University of Colorado
Department of Aerospace Engineering Sciences
ASEN 4018

Fall Final Report (FFR)

Autonomous Localization for GPS-denied Aerial Tracking and Reconnaissance

Team Members

Nolan Stevenson	Nicholas Grant	Brendan Bradley
Carson Kohlbrenner	Yarden Kelmann	Andrew Kabos
Lucia Witikko	Blair Schulze	Zane Vandivere
Nyah Baltazar	Marcus Quintanilla	Thomas Dunnington

Contents

List of Figures

List of Acronyms

ALLIGATR	Autonomous Localization for GPS-denied Aerial Tracking and Reconnaissance
GPS	Global Positioning System
CONOPS	Concept of Operations
UAS	Uncrewed Aerial System
RGV	Robotic Ground Vehicle
FBD	Functional Block Diagram
RGB camera	Red, Green and Blue camera
SD card	Secure Digital card
NLS	Non-Linear Least Squares
2DRMS	Two Dimensional Euclidean Root Mean Square Error
FPS	Frames per Second
HFOV	Horizontal Field of View
VFOV	Vertical Field of View
PWM	Pulse-Width Modulation

Definition of Symbols

$\vec{r}_d$	=	Drone inertial position	[m]	P	=	Image principal point	
$\vec{r}_{rel}$	=	RGV relative position	[m]	c_x	=	Center of image x-axis	[pixels]
$\vec{r}_r$	=	RGV inertial position	[m]	c_y	=	Center of image y-axis	[pixels]
$\hat{x}_d$	=	Drone x-axis		x_f	=	Centroid x-axis coordinate	[pixels]
$\hat{y}_d$	=	Drone y-axis		y_f	=	Centroid y-axis coordinate	[pixels]
$\hat{z}_d$	=	Drone z-axis		x_c	=	Centroid relative x-axis coordinate	[pixels]
s_x	=	Conversion factor x-axis	[mm/pixels]	y_c	=	Centroid relative y-axis coordinate	[pixels]
s_y	=	Conversion factor y-axis	[mm/pixels]	f	=	Camera Focal Length	[mm]
s	=	AR Tag Conversion factor	[m/pixels]	Δl_{avg}	=	Length of AR tag sides	[m]
x_r	=	Relative measurement x-axis	[m]	α	=	Pointing angle x-axis	[rad]
y_r	=	Relative measurement x-axis	[m]	β	=	Pointing angle y-axis	[rad]
E_r	=	Relative measurement E-axis	[m]	θ	=	Drone pitch angle	[rad]
N_r	=	Relative measurement N-axis	[m]	ϕ	=	Drone roll angle	[rad]
h_k	=	RGV dynamic model	[m]	ψ	=	Drone yaw angle	[rad]
J_{NLS}	=	NLS cost function	[m^2]	ϵ_{gps}	=	GPS error	[m]
$\hat{X}_0$	=	RGV state estimate	[m]	ϵ_{att}	=	Attitude error	[rad]
$H(\hat{X}_0)$	=	Dynamic model jacobian					

1 Project Purpose

Authors: Lucia Witikko, Nyah Baltazar

1.1 Project Description and Implications

Communication between ground targets has limitations in difficult-to-map terrain and cooperative localization is desirable to gather information about a network of targets because it can serve as an intermediary link between the targets. Having an intermediary link via unmanned aircraft is advantageous for increased communication and mapping. Unmanned aircraft provide a combination of maneuverability and sensing ability for any mission operating in difficult-to-map terrain due to their variable aerial perspective. Their ability to simultaneously pinpoint and trail multiple targets in a GPS-denied ground field provides significant advantages for many fields including emergency response teams, remote science missions, military applications, and future planetary exploration.

1.2 Concept of Operations (ConOps)

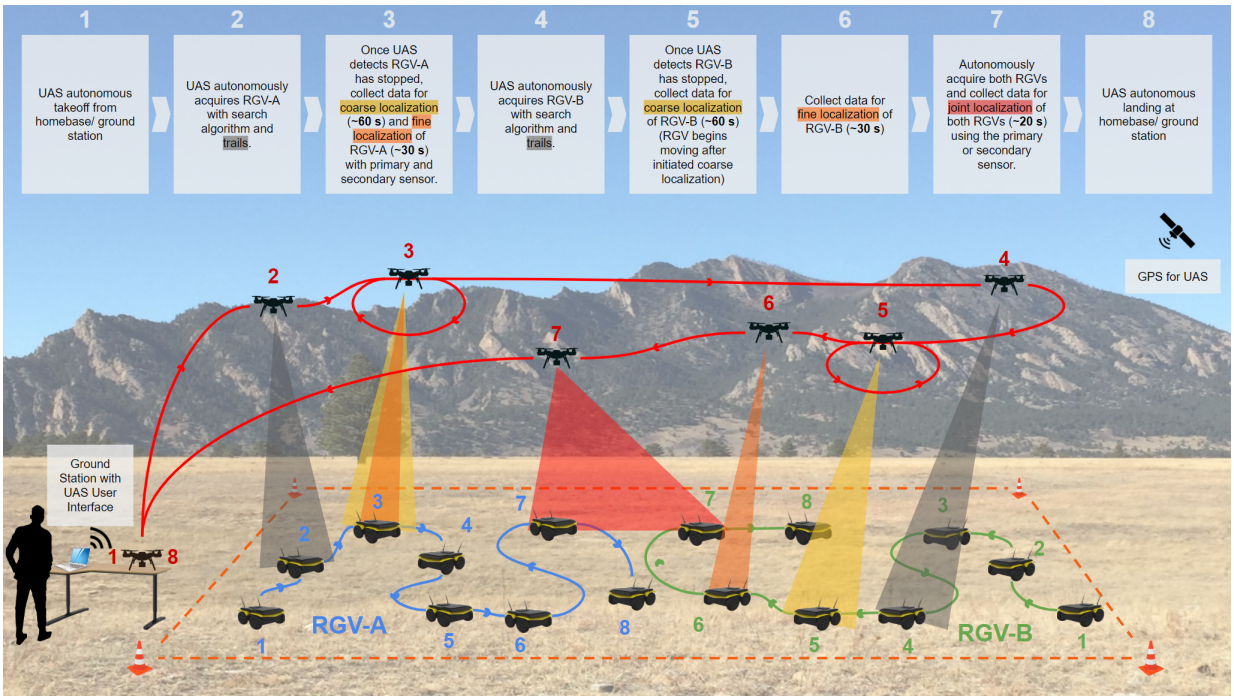


Figure 1: Mission CONOPS

The ALLIGATR unmanned aerial system (UAS) shall relay and save autonomously collected surveillance data of multiple robotic ground vehicle (RGV) targets to a ground station in real time. After deployment, the UAS will independently travel to the predetermined GPS-denied testing range and begin the mission. Location and trailing of these ground vehicles will consist of two main components of drone activity: acquisition and pursuit of a target followed by coarse and fine localization to collect data for the post-flight determination of precise coordinates of the targets. During the acquisition stage, the UAS shall employ its primary sensor to autonomously locate the first target, RGV-A, and begin trailing. Once RGV-A has halted motion, the UAS shall autonomously initiate an orbiting flight path about the target and begin coarse localization data collection for 60 seconds. Coarse localization data shall have accuracy within two meters. Fine localization data collection using the secondary sensor will be initiated after coarse localization data collection to increase the location certainty for RGV-A, lasting 30 seconds. Fine localization data shall have accuracy within one meter. After this sequence is complete, the UAS will disengage from RGV-A and repeat the acquisition, search, and localization data collection process for the second target, RGV-B. Once coarse and fine localization data collection for both targets has been achieved, the UAS will re-acquire RGV-A while keeping RGV-B within surveillance range. During this final stage of joint localization data collection, both RGV targets will be tracked simultaneously and the same onboard sensor shall record their path data for 20 seconds. Joint localization data shall have accuracy within one meter. Once joint localization data collection ends, the mission is complete and the UAS will autonomously return to the home base.

2 Detailed Design

Authors: Andrew Kabos, Nicholas Grant, Yarden Kelmann, Blair Schulze, Zane Vandivere, Marcus Quintanilla

2.1 Functional Block Diagram

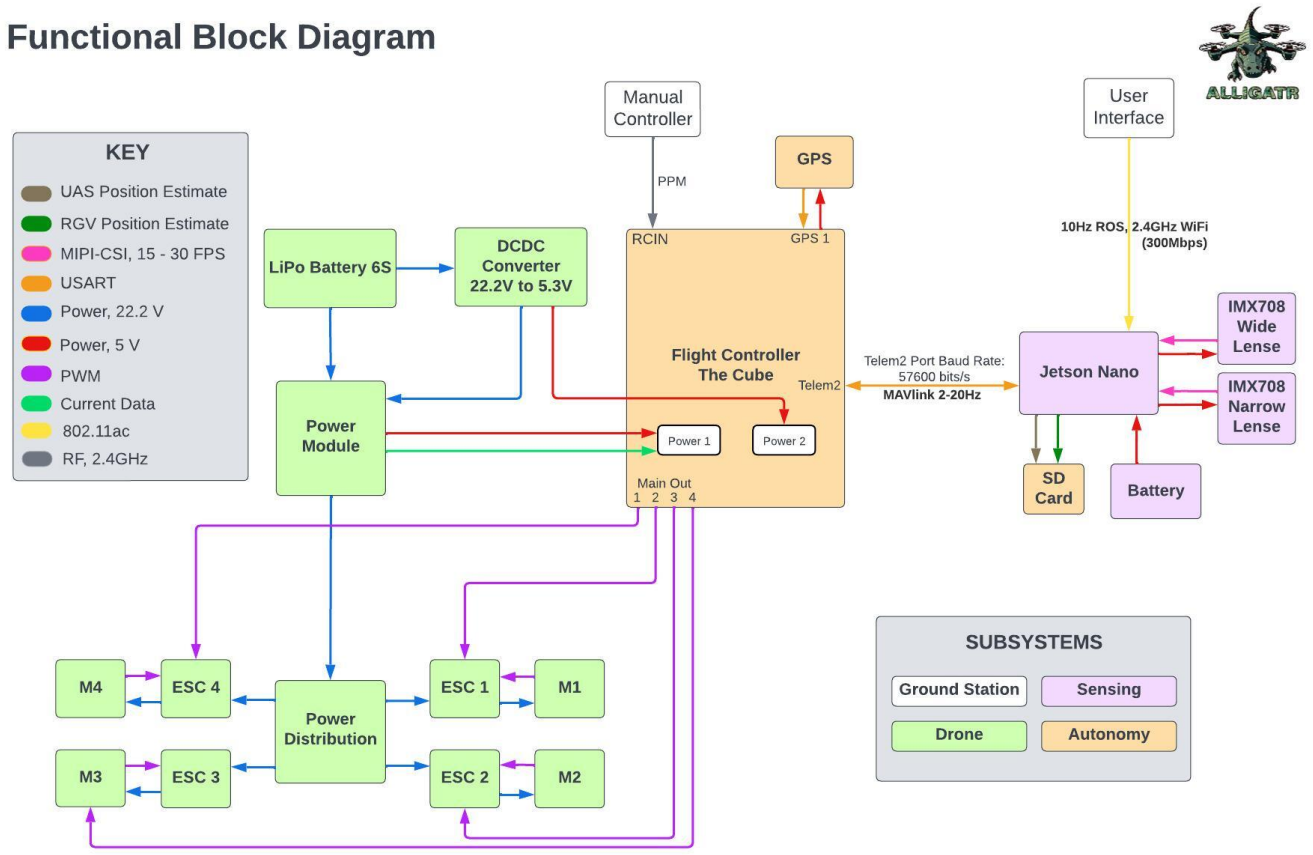


Figure 2: Functional Block Diagram - Full System

This system operates autonomously within the mission boundaries. Data from the primary and secondary RGB camera sensors is fed into the Jetson Nano processor, which uses our mission planning algorithm to determine the current mission phase and necessary drone position commands. The processor then sends motion planning waypoints to the flight controller, which distributes power appropriately to the motors to maneuver the drone.

The electrical system incorporated in this UAS design takes inspiration from the Ardupilot Cube ecosystem documentation. It uses two LiPo 6s batteries to provide the main power to the electronic speed controllers (ESCs), motors of the drone, the Cube Black autopilot module itself as well as its peripherals connected to it including the manual control receiver connected to the RCin port and the Here2 GPS module connected to the GPS 1 port. Power to the Cube autopilot is redundant and regulated using DC-DC buck converters with battery eliminator circuits. The DC-DC converter powers a Mauch power module with an integrated Hall sensor, which connects to the Power 1 port of the Cube as the primary power line. The power module, which is based around an Allegro Hall Sensor ACS-100U, is able to output data regarding the battery voltage and current, both of which are also provided into the Power 1 port of the Cube. Redundant power is provided to the Power 2 port also using a battery eliminator circuit without the Hall sensor connection. Power to the motors and ESCs is unregulated and comes directly from the LiPo batteries. The exact power level sent to the motors is controlled by pulse-width modulated (PWM) signals sent to the ESCs from the Cube's Main Out port. The Jetson Nano processor mission planner that is connected to the cube via USART on the TELEM2 port has its own power source for further redundancy and safety. It receives power from a set of four 18650 Li batteries regulated to a stable 5V. The Jetson Nano then provides power to both our camera sensors through two MIPI-CSI-2 ports, and to a USB to WiFi converter through the USB-A port.

During our initial search phase, the drone follows a predetermined search pattern and takes inputs from the IMX708 wide-angle primary sensor into the processor for analysis. Once an RGV has been located and has stopped moving, the coarse localization phase begins and the inertial position estimates of the RGV are stored on the SD card. The fine localization phase begins once the coarse localization algorithm has been completed, effectively narrowing the error bounds of the inertial position estimation. This entire process is repeated for the second RGV, after which joint localization will follow a similar process using combined primary and secondary sensor data to localize both RGVs simultaneously.

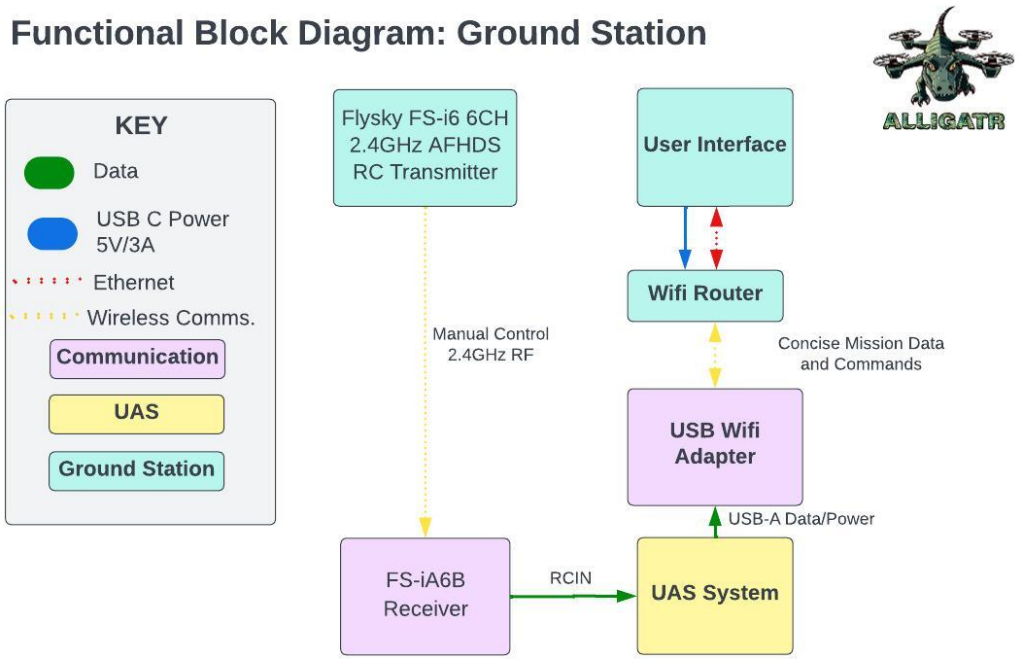


Figure 3: Functional Block Diagram - Ground Station

The ground station of our system will include a manual control override and a user interface that will interact with the USB WiFi adapter connected to our onboard processor. Throughout the mission, status updates will be sent to the UI via this WiFi connection. These updates may include but are not limited to the current mission phase, GPS drone data, and inertial estimates of RGVs. The manual controller will have three switches corresponding to desired UAS flight modes: hover, fully manual control, and fully autonomous input (default).

2.2 Computer Aided Design

Figure 4: Full Assembly

Figure 5: Sensing Housing

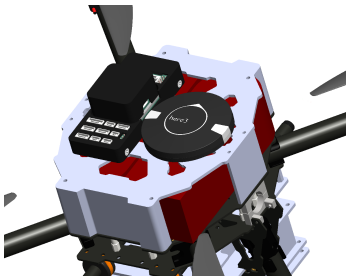


Figure 6: Placement of Batteries (Red), Cube, and GPS

The sensors are placed facing straight down at the bottom of the sensor housing. The Jetson Nano is directly above that. The LiPo batteries powering the propulsion and cube are placed near the center of thrust, with the Pixhawk Cube and GPS placed on top. The electronic speed controllers are placed directly below the blades to ensure sufficient airflow for cooling. Care was taken to design the housing so that it can be printed with no 90-degree overhangs and so wiring can be routed between all of the components.

2.3 Subsystem Design

Subsystem	Purpose
Sensing Package	Acquire RGV inertial positions
Mission Planner	Transition through required phases
Motion Planner	Generate movement waypoints for each phase
Flight Control	Feedback loop to control and stabilize UAS
Power	Allow vehicle propulsive and electrical capability to meet mission objective
Communications	Enable data link between components on UAS as well as between UAS and ground station/UI

Table 1: Subsystem Breakdown

2.3.1 Sensing Package

The purpose of the sensing package is to accurately acquire RGV positions in order to inform guidance of the drone and enable data collection for localization. The sensing package consists of a primary sensor, secondary sensor, and sensing algorithm. Both the primary sensor and secondary sensor are 12MP IMX708 cameras, which connect via ribbon cables to MIPI-CSI-2 camera ports on the Jetson Nano processor. Fig. 8 and Fig. 7 show the trade off between having a high FOV for large sensing area and the associated pixel error. On the software side of this package,

the sensing algorithm is executed on the Jetson Nano, which is also shared with the mission and motion planners. Consequently, sensing data is readily available for programmatic use by the "brain" of the UAS. The sensing algorithm consists of three major components: blob detector, AR tag detector, and localizer. The blob detector and AR tag detector utilize open-source computer vision algorithms to output the pixel coordinates of detected RGVs in the 2D image frame. The localizer function receives a pixel coordinate and transforms it into an inertial coordinate in the ENU frame. The estimated inertial coordinate of the RGV then gets passed on to the mission planner for autonomous guidance and decision making.

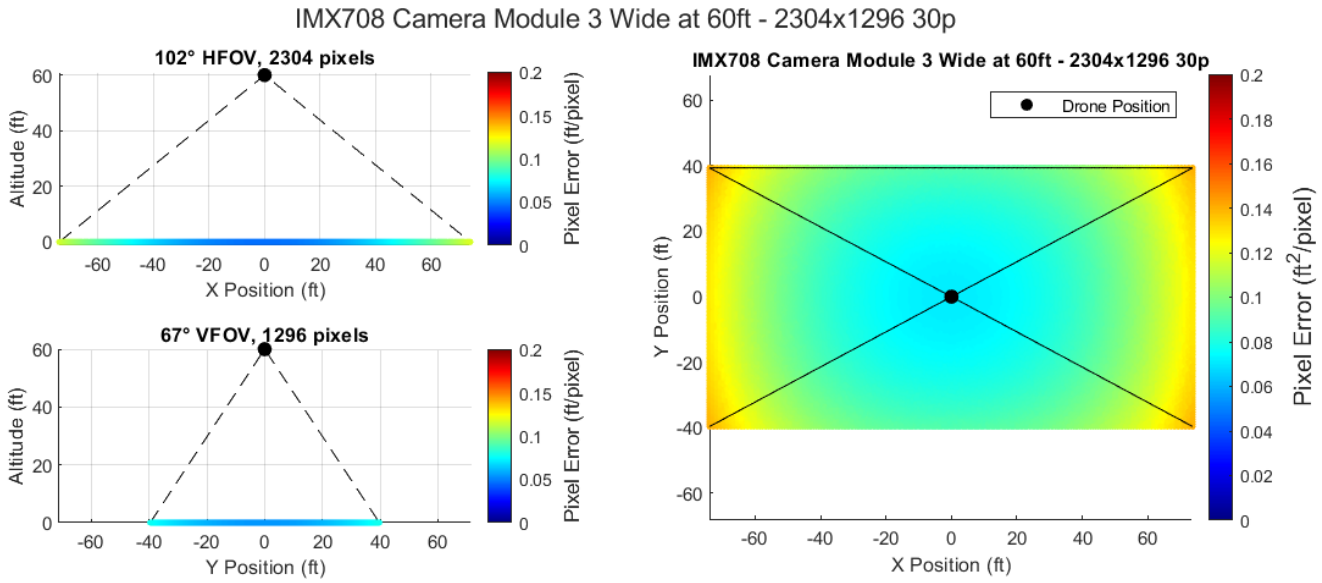


Figure 7: IMX708 Wide FOV Expected Pixel Error

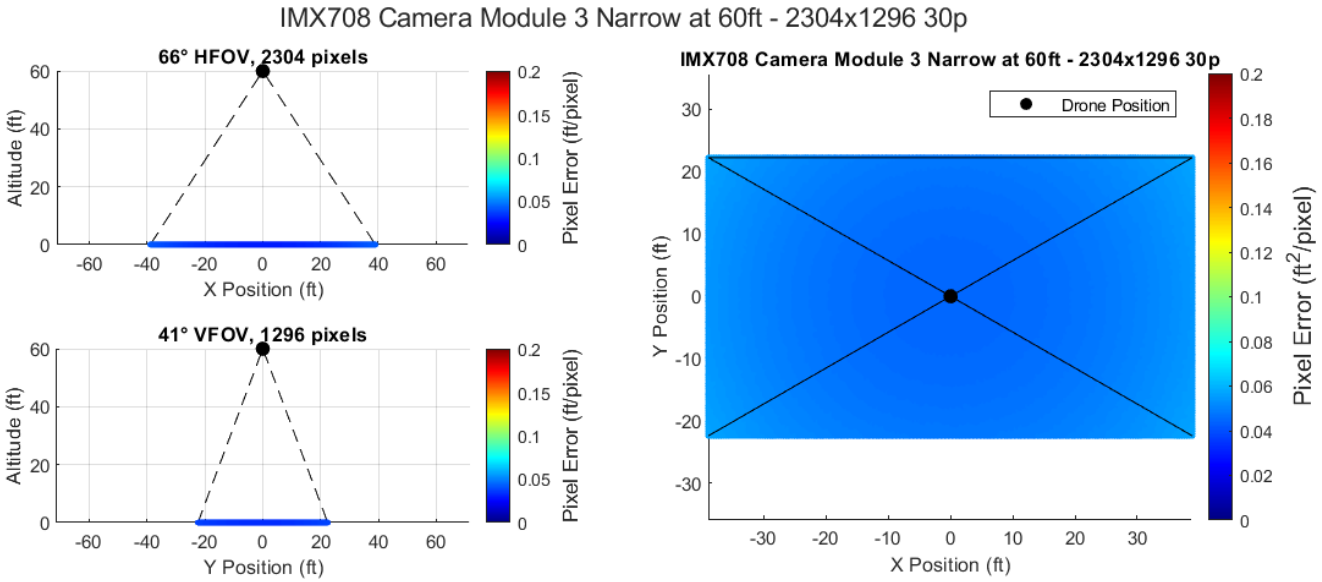


Figure 8: IMX708 Narrow FOV Expected Pixel Error

The two computer vision detectors each have a distinct purpose throughout the mission. The blob detector has a much higher detection rate and is capable of detecting RGVs at higher altitudes. Consequently, the blob detector is used in all phases of the mission for guidance of the drone. On the other hand, the AR tag has a lower detection rate, but it is capable of distinguishing between the two RGVs and is able to acquire orientation data for the customer. The AR tag detector is therefore used for identifying which RGV is being detected when an RGV is acquired, and it

is also utilized for improved RGV state data collection in the fine and joint localization phases of the mission.

2.3.2 Flight Control

In-flight control for the drone is provided by a Cube Black, a robust autopilot package with three redundant integrated inertial measurement units (IMUs) as well as a powerful 32-bit STM32F427 processor and 32-bit STM32F103 co-processor for redundancy. The Cube Black runs open-source ArduCopter autopilot software set to interface with a quadrotor aircraft. This software package allows for pre-flight calibration with any drone setup, allowing the autopilot to take into account physical/mechanical characteristics of the drone such as center of gravity, mass, and motor thrust as well as take in a predetermined set of commands, dictated by the MAVlink (micro aerial vehicle link) protocol. Most notably, waypoints can be set using MAVlink commands to the Cube. Because MAVlink is to be used with a companion computer running ROS, MAVROS libraries will be used for compatibility.

To autonomously control the quadrotor, we will use a custom Guidance, Navigation, and Control (GNC) script to do all the motion and mission planning onboard our companion computer. This script will iteratively feed the autopilot, in *mode guided*, pose waypoints throughout the mission using MAVROS protocol, a ROS communication protocol designed specifically for micro aerial vehicles. The waypoint commands will be calculated based on the current phase, pose, and computer vision estimate of the quadrotor. Additionally, the IMU and GPS data onboard the flight controller is accessible by the companion computer using MAVROS. This allows important data such as pose measurements to constantly be published by the flight controller such that the companion computer can access it. The software diagram below illustrates this flight control system:

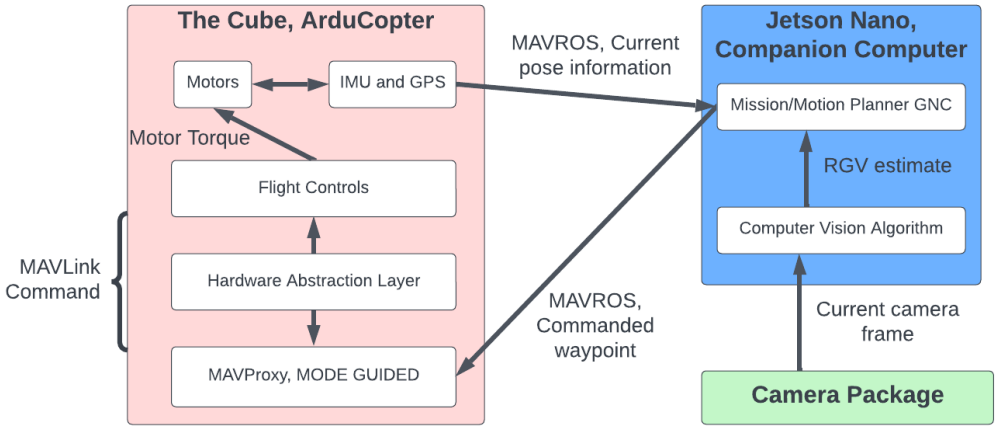


Figure 9: Flow chart for autonomous control of quadrotor

During the flight, the ArduCopter software enables the drone to perform a large set of commands, like moving through this set of commanded waypoints. The waypoints can be commanded to the ArduCopter autopilot through a lightweight control interface called MAVProxy. The advantage of using the control interface MAVProxy is it supports multiple modes of flight: manual, loiter, and guided, where the drone is completely controlled by the manual controller, the drone assumes a stationary hover, and the drone follows the aforementioned waypoints set by the companion computer respectively. Thus, our manual flight controller can have a three-way switch that is easily able to switch between these modes of flight, for safety considerations. While in *mode guided*, the quadcopter can take in commanded waypoints to move to. Specifically, a MAVLink command is sent through a hardware abstraction layer (HAL) incorporated in the autopilot. The HAL leverages the physical stats of the quadcopter to translate the commanded waypoint to a commanded set of motor torques that can be fed through the ESCs to the motors. Therefore, this protocol establishes the baseline for how we will be able to control the quadrotor to follow a waypoint path.

2.3.3 Mission Planner

The mission planner takes in data on the drone position from the flight computer, estimated RGV positions from the computer vision, the current mission phase, and information about time into the phases from itself; and is able to calculate which phase of the mission the drone should be in. This phase is passed to the motion planner, discussed in the next section. This process is shown in the block diagram below. The planner is made up of several sub-functions for each phase that flow into specific other phases. There are eight distinct autonomous phases defined in our design, outlined in the table below. The autonomous planning can be overridden at any point by manual control.

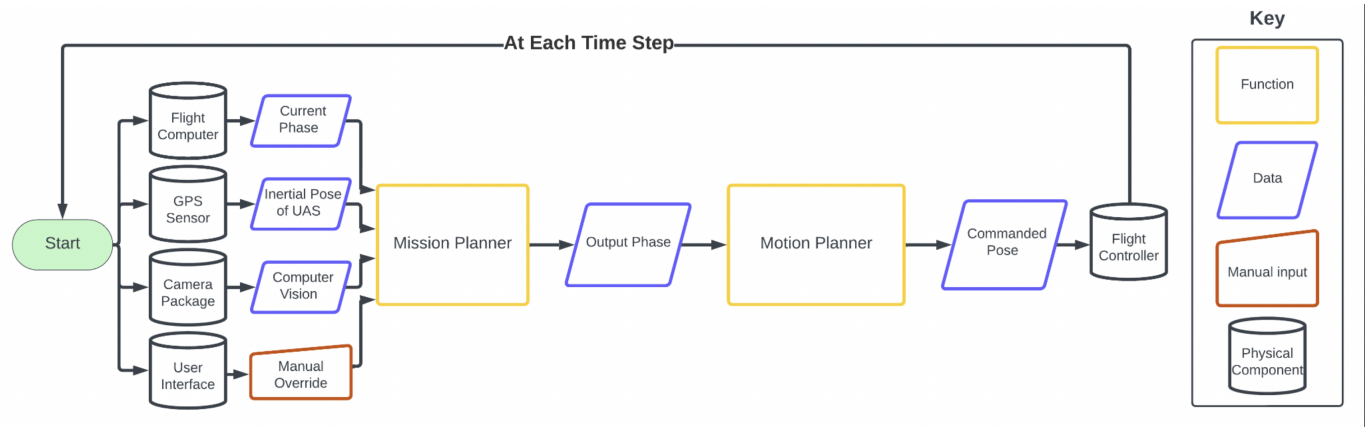


Figure 10: High Level Mission and Motion Planner

Mission Phase	Motion Plan
Takeoff	UAS commands upwards to meet 30 ft height requirement.
Boundary Control	UAS commands towards center of mission environment, to return UAS to mission bounds.
Search	UAS conducts intelligent search pattern across the environment.
Trail	UAS commands towards the most recent RGV estimate.
Coarse	UAS commands a circular motion around the most recent RGV estimate.
Fine	UAS commands directly on top of the most recent RGV estimate.
Joint	UAS commands towards an optimal height and orientation to maximize data collection for both RGVs.
Complete	UAS commands back towards the launch pad, to prepare for manual landing.

Table 2: Mission Phase Descriptions

An example of how this works will now be described for the search phase. All of the phases have a check for moving into boundary control. Search then checks if RGV-A is in view, and if so, if it has been fully coarse and fine localized. If that RGV has been fully localized, it checks if RGV-B has also been fully localized, setting the phase to joint localization if so. If RGV-B has not been fully localized, the phase is set to search, ignoring RGV-A. If RGV-A has been coarsely localized but not fully fine localized, the phase is set to fine localization. If RGV-A has not been fully coarse localized, the phase is set to trail. This is because we cannot begin coarse localization unless the RGV has fully stopped, and it is simpler to just let the trail phase function catch any cases than repeat the same conditionals in search. The same checks are then made for RGV-B. Finally, if neither RGV is in view, the phase is set to search, so the drone will continue along its search pattern. This process is shown visually in the block diagram below. Diagrams for the other phases of the mission can be found in the Appendix, Section 5.5.

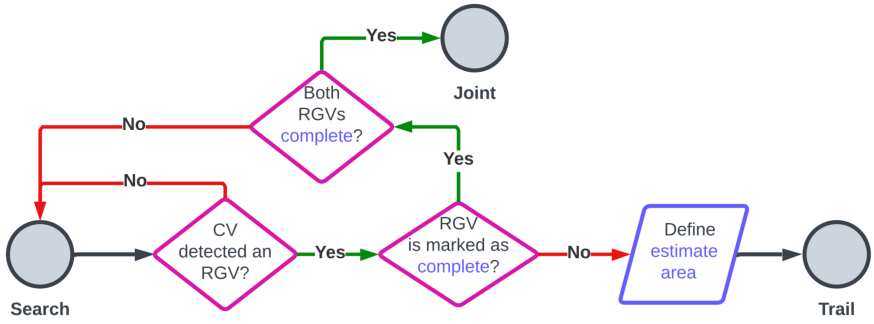


Figure 11: Mission Planner Search Function

2.3.4 Motion Planner

The motion planner is structured similarly to the mission planner, with a sub-function for each phase, as can be seen in Table 2. Each function takes in the estimated current drone state, the estimated current RGV states, and the environment bounds. Certain functions also take in some tracking variables and the drone FOV. For example, the coarse localization phase function uses a theta variable that tracks where in the orbiting flight path the drone is. Each function outputs a waypoint to be passed to the flight controller, with some functions also outputting updated tracking variables. The search function also updates the environment bounds. When the drone first enters the search function, determined using a tracking variable, it sets the waypoint at the closest corner of the environment bounds. The environment bounds are initially stored as the bottom left corner, (x_1, y_1), and the top right corner, (x_2, y_2). However, the search algorithm assumes that the drone moves from y_1 to y_2 and back and forth from x_1 to x_2 in a specific pattern. In order for this to work in the case of the closest corner and thus start point not already being the bottom left corner, the environment bounds are rotated at the start of search such that (x_1, y_1) is always the corner closest to the drone when the search function is initialized. The search algorithm then moves left to right, forward, right to left, forward, and so on until it reaches the edge of the bounds again, where it restarts the search from the beginning. The trail and fine localization phase functions simply set the waypoint as the estimated RGV position plus thirty feet on the z-axis. The boundary control phase sets the waypoint as the exact midpoint of the environment bounds. Finally, the joint localization phase function sets the waypoint to the midpoint between the last known estimated positions of the two RGVs at maximum elevation, then spins around the z-axis until both RGVs are detected. It then adjusts the elevation to the minimum necessary to continue detecting both RGVs, and waits there until enough data has been collected.

2.3.5 Power and Thrust

Sizing the battery that was to power the propulsion system and the cube was primarily a function of thrust required, and therefore vehicle weight. The process of picking rotors capable of delivering the thrust given the vehicle weight was iterative, as this added battery weight, which increased the thrust required, which increased the battery weight. The final vehicle weight, including the batteries and the rotors, was 3,868 grams. For reasonable maneuverability, a maximum thrust-to-weight ratio (T/W) of 2 was desired. This value would allow the vehicle to hover at half throttle, allowing the mission to be primarily run at an efficient power setting. Higher T/W would allow even more efficiency, as hover and therefore movement would be available at a lower throttle. With lots of leftover budget, more leeway was available to pick more expensive and powerful motors. The MN4116 IP45 Navigator was chosen for motors. The manufacturer published static test data with a 16-inch propeller, giving us predictable performance data without relying on first principal models that are limited by inefficiencies of the motors. These are often significant and unpredictable without tests. The propeller-motor combination gives an individual maximum thrust value of 3360 g, where thrust is measured in how much mass the rotor can hover with. This gives a total maximum T/W of 3.5, exceeding our desired T/W and providing additional maneuverability and power efficiency.

Nominal flight will not be conducted at maximum T/W, but at a throttled setting. The battery was sized based on a continuous T/W value of 1.5 for 15 minutes. Choosing to model the flight at a continuous T/W of 1.5 allows for a conservative estimate on battery as the vehicle will be able to not only hover but move for the entire flight duration. The timing requirement of 15 minutes is over double the minimum mission requirement, giving plenty of overhead in terms of higher thrust values throughout the flight.

Maximum Instantaneous Propulsion Current Draw (A)	Total Propulsion Charge Capacity Required (mAh)
8	8120
Additional Power Requirements	**Instantaneous Current Required (mA)**
Cube	2500
Total	2500
Additional Charge Capacity Required (mAh)	625
Battery Selection	**Charge Capacity (mAh)**
LiPo 6S	5200
Battery Number	2
Total Battery Charge Capacitance	10400
20% of unusuable charge	2080
Charge Capacity Required	8120
Spare Power	200

Table 3: Power Budget Analysis

The propulsion amperage was given by the aforementioned manufacturer tests. For a desired T/W of 1.5, each rotor will need to provide 1450 grams of thrust. Referencing figure 12, roughly 8 amps is required to pull that much thrust. This value is converted into mAh required for the duration of the flight. The cube requires 2500 mA per the manufacturer, including the peripherals.

Type	Propeller	Throttle	Voltage (V)	Thrust (g)	Torque (N*m)	Current (A)	RPM	Power (W)	Efficiency (g/W)	Operating Temperature (℃)
MN4116-KV340	T16*8"	40%	23.78	855	0.19	3.70	3386	88	9.72	72℃ Ambient (Temperature: 23℃)
		42%	23.77	932	0.20	4.15	3515	99	9.45	
		44%	23.77	1004	0.20	4.53	3635	108	9.32	
		46%	23.76	1064	0.22	4.86	3736	116	9.21	
		48%	23.75	1141	0.23	5.37	3861	128	8.95	
		50%	23.74	1221	0.25	5.87	3977	139	8.76	
		52%	23.73	1292	0.28	6.39	4102	152	8.52	
		54%	23.72	1369	0.28	6.97	4223	165	8.27	
		56%	23.71	1454	0.31	7.50	4336	178	8.17	
		58%	23.71	1527	0.32	8.12	4466	192	7.94	
		60%	23.69	1629	0.34	8.81	4579	209	7.80	
		62%	23.68	1725	0.37	9.56	4693	226	7.62	
		64%	23.67	1811	0.38	10.26	4816	243	7.45	
		66%	23.66	1913	0.40	11.02	4932	261	7.33	
		68%	23.65	2001	0.42	11.83	5045	280	7.15	
		70%	23.63	2105	0.45	12.73	5154	301	7.00	

Figure 12: Manufacturer Rotor Test Data

Powered by two LiPo 6S batteries, both the propulsion system, the cube, and the cube's peripherals will require 8,120 mAh for the flight duration. With most batteries only discharging 80% of their advertised capacity, 20% of the batteries are deemed unusable and subtracted from the power budget. This leaves an additional 200 mAh leftover. With all the overhead built into the flight time and throttle settings, we are confident the power system will be sufficient enough to meet all mission objectives.

2.3.6 Communications

The ground station portion of the communications subsystem comprises of a WiFi antenna on the drone that will connect to the Jetson Nano via USB, a router on the ground, a laptop on the ground, and a manual controller and receiver. The 802.11ac 300Mbps 2.4Ghz WiFi router interfaces with the WiFi antenna to allow a laptop to interface with the Jetson Nano via MAVProxy. This will allow us to send flight data such as the position, velocity, pose, and tracking state to a laptop. We can then actively load this data into a Matlab App which will allow us to see what tracking state the drone is in and to see what the drone is doing in real-time through the use of visualizations potentially including a flight path of the drone, various flight instruments, and a display showing pictures that are semi-frequently sent from the cameras.

The onboard portion of the communications subsystem mainly consists of the communication between the Jetson Nano processor and the Cube Black. This is done through a USART connection going from the TELEM2 port on the Cube Black to pins 6 (GND), 8 (TX), and 10 (RX) on the Cube Black. The USART connection operates at a baud rate of 57600 bits/s, and the information sent along this line takes the form of a ROS node of GPS data sent from the Cube to the Jetson as well as commands including but not limited to waypoints and orbit commands. All information sent between the Cube and Jetson is sent over the MAVlink protocol with a frequency of at least 2Hz and at most 20Hz. Because the Jetson Nano uses ROS, the MAVlink commands will be transferred through MAVROS. Additionally, the two IMX708 cameras used for both the primary and secondary sensors are connected to the Jetson Nano through two CSI MIPI ports.

3 Engineering Models and Prototyping Results

Authors: Carson Kohlbrenner, Thomas Dunnington, Nolan Stevenson

3.1 Models and Prototyping

Tables 4 and 5 show the engineering models and prototypes we have completed up to this point. The two we will be highlighting in this report are the Localization Method and the Autonomy Simulation.

Engineering Model	Relavant Requirement	Design Selection	Prototyping Completed
Localization Method	Data collected for (coarse, fine, joint) localization shall enable localization in an inertial reference frame with a 2DRMS accuracy of no greater than (2, 1, 1) meter(s).	Sensing Package and Computer Vision Algorithms	Hardware and Software
Autonomy Simulation	UAS shall determine autonomously how to navigate the aircraft to collect localization data on an RGV.	Mission and Motion Planner.	Software
Power Analysis	UAS shall provide its own power source and be untethered.	Batteries, Motors, Processor, and Sensing Package	None

Table 4: Engineering Models Completed

Prototype	Description	Requirements
AR Tags	AR Tag computer vision detection at various altitudes	SNS 3, SNS 3.1
Blob Detection	Blob computer vision detection at various altitudes	SNS 2, SNS 2.1
Raspberry Pi	Blob and AR Tag detection on a Raspberry Pi 4B with an IMX708 camera	UAS 3
Relative Localization	Relative localization algorithms applied to sample drone footage	SNS 6.2, SNS 6.3, SNS 6.4
Inertial Localization	Inertial localization algorithms applied to sample drone footage	SNS 6.2, SNS 6.3, SNS 6.4
Autonomy Simulation	Full mission simulation in MATLAB	FR 1, AUT 2

Table 5: Completed Prototypes

3.1.1 Localization Model Prototyping

Software: To prototype our localization method, we conducted hardware and software prototyping tests. For software prototyping, we developed blob detection and an AR tag detection program that takes in an image and outputs a centroid position of our tracked tag in pixels, along with the orientation and depth for AR tag detection. These programs were first applied to a single picture with an AR tag in it, then to a video, and then to multiple videos filmed with a drone. The results of this prototyping revealed that 1) flickering, or detection failure, may be prevalent for our AR tag detection method and 2) our blob detection program has a tendency to pick up on extra targets that are not intended to show up. Because of these prototyping results, we implemented a detection failure rate in our analyses of our localization method and a way to ignore unidentified targets in our mission planner. We also conducted a relative and inertial localization test where we processed provided drone footage by Professor Ahmed's PhD student Hunter Ray. We applied our algorithms and obtained relative and inertial estimates of the RGVs using our AR Tag and Blob Detection methods.

Hardware: Next, we tested our localization method on hardware that we were expecting to use on our drone. We ran both the blob detection and AR tag detection separately on a Raspberry Pi 4B using an IMX708 camera with a wide angle to test how well they ran. Because we ran into little computational deficiency, the prototyping results suggested that would could potentially run our blob detection, AR tag detection, and mission planner concurrently on a single processor, albeit a Nvidia Jetson Nano which is significantly more powerful than a Raspberry Pi 4B.

3.1.2 Autonomy Prototyping

To prototype our autonomous mission and motion planner, we developed a MATLAB environment where we could simulate a full mission from start to finish. This MATLAB environment also allowed us to add Gaussian error and run Monte-Carlo benchmarks to identify shortcomings and possible edge cases that we didn't plan for in our initial design. A more detailed breakdown of our MATLAB simulation is found in Section 3.3.1. Our prototyping results informed us that our software model described in Section 2.3.3 and Section 5.5 should be capable of handling the required mission, even when there is sensor error.

3.2 Localization Model

The overall deliverable for this project is a data package that can be used to determine the inertial position of both RGVs for coarse, fine, and joint localization estimates. Collecting this data is integral to the success of the missions so we must collect sufficient data to satisfy requirements. The following table includes the three main driving requirements for this model:

ID	Requirement	Parent Req	Child Req	Verification Method
SNS 6.2	Data collected for coarse localization shall enable localization in an inertial reference frame with a 2DRMS accuracy of no greater than 2 meters.	SNS 6	N/A	D
SNS 6.3	Data collected for fine localization shall enable localization in an inertial reference frame with a 2DRMS accuracy of no greater than 1 meter.	SNS 6	N/A	D
SNS 6.4	Data collected for joint localization shall enable localization of both RGVs in an inertial reference frame with a 2DRMS accuracy of no greater than 1 meter.	SNS 6	N/A	D

Table 6: Localization Model Driving Requirements

All of these requirements are related to the accuracy of the localization estimates. For coarse localization we must provide data to localize to an accuracy of less than 2 meters; for fine and joint the accuracy must be less than 1 meter. To show that we are capable of meeting these requirements, we started by developing a localization method that will utilize our sensing package and GPS measurements to determine the inertial positions of the RGVs.

3.2.1 Localization Method

The goal of the localization model is to determine the inertial location of the RGVs in the ENU frame using the drone's GPS and computer vision data from the primary and secondary sensors. Figure 13 shows the inertial frame for localization with an ENU coordinate frame. The position of the drone $\vec{r}_d$ is known from GPS measurements; to determine the RGVs position $\vec{r}_r$, our goal is to calculate $\vec{r}_{rel}$. With this setup, we made assumptions to simplify the localization process. Table 7 details these assumptions.

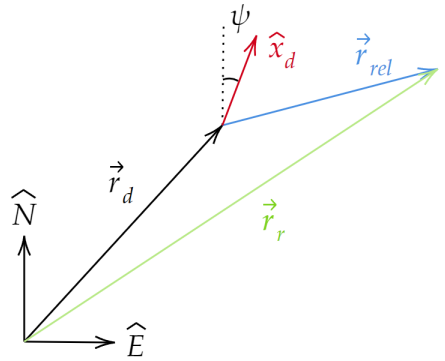


Figure 13: Inertial Frame for Localization

Localization Component	Assumptions
Drone GPS	The GPS module on the drone will provide accurate inertial measurements and the camera will have the same inertial location as the GPS receiver.
Camera	The camera is a perfect pinhole camera with no distortion.
RGVs	The RGVs will be on flat ground at a constant elevation.

Table 7: Localization Method Assumptions

The listed assumptions allow us to simplify localization to a two-dimensional problem where we will determine the E and N coordinates of the RGVs in the ENU frame. It also simplifies the computer vision aspect of the problem as we are assuming we have a perfect camera with a perfect centroid output. The first step in the localization process involves the computer vision algorithm to process an RGB image to get the pixel coordinates of the detected centroid. Figure 26 shows the output from the computer vision with the pixel coordinates x_f and y_f. The relative pixel coordinates from the principal point (P) x_c and y_c are calculated using the CV output and the known coordinates of the principal point c_x and c_y utilizing Equations 1 and 2.

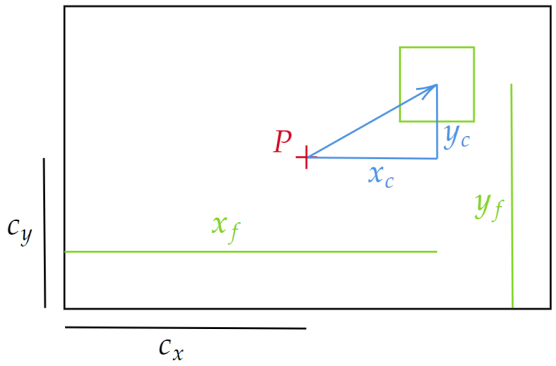
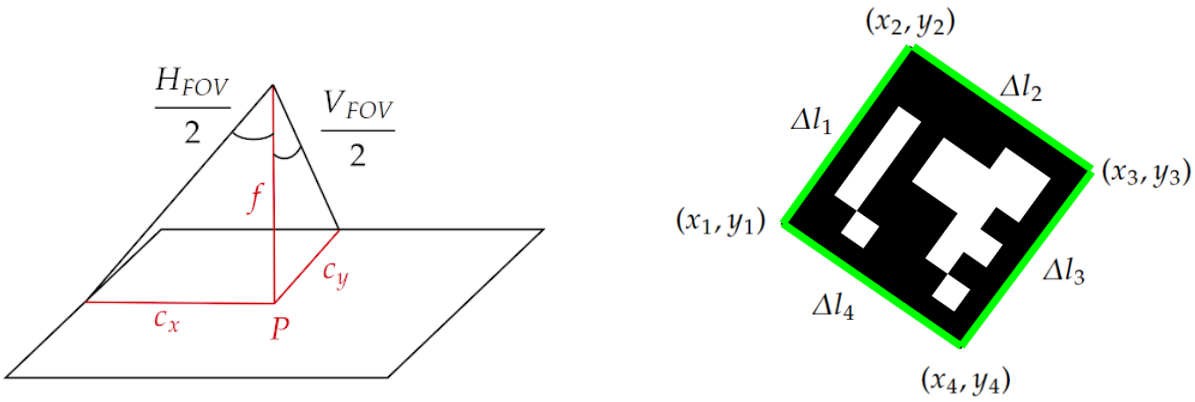


Figure 14: Computer Vision Output

$$x_c = x_f - c_x \tag{1}$$

$$y_c = y_f - c_y \tag{2}$$

Now that we have the pixel coordinates of the RGV, we will develop conversion factors to convert the pixel measurement to a distance. Figure 15a shows the optical center of the camera and the image plane. Using the known field of view and focal length of the camera, we can calculate two conversion factors s_x and s_y which convert a pixel measurement to a distance in mm. This calculation is shown by Equations 3 and 4. This method of conversion is what we will use for our blob detection when an AR tag is not used. For our AR tag detection, we will use a different method to get a conversion factor utilizing the size of the AR tag. Figure 15b shows the dimensions of the AR tag in pixels. Equation 5 is used to calculate the distance in pixels of each side of the AR tag. The average side length in pixels is calculated using Equation 6. The known length of the AR tag, L, in meters is used with the average side length Δl_{avg} to get the conversion factor s as detailed by Equation 7. This conversion factor converts pixel measurements in the image plane to distance measurements on the ground in meters.



(a) **(b)**

Figure 15: a) Image Plane: Focal Length and FOV b) AR Tag Dimensions

$$s_x = \frac{f \tan(HFOV/2)}{c_x} \tag{3}$$

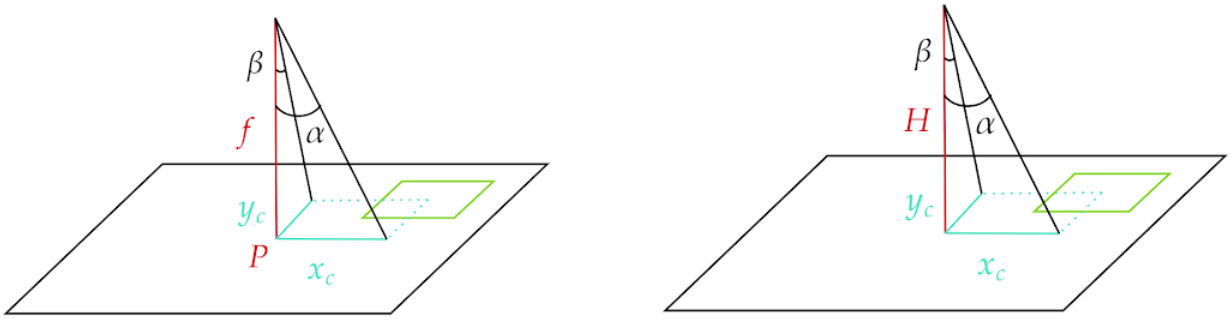
$$s_y = \frac{f \tan(VFOV/2)}{c_y} \tag{4}$$

$$\Delta l_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{5}$$

$$\Delta l_{\text{avg}} = \frac{1}{4} \sum_{i=1}^{4} \Delta l_i \tag{6}$$

$$s = \frac{L}{\Delta l_{\text{avg}}} \tag{7}$$

Using the relative pixel coordinates from the principal point to the detected centroid and the conversion factors, we can calculate two pointing angles. Figure 16 shows the two angles α and β which are the two pointing angles along the x and y axis respectively. For the case with no AR tags, the angles are calculated using a simple trigonometric relationship between the focal length and the relative coordinates as shown by Equations 8 and 9. These equations utilize the relative measurements, which are converted from pixels to millimeters, and the focal length of the camera. For AR tags, Equations 10 and 11 are applied. The relative measurements, which are converted from pixels to meters, and the known height of the drone H are used to calculate the same pointing angles α and β.



(a) **(b)**

Figure 16: a) Image Plane: Pointing Angles b) AR Tag Pointing Angles

$$\alpha = \arctan\left(\frac{x_c s_x}{f}\right) \tag{8}$$

$$\beta = \arctan\left(\frac{y_c s_y}{f}\right) \tag{9}$$

$$\alpha = \arctan\left(\frac{x_c s}{H}\right) \tag{10}$$

$$\beta = \arctan\left(\frac{y_c s}{H}\right) \tag{11}$$

Once the α and β pointing angles are calculated, the next step is to account for the roll and pitch of the drone. Our current design will have both the primary and secondary sensors oriented such that the camera is directed along the drone's z-axis. For the majority of the mission, when the drone takes data, it will be steady with the z-axis pointed directly along the inertial U-axis in the ENU frame. However, there will be perturbations when taking data, and for our orbiting flight path, there will need to be roll and pitch angles to achieve the desired path. As such, we will account for these angles when determining the relative position of the RGV from the drone's position. Figure 17 shows the two pointing angles with a given roll (ϕ) and pitch (θ) of the drone. The principal point projection is shown in red by point P and the height of the drone is shown by H. To determine the relative position of the RGV from the drone, the two coordinates x_r and y_r need to be calculated. By using the pointing angles, as calculated previously, and the roll and pitch of the drone as provided by the onboard IMU, the relative coordinates are calculated using Equations 12 and 13.

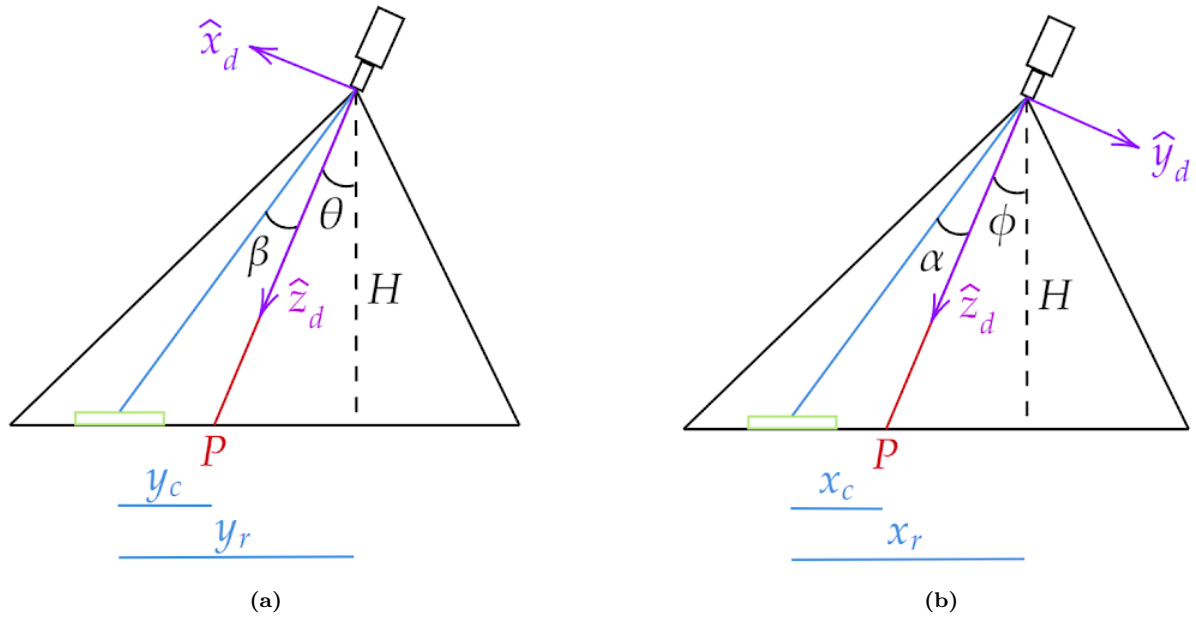
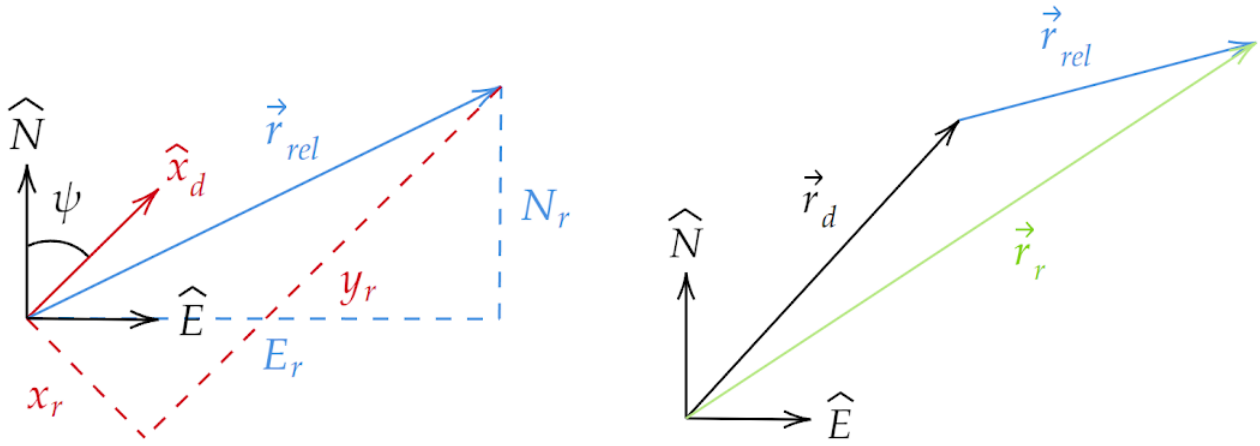


Figure 17: a) Relative measurements with known pitch angle b) Relative measurements with known roll angle

$$x_r = H \tan\left(\alpha + \phi\right) \tag{12}$$

$$y_r = H \tan\left(\beta + \theta\right) \tag{13}$$

The relative coordinates x_r and y_r represent the relative location of the detected RGV in the drone frame. The IMU measurements from the drone will provide the yaw angle ψ which is the heading from north. This angle can be used to determine the relative coordinates E_r and N_r in the ENU frame. This coordinate transformation is shown in Figure 18a and utilizes a rotation matrix about the drone's z-axis as shown by Equation 14. Once the relative measurements in the ENU frame are obtained, the final inertial location of the RGV is calculated using the known position of the drone $\vec{r}_d$ and the relative position $\vec{r}_{rel}$ as detailed by Equation 15.



(a) **(b)**

Figure 18: a) Rotation into the ENU frame b) Inertial calculation of RGV position

$$\vec{r}_{rel} = \begin{bmatrix} E_r \\ N_r \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{14}$$

$$\vec{r}_r = \vec{r}_d + \vec{r}_{rel} \tag{15}$$

3.2.2 Localization Model Analysis

To determine if our localization model will meet the accuracy requirements necessary for mission success, we simulated the coarse and fine localization phases using a conservative estimate of error and benchmarked our localization results. To setup the simulation applicable for Monte Carlo randomization for benchmarking, we made the following assumptions, where all random values are Gaussian:

Simulation Component	Assumptions
Drone Flight Path	Randomly chosen control from set of *forward, hover,* and *circle.* Speed of maneuvers chosen as a random constant velocity between 0-5 m/s.
RGV Path	RGV always is stationary for the duration of the simulation.
Camera Model	Induced detection failure rate of 25% to simulate inconsistencies in blob detection and AR tag detection. No distortion accounted for in sensing and detection.

Table 8: Localization Analysis Assumptions

With these assumptions in place, the drone had a probability of 75% to take a measurement of the RGV's position for each timestep through our localization model described in Section 3.2.1. This probability was chosen based on our AR tag detection program prototyping results. A timestep was defined by the frequency of image capture from the camera, so a camera that records at **30 frames per second** (fps) had a timestep of 1/30 seconds. Our simulation did not use computer vision, so we assumed the centroid position was already known if the drone successfully took a measurement. Error was propagated through our localization model through Eq. (12) & Eq. (13) as follows:

$$x_r = (H + \epsilon_{\text{gps}}) \tan(\alpha + \phi + \epsilon_{\text{att.}}) \tag{16}$$

$$y_r = (H + \epsilon_{\text{gps}}) \tan(\beta + \theta + \epsilon_{\text{att.}}) \tag{17}$$

Using the datasheet stated errors for our *Here2* GPS module and our *Cube Black* flight controller, the errors for altitude and attitude were set as such: $\epsilon_{\text{gps}} = \pm$ 8.2 ft and $\epsilon_{\text{att.}} = \pm\ 2^\circ$

After a full simulation, a vector of all the measured locations from each frame was stored in a stacked column vector, denoted y_k. With this data, we could now use non-linear least squares (NLS) to analyze if our measurements can give an estimated location of the RGV within requirements.

We can represent the measured vector y_k as a vector of the true RGV state, $x_{\text{RGV}}(x_0)$ and $y_{\text{RGV}}(y_0)$, and the Gaussian error, $v_{x,k}$ and $v_{y,k}$. To compare our measurements when we do not know the true position of the RGV, we define $h_k(X_0)$ in Eq. (18), which is the RGV's propagated state through time with an initial guess of the RGV's state, X_0. By defining a cost function $J_{NLS}(\vec{y}_k, X_0)$ in Eq. (19), we can compare our initial guess to the measured values. Our goal is to minimize the cost function by finding the best initial guess of the RGV's state, as defined in Eq. (20). By using the Jacobian Eq. (21), which was calculated numerically in this simulation, we can define a step of adjustment towards a new and better initial prediction using Eq. (22). By iterating through this guessing process stated in Eq. (23) using the Gauss-Newton method, our cost function will reach a local minimum and cease iteration once our next guess is less than a specified ϵ tolerance better than the previous iteration.

$$h_k(X_0, v_k) = \begin{bmatrix} x_{\text{RGV}}(x_0) + v_{x,k} \\ y_{\text{RGV}}(y_0) + v_{y,k} \end{bmatrix} \tag{18}$$

$$J_{NLS}(\vec{y}_k, X_0) = ||\vec{y}_k - \vec{h}_k(X_0)||^2 \tag{19}$$

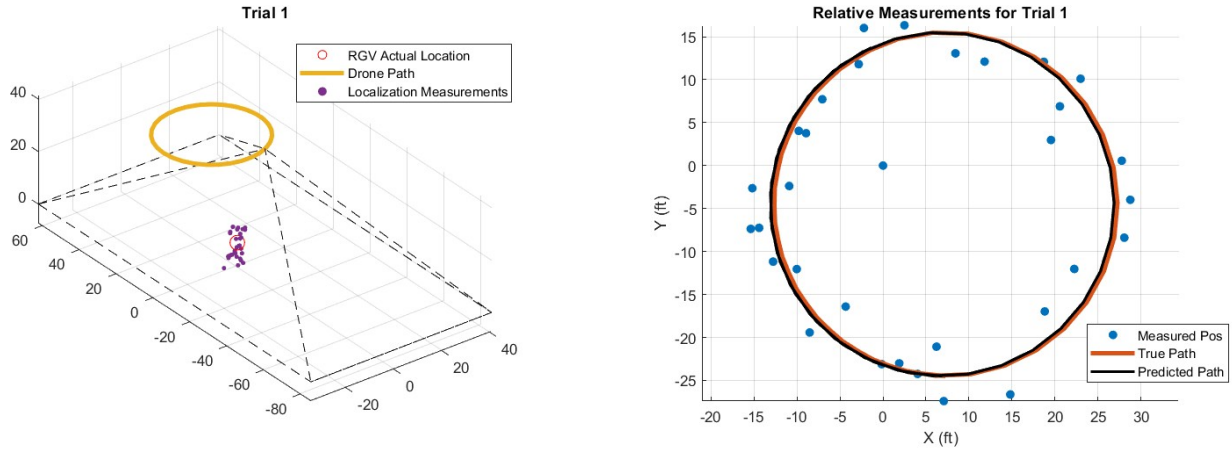
$$\hat{X_0} = \arg\min[J_{NLS}(\vec{y}_k, X_0)] \tag{20}$$

$$H(\hat{X_0}) = \frac{\partial \vec{h}_k}{\partial X_0}|_{\hat{X}_0} \tag{21}$$

$$\begin{aligned} \Delta\hat{X} =& (H^T(\hat{X}_{0,\text{old}})H(\hat{X}_{0,\text{old}}))^{-1} \\ & \times (H^T\hat{X}_{0,\text{old}})[\vec{y}_k - \vec{h}_k(\hat{X}_{0,\text{old}})] \end{aligned} \tag{22}$$

$$\hat{X}_{0,\text{new}} = \hat{X}_{0,\text{old}} + \Delta\hat{X} \tag{23}$$

An example of a full simulation can be seen in Fig. 19a. This figure shows the drone taking a circular path where the dashed black lines are the camera's projected field of vision. The purple dots are where the drone measured the RGV at a given timestep. After using NLS to make our best guess, we compared the predicted state to the actual state. Fig. 19b shows the RGV guess and true state from the drone's relative frame.



(a) **(b)**

Figure 19: a) Example of a simulated flight path and camera sensing area b) Measured relative path of the RGV compared to the actual RGV relative path

By running many iterations of this simulation with Monte-Carlo randomization described in Table 8, we could get a relevant benchmark of the accuracy of our localization method. We found that the most relevant design factor to take into consideration was the frames per second of the camera. Our choice of FPS directly impacted the number of measurements that could be made over a fixed simulation time of ten seconds. Fig. 20 shows the results of running 25 simulations per selected FPS.

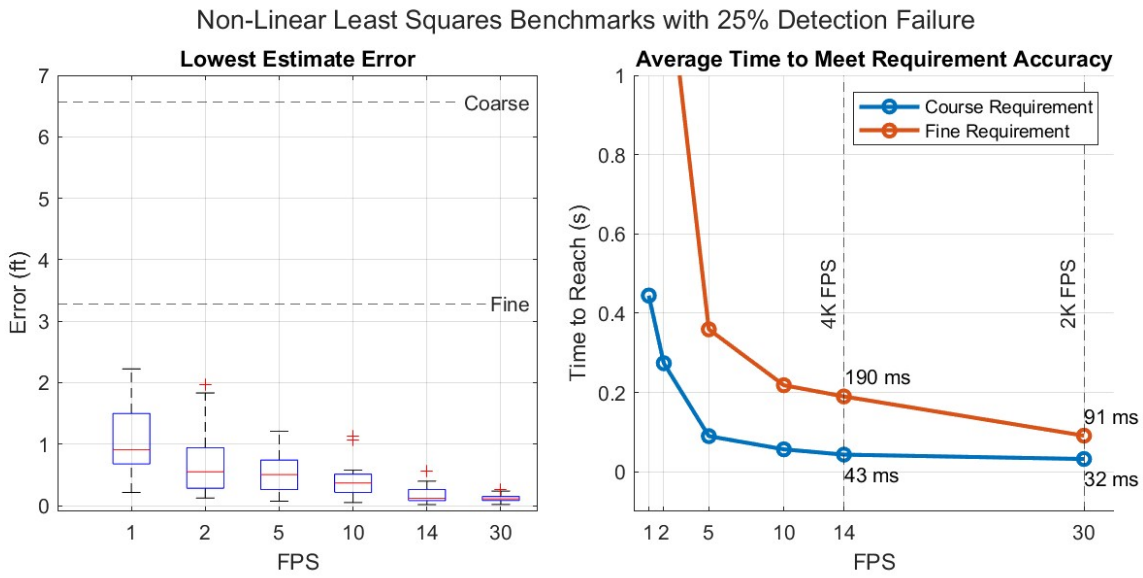


Figure 20: Benchmarking Results of Localization Model

Important findings that we noticed from our benchmarking results are as follows:

- Even with a staggeringly low FPS, our localization model still can reach coarse and fine error requirements (2 m and 1 m respectively). Not a single simulation localized with an accuracy greater than the fine requirement, satisfying 2DRMS requirements.
- The time it took to satisfy accuracy requirements decreased with a higher FPS. Our camera can localize a stationary RGV in under 200 ms at 14-30 fps, which is important because at this rate we can assume a real RGV traveling at its max speed of 2 m/s could not travel a significant enough distance for our localization model to lose accuracy. This result informed that our camera choice should at minimum capture at 14 FPS. For our assumptions to hold.

3.2.3 Localization Model Demonstration

To show that this localization model will work for our mission, we conducted a test with sample drone footage provided by Professor Ahmed's PhD student, Hunter Ray. The footage was collected by a DJI Mavic 3 drone with a 20-megapixel CMOS camera. The drone also had an onboard GPS module which allowed us to simulate all of the inputs to our localization model. For this test, we placed an ArUco tag on the RGV and used our AR tag detection computer vision algorithm to acquire the relative position of the RGV. We then used the GPS position data of the drone to get inertial estimates of the RGVs in the ENU frame. Figure 21 shows the detection of the AR tag. The green vector is the drone x-axis, black is North, and white is East. The light blue lines represent the relative position measurements x_r and y_r.

Figure 21: AR Tag Detection. Drone Footage courtesy of Hunter Ray.

Using our localization method, we were able to process the image and GPS data to get inertial estimates of the RGVs. Figure 22 shows the path of the drone in the ENU frame with RGV-A measurements in red and RGV-B in blue. Figure 23 shows a zoomed-in view of the inertial measurements of both RGVs. For this test, both RGVs were stationary and were marked with AR tags which allowed us to localize and uniquely identify each RGV during the mission. The origin of the ENU frame was selected to be the first location of the drone when the footage began. Our final estimates in the inertial frame were: RGV-A (-12.44 m, -9.78 m) and RGV-B (-16.76 m, -1.22 m). The 2DRMS errors in these estimates were 0.32 m for RGV-A and 0.23 m for RGV-B. Both of these errors are less than the required 1 meter of accuracy for fine localization. This test shows that we will be able to meet the requirements as listed in Table 6 and demonstrates the feasibility of our algorithms.

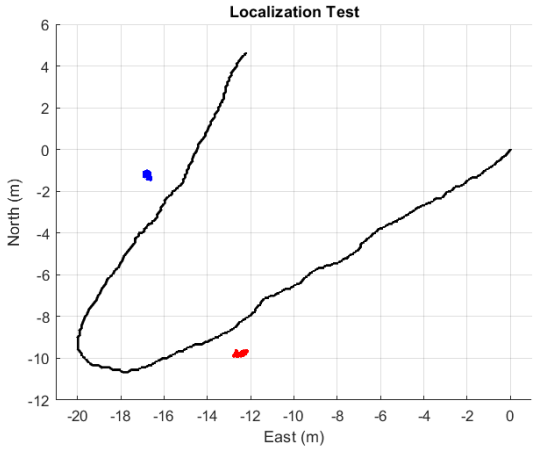
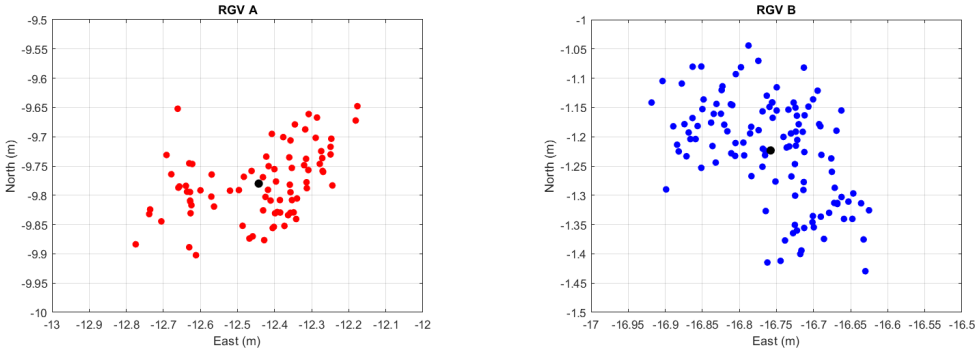


Figure 22: Drone Path



(a) **(b)**

Figure 23: a) RGV-A inertial measurements b) RGV-B inertial measurements

3.2.4 Localization Model Uncertainty

While this model uses reasonable assumptions, there is still uncertainty associated with our results. Table 9 lists the assumptions and the differences we expect in our actual flight versus our model. In our model, we assumed equipment noise extremes with the GPS and attitude error. In reality, this error will be different which may affect our estimation algorithms. However, we did assume large values so we expect the actual error to be less in reality. The camera model we developed assumed a perfect pinhole camera with no distortion. In reality, our cameras will have distortion near the boundary of the image. This can cause errors in the localization method and will require calibration of the camera prior to use. We also assumed 25% detection failure of our computer vision, however, we did not account for false positives. If the computer vision algorithm marks a rock as a location of an RGV we will have outliers in the data set that we do not account for in this model. Lastly, we assumed the RGV was stationary while in reality, they will be driving around. We showed with Figure 20 that we can localize the RGV in a very small amount of time which reinforces our assumption of stationary. However, our dynamic model did not account for the motion of the RGV and will thus need to change slightly in our actual flight.

Assumption	Reality	Effect
Equipment Noise Extremes	Ranges may be higher or lower	Different error with marginal recorded effect on accuracy
Perfect camera with no distortion	Cameras, especially wide angle, will have distortion near the boundary	Increased error in localization, required camera calibration
Accurate Computer Vision Output	Blob detection may detect objects that are not RGVs	Outliers in the data set affect the localization estimate
Stationary RGV	RGV can move during localization	Requirements met quickly, can approximate as stationary

Table 9: Localization Model Uncertainty

3.3 Autonomy Simulation

Our mission must be accomplished using a nearly fully autonomous system, capable of intelligently planning where and how to move throughout the environment to maximize data collection. To ensure our mission will be capable of completing the required phases for autonomous flight, we modeled the mission and motion planner software in MATLAB. The following table highlights the main driving requirements for this model:

ID	Requirement	Parent Req	Child Req	Verification Method
AUT 4	UAS shall transition between mission phases.	FR 1	AUT 4.1-4.6	D
AUT 6	UAS shall autonomously execute maneuvers to allow trailing, staying within 40 ft ground distance of acquired RGV ground targets.	FR 1	AUT 6.1	D
AUT 7	UAS shall autonomously execute maneuvers that permit data collection for coarse RGV target localization.	FR 1	AUT 7.1	D
AUT 8	UAS shall autonomously execute maneuvers that permit data collection for fine RGV target localization.	FR 1	AUT 8.1	D
AUT 9	UAS shall autonomously execute maneuvers that permit data collection for joint RGV target localization.	FR 1	AUT 9.1	D

Table 10: Autonomy Simulation Driving Requirements

3.3.1 MATLAB Simulation

The MATLAB simulation begins by randomly spawning two point mass RGVs in a 150x150 ft^2 environment. These RGVs start the simulation with a random velocity between 0 and 2 m/s (defined as max RGV speed). Then, as simulation time increases, the RGVs have a 5% chance to stop motion for 60 seconds, after which they will once again start moving with a random velocity in a random direction. Through this, we can emulate the RGV motion we will experience in the mission. The UAS system spawns at (0,0,0) ft in the environment, where it is initialized with a phase of *Takeoff*. The UAS motion, as defined by Table 2, was simulated using point mass kinematics moving towards commanded waypoints with the ode45 solver. To emulate the UAS's computer vision, a *Camera Vision* rectangular box was defined based on the UAS's current height and orientation using the following relationship:

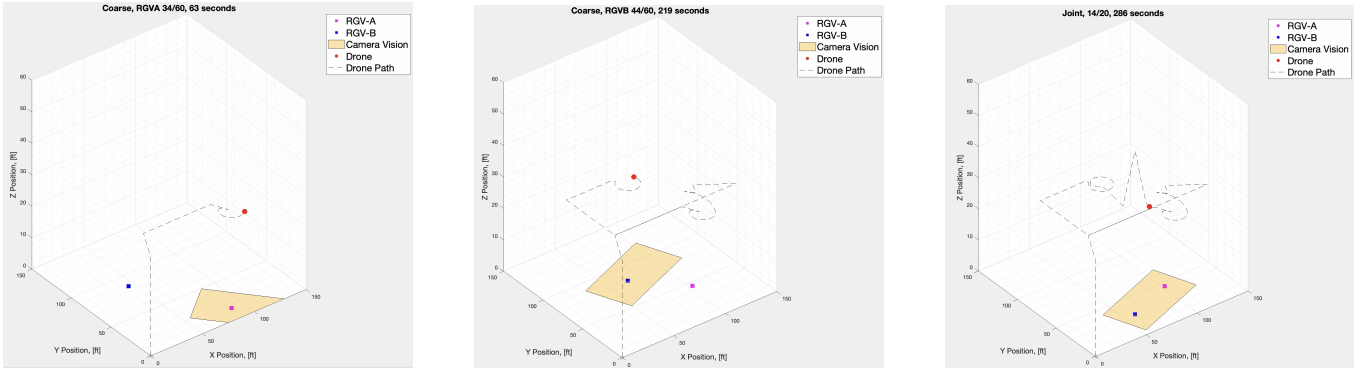
$$\text{wideMag} = \tan\left(\frac{\text{wideFOV}}{2}\right) * \text{UAS}_z \tag{24}$$

$$\text{narrowMag} = \tan\left(\frac{\text{narrowFOV}}{2}\right) * \text{UAS}_z \tag{25}$$

$$\text{verticesOfBox} = \begin{bmatrix} x_1, & y_1 \\ x_2, & y_2 \\ x_3, & y_3 \\ x_4, & y_4 \end{bmatrix} = \begin{bmatrix} \text{-wideMag}, & \text{-narrowMag} \\ \text{wideMag}, & -\text{narrowMag} \\ \text{wideMag}, & \text{narrowMag} \\ \text{-wideMag}, & \text{narrowMag} \end{bmatrix} * \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} + (\text{UAS}_x, \text{UAS}_y) \tag{26}$$

Where the UAS has a current location (x, y, z), yaw angle ψ, and camera stats wideFOV and narrowFOV. The rectangle created by *verticesOfBox* can then be used to determine when an RGV is within the bounds of the box. In this way, we can emulate ideal computer vision by getting the RGV's inertial position when an RGV is within the FOV box.

The simulation runs in a while loop, incrementing environmental time at each step. The while loop only breaks when the environment time becomes greater than 8 minutes (as restricted by requirements) or the UAS reaches the *Complete* mission phase. The following figures are screenshots from the MATLAB simulation of the UAS autonomously planning through the required phases:



(a) **(b)** **(c)**

Figure 24: a) Coarse localization on RGV-A b) Coarse localization on RGV-B c) Joint localization on both RGVs

As seen in the figures above, the UAS, the red dot, is able to search the environment to find both RGVs and initiate the required orbiting coarse phase and fine phase. For joint localization, the flight path follows the functionality described in the Motion Planner subsystem, Section 2.3.4, where the UAS moves to the maximum z height and the (x, y) midpoint of the two RGVs. Once both RGVs are in the camera's FOV, the UAS elevation and orientation is optimized to see both RGVs while being as low as possible, as seen in the elevation drop in (c).

3.3.2 Incorperating Error into Simulation

While the initial MATLAB simulation was able to successfully complete the mission, this was done using ideal computer vision. Every time an RGV was within the field of view the UAS was able to acquire its position. However, this is not a valid assumption for our mission; we know through testing our computer vision algorithms that there will be error in detecting the RGVs. Specifically, we quantify two types of computer vision error, *detection failure* and *false positives*. Detection failure is when the RGV is within the camera's field of view, yet, the computer vision algorithm fails to detect the RGVs location for that given camera frame. A false positive is when the computer vision algorithm detects that a given location is an RGV when it is not. To mitigate the effects of these errors, we adjusted the original high-level mission planner structure, Fig. 10, to include a *computer vision smoother* and *phase smoother*:

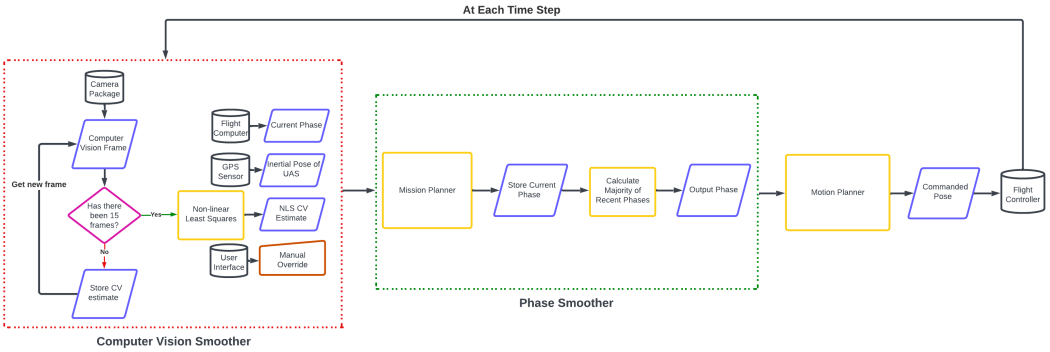


Figure 25: Adjusted High Level Mission and Motion Planner

The computer vision smoother works by not running the mission planner at every camera frame: instead, it takes the non-linear least squares estimation from the prior 15 camera frames. This allows the computer vision algorithm to experience detection failure for a certain amount of frames, but the overall output to the mission planner is a clean estimation of RGV position. We have proved that we can do non-linear least squares estimation with detection failure in the prior mentioned localization model, specifically Fig. 20.

The phase smoother works by looking at a certain number of past phase outputs and determining the majority output. For example, if 3 of the last 5 phase calculations say an RGV is detected and we should switch to the trail phase, the trail phase will be officially switched to. This method helps prevent false positives from impacting the mission if we assume an optimistic false positive error rate, where the rate of false positive error is less than 50%. If so, it is statistically unlikely that false positive detections will ever switch the mission planner to a phase it shouldn't switch to, as we take the majority output.

We were able to implement the above software structure into the MATLAB simulation and verify that these methods are able to prevent computer vision failure from affecting mission success. To verify this, we benchmarked the simulation by varying the percentage chance that a single computer vision frame experiences detection failure. For this benchmark test, we assumed a 30 FPS camera and a mission planner that runs every 0.5 seconds; this means the computer vision smoother input into the mission planner looks at the past 15 frames. The results are shown below, where success is taken as the UAS moving through all required phases in under 8 minutes:

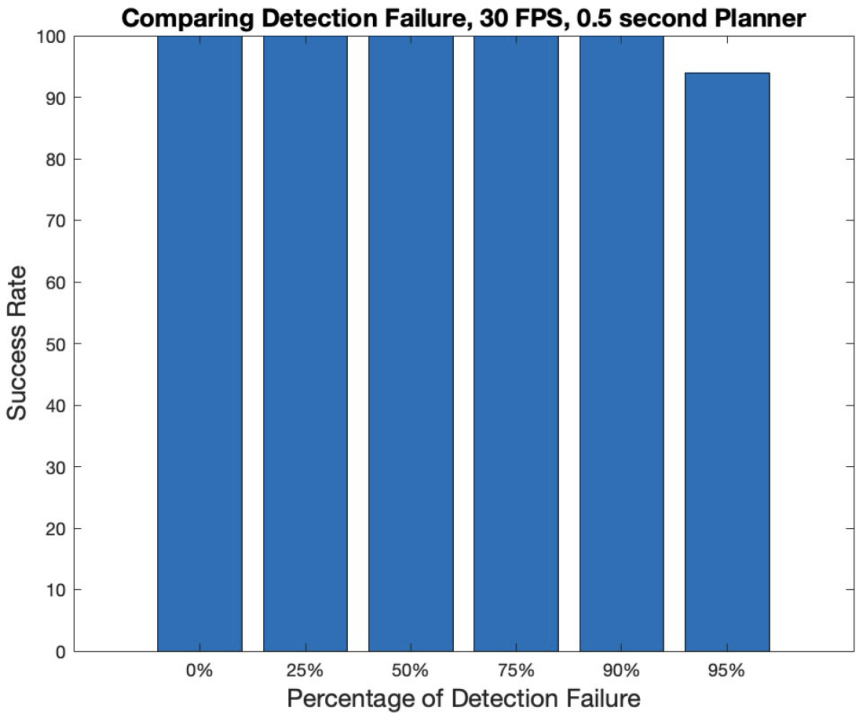


Figure 26: Benchmarking Success vs Detection Failure with a Computer Vision Smoother

As evident in the results above, the computer vision smoother was a massive success in mitigating the consequence of CV detection failure. This gives us great confidence that when we deploy this software algorithm into our actual system with noisy data, we will be able to take into account output error and uncertainty from the camera package.

4 Spring Tests, Schedule, Budget

Authors: Brendan Bradley, Carson Kohlbrenner, Nyah Baltazar

4.1 Spring Testing

Test	Description	Location	Requirements
Endurance/Battery Test	Determine battery life under varying conditions	Aero Building	UAS 2, UAS 3
Software in the Loop Test	Ensure our software works	Aero Building	SNS 1.2
Computer Vision Test	Calibrate for distortion, and verify CV algorithms	Aero Building	AUT 7, 8, 9
Wi-Fi Communications Test	Ensure Wi-Fi communications function at max range	East Campus	UAS 7.2, SNS 5.1
Radio Communications Test	Ensure radio communications function at max range	East Campus	UAS 7.2
Hardware in the Loop Test	Ensure that our hardware can handle computations	Aero Building	UAS 2.2
Manual Control Test	Demonstrate safe flight and adjust controls	ASPEN Lab	FR 2
UI Functionality Test	Adjust for ease of use and data display	Model Airport	FR 6, AUT 11.1, 11.4, SNS 1
Searching Test	Prove we can autonomously search for RGVs	Model Airport	AUT 4.1, 5.1, 2
Trailing Test	Prove we can trail RGVs autonomously	Model Airport	AUT 4.2, 6
Coarse Localization Test	Prove we can execute maneuvers to gather coarse data	Model Airport	AUT 4.3, 4.4, 7
Fine Localization Test	Prove we can execute maneuvers to gather fine data	Model Airport	AUT 4.5, 8
Joint Localization Test	Prove we can execute maneuvers to gather joint data	Model Airport	AUT 4.6, 9
Complete Test	Full run through of the mission	Model Airport	All Requirements

Table 11: Spring Testing

Table 11 lists the tests that we will complete in the upcoming semester. Our key subsystem test is our computer vision test, this is where we will be calibrating our cameras for distortion and testing our algorithms with our own cameras and data to determine exactly what it takes to localize within 1 meter for both AR tag and blob detection. We will perform this test at the Aero building using a checkerboard to account for distortion and testing our cameras with varying distances from the AR tags, different sized AR tags, and at varying angles using the different balconies. It will verify the requirements AUT 7, 8, & 9, because those requirements are the parent requirements for all of the requirements for actually gathering data using our sensors. Moving on to our key system tests our most critical tests are the complete test, the UI functionality test, and the manual control test. The complete test is our most critical in which we will do a full run-through of the mission and ensure that everything is working in concert. We will be performing it at the model airport here in Boulder, however, if we cannot do that for some reason we have South Campus as a backup. By running through the entire mission we will verify every requirement. For our UI functionality test, we will have the aircraft takeoff and begin the mission through the UI and we will have the UAS land through the UI as well. We will also make sure that the aircraft state and position are relayed to the UI so that we meet requirements. We will be using the results of this test to adjust for ease of use if necessary. This test will also occur at the model airport. For our manual control test we will make sure we can switch between manual and autonomous control, as well as making sure our drone can safely fly and potentially adjusting controls if needed, this is to make sure that we can meet our safety requirements. This test will occur indoors, most likely in the ASPEN Lab. When testing our biggest safety risk is crashing and either causing property damage, a fire, or injuring somebody. To mitigate these risks we are doing extensive analysis before we can fly for the first time, we will have a safety check before each flight, we will not approach the UAS while the rotors are spinning, and we will have a fire extinguisher ready when testing.

4.2 Spring Schedule

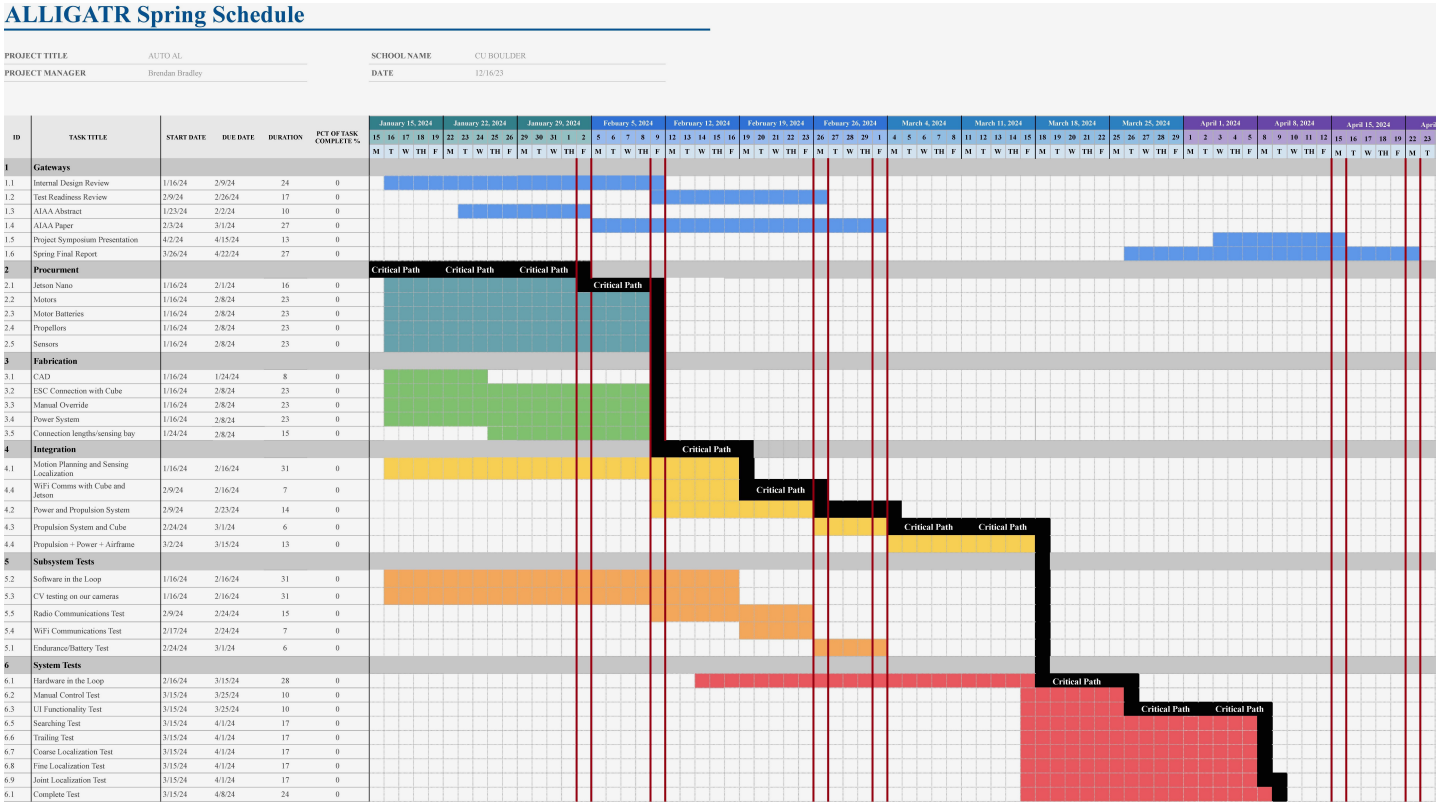


Figure 27: Spring Semester Gantt Chart

For our Gantt Chart, the critical path is shown by the black boxes and is mainly affected by procurement, the integration of the full UAS system, and finally our system tests. The path took this route because we can only begin to integrate the parts and build our system after we obtain the necessary parts. As parts come in, fabrication of the different subsystems, specifically the power system and sensing bay, can happen concurrently. Since most of our system tests require flying, they can only be performed after the entire system has been integrated and after the necessary flight training. When creating the chart, margins were mainly added to procurement and fabrication. This is because the full UAS integration is expected to be the most difficult and we wanted to ensure that we could perform all the testing we need to do. Because we can obtain most items in less than a week, this will accelerate our fabrication timeline and we will likely be able to start the integration process earlier than what is outlined. We also tried to have the end dates for tasks align with when the milestones are due, in order to have as much information as possible to present.

4.3 Spring Budget

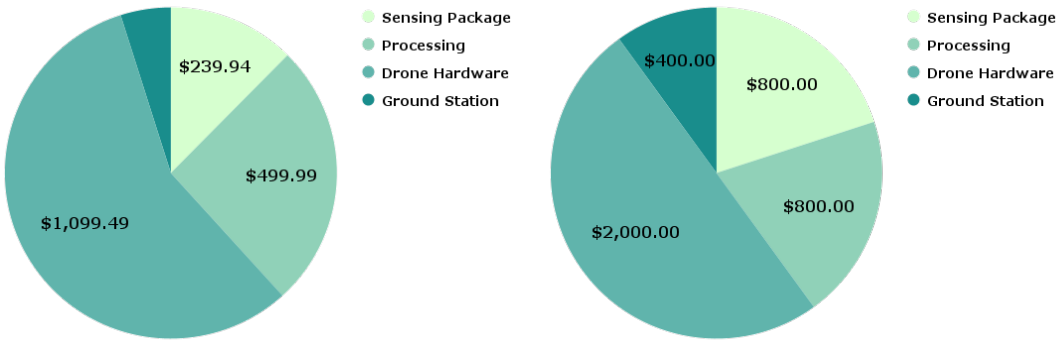
Component	Quantity	Price	Lead Time
Motors (Navigator 340 KV)	4	$399.96	< 1 Week
Flight Controller (Cube Black)	1	$330.00	< 1 Week
Motor Batteries	2	$235.72	4-10 Days
Telemetry Comms	1	$226.00	< 1 Week
Rotor ESC (Air 40)	4	$159.96	< 1 Week
Drone Chassis (Tarot 650)	1	$158.88	< 1 Week
Processor (Jetson Nano)	1	$150.00	< 1 Week
GPS (Here 2)	1	$80.00	< 1 Week
Primary Camera (IMX708)	1	$59.98	< 1 Week
Manual Controller	1	$55.60	< 1 Week
Secondary Camera (IMX708)	1	$45.99	< 1 Week
Power Module Mauch	1	$40.00	< 1 Week
WiFi Router	1	$39.00	< 1 Week
WiFi Module	1	$21.00	< 1 Week
Propellers	4	$28.11	< 1 Week
Power Module Battery	1	$27.98	< 1 Week
Power Module Case	1	$21.00	1-2 Weeks
SD Card	1	$19.99	< 1 Week
TOTAL		$2,104.16	< 1 Week

Table 12: Total Drone Cost

Table 12 has a detailed breakdown of all the items needed for the current design of our drone. Over the course of the Fall semester, we have already procured parts through our initial $500 budget and the electronics shop. The Table 13 has all the parts we currently have, as well as the remaining purchasing cost of the drone for the spring semester. The only in-house manufacturing we will be doing is for the electronics housing that we will be 3D printing for an estimated **$16.10**.

Component	Quantity	Price
Motors (Navigator 340 KV)	1	$99.99
Flight Controller	1	$330.00
Telemetry Comms	1	$226.00
Rotor ESC (Air 40)	1	$39.99
Drone Chassis (Tarot 650)	1	$158.88
GPS (Here 2)	1	$80.00
Primary Camera (IMX708)	1	$59.98
Secondary Camera (IMX708)	1	$45.99
SD Card	1	$19.99
Procured Value		$835.82
Remaining Cost		$1,099.20

Table 13: Purchased Components



(a) **(b)**

Figure 28: a) Estimated Total Cost Margins b) Allocated Margins for Full Budget

5 Individual Report Contributions

5.1 Project Leads

Project Manager - Brendan Bradley:

- Handled scheduling and planning for the semester, to make sure we effectively used our time and were on track for milestones.
- Designed our testing plans for the spring semester
- Studied FAA and school regulations to ensure that we legally operate the aircraft and have multiple locations to fly.
- Rewrote requirements to make them testable and have the project be more feasible.

Systems Engineer - Nyah Baltazar:

- Requirements development throughout the semester and updating them to make them singular and testable.
- Helped author Project Purpose and CONOPS. Created Gantt Chart and contributed to scheduling and planning for the spring semester.
- Developed Full-System Functional Block Diagram to help interface different subsystems.
- Contributed to the aircraft electrical design by researching components and integration, and created a wiring diagram.

5.2 Sensing Team

Sensing Team Lead & CFO - Carson Kohlbrenner:

- Focused primarily on the sensing package, making sure it integrated well with entire drone. Effort was mainly spent on selecting and analyzing the error of the best hardware for the sensing package such as the processor, cameras, and interconnections with the drone.
- Conducted hardware testing with a Raspberry Pi 4B and IMX708 wide angle camera.
- Created the Non-Linear Least Squares Monte Carlo simulation and code.
- Kept track of the budget and made all purchases.

Primary Sensor Lead - Yarden Kelmann:

- Developed blob detection method and prototyped the computer vision algorithm with drone footage.
- Prototyped the sensing algorithm component of the sensor package.

Secondary Sensor Lead - Thomas Dunnington:

- Developed the AR Tag detection method and prototyped the computer vision with sample drone footage.
- Developed the localization method and wrote the corresponding section in the engineering model.
- Analyzed drone footage and GPS data to get RGV inertial estimates. Wrote about findings in the localization model demonstration section.

5.3 Autonomy Team

Autonomy Team Lead - Nolan Stevenson:

- Responsible for ensuring autonomy team efforts aligned and integrated with sensing and drone team efforts.
- Lead software developer for MATLAB simulation.
- Flight control protocol and software structure.

Mission Planning Lead - Lucia Witikko:

- Designed and wrote code for mission planning phases (search, trail, coarse, fine, and boundary control).
- Helped with aspects of mission and motion planning code as well as the MATLAB simulation.
- Wrote ConOps section and helped author purpose section.
- Created initial report configuration and maintained its organization.

Motion Planning Lead - Zane Vandivere:

- Designed and wrote code for search, trail, coarse, fine, and boundary control phase motion.
- Contributed to mission planner code and MATLAB simulation design

5.4 Drone Team

Drone Team Lead - Marcus Quintanilla:

- Contributed to design decisions including chassis, motors, propellers, batteries, and processor.
- Assembled drone chassis and ensured the chassis was balanced and the motor mounts were level.
- Contributed to mass and power budgets.
- Helped with research for communications, ground station user interface, and mission planning software.

Communications Lead - Andrew Kabos:

- Helped with design decisions regarding the layout of the power system to be used in the UAS.
- Helped with research for communications integration between the ground system and Mission Planner as well as Mission Planner and Autopilot.
- Helped with functional block diagrams.

CAD & Power Budget Lead - Nicholas Grant:

- Calculated aircraft power budget, ensuring sufficient battery capacity for vehicle to complete all mission objectives.
- Designed the aircraft's Computer Aided Design model, including modeling and placing component and component housing.
- Aided in design of aircraft electrical system, including component research and analysis.

Integration Lead - Blair Schulze:

- Assisted in the development of system functional block diagram and ground station interface package
- Gathered electrical and power requirement data for subsystem components
- Helped research and integrate communication lines between subsystem designs

Appendix 1: Requirement Flow-down

Table 14: Level 0 Requirements

ID	Requirement	Parent Req	Child Req	Verification Method
FR 1	UAS shall determine autonomously how to navigate the aircraft to collect localization data on a RGV.		AUT 4-10, UAS 3, UAS 5-6, UAS 9, SNS 2-7	A
FR 2	UAS shall respond to flight operator inputs provided through a manual controller.		AUT 12, UAS 4	D
FR 3	The aircraft shall take off from and land at a designated home base that is located at least 30 ft from the outer perimeter of the mission area.			D
FR 4	UAS shall perform all autonomous motion within the designated mission environment in accordance with the CU FAA certificate of authorization (2023-WSA-13256-COA).		AUT 1-3	A
FR 5	UAS shall provide its own power source and be untethered.		UAS 2	I
FR 6	All sensor data shall be logged on-board the UAS with necessary data displayed on the ground UI.		AUT 11, UAS 7-8, SNS 1	D

Table 15: Level 1 Requirements

ID	Requirement	Parent Req	Child Req	Verification Method
AUT 1	All UAS motion, besides takeoff and landing, shall stay within the operating environment	FR 4	AUT 1.1, 1.2	D
AUT 2	UAS shall perform all autonomous motion in a safe and controlled way.	FR 4		I
AUT 3	UAS shall autonomously recognize when the mission has been completed.	FR 1	AUT 3.1	D
AUT 4	UAS shall transition between mission phases.	FR 1	AUT 4.1, 4.2, 4.3, 4.4, 4.5, 4.6	D
AUT 5	UAS shall autonomously execute maneuvers to allow searching of the 150 x 150 ft environment.	FR 1	AUT 5.1, 5.2	D
AUT 6	UAS shall autonomously execute maneuvers to allow trailing, staying within 40 ft ground distance of acquired RGV ground targets.	FR 1	AUT 6.1	D
AUT 7	UAS shall autonomously execute maneuvers that permit data collection for coarse RGV target localization.	FR 1	AUT 7.1	D
AUT 8	UAS shall autonomously execute maneuvers that permit data collection for fine RGV target localization.	FR 1	AUT 8.1	D
AUT 9	UAS shall autonomously execute maneuvers that permit data collection for joint RGV target localization.	FR 1	AUT 9.1	D
AUT 10	UAS shall autonomously transmit the necessary information to the user via a ground UI.	FR 6	AUT 10.1, 10.2, 10.3	D/I
AUT 11	UAS shall be able to initiate and disengage the autonomous flight via a ground UI.	FR 2	AUT 11.1, 11.2, 11.3, 11.4	D
AUT 12	The system shall be fully Robot Operating System (ROS) compatible.	FR 2		I
UAS 1	UAS shall accommodate for the weight and size of the sensor payloads.	FR 5	UAS 1.1, 1.2	A
UAS 2	UAS shall provide necessary electrical power to all systems for a 15-minute flight.	FR 5	UAS 2.1, 2.2	T/A
UAS 3	UAS shall provide the necessary computational power for all operations during a 15-minute flight.	FR 1		A
UAS 4	UAS shall provide the framework for manual flight.	FR 4	UAS 4.1,	I
UAS 5	UAS shall provide the framework for autonomous flight.	FR 1		I
UAS 6	UAS shall have all necessary sensors to determine it's attitude and inertial position.	FR 1		T/A
UAS 7	UAS shall be capable of transferring and recieving all necessary data to and from the ground station UI.	FR 6	UAS 7.1, 7.2	T
UAS 8	UAS shall be capable of storing data that will be accessible at the end of the flight.	FR 6		T

Table 16: Level 1 Requirements Cont.

ID	Requirement	Parent Req	Child Req	Verification Method
UAS 9	UAS shall adhere to all relevant FAA and CU airworthiness regulations.	FR 4	UAS 9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 9.7	I/A
SNS 1	UAS shall display live aircraft states and desired RGV tracking artifacts to the operator at a ground control station through a UI.	FR 6	SNS 1.1, 1.2, 1.3	I/D
SNS 2	UAS shall use an RGB camera as its primary sensor.	FR 1	SNS 2.1	I
SNS 3	UAS shall have an onboard secondary sensor.	FR 1	SNS 3.1	I
SNS 4	UAS shall be able to identify both RGVs and differentiate between them	FR 1	SNS 4.1, 4.2, 4.3, 4.4	D
SNS 5	UAS shall maintain mostly uninterrupted visual detection of a selected RGV target within its RGB camera sensor field of view when following an RGV.	FR 1	SNS 5.1	D
SNS 6	UAS shall not communicate with RGVs.	FR 1	SNS 6.1, 6.2	I
SNS 7	UAS shall adhere to all FCC regulations.	FR 4	SNS 7.1	I/A

Table 17: Level 2 Requirements

ID	Requirement	Parent Req	Child Req	Verification Method
AUT 1.1	All motion planning shall include horizontal plane bounds on UAS movement so it stays within the 150 x 150 ft mission area.	AUT 1		A
AUT 1.2	All motion planning shall include vertical axis bounds on UAS movement so it stays between 30 and 60 ft.	AUT 1		A
AUT 3.1	UAS shall initiate the landing phase after mission completion.	AUT 3		D
AUT 4.1	UAS shall recognize when a height of 30-60 ft relative to the ground is reached during the takeoff phase and will begin searching for RGVs.	AUT 4		D
AUT 4.2	UAS shall recognize when an RGV has been acquired and will begin trailing the motion of the RGV.	AUT 4		D
AUT 4.3	UAS shall recognize when a tracked RGV stops its motion, characterized by a speed of 0 m/s, and will then begin acquisition of data for coarse localization.	AUT 4		D
AUT 4.4	UAS shall recognize when an RGV has gathered data for coarse localization for at least 60 seconds and will then begin acquisition of data for fine localization.	AUT 4		D
AUT 4.5	UAS shall recognize when all RGVs have had data gathered for fine localization for at least 30 seconds and will then begin acquisition of data for joint localization on all RGVs.	AUT 4		D
AUT 4.6	UAS shall recognize when all RGVs have had data gathered for joint localization for at least 20 seconds and will register as mission complete.	AUT 4		D
AUT 5.1	UAS shall perform a searching flight path maneuver to allow visual searching of the entire 150 x 150 ft environment.	AUT 5		D
AUT 5.2	UAS shall avoid to recognize any prior localized RGVs while searching, to ensure the UAS only acquires new RGVs.	AUT 5		D
AUT 6.1	UAS shall be able to follow RGV motion at speeds of up to 2 m/s.	AUT 6		D
AUT 7.1	UAS shall perform an orbiting flight path maneuver around the RGV to gather data for coarse localization within 2 meters	AUT 7		A
AUT 8.1	UAS shall perform an informative flight path maneuver to gather data for fine localization 1 meter	AUT 8		A
AUT 9.1	UAS shall perform an informative flight path maneuver to gather data for joint localization of RGVs within 1 meter	AUT 9		A
AUT 10.1	UI shall display a 2D map of the 150 ft x 150 ft environment.	AUT 10		I
AUT 10.2	UI shall display the current drone UAS location relative to the environment.	AUT 10		I
AUT 10.3	UI shall display an estimated state for acquired RGV location(s) at each stage of localization.	AUT 10		I
AUT 11.1	UI shall have the ability to initiate the UAS mission, beginning the startup sequence.	AUT 11		D
AUT 11.2	UI shall have the ability to activate a hover mode for the UAS, stopping all motion and mission operations until deactivated.	AUT 11		D
AUT 11.3	UI shall have the ability to abort the UAS mission, commanding the UAS to return to base and land.	AUT 12		D
AUT 11.4	UI shall be able to switch the UAS control method to a manual handheld controller, deactivating the autonomous controller.	AUT 11		D
UAS 1.1	UAS shall provide adequate structural support and impact absorption for hardware protection.	UAS 1		A

Table 18: Level 2 Requirements Cont.

ID	Requirement	Parent Req	Child Req	Verification Method
UAS 1.2	UAS shall be equipped with proper mounting solutions for sensors.	UAS 1		I
UAS 2.1	UAS shall have a battery that is capable of safely providing power to all systems throughout the entire flight.	UAS 2		A/D
UAS 2.2	UAS or ground station shall provide computational power that handles all computational requirements for drone operation.	UAS 2		I/A
UAS 4.1	Systems and motors shall be calibrated to provide specific movement given an input.	UAS 4		D
UAS 7.1	UI design shall allow for sending and recieving of signals to and from the UAS.	UAS 7		D
UAS 7.2	UAS shall be able to send and recieve all signals over any distance within the operational boundaries.	UAS 7		D
UAS 9.1	UAS shall weigh no more than 55 pounds.	UAS 9		I
UAS 9.2	UAS propellers must not interfere with eachother or the UAS itself. (AM 4.2.2.2 (b))	UAS 9		D
UAS 9.3	UAS transmitters and recievers shall have a diverse array of antennas. (AM 4.2.2.5 (b))	UAS 9		I
UAS 9.4	Ground station tansmitters and recievers shall have a diverse array of antennas. (AM 4.2.2.5 (b))	UAS 9		I
UAS 9.5	UAS shall be able to remain stable while in up to 10mph winds.	UAS 9		D/A
UAS 9.6	UAS shall be stable about all axes during operation.	UAS 9		A
SNS 1.1	All data collected during the mission shall be saved and made available for post processing.	SNS 1		I
SNS 1.2	Logged data shall be referenced to the aircraft's state at the time of recording.	SNS 1		I
SNS 1.3	An onboard communications link shall transmit telemetry and sensing data to the ground station during flight.	SNS 1		I
SNS 2.1	Data collected for coarse localization shall be acquired exclusively by the primary sensor's RGB sensing capability.	SNS 2		I
SNS 3.1	Data collected for fine localization shall be acquired exclusively by the seconday sensor.	SNS 3		I
SNS 4.1	Each RGV shall have data acquired for coarse localization during the course of the mission.	SNS 4		D
SNS 4.2	Each RGV shall have data acquired for joint localization during the course of the mission.	SNS 4		D
SNS 5.1	Primary sensor shall provide sensing/tracking data to the ground station during operation.	SNS 5		D
SNS 6.1	Any beacons mounted to an RGV shall be mounted before the mission and shall be removable after the mission.	SNS 6		I
SNS 6.2	Data collected for coarse localization shall enable localization in an inertial reference frame with a 2DRMS accuracy of no greater than 2 meters.	SNS 6	N/A	D
SNS 6.3	Data collected for fine localization shall enable localization in an inertial reference frame with a 2DRMS accuracy of no greater than 1 meter.	SNS 6	N/A	D
SNS 6.4	Data collected for joint localization shall enable localization of both RGVs in an inertial reference frame with a 2DRMS accuracy of no greater than 1 meter.	SNS 6	N/A	D
SNS 7.1	All radio frequency transmission shall have a frequency within a permissible bandwidth as defined by the FCC (2.4 or 5 GHz).	SNS 7		I

Appendix 2: Backup Images

5.5 Mission Phase Subfunctions

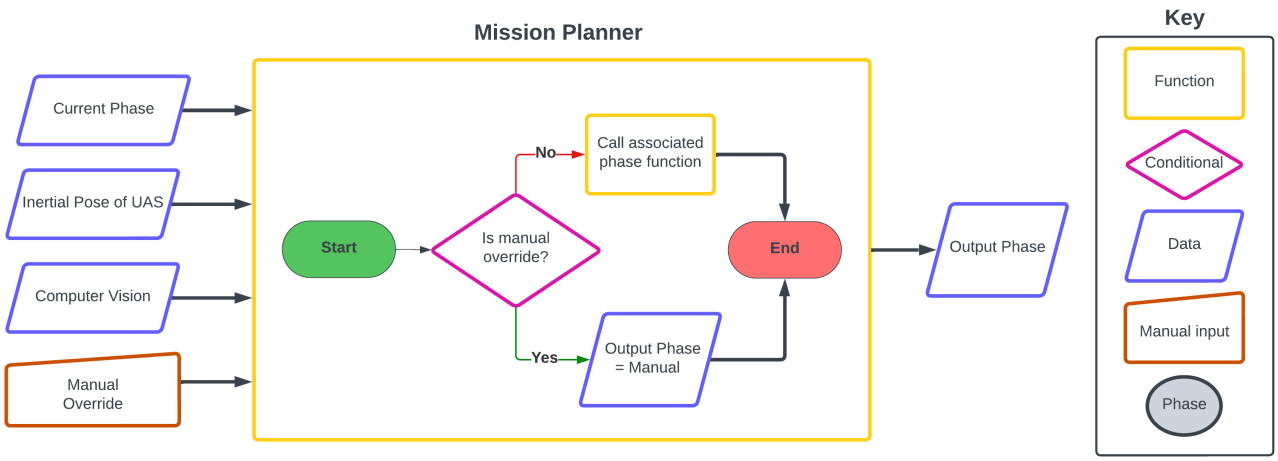


Figure 29: Mission Planner subfunction struture.

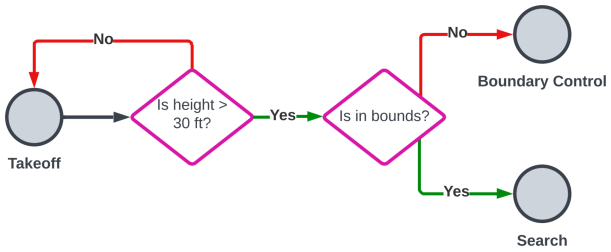


Figure 30: Takeoff phase conditionals.

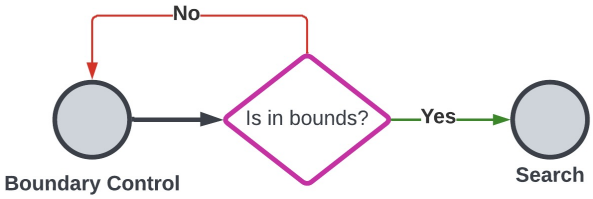


Figure 31: Boundary Control phase conditionals.

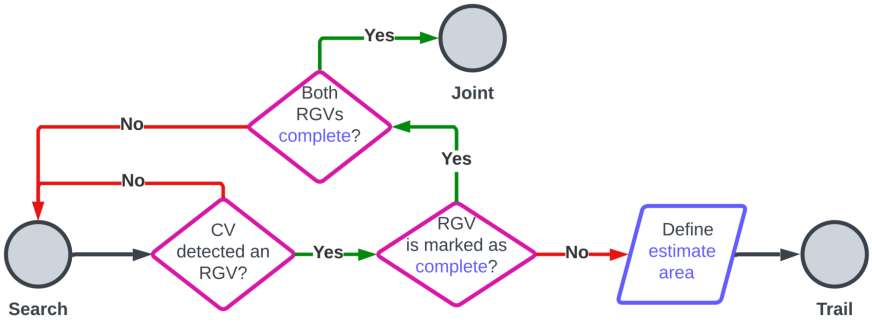


Figure 32: Search phase conditionals.

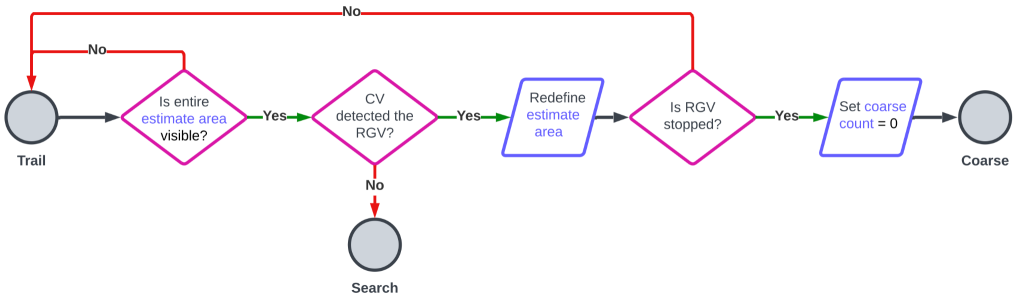


Figure 33: Trail phase conditionals.

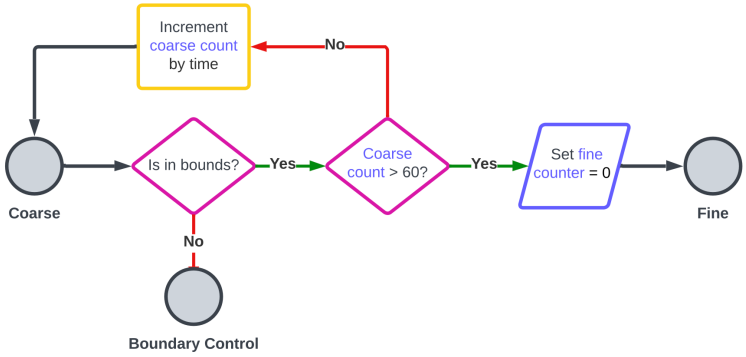


Figure 34: Coarse phase conditionals.

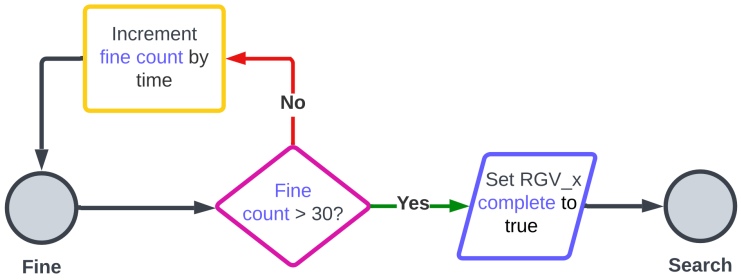


Figure 35: Fine phase conditionals.

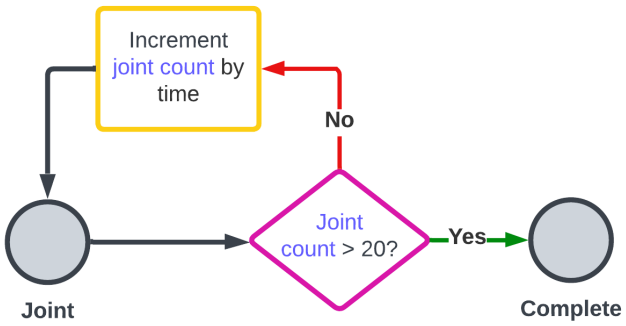


Figure 36: Joint phase conditionals.

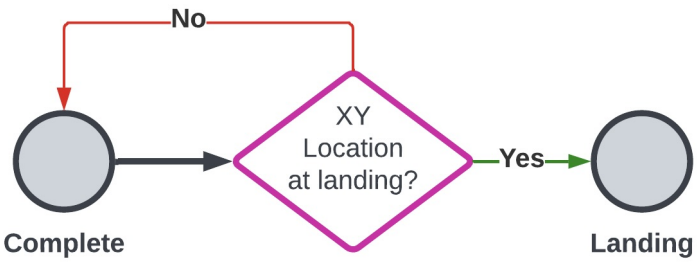


Figure 37: Complete phase conditionals.

Appendix 3: References

ALLIGATR - System Design and Requirement Proposal (SDRP)

Federal Aviation Administration - Recreational Flyers and Community-Based Organizations, https://www.faa.gov/uas/recreational_flyers

Professor Ahmed - RFP Autonomous Aerial Localization of Ground Vehicles for Assisted Navigation https://canvas.colorado.edu/courses/97383/files/66980210?wrap=1

S. Sanyal, S. Bhushan and K. Sivayazi, "Detection and Location Estimation of Object in Unmanned Aerial Vehicle using Single Camera and GPS," 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T), Raipur, India, 2020, pp. 73-78, https://doi.org/10.1109/ICPC2T48082.2020.9071439

University of Colorado Boulder - Unmanned Aerial Systems Airworthiness Certification Manual https://www.colorado.edu/isc/node/197/attachment