Aerial Robot Design for Ground Robot Interaction and Navigation without Landmarks

Aaron Miller, Levi Burner
Robotics and Automation Society - University of Pittsburgh

ABSTRACT

A design for an autonomous drone's mechanical, electrical, and software system capable of finishing the International Aerial Robotics Competitions Mission 7a will be presented. The solution focuses on the ability to herd a group of ten ground robots across a single side of an arena while avoiding tall cylindrical obstacles. The sensors used for orientation and position estimates will be described as well as their intended purposes and why they were selected. In addition, a monitoring system will be described that enables safe recovery of the drone during software failure. The drone is of a quadrotor design, 107cm across and weighs over 4kg. Five cameras, a planar lidar, two ground distance sensors, ground contact switches, an accelerometer, a gyroscope, and a magnetometer are used for state estimation and target tracking. A real time flight controller is used for active stabilization, while a Jetson TX2 and a ground station are used for image processing, high level control, and planning.

INTRODUCTION

The Problem

Mission 7a of the International Aerial Robotics Competition requires teams to develop autonomous aerial vehicles capable of navigation in an indoor, GPS-denied environment without nearby landmarks amenable to conventional SLAM algorithms. The aerial vehicles must interact with randomly moving ground robots to direct them toward one side of the competition arena, while avoiding dynamic obstacles. Mission 7b builds on these design goals by requiring two drones to compete in the arena at the same time, avoiding collisions with each other and directing ground robots toward separate goals [4].

Overall Solution Concept

Our drone is a quadrotor with four side cameras, one bottom camera, a long-range time of flight LIDAR paired with a short-range time of flight LIDAR for height measurement, landing gear with contact switches for landing detection, a 360 degree planar laser scanner, a Teensy 3.2 microcontroller for digital and analog sensor interfacing, an NVIDIA Jetson TX2 for onboard computer vision and high level autopilot functionality, and a Seriously Pro F3 Evo flight controller for real time control and stabilization.

The autonomous intelligence primarily runs on the Jetson and uses ROS to support high level autonomous control. The Jetson then communicates its desires to the real time flight controller.

Communication with a base station is achieved via WiFi. The control system is completely segmented into independent nodes through ROS. Any combination of nodes may be chosen to run on the drone or the ground station as deemed appropriate.

At any time a human pilot can override the autonomous system. Additionally, a separate wireless system is available to remove power from the ESCs during an emergency situation.

See Figure 1 for a diagram of the design.

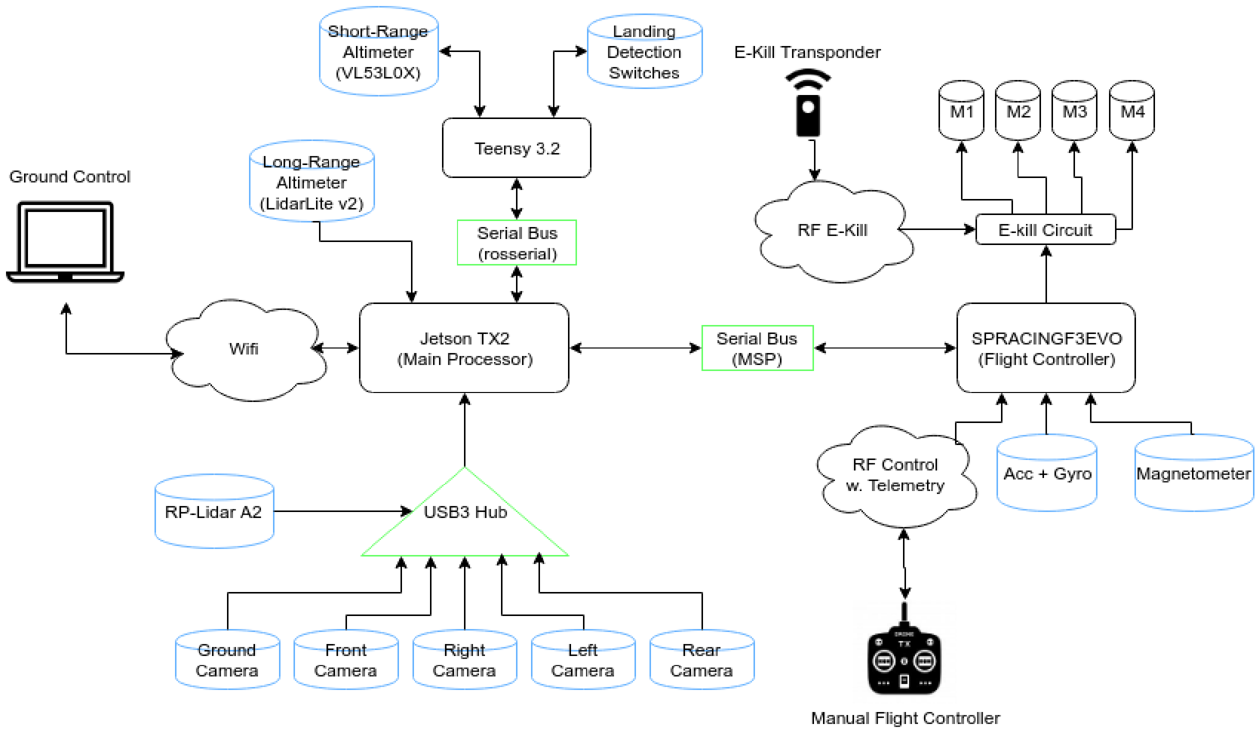

Figure 1. System Architecture

Milestones

May 2016	Initial system design began
September 2016	Team formation
September 2016	High-level software design
October 2016	Simulator developed
February 2017	First autonomous takeoff
May 2017	Final hardware revision assembled

VEHICLE DESCRIPTION

The vehicle frame is entirely custom-made, with the core being an X-shaped carbon fiber tube configuration. Carbon fiber was chosen for its high strength-to-weight ratio. The four carbon fiber tubes are held together at the center of the drone between two carbon fiber plates, with all of the mounting brackets being 3D-printed. On the outward ends of the frame tubes, 3D-printed motor mounts support the motors, with attached prop guards enclosing the motors and propellers for protection. Two rings of copper wire encircle the upper body of the drone around the perimeter of the prop guards. Landing gear is mounted on the bottom of the carbon fiber rods, which extend downward from each of the four motor mounts. These rods are attached to each other at the bottom with wooden spars that form a bounding box around the lower body and allow us to block the path of a ground robot. Finally, a button pusher assembly made from soft sponges extends downward from the central frame.

Propulsion System

Propulsion is provided by four brushless electric motors (KDE2814XF-515 from KDE Direct) paired with KDEXF-UAS35 ESCs and KDEXF-CF125-TP 12.5 inch triple-bladed carbon fiber propellers. The whole system provides 10.4kg of thrust at maximum throttle, and an efficiency of approximately 8.8 g/W at hover.[2] The propulsion system is powered by a 6 cell, 8Ah, 10C discharge MultiStar LiPo battery.

Guidance, Navigation, and Control

Stability Augmentation System

The SAS used is a standard SP Racing F3 Evo flight controller running a modified version of the Cleanflight open-source software. The Jetson communicates with the flight controller using the Multiwii Serial Protocol to send orientation and throttle commands and receive readings from the sensors on the flight controller, including an orientation estimate and accelerometer readings. The flight controller uses a simple cascading PID controller to send commands to the motors so that the drone maintains the commanded orientation. The F3 Evo and Cleanflight are known for their use on racing drones. Together, they offer stability during demanding maneuvers. In addition, Cleanflight was written in a simple manner that was easy to modify.

Navigation

The drone has a multi-layer navigation stack. Figure 3 details the relationships between each layer. At the highest level is a motion planner node, which allows other nodes to request high-level tasks, such as navigation to waypoints or various interactions with ground robots. Each of these tasks is capable of generating velocity profiles, which are then modified by the obstacle avoidance algorithm to guarantee their safety. These safe velocities are then handed to the low level motion controller, as shown in Figure 3.

Control

The lowest level of the autopilot that runs on the Jetson is contained in the Low Level Motion Controller node. This node contains an in-flight velocity controller, which uses PID controllers, along with models of the drone's dynamics, to generate orientation and throttle commands for the flight controller. A crucial requirement for this system is a model of the amount of thrust generated for a given throttle command. To develop the model, a single battery-ESC-motor-prop combination was mounted horizontally in a testing apparatus, with a lever attached to a load cell so that the thrust output of the motor could be measured by a computer. The tests were run at multiple battery voltages so that the model would remain accurate during a long duration flight. A polynomial surface was then fit to the resulting data (See figure 2), resulting in a function mapping throttle commands and battery voltage to output thrust.

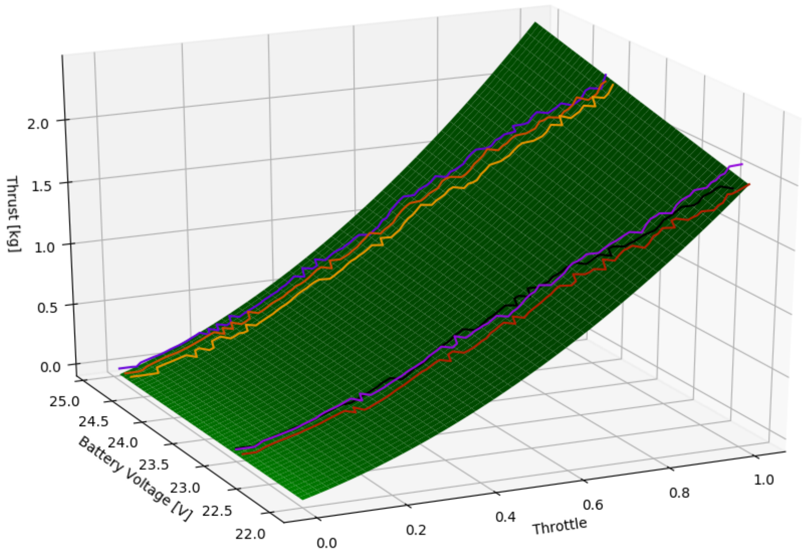

Figure 2. Thrust data with a least-squares fit polynomial surface

Flight Termination System

Flight termination needed to be simple and power efficient. Four power MOSFETs control the current to each of the four electronic speed controllers. The MOSFETs are controlled by a timing circuit that listens to the output of a dedicated radio receiver and controller. When a button on the controller is pressed, the pulse width from sent from the receiver is modified. The safety circuit detects this and turns off the MOSFETs, which remove power from the electronic speed controllers. The drone is then rendered ballistic. The safety timing circuit is equivalent to the officially recommended design and uses D-flip flops and an RC circuit for timing the radio pulses [1].

An additional feature of the flight termination system is allowing a separate logic input for the control computer. The control logic line and the dedicated radio's signal are used as inputs to an AND gate. The output of the gate is used to drive the gates of the power MOSFETs. Either the auto-pilot or dedicated radio can disable power at any time; however,

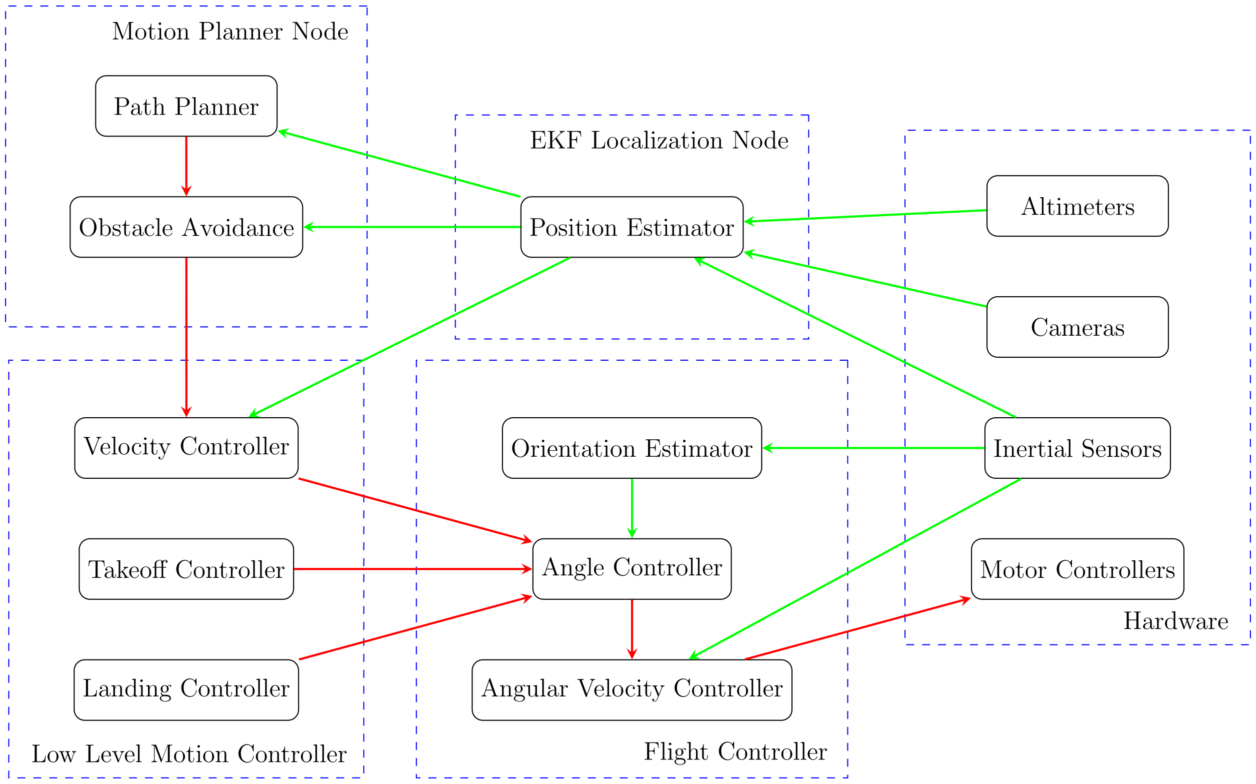

Figure 3. Control system architecture

both have to be in agreeance to allow power to the motors due to the AND gate. This system was designed so that the propulsion battery could be connected by an operator without power being provided to the propellers. When all operators are a safe distance from the drone, and the flight termination radio is active, the control computer can allow the MOSFETs to be turned on.

PAYLOAD

Sensor Suite

GNC Sensors

The SP Racing F3 Evo has an MPU9250 onboard, which contains an accelerometer, a gyroscope, and a magnetometer. These sensors are fused on the SP Racing F3 Evo itself to produce a 3D orientation estimate. The quadcopter has two altimeters: first, it has a LIDAR-Lite v2 time-of-flight laser rangefinder that works up to an altitude of 40m, but is very noisy at altitudes below approximately half a meter [3]. For this reason, the quadcopter also has a short-range ST VL53L0X time-of-flight laser ranging sensor with a range of up to 2m, as well as a microswitch on each landing gear strut that closes upon close proximity with the ground [5].

To detect position, the images from the downward-facing camera are processed on the Jetson using CUDA accelerated OpenCV algorithms to extract the locations of the gridlines. The image processing pipeline is as follows:

1. Resize so that the features comprise a similar number of pixels at all altitudes
2. Canny edge detector
3. Hough line transform
4. Custom line-processing algorithm that uses least-squares fitting to find the most likely grid orientation based on the observed lines and the previously estimated state of the drone

This provides more stable long-term position estimates than naïve optical flow-based velocity estimation because the results are absolute positions instead of velocities, so there is no drift due to integration of noisy data. One disadvantage of this approach is that the system can occasionally miss a grid cell, resulting in the position estimate being 1m off for the remainder of the run. Testing in the simulator showed that this only occurs when the drone travels at speeds significantly higher than are realistic during the mission.

All of these observations are fed into robot_localization's ekf_localization_node, which implements a 15-DOF Extended Kalman filter based on a 3D kinematic model. Position, velocity, and acceleration estimates are provided by the filter for all three translational degrees of freedom.

Mission Sensors

All five cameras on the drone are used for target identification. The bottom camera has a wide angle lens and is used to detect the precise relative position and relative velocity of a target when attempting to land on or in front of the target.

For threat detection, the drone has an RP-LIDAR A2 scanning LIDAR, which provides a full 360° view of the area around the drone. The scan from this LIDAR is then processed into a point cloud, which is matched to a model of the obstacle robots using nonlinear least-squares fitting on clusters of points within the arena to find likely obstacle locations.

The bottom and side cameras provide complete visibility below the horizontal plane on all sides of the drone, except for small areas blocked by the landing gear. Because of this, the hardware is capable of providing robust omni-directional obstacle detection in the future when necessary for Mission 7b as soon as the appropriate software is developed.

Communications

The quadcopter has a standard 2.4GHz radio receiver connected to a FrSky Taranis remote control, which can be used to arm and disarm the motors, take the drone into and out of autopilot mode, set limits on the throttle, and take over from the autopilot at any time.

The hardware E-Kill switch is controlled by a separate radio receiver/transmitter pair.

The Jetson also has built-in WiFi, which is used to connect to a local network, to which multiple ground computers can be connected. This allows for communication between ROS nodes on the ground computer and ROS nodes on the Jetson, which makes debugging easier and enables us to use the ground computer as a control center. This also allows for nodes that are determined to be too computationally intensive to be run on the ground computer.

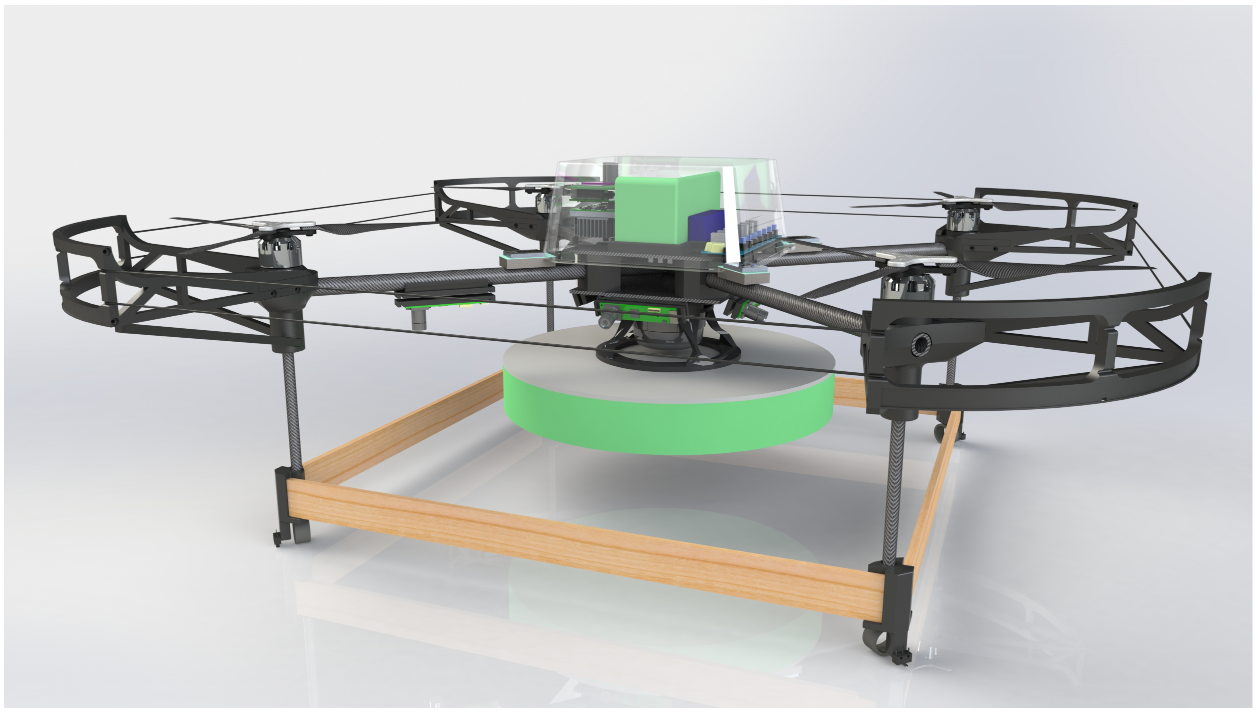

Figure 4. Render of our drone's CAD model

Power Management

A single 6 Cell 22.2V 8Ah LiPo battery is used to power the propulsion system. A second 3 Cell 11.1V 1Ah LiPo battery is used to power the more sensitive electronics, such as the Jetson, flight controller, and cameras. The safety circuit is also powered using the low power electronics battery, while the MOSFETs control the power from the propulsion system battery to the motors. Having two batteries allowed powerful and efficient motors that required higher voltages than the control electronics to be used. The minimal electrical coupling between the propulsion system power and the control electronics was considered advantageous since voltage and current transients were less likely to impact sensitive components. Additionally, it eliminated the need for the control system to be rebooted when swapping the propulsion battery.

Voltage monitoring of the propulsion battery and control electronics battery is done using the Teensy's analog pins. Simple voltage dividers made of resistors lower the voltage to a measurable level and protect the electronics from voltage spikes and current transients.

OPERATIONS

Flight Preparations

The drone is checked over both by the software and by a human pilot prior to takeoff. In addition, several physical procedures must be completed prior to takeoff.

Before connecting the propulsion battery, the following checklist must be completed:

1. Frame is checked for any integrity issues
2. Electrical systems are looked over for potential breaks in the wiring
3. Battery level is checked
4. Radios are turned on
5. Bystanders informed of the event about to take place.
6. Control battery connected
7. Wifi connection and SSH session established

Once completed, the propulsion battery is connected.

The checklist for autopilot initialization is as follows:

1. Flight termination is tested through an indicator LED and the dedicated radio
2. A Bluetooth connection is made with the flight controller and used to verify the human operator's radio connection commands are being received
3. Arming switch on the human operator's radio is engaged
4. Autopilot switch on the human operator's radio is engaged
5. Autopilot software is started via SSH session
6. Autopilot software takes control of the flight controller

After these are completed, the autopilot is free to initiate takeoff and start the mission.

Automated Checks

The system automatically performs a variety of checks before autopilot engagement. First, the software enforces a startup order that will halts each ROS node until all of the nodes it depends on are ready. This means that nodes associated with the high-level functionality of the autopilot will not start any autonomous behavior until lower-level nodes (such as sensors and communications with the flight controller) have come online successfully.

Furthermore, the system performs a variety of sanity checks before takeoff. If the drone is not flat on the ground or the motors are not disarmed, takeoff will not be allowed to start.

While flying, automated validity checks are performed by high level motion tasks during their startup and operation. For instance, while in flight, a minimum maneuvering height is enforced to prevent accidental contact between the feet and ground. Similarly, checks are made when landing to make sure the drone is actually in the air and in a state capable of executing an automated landing sequence without damaging on board hardware.

Man/Machine Interface

The system has multiple points of interaction with human pilots and observers. First, the hardware kill switch is controlled by its own radio transmitter/receiver pair. This allows a single observer to quickly cut power to all motors on the drone in case of an emergency.

Next, the flight controller is attached to a standard radio transmitter/receiver pair allowing a human pilot to fly the drone manually. This transmitter has the ability to arm or disarm the motors, enable or disable commands from the autopilot on the Jetson, and immediately override any commands from the autopilot with commands from a human pilot.

Finally, the ground computer interfaces with the autopilot through ROS, allowing high-level commands to be sent to the drone. This also allows for continuous monitoring of the autopilot's internals, such as its incoming sensor data and current state estimate.

RISK REDUCTION

Vehicle Status

The drone has several software systems that allow for monitoring of the system and safe shutdown. First, the software system includes a Node Monitor which maintains a connection with all important ROS nodes in the system. In the case of a failure in one node, the Node Monitor sends a message to the other nodes to activate a safety response. These safety responses are hierarchical, so if a high-level node goes into a failure state, then a lower level node will execute a controlled descent. At the bottom level of the hierarchy, the flight controller communication node will drop the throttle to a value which will slow the quad's fall.

The drone has a compression spring mounted on the bottom of each foot, capable of absorbing approximately 1.1 Joules each, to protect against shock on landing. To limit the amount of vibration from the motors, the motor mounts are isolated from the frame by a thin layer of rubber, and the propellers are balanced by the manufacturer. The remaining vibrations from the motors during flight were observed not to interfere greatly with sensitive instruments such as the accelerometer, so it was determined that dedicated vibration isolation hardware is not necessary in that case. However, the cameras have rolling shutters which are prone to shearing effects from vibration, so they are isolated from the frame by a layer of gelatinous material.

The most EMI-sensitive component on the drone is the magnetometer, so wires were intentionally routed away from that area to avoid any interference. The high-current wires that carry power from the battery to the ESCs were also routed over the top of the drone, far away from all other components, to prevent interference.

Safety

The drone has both hardware and software safety precautions. On the hardware side, the drone has a system of propeller guards around its perimeter, reducing the chance of any object entering the reach of the propeller and causing harm to itself, the propeller, or the motor. These prop guards can be seen in Figure 4. The physical drone also has stiff wire running around the prop guards to enclose the entire perimeter.

The status monitoring software described in the above Vehicle Status section also provides safe shutdown mechanisms in the case of various system failures. Additionally, failures are relayed to the ground monitoring station so that a human pilot can be aware of the safety event and choose to take control of the drone instead of allowing the automated safety responses to complete the landing maneuver.

Modeling and Simulation

The entire mission is simulated using MORSE, the Modular OpenRobots Simulation Engine, which is built on Blender and the Bullet physics engine. MORSE is programmable in Python and interfaces with ROS. All of the drone's sensors are simulated and available through ROS just as they would be on the physical drone. The ground robots are also simulated with the same movement characteristics that they will have in the competition. Ground truth values for various portions of the environment state are also available, such as the drone's pose and the ground robots' locations. All of this allows for various parts of our software stack (such as localization or computer vision) to be swapped out and tested independently. This allows us to catch many bugs and design more robust software before testing on our expensive hardware.

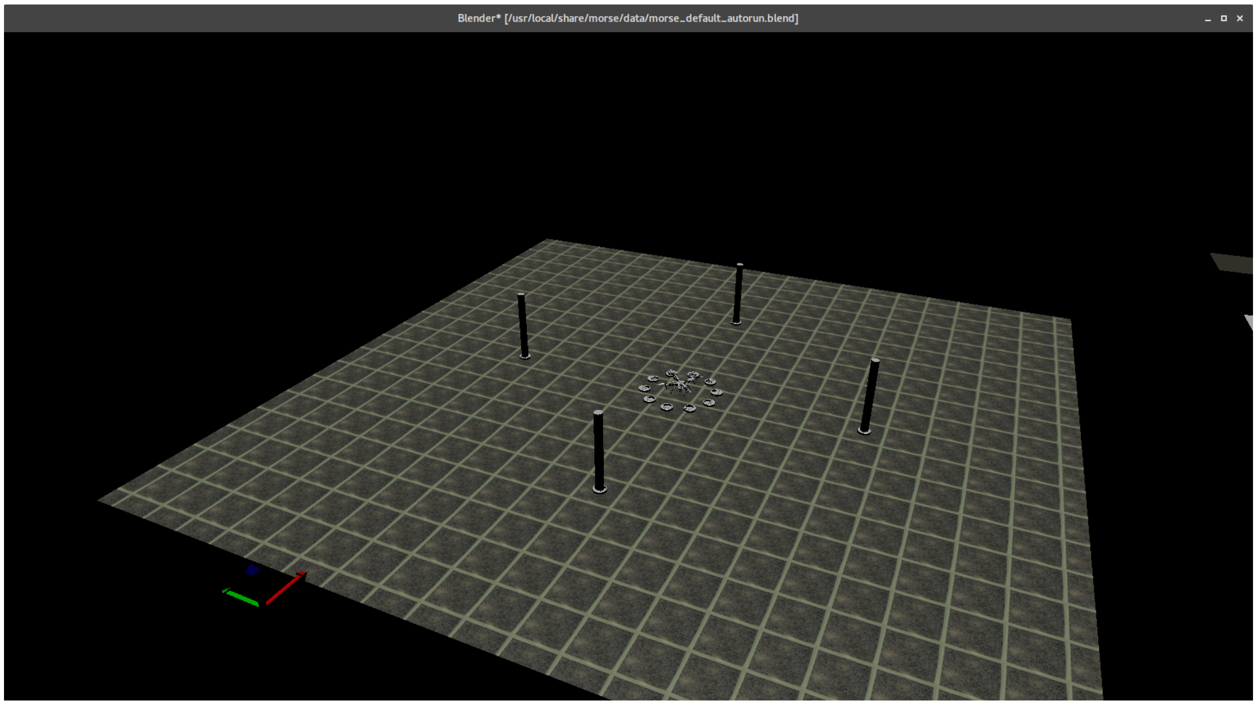
Blender* [/usr/local/share/morse/data/morse_default_autorun.blend]

Figure 5. The simulator, with the drone in the center of the arena and the ground robots in starting positions

Testing

Simulations were used extensively to test the navigational stack. Standardized tests were developed for individual components of the navigation stack to allow easy evaluation of changes. Larger integration tests allowed multiple parts of the navigation stack to be tested in simulations with a variety of different consecutive maneuvers. These same tests were then run on the real drone to validate the controller designs on physical hardware.

To test hardware, small test programs were written. Most sensors have a program that can be quickly launched to validate a sensor's operation and allow for in-depth testing.

To test the flight controller, the drone was flown in an outdoor environment with plenty of space to perform maneuvers more aggressive than will be seen in competition. These tests formed a basis for how aggressive the autopilot software is allowed to be when requesting maneuvers. This also allowed us to confirm that our battery has enough capacity to sustain flight for well over the ten minutes required for the mission.

Full system testing on the physical drone at the time of writing has been restricted to takeoff, hover, and landing, which have been successfully achieved. Future system testing of the navigational stack will be conducted inside of a 10 foot by 20 foot tent to protect bystanders. Full scale tests will be conducted in a basketball court with conditions as close as possible to those in the competition arena.

CONCLUSION

Our algorithms and hardware have been demonstrated to provide effective approaches for many of the problems relevant to Mission 7 through simulation and testing. Further testing is needed, but these approaches seem likely to translate to effective strategies for the physical drone and success at competition.

ACKNOWLEDGEMENTS

We want to thank the University of Pittsburgh Swanson School of Engineering for providing space and resources during our design process. In addition, the University of Pittsburgh was a key part of funding this ambitious project. Finally, we'd like to thank our corporate sponsors, Rockwell Automation, KDE Direct, Slamtec, and SolidWorks, for their financial support and discounted pricing. Without them, we would not have been able to make this drone.

Finally, we want to thank Andrew Saba, without whom there would be significantly fewer commas.

References

[1] *International Aerial Robotics Competition Safety Switch Design.* URL: http://www.aerialroboticscompetition.org/downloads/killswitch.zip.

[2] *KDE Direct XF CF Brushless Performance Testing - KDE2814XF-515.* URL: https://cdn.shopify.com/s/files/1/0496/8205/files/KDE_Direct_XF_CF_Brushless_Performance_Testing_-_KDE2814XF-515.pdf?17184393026696725656.

[3] *Lidar Lite v2 Datasheet.* URL: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/lidarlite2DS.pdf.

[4] *Official Rules for the International Aerial Robotics Competition - Mission 7.* 2017. URL: http://www.aerialroboticscompetition.org/downloads/mission7rules_013017.pdf.

[5] *ST VL53L0X Time of Flight Rangefinder Datasheet.* URL: http://www.st.com/content/ccc/resource/technical/document/datasheet/group3/b2/1e/33/77/c6/92/47/6b/DM00279086/files/DM00279086.pdf/jcr:content/translations/en.DM00279086.pdf.