Autonomous Aerial Robot Localization and Target Detection and Manipulation in a GPS-denied Environment

Alex E. Bennett
University of Louisville
Alexander A. Rickert
University of Louisville
William B. Crawford
University of Louisville
Elijah D. Lamppin
University of Louisville
Devon D. Warman
University of Louisville

ABSTRACT

In this paper, a solution for Mission 7a of the 2017 International Aerial Robotics Competition is presented by Redbird Robotics of the University of Louisville. This competition poses a task that requires an autonomous aerial robot, without GPS, to herd 10 ground robots into a corral without colliding into moving obstacles or leaving the confines of the arena. Redbird Robotics' solution is to build a single quadcopter that utilizes multiple onboard cameras for ground robot localization, an optical flow module, and a 1-dimensional LIDAR unit for accurate position estimation. The flight of the quadcopter is controlled through an onboard NVIDIA Jetson TX2 running ROS. This computer handles communication with the chosen Pixhawk flight controller running the PX4 flight stack through MAVROS and the MAVLink protocol. Real-time monitoring of the quadcopter is maintained from a ground station running a ROS node that communicates with the aerial robot over a Wi-Fi network.

INTRODUCTION

Problem Statement

In Mission 7a of the International Aerial Robotics Competition (IARC), 10 ground robots and 4 moving obstacles are placed on a 20-meter by 20-meter surface with a grid spaced at 1-meter increments. The objective is to use an autonomous aerial robot to herd all of the ground robots across a predetermined boundary of the playing field within 10 minutes. Herding is achieved through contact with the top or front of the ground robot. The established penalties are such that the aerial robot may not strike the obstacles more than twice, may not leave the bounds of the playing field for more than 5 seconds, and each ground robot that leaves the bounds of the field incurs a deduction of points from the team's final score. The problem faced by teams is localization of the aerial robot while denied GPS and thus localization of the ground robots.

Conceptual Solution to Solve the Problem

The vehicle for the proposed solution is a custom designed and built X-configuration quadcopter. Given the large area of the field, repeated floor pattern, and lack of static objects for reference, the PX4Flow was chosen to localize the aerial robot using optical flow. The PX4Flow was chosen due to its documented accuracy and reliability [1]. Using the calculated location of the vehicle given by the flow module, 3 downward facing cameras attempt to map the ground and obstacle robot locations. Cascade and Blob Detection tools from the OpenCV library are used to identify targets on the ground. Many studies have been documented using these tools to successfully identify and track a variety of objects [2] [3].

All processing is handled onboard by an NVIDIA Jetson TX2 computer. The Jetson was chosen due to its immense processing power. It is outfitted with an HMP Dual Denver 2 and Quad ARM A57 dual-processor setup along with an NVIDIA Pascal GPU and 8 GB of LPDDR4 RAM [4]. The capability brought forth by the Jetson allowed for the implementation of more intensive algorithms and data processing that would not be possible on similarly marketed platforms. The onboard computer runs Robot Operation System (ROS). This allows for the integration of several custom software packages that range in responsibility and serve as the implementation of the proposed problem solution [5].

Figure 1 below outlines the system architecture of the proposed solution.

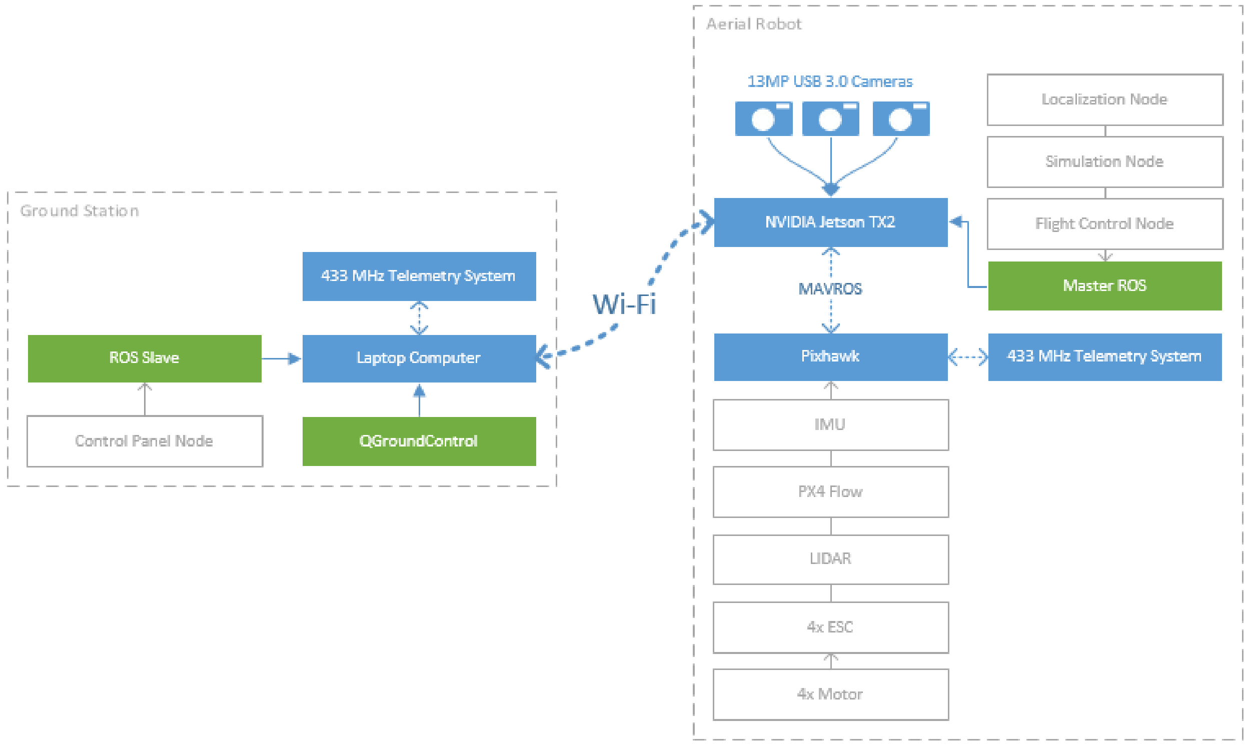

Figure 1. Block diagram of the system architecture.

Yearly Milestones

January 2017	• Redbird Robotics is created with intentions to compete in the IARC
	• Redbird Robotics recruits 10 new members
February 2017	• Conceptual design of quadcopter and flight logic established
March 2017	• Preliminary design of quadcopter is purchased and assembled
	• First manual flight of quadcopter
	• First autonomous position hold
April 2017	• Major design change to flight control
	• Final design of quadcopter established
May 2017	• Final design of quadcopter purchased and assembled
	• Positive detection and tracking of ground robots
	• Technical paper written
June 2017	• Technical paper submitted

AERIAL ROBOT

Propulsion and Lift System

The aerial robot is propelled using four MN4010-KV580 Navigator Series T-Motors with 12x3.8 propellers. With this configuration, up to 6.8kg of total thrust can be provided at maximum power while only pulling 18.5A per motor. The electronic speed controllers for the brushless motors are Air 40A 600Hz 2-6S ESCs from T-Motor. The craft is powered using a 14.8V 10C 10,000mAh battery. This battery provides sufficient power for the propulsion system and all onboard electronics. When combined, the propulsion and power systems will provide the aerial robot with a 15-minute flight time.

Custom Frame

For the competition, a custom quadcopter frame was designed and fabricated. The mounting plates for the body are made from 3.125mm thick carbon fiber sheets that were cut per required specifications for mounting the battery, flight controller, voltage regulator, onboard computer, cameras, and other necessary components. The arms and legs selected are carbon fiber tubes with outer diameters of 16mm. Carbon fiber was selected for its low density as well as its high isotropic material strength and stiffness properties. The length of the arms was determined by performing a stress analysis on the carbon fiber tubes at various lengths. The tubes act as a cantilever beam fixed to the carbon fiber sheets via aluminum tube clamps. The aluminum tube clamps were fastened to the top and middle carbon fiber sheets using M3 fasteners with lock nuts and lock washers. The same aluminum tube clamps are used to clamp carbon fiber motor mounts on the ends of the carbon fiber tubes. A thrust load of 1700g was applied to the end of the tubes based on the motor characteristics given by the manufacturer and in-house testing. With a maximum stress value of 32.8 MPa at a length of 430mm the stress on the arms was determined to be negligible. The length of the arms was finalized based on testing for a balance of agility and stability. An FEA was then performed on the carbon fiber sheets in order to ensure the rigidity of the frame.

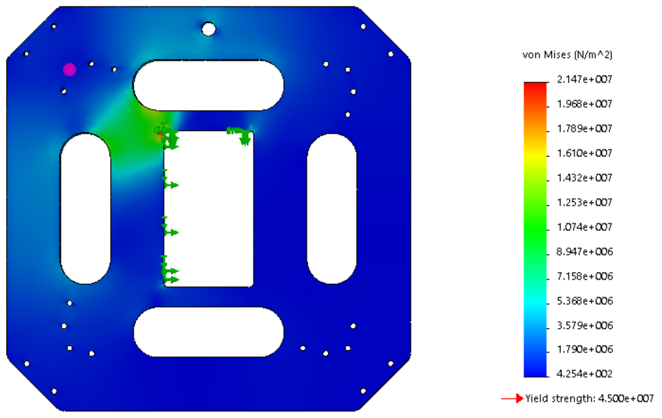

Figure 2. Von Mises stress plot on the top carbon fiber plate.

A 3D modeled representation of the custom quadcopter design is shown below in Figure 3 without many of the onboard electronics. Without these components, the craft is estimated to weigh 1.94kg. A triangular prism composed of five carbon fiber plates was used to mount three cameras used for computer vision. A leg with a wide circular foot will be used to interact with the top plate of the ground robots.

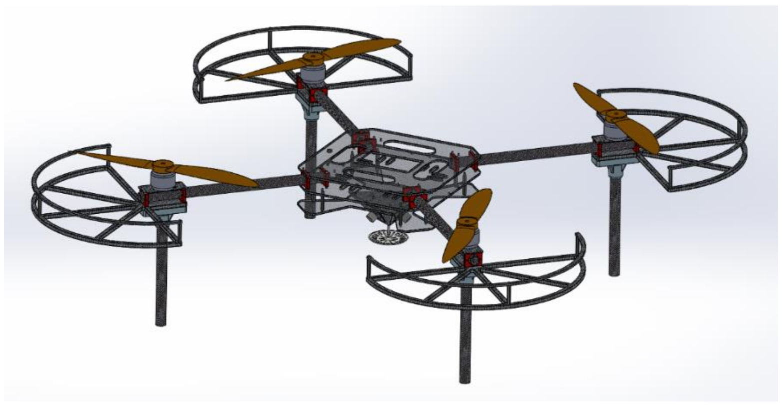

Figure 3. A model of the custom frame without flight electronics.

Guidance, Navigation, and Control

Stability Augmentation

Stability augmentation is performed by the onboard Pixhawk flight controller. The Pixhawk is equipped with several sensors including an inertial mass unit (IMU) that allow it to respond to vehicle instabilities and maintain a safe flight. The Pixhawk fuses all information it receives from its sensors through the use of multiple extended Kalman filters to create an accurate real-time model of the system. This information is then used by several proportional-integral-derivative controllers to stabilize the vehicle and control its movement in its environment.

Navigation

Navigation of the aerial robot is controlled through an independent ROS node referred to as the flight planning node that handles all decision making and trajectory planning using information gathered from other ROS nodes including onboard simulation and vehicle localization.

Upon beginning its flight, the aerial robot will initially takeoff to an altitude of 2.5m in an effort to avoid the obstacle ground robots until navigation at a lower altitude is required. The previously mentioned onboard simulation will constantly gather information about the environment from localization and assign a priority to all ground robots detected on the arena that relates to the likelihood of that robot scoring, i.e. moving across the goal line. This process is discussed more in the Target Identification section. The flight decision making happens simultaneously and runs in a loop for the duration of the flight. The constant goal of the flight planner is to identify which ground robot has the highest priority and then maneuver the aerial robot in a way that guides the ground robot to the desired location. This logic is shown in Figure 5 below.

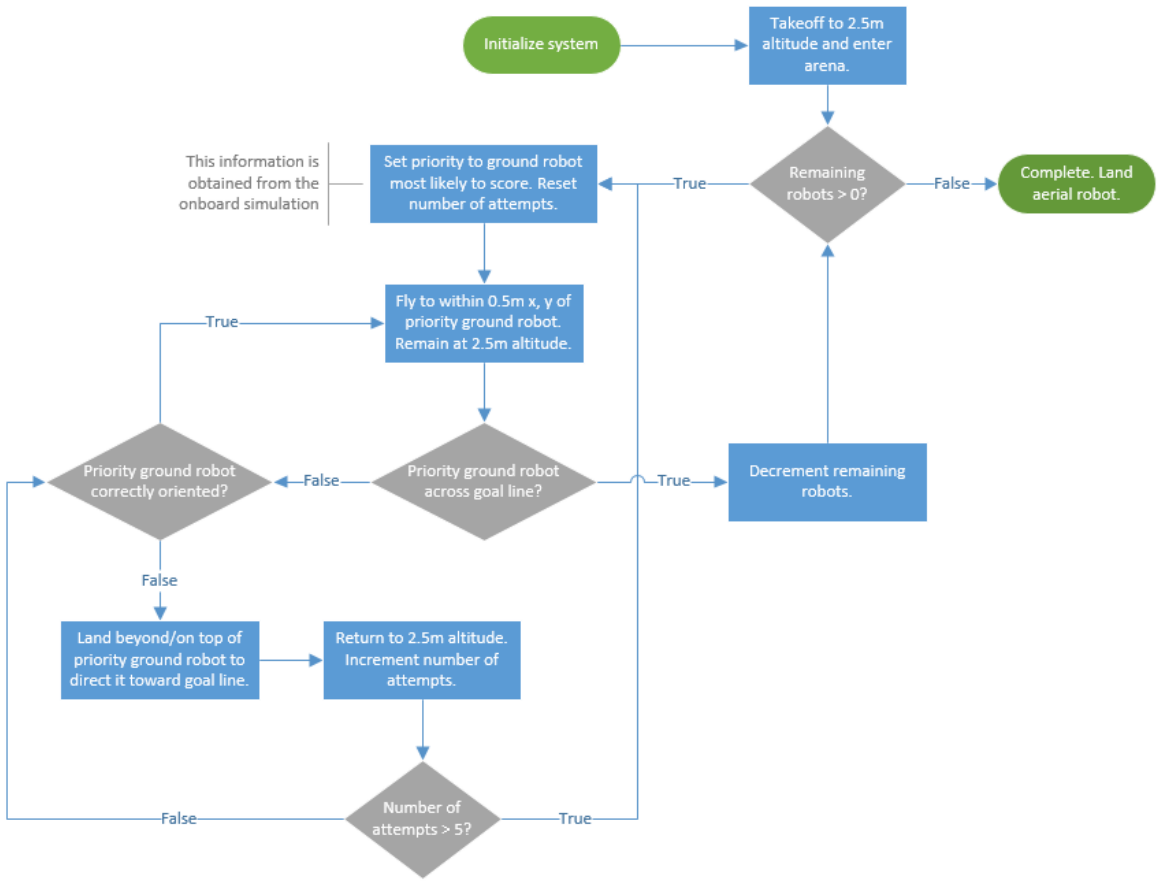
Initialize system
Takeoff to 2.5m altitude and enter arena.
This information is obtained from the onboard simulation
Set priority to ground robot most likely to score. Reset number of attempts.
True
Remaining robots > 0?
False
Complete. Land aerial robot.
True
Fly to within 0.5m x, y of priority ground robot. Remain at 2.5m altitude.
Priority ground robot correctly oriented?
False
Priority ground robot across goal line?
True
Decrement remaining robots.
False
Land beyond/on top of priority ground robot to direct it toward goal line.
Return to 2.5m altitude. Increment number of attempts.
False
Number of attempts > 5?
True

Figure 4. Simplified figure of control system architecture.

Flight Termination System

The flight termination system is initiated after the following conditions have been met:

1. The flight planning ROS node determines that all ground robots have successfully crossed into the green corral. A flight script will then navigate to the center of the arena and initiate a controlled landing.
2. The flight planning ROS node determines that 10 minutes have elapsed and thus a controlled landing is automatically initiated.

3. The IARC officials end the match due to all ground robots having left the bounds of the arena or the aerial robot contacts the moving obstacles more than twice. A controlled landing is initiated from the ground station by team members.
4. The battery of the aerial robot can no longer provide power to maintain flight. The vehicle safely lands at its current position.
5. The aerial robot has left the bounds of the arena for more than 5 seconds or has gone farther than the 2-meter buffer space. If the aerial robot is exhibiting stable behavior, the flight will be terminated by means of a controlled landing initiated from the ground station by team members. If the aerial robot is exhibiting unstable behavior or is on a potentially harmful trajectory the flight will be terminated by means of the remote kill switch.
6. If at any time the aerial robot exhibits unstable behavior that is potentially harmful the remote kill switch is activated.

PAYLOAD

Sensor Suite

GNC Sensors

The sensors involved with guidance, navigation, and control are contained within the flight controller and optical flow module. The flight controller is the Pixhawk 2.4.6 running the PX4 Flight Stack and the optical flow module is the PX4Flow. The flight controller uses multiple sensors within it to stabilize the yaw, pitch and Rolland externally utilizes the Lidar Lite v3 to interpret altitude. The optical flow module relays the current position of the aerial robot in 2-dimensional space. These coordinates are posted to the localization node within ROS.

Mission Sensors

The aerial robot utilizes three cameras and the OpenCV library to detect and identify each of the 10 ground robots and four obstacle robots. Each camera is mounted to the bottom of the quadcopter facing 45 degrees below horizontal at 120° radial horizontal spacing. The cameras are connected to the NVIDIA Jetson TX2 via a powered USB 3.0 hub. Each camera is capable of a 13MP resolution [6] and is outfitted with a lens that increases the field of view (FOV) to 90° vertically and 120° horizontally.

TABLE 1. AERIAL ROBOT SENSORS

Task	**Sensor Type**	**Sensor Model**	**QTY**	**Description**
Computer Vision	Camera	See3CAM_CU130, AR1820HS	3	Detects objects, grid, and corral
Optical Flow	Camera	MT9V034	1	Determines 2-dimensional position in space
	Gyroscope	L3GD20	1	
Flight Controller	Gyroscope	STMicro L3GD20H	1	Vehicle attitude calculation
	Accelerometer	STMicro LSM303D	1	
	Magnetometer	InvenSense MPU 6000	1	

	Barometer	MEAS MS5611	1	Altitude estimation
	Lidar	Lidar Lite v3	1	

Target Identification

Each frame from the three onboard USB 3.0 cameras is analyzed using tools from the OpenCV library. One of these tools is a multiscale cascade detector and the other is called a blob detector. Various other functionalities of the OpenCV and NumPy libraries such as color space conversion, image resizing, matrix manipulation, and others used to identify and track objects.

The process for utilizing a multiscale cascade detector starts by creating a custom made Local Binary Pattern (LBP) cascade. The cascades are made using positive images, or images containing the object being tracked, and negative images, or images that do not contain the object being tracked. Machine learning is utilized to create an XML-formatted cascade that contains a profile of the object. Using the current frame obtained from the onboard cameras and the custom-made cascade, the cascade detector outputs a bounding box around anything that fits the cascade profile. A bounding box is the smallest rectangle that encapsulates every point that is determined to be part of the object being tracked.

The process for utilizing a blob detector begins by first determining a threshold of color, between which the color of the desired object is found. For this process, a frame obtained from the onboard cameras is converted to the hue, saturation, value (HSV) color space. The high reflectivity of the top plate on the ground robots lead to the implementation of the glare resilient HSV color space. Using the custom-made thresholds, and the current frame from the onboard cameras, the blob detector outputs what are referred to as keypoints. Keypoints are a group of pixels calculated to be a part of the desired object.

Using these tools, a process for identifying the ground robots was created. A first scan utilizes an LBP cascade to look for a fully assembled ground robot. It then scans the frame again using a LBP cascade to look for the top plates of the ground robots and then it is searched again using blob detection. This triple detection helps to assure that all robots are identified and helps with any false positives that may occur. The coordinates and color of the ground robot detections are assigned arbitrarily to 10 robot objects within the program. The program then looks at the next frame but searches only regions of interest (ROI) that may contain ground robots. A region of interest is a small portion of the whole frame that is generated based upon the last known coordinates of a ground robot. If a robot is not detected in its ROI, a new ROI is created based upon its last known vector which increases the area of the ROI in the direction of its vector. The area of the ROI is then incremented based upon number of frames it has not been detected. After 5 frames where the object was not detected, it is then considered lost and the whole frame will be searched. Figure 5 below shows an outline of this logic.

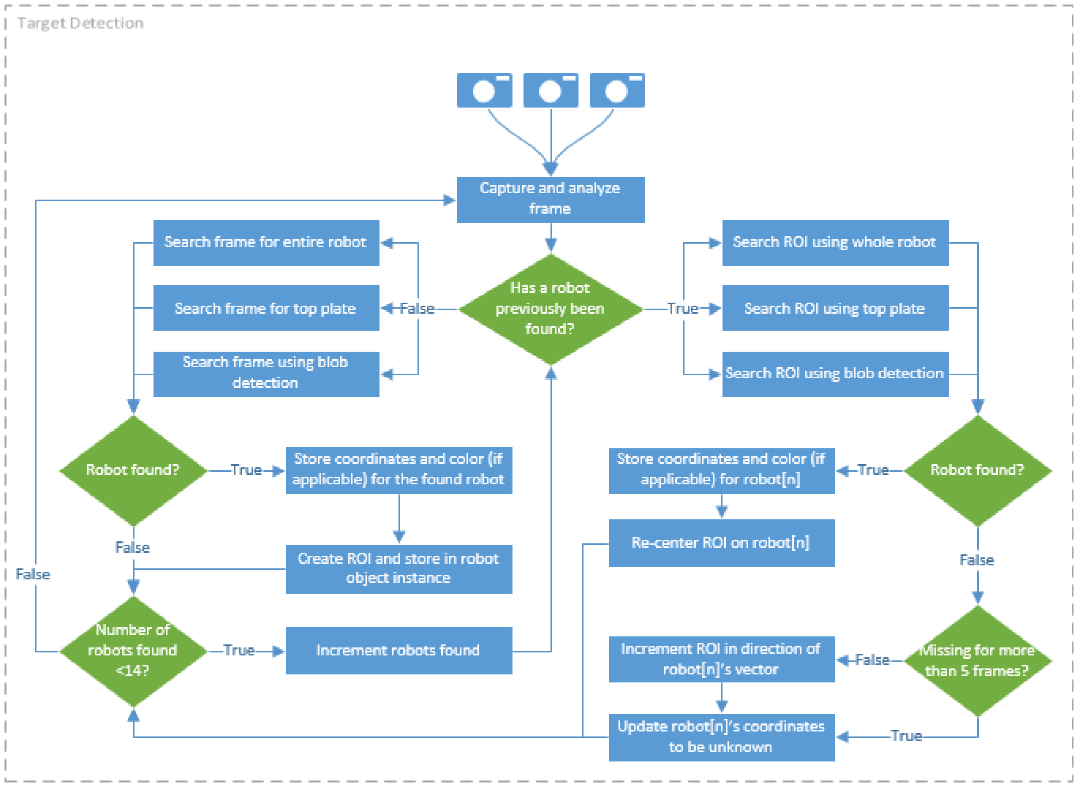
Target Detection
Capture and analyze frame
Has a robot previously been found?
False
True
Search frame for entire robot
Search frame for top plate
Search frame using blob detection
Search ROI using whole robot
Search ROI using top plate
Search ROI using blob detection
Robot found?
True
False
Store coordinates and color (if applicable) for the found robot
Create ROI and store in robot object instance
Number of robots found <14?
True
False
Increment robots found
Robot found?
True
False
Store coordinates and color (if applicable) for robot[n]
Re-center ROI on robot[n]
Missing for more than 5 frames?
False
True
Increment ROI in direction of robot[n]'s vector
Update robot[n]'s coordinates to be unknown

Figure 5. Outline of target detection logic.

All data pertaining to the localization of the ground robot and obstacle robots is posted to the localization node within ROS. Each ground robot and obstacle robot is assigned and integer identification number, 2-dimentional Cartesian coordinates and vector, and an integer color identification.

On top of real-world mapping, a custom simulator written in Python 2.7 runs as a ROS node onboard the aerial robot and integrates actual environmental data including known ground robot positions with simulated data in order to best predict the paths of ground robots and allow appropriate flight planning. Upon gathering data published by the localization node, the simulator first compares its calculated positions of all ground robots to the known positions and calculates a confidence level for the individual robots that represents its certainty in the position and vector information it publishes to its own related topics. This becomes especially useful in the event that a collision is detected in the simulator and it can no longer be certain of the involved ground robots' new position and vector—thus the confidence level of that ground robot is lowered. If the necessary data is available and confidence in that data is high, the simulator will refresh the position and vector information of its simulated robots in order to obtain a more accurate representation of the arena.

This combination of simulation and real-time mapping is integral in providing the flight planner with enough information to perform its given task.

Threat Avoidance

Data used for threat avoidance will be collected from the three downward facing cameras, interpreted, and posted to the localization ROS node. This data will include a unique integer robot identification, an integer color identifier, x and y float coordinates, and a float vector for each identified robot. Based on these values, the simulation will attempt to predict movement as previously noted. The real-time tracking and prediction of all ground robot positions allows for avoidance of obstacles and other threats when the flight planner develops a path to fly.

Communications

The aerial robot utilizes three RF links on two different center frequencies for all communication. These links include a 433 MHz telemetry system using Holybro-branded telemetry transceiver modules, a 2.4 GHz Wi-Fi link, and a 2.4 GHz RC transmitter/receiver link.

The 433 MHz telemetry system is used to communicate critical vehicle state information to the ground station such as erroneous control system behavior, loss of RC link, battery life, etc. The 2.4 GHz Wi-Fi link allows communication of the ground station to the master ROS node hosted by the aerial robot flight computer. Lastly, the 2.4 GHz RC link allows for manual override of the vehicle during flight by a trained pilot. This system also utilizes frequency hopping to maintain signal integrity in environments that are saturated with noise from other RF sources.

Power Management System

The aerial robot is powered by a single 4S lithium polymer (Li-Po) battery with a nominal voltage of 14.8V and a 10000mAh capacity. This power configuration combined with the chosen propulsion system will provide the aerial robot an estimated 15-minute flight time. Power is distributed to various electronics on the aerial robot through the use of a single primary power distribution board and a secondary inline power regulation module for the flight controller. The main draw of power is from the motors which pull 18.5A each under full load. These connect directly to the battery through the power distribution board. The power distribution board also has onboard regulation for 12V and 5V power rails. These are used to supply the flight computer and cameras. Lastly, the secondary inline power regulation module is designed and used for the Pixhawk flight controller. This module also provides voltage and current monitoring capabilities which in turn allows for the aerial robot to safely perform its mission without risk to itself or its battery.

OPERATIONS

Flight Preparations

All flight preparations are enacted through a series of checklist items as shown below. Before moving to any full-scale flight test, adequate testing of all components including flight scripts is done computationally to ensure no erroneous behavior that could lead to physical damage to the aerial robot or its environment and operators.

Checklist

Pre-flight

- Mechanical
    - ___ Fasteners secure
    - ___ Vehicle frame undamaged

    - ___ Propellers undamaged, secured
    - ___ Motors spin freely
    - ___ All wiring secure, undamaged
- Electrical
    - ___ Vehicle battery charged
    - ___ RC controller battery charged
- Flight-specific
    - ___ Flight objectives outlined

Pre-arm

- Electrical
    - ___ Battery voltage monitor attached
    - ___ RC controller on, connected
    - ___ Onboard computer on, software running (ROS)
    - ___ Ground station software open, connected (ROS, QGroundControl)
- Safety
    - ___ Trained operator holding RC transmitter
    - ___ Trained operator holding kill-switch remote
- Flight-specific
    - ___ Flight area clear
    - ___ Bystanders aware of flight goals

Post-flight

- Electrical
    - ___ Onboard computer safely shutdown
    - ___ Battery disconnected, properly stored
- Flight-specific
    - ___ Ground station software shutdown, logs saved

Man/Machine Interface

The aerial robot interfaces with humans through a ground station, an RC digital telemetry radio system, and a kill switch. Using a custom-built GUI and integration into ROS (referred to as the Redbird Control Panel, or RCP), the ground station provides the capability start and stop flights and to override the aerial robot activity in the event of an emergency. In addition, the ground station provides flight data such as altitude, velocity, and position of the aerial robot to the user. Coupled with the RCP is the use of QGroundControl (QGC) which is an open-source software package developed to work seamlessly with the PX4 flight software. An FrSky Taranis Plus+ RC transmitter also provides a trained operator the ability to manually control the aerial robot for testing as well as flight script overriding in the event of an emergency.

The emergency kill switch provides a way to cut all power to the aerial robot motors. This stops all motors causing the vehicle to safely fall to the ground, thus avoiding damage to property or bystanders.

The vehicle requires three people to operate, one to pilot the aerial robot using the RC transmitter, another person to monitor flight data through the ground station, and another to control the kill switch in the event that an emergency shutdown is required.

RISK REDUCTION

Vehicle Status

Vehicle status is monitored through the aforementioned RCP and QGC software packages. When combined, they allow for real-time monitoring of all vehicle flight conditions ranging from vehicle attitude to battery voltage levels and current draw. They provide the operator quick insight into the status of flight scripts and onboard instrumentation and ultimately allow for quick reaction to off-nominal flight conditions.

Shock/Vibration Isolation

To reduce interference with electronic sensors from the vibrations of the propellers and motors, electrical tape is used at the interface of the clamps and arms. Additionally, the electronics are attached to the vehicle plates with soft foam, and rubber grommets are used on all onboard computer standoffs. All structural fasteners on the frame assembly are fastened using lock washers and lock nuts in order to prevent components from vibrating loose.

EMI/RFI Solutions

Cable routing and antenna placement onboard the aerial robot has been considered very carefully throughout the design and assembly phases. Care has been taken to ensure isolation of critical components from other components that produce a large amount of electromagnetic interference—most notably the DC voltage supply lines going to the motor electronic speed controllers. These lines have been routed away from all other electronics and shielded when necessary and LC filters have been included.

Modeling and Simulation

Flight simulations of the aerial robot are performed using a combination of a custom simulator previously discussed in Threat Avoidance as well as through the use of a simulator inside of Gazebo built by the developers of the PX4 flight stack. Gazebo is primarily used to simulate the movement of the vehicle itself, therefore allowing for intuitive monitoring of the attitude of the vehicle in relation to the flight path being chosen by the flight planner. The custom simulator is used to simulate the objective of the aerial robot and how it has interacted with all involved ground robots through its own collision monitoring.

All physical components of the quadcopter were modeled and assembled in SolidWorks, a 3D CAE software that is provided to University of Louisville engineering students. SolidWorks can also be used to mesh components and run stress tests without putting the actual craft at risk for damage.

Testing

All testing of the aerial robot and associated components is done in a controlled environment with minimal human involvement. At each stage of development, thorough computational simulations are performed before the team engages in full-scale flight testing. As previously noted in the checklist, all bystanders to physical tests are briefed on the objectives of the flight beforehand to ensure that they are aware of off-nominal flight conditions and can react accordingly.

When testing custom-designed mechanical components of the vehicle, structural analysis and computational finite element analysis (FEA) are used to verify the integrity of the components.

CONCLUSION

The advantage of this design is the combination of tools used to accurately locate targets on the ground, use of a simulated environment to predict movement patterns, and communication through nodes in ROS. The 3 levels of target detection allow for fewer false positives and the wide angle of the cameras allow for more information to be extracted. The robot simulator allows the aerial robot to predict the movements of ground robots that are not in view and choose optimal targets to interact with. ROS allows for seamless communication between all parts of the aerial robot both simply and efficiently.

ACKNOWLEDGMENTS

We would like to extend our sincerest thanks to our many team sponsors including Material Handling Systems, Inc., Luckett & Farley, LLC., Thermal Equipment Sales, Inc., and Genscape, Inc. Without the contributions of these companies our work would simply not have been possible. The kindness shown to us has gone on to inspire team growth, the advancement of our members' skillsets, and outreach to our community.

REFERENCES

[1] D. Honegger, L. Meier, P. Tanskanen and M. Pollefeys, "An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications," ETH Zurich, [Online]. Available: https://pixhawk.org/modules/px4flow. [Accessed 01 03 2017].

[2] K. Sinhal, "Training a Better Haar and LBP Cascade Based Eye Detector Using OpenCV," 23 01 2017. [Online]. Available: https://www.learnopencv.com/training-better-haar-lbp-cascade-eye-detector-opencv/.

[3] S. Mallick, "Blob Detection," Learn OpenCV, 02 2017. [Online]. Available: https://www.learnopencv.com/blob-detection-using-opencv-python-c/.

[4] NVIDIA, "Embedded Download Center," 2017. [Online]. Available: https://developer.nvidia.com/embedded/downloads.

[5] "Why ROS?," ROS, [Online]. Available: http://www.ros.org/core-components/.

[6] e-con Systems, "See3CAM_130 - Technical Documents," [Online]. Available: https://www.e-consystems.com/doc_13MP_autofocus_USB3_Camera.asp. [Accessed 18 May 2017].