A Novel UAV for Interaction with Moving Targets in an Indoor Environment

Aaron Miller, Levi Burner, Evan Becker, Ritesh Misra, Andrew Saba, Liam Berti
University of Pittsburgh - Robotics and Automation Society

ABSTRACT

A novel design for an autonomous drone capable of reliably interacting with ground robots in a GPS-denied environment will be presented. A variety of ideas from various areas of robotics were applied to solve the many aspects of this problem. Motion controllers and plant models were derived to allow for execution of fast trajectories. Sensors for state estimation were carefully selected and fused in an extended Kalman filter to support control at high speeds. Custom vision algorithms were developed for localizing the drone in the arena and detecting the target robots, and a custom target tracking filter was designed. All of these were implemented in a custom software stack that allows for quick integration and experimentation with new designs and features. Together, they form a full system which is capable of performing all of the tasks required for Mission 7a of the IARC.

INTRODUCTION

Statement of the Problem

Mission 7a of the International Aerial Robotics Competition (IARC) requires teams to develop autonomous aerial vehicles capable of herding ground robots in an intended direction and avoiding dynamic obstacles. Further, it must navigate an indoor environment without GPS or SLAM. Completion of the mission requires state estimation, target identification, and obstacle detection via optical methods. In addition, robust and repeatable control systems are necessary for target interaction.

Conceptual Solution to Solve the Problem

The aerial vehicle discussed in this paper is a quadrotor. The main computer is an NVIDIA Jetson TX2 which processes images from the bottom and side facing cameras and performs all navigation. A second Jetson is used to execute additional computer vision algorithms. A Seriously Pro Racing F3 Evo flight controller is used for flight stabilization. Finally, a Teensy 3.2 and an Arduino Nano are used for collecting sensor data and relaying it to the main Jetson.

The main Jetson handles motion planning, position estimation, and control of the pitch, roll, yaw, and thrust of the UAV. ROS is used for algorithm development and abstraction between software components.

See Figure 1 for a diagram of the design.

Yearly Milestones

In 2017, the University of Pittsburgh entered for the first time in the IARC. An early version of the software system had been completed, which allowed for basic velocity-based translation

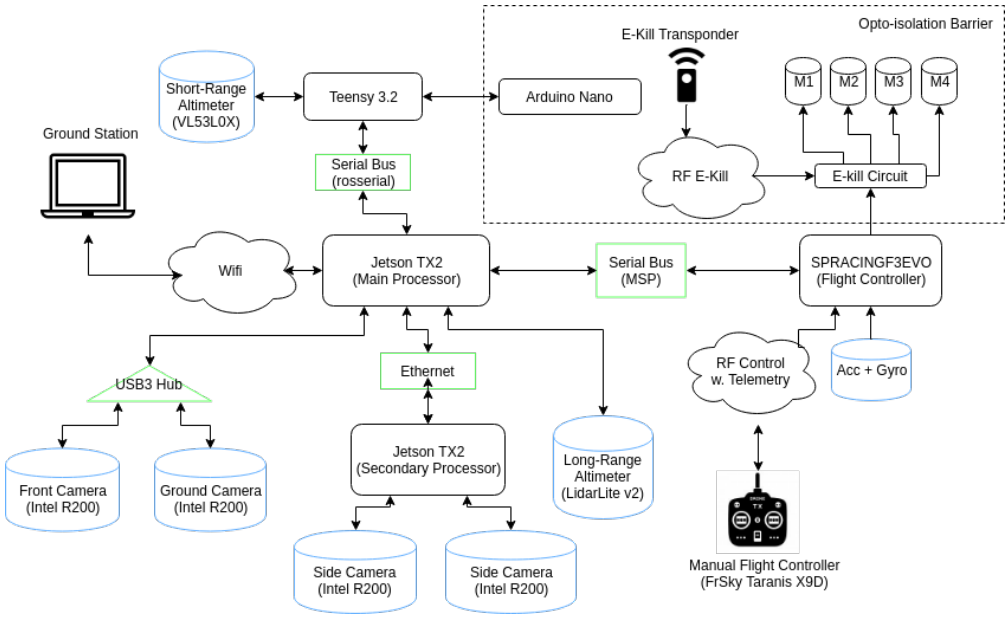

Figure 1. System Architecture

and height hold to be demonstrated. Significant efforts were made in estimation of target and obstacle positions, but the guidance system could not use the data. Thus, only achievement of velocity set-points was demonstrated.

The team set out to accomplish as much of Mission 7a as possible for the competition's final year. Significant effort was applied to improve the motion control and state estimation in order to enable precise and repeatable target robot interaction. Additionally, the obstacle detection and target identification algorithms were redesigned. Finally, a concerted effort was put into motion planning to achieve robust obstacle avoidance. At the 2018 event, we expect to autonomously interact with multiple target robots and guide at least one across the arena.

AERIAL VEHICLE

The aerial vehicle was designed to operate as a re-configurable medium-lift platform. The frame is constructed from carbon fiber plates and tubes. 3D Printed ABS brackets are used to attach all carbon fiber components. The brackets were designed to break in the event of a crash, as they are easy to replace. The complete vehicle weighs approximately 3.8 kg, and is pictured in Figure 2.

Propulsion and Lift System

Propulsion is provided by four brushless motors (KDE2814XF-515 from KDE Direct) paired with KDEXF-UAS35 Electronic Speed Controllers (ESCs) and 12x6 dual-bladed APC propellers. The propulsion system is powered by a pair of 6S1P 5.2Ah 15C Tattu LiPo batteries in parallel. This provides a maximum thrust of 10 kg; the nearly 2.5 to 1 thrust/weight ratio allows for responsive flight.

Figure 2. Image of the drone showing the frame, target bumpers, propulsion, batteries, and landing gear.

Guidance, Navigation, and Control

Stability Augmentation System

The flight controller runs a custom version of the Cleanflight open-source software. It contains a built-in gyroscope and accelerometer to sense the attitude of the UAV. Stabilization of orientation is supported using a pure gain controller on the angular error, cascaded with angular rate PID controllers. In order to improve disturbance rejection, Cleanflight was modified to cascaded PID controllers to achieve stabilization of orientation.

Guidance

The software system includes a multi-layer guidance 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 interactions with ground robots. Each of these tasks is capable of generating dynamically consistent motion profiles to achieve their goals. These profiles are generated by a search-based planner with built-in obstacle avoidance, or by a naive local planner passed through an obstacle avoidance algorithm. These safe plans are handed to the low level motion controller shown in Figure 3.

The tasks for larger actions use a heuristic search-based planner with motion primitives derived to minimize a linear combination of time-to-goal and control effort. The details of the planner can be found in [3]. Modifications were made so that planning can occur in real time on the Jetson. This allows the guidance software to generate trajectories that both respect the jerk limits of the vehicle and avoid obstacles.

Navigation

Estimation of the drone's pose and its derivatives is separated into two problems: estimation of orientation and estimation of translation. Estimation of orientation and rotation rate are done entirely by the flight controller. Estimation of translation, including velocity and acceleration, is accomplished by our software stack. This is done using an extended Kalman Filter (EKF) based on the implementation in [4], modified to allow for decoupled estimation

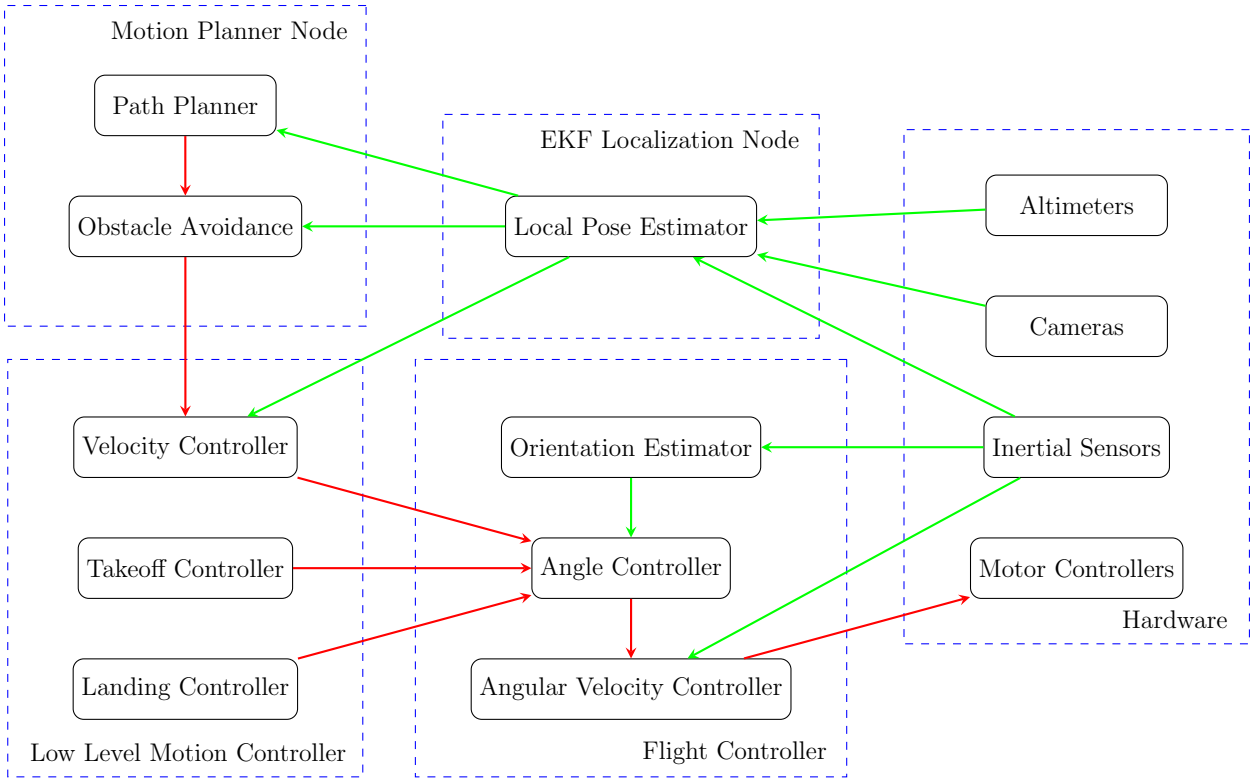

Figure 3. Control system architecture (From [1])

of orientation and translation as described above.

Multiple sensors are fused in the EKF to provide good estimates of accelerations, velocities, and positions. These sensors include two time-of-flight LIDAR altimeters, the accelerometer on the flight controller, optical flow measurements, and global position estimates from the cameras.

Control

A custom controller for following desired trajectories (referred to as motion profiles) was designed and implemented to run on the Jetson. It was coupled with a custom plant model that accounts for the dynamics of rotor spin-up and spin-down. The system resulted in robust and repeatable setpoint tracking, which are both necessary for reliable target interaction. This solution improved upon popular drone flight control systems such as PX4 and Ardupilot by adding an interface to specify acceleration setpoints.

The controller was derived from a position-velocity controller. PID was used to achieve velocity setpoints, and pure gain control was used to generate velocity corrections for positional errors. To prevent the velocity error from being numerically differentiated and injecting significant noise into the controller, the k_d term was replaced with a multiplier on the error in acceleration. Figure 4 details the full controller design. It shows how the feed-forward setpoints are fed into the controller at each stage and how the error compensation is calculated.

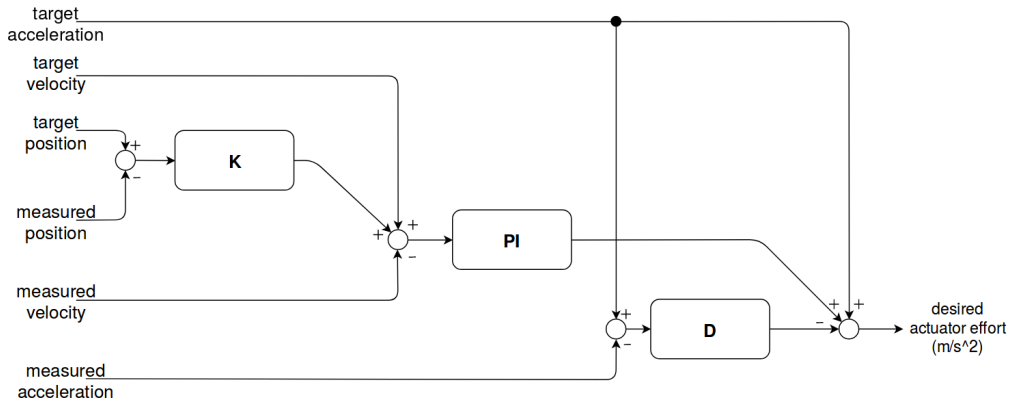

Figure 4. Controller implementation that is used for control of each axis. Estimates of state come directly from the EKF and the setpoints come from the guidance portion of the software.

An instance of this controller is run separately for each axis of the drone in the global frame. The resulting 3D acceleration vector is then used in a mixer stage which sets the pitch and roll of the drone to match the direction of the acceleration vector.

A custom plant model is used to predict the voltage necessary to produce a desired thrust based on an estimate of the current thrust. The model was designed from the ground up to allow for quick and easy characterization of any propeller, brushless motor, and electronic speed controller. To construct the model, a dynomometer test stand was used to measure the effect of unit step throttle changes at all operating points. Over 100 steps were recorded for each propeller-ESC-motor combination candidate. The responses were used to develop exponential functions that fit the thrust change as a function of time. This set of functions was then used to generate a map of starting thrusts, ending thrusts, and required voltages to achieve a desired transition within a single controller time-step (Figure 5).

Flight Termination System

Flight termination needed to be simple and power efficient. A bank of power MOSFETs controls the current to each of the electronic speed controllers that drive the motors. The MOSFETs are controlled by a timing circuit that listens to the output of a dedicated radio receiver and controller. When a switch on the controller is flipped, the pulse width sent from the receiver is modified. The safety circuit detects this and turns off the MOSFETs, which then disconnect the electronic speed controllers from the battery, rendering the drone 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. [2].

In additional to the power based flight termination system, the flight controller will terminate the autopilot's control and accept commands from a human "safety" pilot upon receiving the appropriate commands via the safety pilot's controller. The safety pilot can then fly the drone as they would fly a normal drone. This allows for more graceful recovery from errors than the kill switch solution.

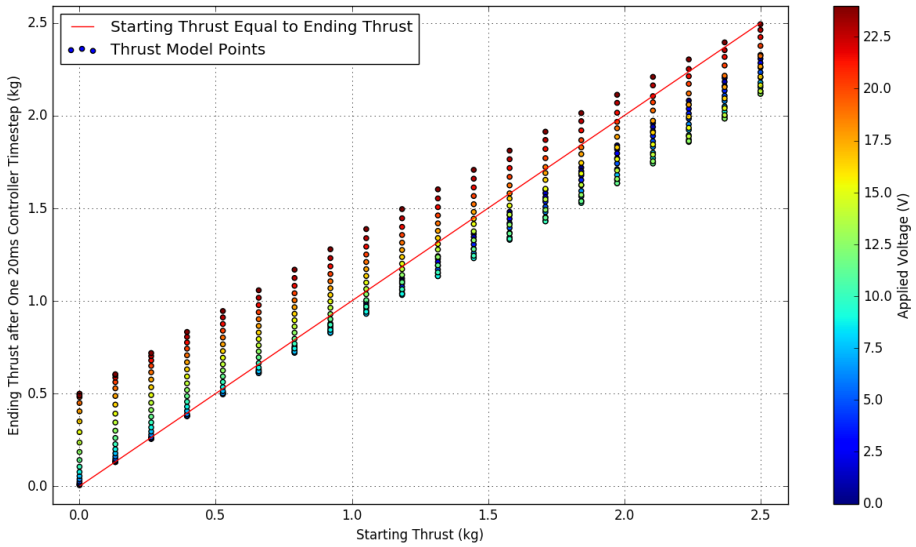

Figure 5. Thrust model for the KDE2814XF-515 from KDE Direct, KDEXF-UAS35 ESC, and 12x6 dual-bladed APC propellers with 20 ms update intervals assumed.

PAYLOAD

The following section will describe the sensors and computers used to accomplish the goals of Mission 7a.

Sensor Suite

GNC Sensors

The height measurements are made with a LIDAR-Lite v2 time-of-flight laser rangefinder as well as a ST VL53L0X TOF ranging sensor. The LIDAR-Lite works up to a height of 40 meters, but is very noisy at altitudes below a half-meter. The VL53L0X, on the other hand, has little noise, but does not work above approximately one meter. The combination of these two sensors gives accurate height estimates at all altitudes allowed by the competition rules.

Additionally, several sensors on board the flight controller are used for state estimation. First, the flight controller's onboard accelerometer and gyroscope are fused by the flight controller in a Mahony filter to estimate the drone's orientation and rotation rate. The accelerometer data is also used for estimation of position and its derivatives in the EKF localization ROS node. (Figure 3).

Finally, the drone has a downward-facing camera on the bottom that is used for drone and ground target state estimation. An optical flow implementation determines the UAV's velocity while rejecting flow vectors near the target robot, as shown in Figure 6.

The bottom camera also runs a detector that is capable of estimating the location of the grid relative to the drone, providing an estimate of absolute position that is fused into the

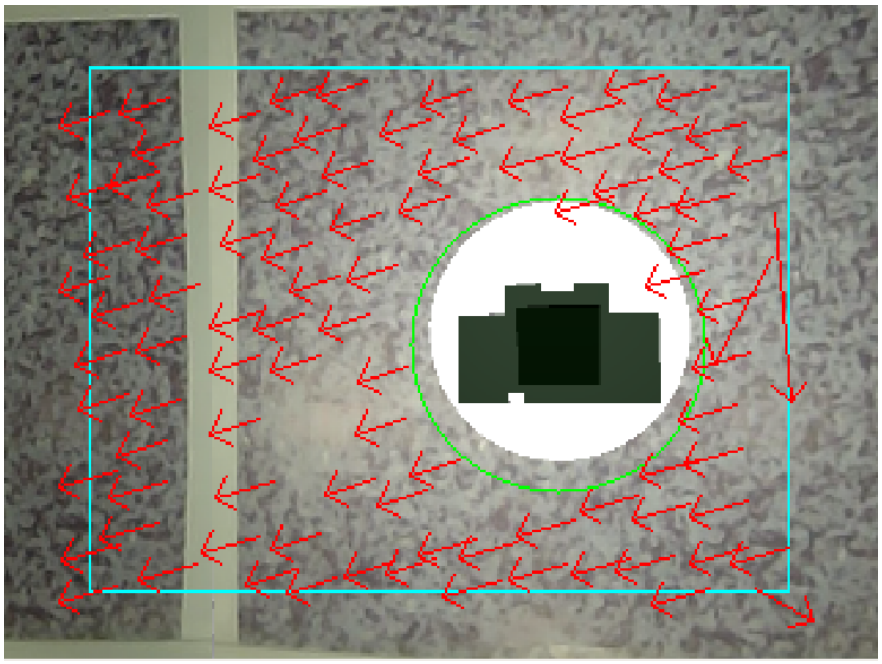

Figure 6. The flow vectors detected by the optical flow algorithm with the drone in flight. Displacements between the previous camera frame and the current frame are shown in red. Vectors with endpoints outside of the blue box are rejected because they are likely to be false matches to features that are no longer in frame. Vectors with endpoints inside the green circle (the estimated location of the target robot) are also rejected, because the target's velocity does not correspond to the drone's velocity.

EKF localization node, eliminating any drift in the odometry estimates.

Mission Sensors

To identify targets and threats during the mission, multiple Intel RealSense R200 stereoscopic cameras are used. One of these is the bottom camera mentioned in the previous section; others are located on the sides of the drone for long range target identification and threat detection.

Target Identification

The RGB images from the R200 cameras are used in two different target detection algorithms. The first target detection algorithm, utilizing the bottom camera, thresholds the input image to look for the color of the red and green top plates on the targets, does some filtering on the result, and then detects blobs in the resulting image which roughly match the expected shape of the top plate by computing moments of each blob. The moments of the blob can then be used to estimate the target's orientation and a confidence in that orientation up to a rotation of 180 degrees. To further disambiguate the orientation, the algorithm computes the average color in each of the four corners of the bounding box around the top plate and determines which corners are white and which are red or green.

For target detection with the side cameras, we use a modified version of the TinyYOLO convolutional neural network [5]. This is able to detect target robots without using hand-crafted features such as top plate color, which are unreliable for long-range target detection. Our testing showed that the TinyYOLO-based detector is capable of detecting targets at ranges up to the other side of the arena with high precision and recall.

All detected targets from the cameras are fed to a filter. This filter is actually a collection of filters, one for each target the drone is currently aware of. Each detection is matched to a target that is currently being tracked or determined to correspond to a new target that has not been previously seen. Once the detection is matched to a filter in the collection, it is integrated with the other measurements in that filter. Each individual filter is a state machine which tracks the target's state (i.e. whether it is moving or not), and depending on its state, fuses measurements in one of two Kalman filters. The first is used when the target's orientation is not well known. It estimates in the state space of the target's position and velocity. The second is applied once there is an estimate of orientation, performing estimates in the state space of position, yaw, and yaw rate.

Threat Avoidance

Depth images from Intel RealSense cameras are used to detect obstacles in the arena (the PVC pipes mounted on obstacle robots). Points from the cameras are clustered using DBSCAN, and then each cluster is fit to a model of the obstacle robot. In a similar method to that described for the target robots, the obstacle detections are fed into a filter which generates smooth estimates of their positions and velocities over time.

Communications

The UAV utilizes a FrSky Taranis X9D transmitter to allow an operator to communicate with the on-board flight controller in the event that the autonomy software fails. The safety switch communicates over 2.4GHz with a FlySky FS-GT2B transmitter. Additionally, 2.4GHz and 5.0GHz WiFi and Bluetooth are used to provide communication from the on-board computer to a ground station.

Power Management System

Two 6 Cell 22.2V 5.2Ah 15C discharge LiPo batteries are used to power the propulsion system and safety circuit. A second 3 Cell 11.1V 1Ah LiPo battery is used to power all other electronics. The two electrical systems are isolated from each other and communication over the barrier is achieved through opto-isolation IC's.

Separate electrical systems allow sensitive electronics associated with autonomy to be isolated from noisy transients induced by the electronic speed controllers. This improves the reliability of the drone and prevents wiring mistakes in the propulsion electronic system from damaging lower voltage components.

OPERATIONS

In this section, the preparation steps for launching the drone will be described.

Flight Preparations

Prior to connecting the propulsion battery, the following checklist must be completed:

1. Frame and wiring harness is checked for any failures
2. Battery level is checked
3. Radios are turned on
4. Control battery is connected
5. Flight controller is calibrated and checked manually
6. Propulsion battery is connected

The accomplishment of the previous checklist establishes flight readiness of the drone. The startup of the autopilot is accomplished using the following checklist:

1. Flight termination capability is tested through an indicator LED and the safety radio
2. Human operator flips the arming and autopilot switches on, enabling autonomy
3. Autopilot software is started from the ground station

Automated Checks during Flight and Startup

The software stack runs a variety of checks prior to takeoff. It does this by enforcing a startup order that will halt each ROS node until all of its dependencies (including sensors and other nodes) are ready. This means that no autonomous behavior can begin until all sensors and nodes are available to the system and working properly. If any problems are found, the initialization is canceled and the autopilot terminates.

Man/Machine Interface

There are three sources of man-machine interaction: an RC kill switch, an RC safety pilot controller, and a WiFi connection to a ground computer used for monitoring. The kill switch allows flight to be terminated at any time as described in the Flight Termination section. Similarly, the safety pilot controller can be used to take over control from the autopilot at the flight controller level. Finally, the ground computer interfaces with the autopilot through ROS, allowing high-level commands to be sent to the drone. The ROS software then communicates back status information and sensor data (only used for monitoring purposes). An example of monitoring some the drone's estimates of the arena on the ground computer is shown in Figure 7.

RISK REDUCTION

The methods of reducing risk during development and flight are described in this section.

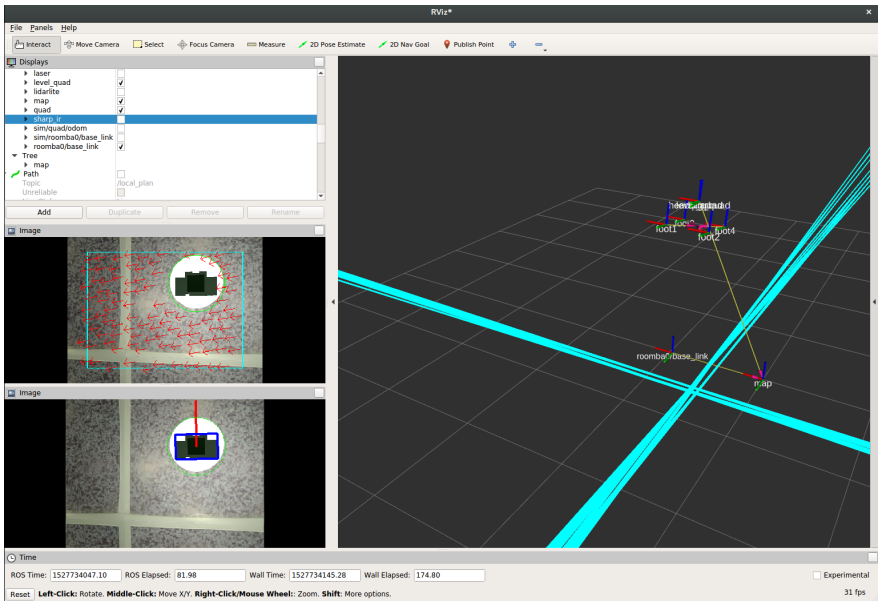

Figure 7. RViz display showing a visualization of what the drone is seeing. Estimated target location is shown in the bottom left camera image, flow vector estimation is shown in the top left camera image, and the estimated location of gridlines, the drone, and the target robot is shown in the 3D view on the right.

Vehicle Status

As discussed in our paper last year [1], we utilize a custom safety monitoring system. This "Node Monitor" maintains bonds with critical software components, and in the event these bonds are broken, sends messages to other nodes in the system to activate a safety response. The safety responses are hierarchical, meaning that if a high-level node goes into a failure state, then a lower-level node will execute a safe descent. At the bottom level of the hierarchy, the flight controller communication node will drop the throttle to a value which will slow the drone's fall while maintaining a level orientation. This year's software stack builds upon this system by adding more nodes into the Node Monitor system, ensuring that in the event of a problem with any sensor, the UAV can respond safely.

Furthermore, automated checks are performed by all high level guidance actions to verify that a variety of preconditions are satisfied. During translation, further checks are continuously executed to ensure maximum velocities are not exceeded and a minimum height limit for translation is not violated. All other tasks perform similar checks, but the details will not fit in this paper. If any check is violated at any time, a specially designed recovery action can be engaged in order to maintain the drone in a safe state and attempt to recover from the error without terminating autonomy.

Shock & Vibration Isolation

Because our propellers are well balanced, most of the on-board equipment did not need to be vibration isolated. The exception was the flight controller, which contains an accelerometer and a gyroscope whose measurements are very sensitive to noise. To reduce noise, the flight

controller unit is mounted with rubber standoffs.

The drone has a compression spring mounted on the bottom of each foot to protect against shock on landing.

EMI-RFI Solutions

EMI and RFI are managed through conventional methods. Radio devices and antennas are provided with a mostly unobstructed line of sight to operators and the ground station transceiver. Special care is taken to ensure that conducting materials, such as carbon fiber, are not encapsulating any RF device. Traces on custom circuit boards are made so that high frequency traces do not cross underneath integrated circuits, meander excessively, or run closely parallel to other traces.

Safety

Safe shutdown mechanisms are available for each point of software failure as provided by the status monitoring software described in the Vehicle Status section. Failures are also relayed to the ground station so that operators are made aware of the situation. If autonomous failure resolutions are not sufficient, either the human safety pilot can take control or the safety switch can be engaged.

Modeling and Simulation

The mission is simulated using MORSE, a software package that is built on Blender and the Bullet physics engine. The simulator publishes to the same ROS topics as our hardware sensors, allowing us to debug undesirable behavior without having to put actual hardware at risk. The ground robots are simulated with the same movement patterns that they will exhibit at competition. The simulator can also be configured to provide ground truth values for a variety of sensor outputs, such as the drone's pose or the poses of the ground robots, allowing us to isolate and debug faults within different parts of our system without running the entire software stack.

Testing

Simulations were used to test the guidance systems. Unit tests were developed for individual components, while larger integration tests allowed many parts of the stack to be tested with a variety of different sequences of maneuvers. These same tests were then run on the real drone to validate the controller designs on physical hardware. Additionally, sensors have programs that can be quickly launched to validate their operation and allow for in-depth testing.

A Crazyflie 2.0 mini drone was used as a safe and efficient platform for controller testing because of its small mass (27g), crash resilience, and interchangeable parts. The same software that runs the main drone can run on a laptop to control the Crazyflie. This allowed controller changes to be quickly tested in realistic settings. Thus, problems with the real drone that were not reproducible in the simulator could be debugged on the Crazyflie.

Full system testing at the time of writing has been restricted to an indoor space. The floor pattern is similar to that of competition, but the space is smaller. Takeoff, translation, landing, and target interaction have been successfully run many times. Future testing will take place on official arena-style flooring that is two thirds of the size of the full arena area. Interactions with four targets and two obstacles will be tested in this arena.

CONCLUSION

Between the 2017 and the 2018 competition, significant improvements have been made in controls, navigation, and target detection. As a result, repeatable navigation and motion profile tracking has been achieved, and reliable target tracking and interaction has been demonstrated. The remaining time before competition will be used to demonstrate more advanced behaviours such as guiding targets across the finish line. A serious challenge will be the inability to test in a full arena due to budget constraints. However, a 2/3 scale arena and approximately half the total targets and obstacles will be available for testing. It is expected that at the 2018 competition, interaction with multiple targets will be achieved and at least one target will be guided across the finish line.

ACKNOWLEDGMENTS

We would like to thank the University of Pittsburgh Swanson School of Engineering for providing space and resources. We also thank Dr. Samuel Dickerson and staff members Jim Lyle and Bill McGahey for their assistance and expertise. Additionally, the University of Pittsburgh and the Electrical and Computer Engineering Department were an essential part of funding this ambitious design. Support was provided in part by the University of Pittsburgh Center for Research Computing through resources provided. Finally, we'd like to thank our corporate sponsors, Rockwell Automation, KDE Direct, and Solidworks, for their financial support. Without their support this project would never have gotten off the ground.

References

[1] *Aerial Robot Design for Ground Robot Interaction and Navigation without Landmarks*. 2017. URL: http://www.aerialroboticscompetition.org/assets/downloads/2017SymposiumPapers/UniversityofPittsburgh.pdf.

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

[3] S. Liu et al. "Search-based motion planning for quadrotors using linear quadratic minimum time control". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 2872–2879. DOI: 10.1109/IROS.2017.8206119.

[4] T. Moore and D. Stouch. "A Generalized Extended Kalman Filter Implementation for the Robot Operating System". In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[5] J. Redmon and A. Farhadi. "YOLOv3: An Incremental Improvement". In: *ArXiv e-prints* (Apr. 2018). arXiv: 1804.02767 [cs.CV].