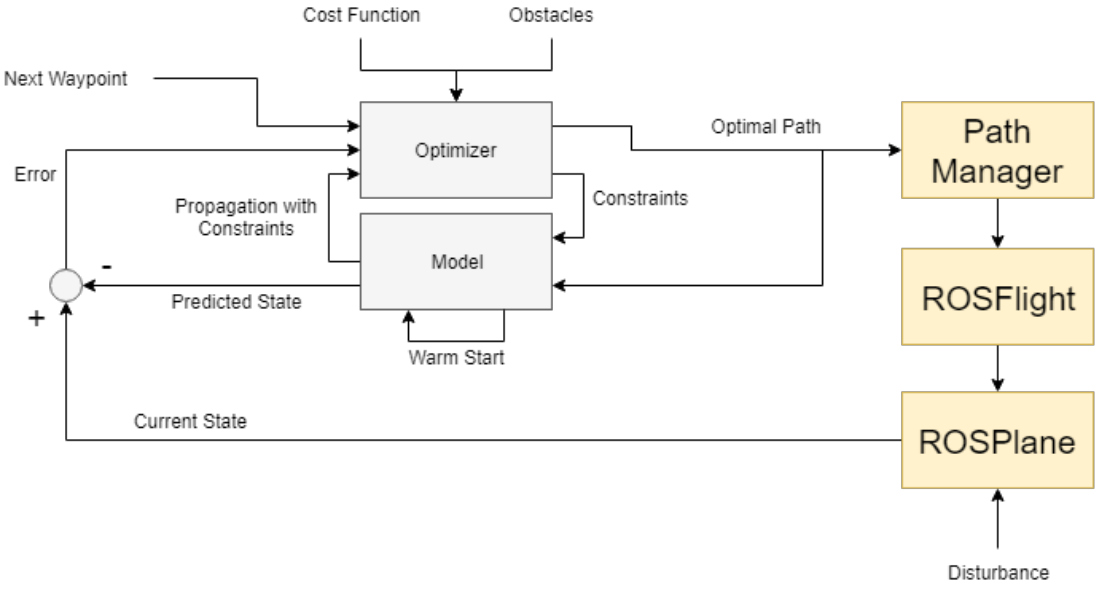

Figure 1: Fast MPC algorithm block diagram

First, waypoint and obstacle locations will be fed into the optimizer, which will then create a set of ideal intermediate waypoints. These waypoints will then be sent to the path manager, which in turn creates a path for the plane to follow. This is in turn sent to ROSPlane, which actuates the ailerons and other controls of the plane to execute the path.

Last year's plane avionics configuration already includes elements of this structure. Fig. 2 illustrates the current state of the architecture and what remains to be implemented.

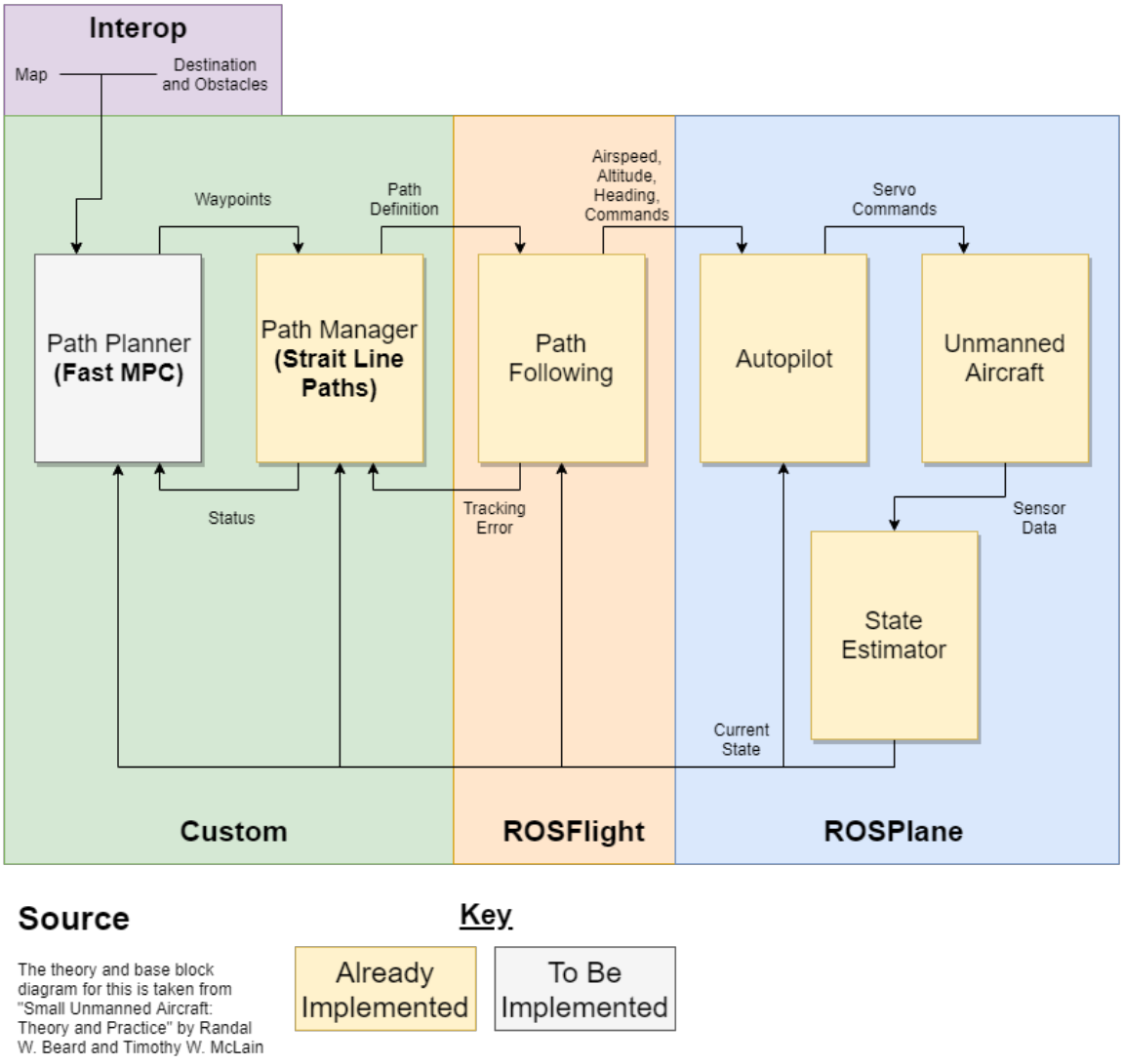

Figure 2: Autopilot software architecture block diagram

Visual Object Classification

Concept Selected: Autonomous Without Gimbal

To meet team requirements and maximize possible score, a fully autonomous image recognition system without use of a gimbal was chosen. Building this side-by-side with last years manual image submission system provides redundancy and the highest chance of success in this portion of the competition. The following broad overview is taken from

DJ-001:

> Autonomous w/o gimbal is end-to-end autonomy with no gimbal stabilization. It includes automatic recognition and submission to judges. It does not include targeting of objects outside bounds. As such it will require the UAS banking to view the off axis object.

While autonomous with gimbal stabilization was initially the preferred choice, the concept selection matrix highlighted the issues with this concept. The gimbal would add additional weight and more potential points of failure to the UAS. The stabilization provided by the gimbal is not significant enough given the robustness of the industrial camera system purchased by last years team. Aiming the vision system out of bounds is the one detriment to having no gimbal, however the reduced weight and complexity more than make up for this.

As shown in fig. 3, the autonomous vision system will be built as an add-on to the already existing manual system. Using OpenCV and image filtering techniques, key features (letter, color, shape, location) will be identified and associated with an image. A database will keep track of identified results. Once an adequate threshold of duplicate results are identified in sequence, the system will submit these results to the judge server using the interop relay. 10-15 images are streamed per second, which should allow multiple shots of the target feature.

Streamed images will also be forwarded to the frontend GUI for manual identification and submission, as was done by last year's team. Multi-submission systems such as this incur no penalty and allow us to maximize potential score.

The OpenCV image processing block algorithm is defined in more detail in the vision prototyping document.

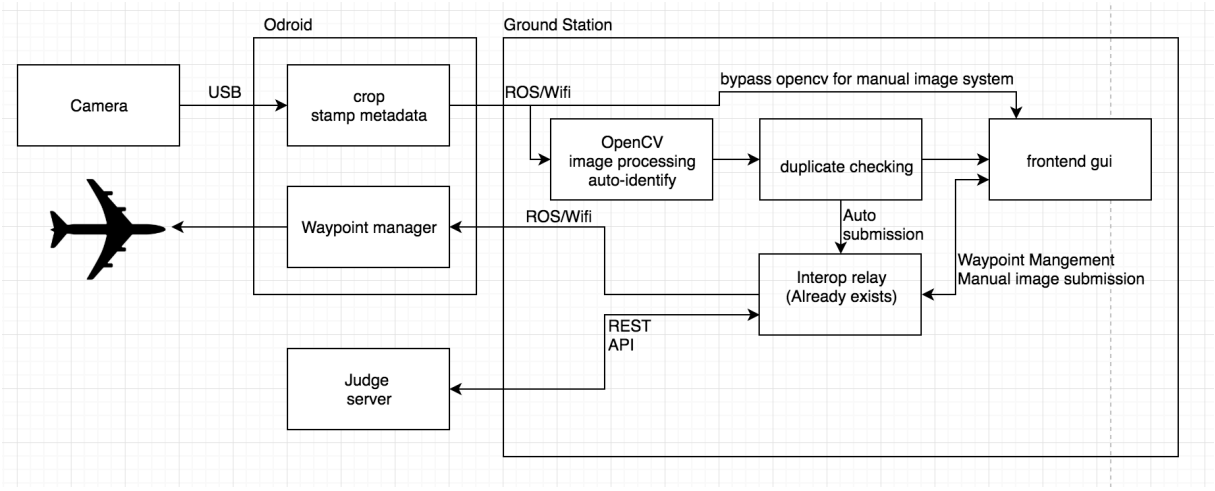

Figure 3: Autonomous image system block diagram

Prototype

Controls

ID	Rev.	Date	Description	Author	Checked By
PR-001	0.1	09-17-18	fall camp draft	John Akagi	Brady Moon

```
init_obstacles();
waypoint = rrt();
current_waypoint = 0;
transmit_waypoints(waypoint[current_waypoint]);

while(mission_finished() == false){

   unscented_kalman_filter(obstacles);

  if(detect_collision()) {

    waypoint[current_waypoint] = fast_mpc();
    transmit_waypoints(waypoint[current_waypoint]);
  }

  if(hit_waypoint(waypoint[current_waypoint])) {

   current_waypoint++;
   transmit_waypoint(waypoint[current_waypoint]);
  }
}
```

This code is intended to outline the code that will be used to control the UAV for obstacle avoidance. At the beginning of the flight, the initial path will be calculated in order to avoid static obstacles. During the flight, deviations to the pre-planned flight path will be recalculated in order to avoid dynamic obstacles while still maintaining the flight path.

Automated Visual Target Detection System

ID	Rev.	Date	Description	Author	Checked By
PR-002	0.1	09-14-18	fall camp draft	Jake Johnson	Connor Olsen

```
Automated Imaging Sequence:
  #Process for detection while continuously taking new frames (at a rate > 10
     fps)

def main():
   convertToHSV(image)
   eliminateBackground(image)
   arrayOfBlobs = findBlobs(image)
   for each blob in arrayOfBlobs
   if blob within size range
      identifyFeatures()
         sendToJudges()

def convertToHSV(image):
  # converts image to HSV colorspace

def eliminateBackground(image, bounds):
  # threshold by color to eliminate background

def findBlobs(image):
  # detect points of interest
  # return False if the blob is not large enough
  maskImage()

def identifyFeatures():
  findShape()
  findText()
  findTextColor()
  findShapeColor()
  findGeolocation()

def maskImage():
   # mask image with threshold

def findShape():
   # if no obvious shape found = false
   # return shape

def findText():
   # if not found = false
   # return shape
```

```
def findTextColor():
   # return text color

def findShapeColor():
   # return shape color

def findGeolocation():
   # use plane attitude and geometry to determine location of target

def sendToJudges():
   # Publish to ROSTopic for interop-relay

# Process for detection while continuously taking new frames (at a rate > 10
    fps)
Threshold by color (eliminate background)
Find blobs (interesting features)
If blob is in a good size range (feature detected)
  Mask image with threshold
  Find shape
  If no obvious shape found, return
  Find text
  If not found, return
  Find color of text
  Find color of shape
  Use plane location and geometry to determine location of object
  Send to judges
Else, not detected
```