Autonomous Quadrotor for the 2017 IARC by Team Elikos

Christophe Bédard, Riad Mohamed Gahlouz, Pierre-Yves Lajoie,
Arnaud Paré-Vogt, Justine Pepin, Eva Terriault
Team Elikos – Polytechnique Montréal

ABSTRACT

The aim of this paper is to present team Elikos' system design for a fully autonomous vehicle capable of solving IARC mission 7a. Using various sensors including GNC sensors, cameras, and lasers, the vehicle shall be able of robot interaction while avoiding obstacles, and present autonomous navigation capabilities based on computer vision and sensors analytics work. This paper emphasises the work of team Elikos for the 2016-2017 year.

INTRODUCTION

Statement of the Problem

Mission 7a of the International Aerial Robotics Competition involves a sheep and shepherd problem wherein the team's aerial robot must herd terrestrial robots, hereinafter referred to as targets, by either triggering the top touch paddle or the bump sensor on their front side. The targets must be herded towards a green line within a 20x20-meter arena while dodging obstacle robots made of large PVC pipes roaming the arena in a circular motion. Mission completion is achieved when at least 4 targets, which have been interacted with, cross the green line.

Yearly Milestones

Team Elikos participated in the 2014 IARC edition with a Turnigy Talon V2 quadrotor frame modified to hold cameras and an embedded computer as well as wireless communication peripherals. The system was capable of relative position estimation through optical flow and was controlled by an automated ground station. In the 2015 IARC edition, our performance featured an aerial vehicle capable of target identification and pursuit, as well as an improved positioning system using SLAM and a completely revisited platform. Last year, Elikos presented a vehicle with an improved platform, target identification and pursuit, as well as a revisited positioning system. Some major work had also been put in the obstacle avoidance, interaction with grounds robots and functional artificial intelligence modules, although the IARC performance did not demonstrated all those functionalities.

Thereby, this year's progress for team Elikos follows this work. As described in the present paper, we intend to demonstrate interaction with ground robots as well as stable and fully functional positioning system. Furthermore, the decision-making module of our autonomous vehicle has been greatly improved, thus connecting the different modules and bringing us one step closer to resolving the mission. Finally, some work have been put in the obstacle avoidance module, although we do not plan that the full integration with the rest of the operations will be accomplished by the time of the 2017 IARC edition.

Conceptual Solution to Solve the Problem

This year's solution for team Elikos has been developed based on challenges brought by the nature of Mission 7 itself as well as previous years learnings, as listed below:

GPS-denied environment: To compensate for the sterile environment as state by the competition rules - i.e. without externation navigation aids - the arena is filled with a 1-by-1-meter grid. As opposed to previous years, where we focused our work on the surrounding environment's visual features (ceiling and walls), this year's work is entirely based on the grid's features, and entirely developed by our team. With this line of work, we were able to come with a solution specifically focused on the mission's specifications, especially regarding the indoor aspect as well as the movements of our vehicle. Furthermore, we were able to test our solution both in virtual and physical environments almost identical to the competition's. Our solution is described in the *Stability Augmentation System* section.

It is important to mention that the position estimation from optical flow integration can quickly diverge from ground truth since every iteration adds small errors to the estimation. Thereby, this estimation, obtained by the PX4Flow Smart Camera, is fused with our visual solution, as explained in the *Control System Architecture* section.

Robots interaction: The second biggest challenge comes from the interaction with the robots, requiring us to perform nearly aggressive vertical movements, and to approach the ground significantly. A direct consequence of this is that we lose all of our visual localization when doing so, both from the grid's features and from the optical flow - since the lens of the PX4Flow camera is fixed focus. Furthermore, the ground effect can make the vehicle drift quickly with no means of it knowing so.

We tackled this problem by bypassing some modules of our usual pipeline in order to program those delicate maneuvers, as described in section *Target Interaction.*

High-level commands and strategy: For the first time this year, we implemented a strategy module calculating the next robot we should interact with. The strategy is based on the position of the robots in relation to the green and red lines. The key aspect of this functionality lies in the ability to know - or estimate - the state of every ground robot. This challenge was tackled by the tracking module, as described in section *Guidance, Navigation, and Control.*

Overall hardware architecture: This year's solution is very similar to previous years regarding the hardware components. Indeed, we still use an onboard computer based on an Intel i5 processor and the Pixhawk open source flight controller, as those are high-performance off-the-shelf products. The Pixhawk offers an excellent development support and gives us the flexibility that we need. As mentioned before, we use the PX4Flow for optical flow position estimation, and the down-facing LIDAR-Lite laser for altitude measurement. Figure 1 presents an overview of our vehicle's hardware architecture.

This year's biggest alteration lies in the integration of mechanical switches under the landing gear to detect both interactions with ground robots and takeoff and landing operations.

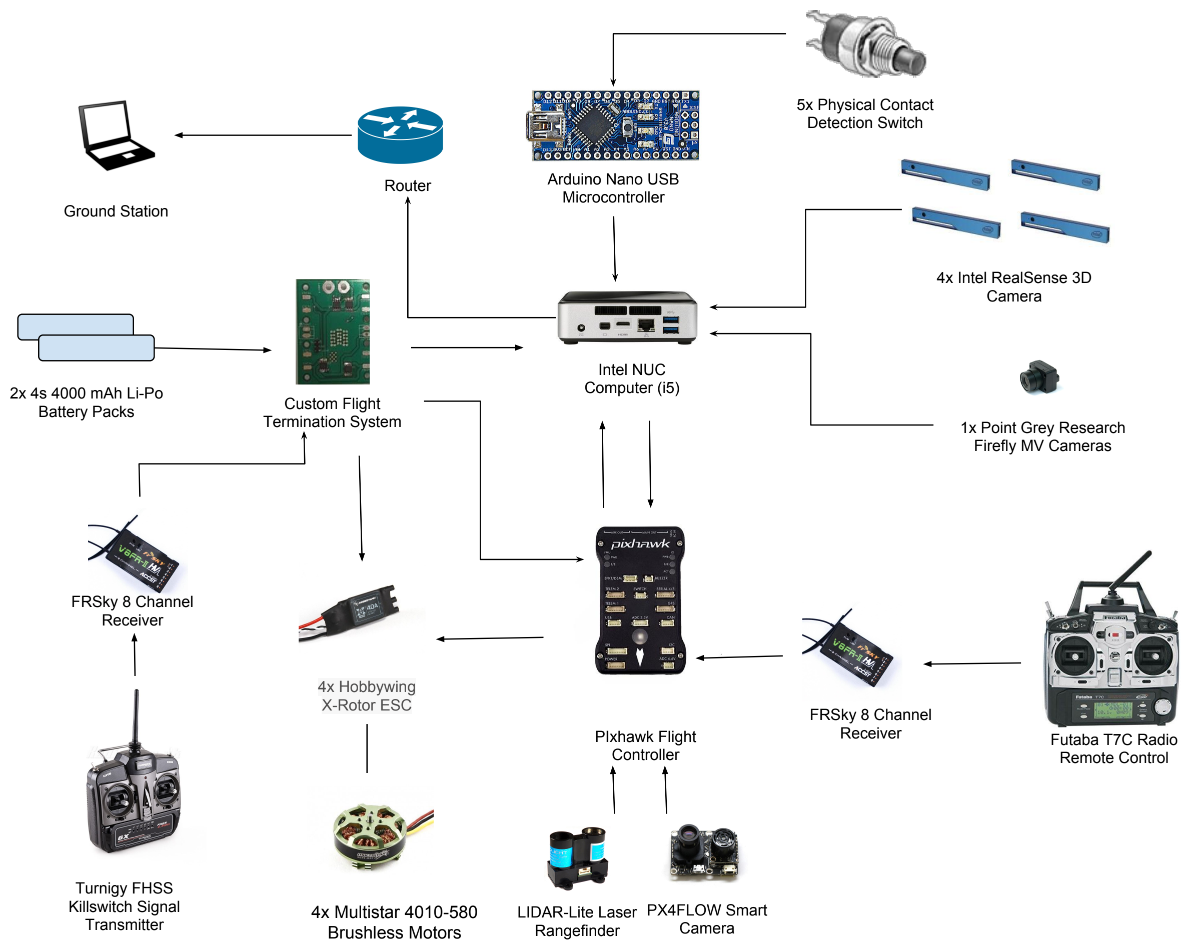
5x Physical Contact Detection Switch
Ground Station
Router
Arduino Nano USB Microcontroller
4x Intel RealSense 3D Camera
2x 4s 4000 mAh Li-Po Battery Packs
Custom Flight Termination System
Intel NUC Computer (i5)
1x Point Grey Research Firefly MV Cameras
FRSky 8 Channel Receiver
4x Hobbywing X-Rotor ESC
Plxhawk Flight Controller
FRSky 8 Channel Receiver
Futaba T7C Radio Remote Control
Turnigy FHSS Killswitch Signal Transmitter
4x Multistar 4010-580 Brushless Motors
LIDAR-Lite Laser Rangefinder
PX4FLOW Smart Camera

Figure 1. Overall hardware architecture.

AIR VEHICLE

Propulsion and Lift System

Our air vehicle, shown in Figure 2, is an improved iteration of our previous vehicles, being a X configuration quadrotor with a 500-millimeter frame (diagonally) providing great structural simplicity and stability as well as easy mounting of all necessary sensors due to its large surface area. Its 191-gram carbon fiber sandwich panel makes it very light while providing good structural rigidity as a result of the greater statical moment of area. The 12”x4.5 tri-blade folding propellers, 45 mm diameter Turnigy Multistar 4010-580 brushless motors, 30 A Hobbywing ESCs, and 8,000 mAh 4S 45 90C Turnigy lithium polymer batteries can generate a total of 5500 g of thrust, thus accommodating payloads from 2.5 kg up to 3 kg at 50 % thrust with usual flight times of about 11 to 13 minutes.

Guidance, Navigation, and Control

Stability Augmentation System

Our vision localization system is based on feature tracking of the arena’s grid and thereby produces a position and yaw estimate of the vehicle’s state. This estimate is then used in

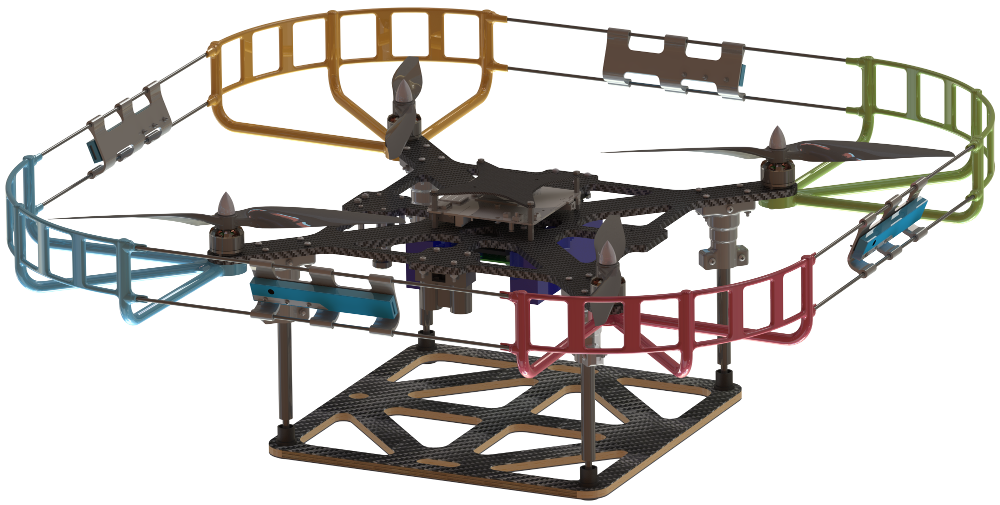
Figure 2. Propulsion and lift system.

the FCU's position estimate as explained in the Control System Architecture section. The procedure goes as follows.

1. *Intersection Detection* In order to detect the line intersections, the image's perspective is removed, and the white lines are detected with the Canny algorithm [1] and merged based on their position and angle. The intersections are then trivial to detect.
2. *Pose calculation* Calculate the camera's pose relative to the intersections using the *PnPRANSAC* algorithm available in *OpenCV* [6].
3. *Prediction* The points positions are predicted based on the vehicle's estimated state i.e. its pose, orientation and linear acceleration. The prediction is generated using an unscented Kalman filter. The filter also integrates data from other sensors (e.g. the IMU) in order to make it more reliable.
4. *Matching* The points and their predictions are matched, based on the smallest euclidean distance.
5. *FCU communication* Calculate the absolute position of the drone and send it to the FCU. At the same time, the calculated position is used to refine the prediction of the next set of points.

Navigation

Figure 3 presents the navigation pipeline for the path planning. The first modules are dedicated to targets detection and tracking to provide accurate information about the state of the ground robots in a way that the decision making module can determine which target trajectory needs to be modified.

Target Detection

There are three layers in the target identification process, as shown in Figure 4. Each of them improves the certainty of a detection.

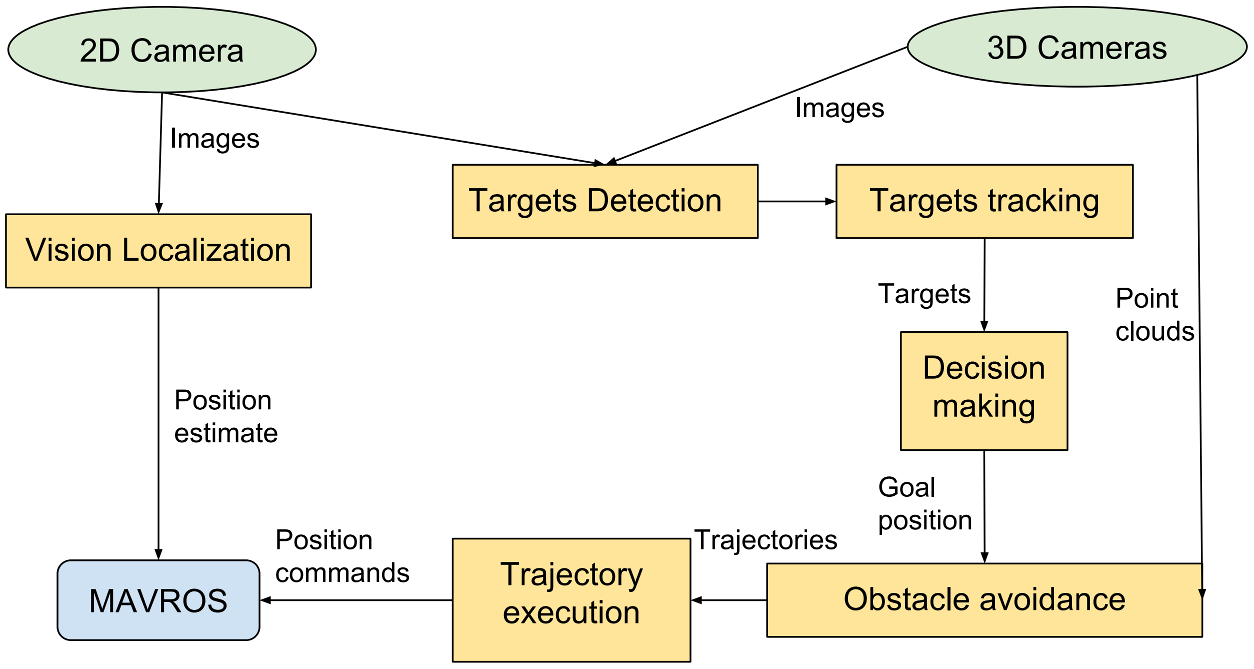

Figure 3. Software architecture.

The first layer consists of color blob detection using HSV filters included in OpenCV to extract the colors of the target robots (red and green) in the images captured by the cameras. Blur, dilations and erosions are applied to the image to filter the results by reducing noise. Since this technique demands a very accurate color calibration, a remote calibration tool was created for the cameras that will be used prior to our performance to adjust to the environment's lighting. The results are then passed to the second layer.

The second layer is a blob shape detection. It compares the first layer extracted blobs' shapes to the known rectangular target's shape to reduce false positive detections. The shape of the detected blobs is estimated using the "Structural Analysis and Shape Descriptor" [8] algorithms available in the OpenCV library.

The third layer, which is not related to the first two layers, is a circle detection. It is performed into a grayscale image using a circle Hough Transform [9] available in OpenCV computer vision library. The results are compared with the second layer's filtered results to improve the certainty of a detected target. Since the camera footage has perspective in it, we used OpenCV's perspective removal tools to remove that perspective to make the Hough Circle Transform algorithm effective.

The found circles' positions are then compared to the second layer's detected targets and the matching results are then output to other systems. Figure 4 presents a schematization of the detection pipeline.

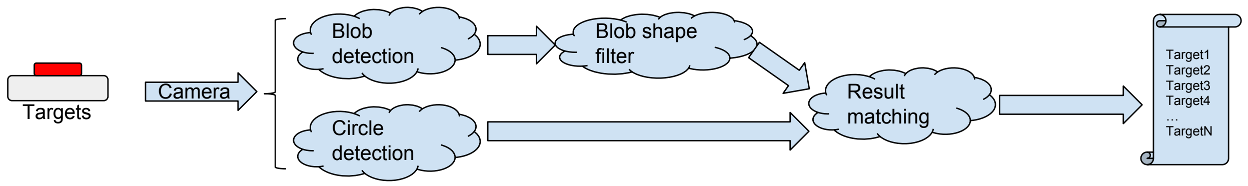

Figure 4. Target detection pipeline.

Target Tracking

The main idea behind this module is that an invisible target must not be non-existent and without any level of priority to the eye of the decision maker module. Therefore, we estimate the position of the hidden robots by using simple Kalman Filters. We use the classical 2D filter for each of the targets. Of course, with time passing, the accuracy of the tracking can drastically drop since the ground robots are moving according to an almost random pattern. However, we suppose that the relatively small arena will make it possible to track quite well even then, because a simple run back and forth can allow the drone to update quite quickly the robots' tracking data.

Decision Making

Three different strategies are implemented : Target Research, Aggressive and Preventive. In the Target Research strategy, the UAV will hold its altitude and follow the ground robot with the highest priority. The two others strategies are focused on interaction with ground robots and mission completion. The difference between them is that the aggressive one will lead the ground robots toward the green line and the preventive one will avoid any ground robot to cross the red lines. Once the target is chosen, the decision making module will switch between different behaviors based on different factors such as the positions of the selected target, the UAV itself and the desired outcome. Those behaviors can be, for example, following the target or interacting with the robot using the front side bumper or the top touch paddle. A setpoint is finally produced and sent to the obstacle avoidance module.

Obstacle Avoidance

To avoid any collision with the obstacle robots we use three Intel Realsense R200 stereo cameras positioned on the front, left and right sides of the UAV. The Intel Realsense R200 infrared stereo sensor has a range of 3 to 4 meters indoor. We use the MoveIt! package [2], originally designed for manipulator robots to perform three dimensional obstacle avoidance. Many modifications were needed to adapt this path planner to a UAV with 6 degrees of freedom. This package can be adapted to any kind of 3D sensor. Through this path planning approach, our threat avoidance system is more preventive than reactive and thus can find the optimal path between two points and reduce the needed time to complete the mission. Also, unlike a two dimensional approach, the use of stereo cameras solves the fully 3D threat avoidance problem that we will face during the part b of mission 7 when two drones will fly at the same time. When the path is correctly generated, the trajectory execution module follows it by sending the intermediate positions to the PX4 autopilot through the MAVROS interface.

Target Interaction

As the vehicle approach the ground, the multiple sensors become unusable, including the PX4Flow and the stereo cameras. Thus, we implemented a small module that allows us to bypass the obstacle avoidance with the help of preprogrammed sequences that will take over when interacting with ground robots. Some crucial aspects of the environment had to be taken into consideration, including the targets and obstacles movements, both of those being out of reach from our sensors. Moreover, we have to be careful about the drone unpredictable moves when flying at low altitude, as safety comes first when compared to the success of the interaction. Finally, the transition between this blinded mode and the normal decision maker module has to be smooth as some delicate operations cannot be interrupted.

Control System Architecture

As explained above and shown in Figure 5, our vision-based pose estimation is used to feed the Local Position Estimator (LPE) included in the PX4 flight stack [3]. This sensor fusion also uses the LIDAR-Lite for altitude measure, the PX4Flow optical flow as well as IMU data, all of those being either connected or embedded on the Pixhawk flight controller. The estimated position is then returned to the software system through the TF package [4] in ROS designed for coordinate frames tracking via MAVROS. Then, the decision making, obstacle avoidance and trajectory execution modules can use the position of the quadrotor to perform their computation. The position estimate is also used by the control system (position then attitude control) to manage properly the motors.

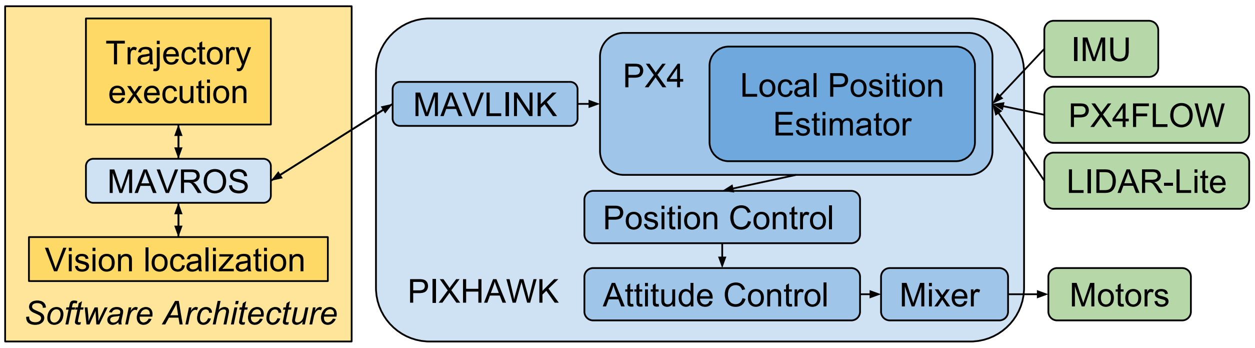

Figure 5. Control system architecture.

Flight Termination System

The flight termination system, commonly called kill switch, simply cuts off the positive voltage to the propulsion system, since cutting the ground, as proposed in the reference design, caused instabilities with other components on the quadcopter. It does not cut power to the onboard computer to avoid restarting it every time and possibly corrupting data.

The kill switch module is controlled by a RF module which was made on a different board, in order to easily change it without modifying the whole design of the kill switch. This RF module is connected to a receiver which is linked to the kill switch transmitter.

PAYLOAD

Mechanical System

Shown in Figure 6, one of the main improvements made to the airframe is the addition of a new landing gear. This new component is essential, as the previous landing gear was composed of only four contact points which didn't provide enough surface area to easily interact with the ground robots. Therefore, we added a carbon fiber sandwich panel with balsa under the structure. It greatly improves the quadcopter's ability to physically interact with the ground robots, and adds another layer of protection for the components located under the quadcopter without adding much mass.

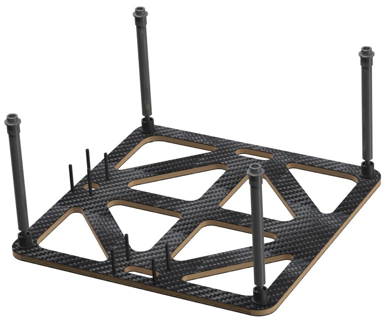

Figure 6. New landing gear for robot interaction.

Sensor Suite

GNC Sensors

The GNC sensor suite used for control is comprised of the Pixhawk's inertial measurement unit (accelerometer, gyroscope, barometer, and compass), a PX4Flow optical flow camera, an external compass, and a LIDAR-Lite laser altitude sensor. The PX4 flight stack running on the Pixhawk is responsible to fuse all of this data, as explained in the *Control System Architecture* section.

Mission Sensors

The mission sensors include the four Intel RealSense 3D camera used for robot and obstacle detection and the Point Grey Firefly camera for visual localization. Their use is described in the *Guidance, Navigation, and Control* section.

We have also integrated switches under the landing gear surface. With one switch under each landing leg, we can detect when legs touch the ground individually. This feedback is useful for autonomous landings or takeoffs. With another switch in the middle of the landing gear surface, we can detect physical interactions with ground robots. These switches are connected to an Arduino Nano, which is connected to the onboard computer through USB. The switch states are relayed to the computer through ROS messages with a `rosserial_arduino` node and are then processed by the decision-making module.

Communications

Robot Operating System (ROS) is used to coordinate software communication between sensors, data processing, and decision-making modules. With its architecture, ROS allows for decentralized computation, which makes offboard processing possible through a Wi-Fi connection. However, offboard processing of the sensor data requires a high-bandwidth connection, but the wireless bands are cluttered by the teams running their own network. Therefore, during the IARC performance, the offboard computer is only used for monitoring through a 5 GHz network.

An RC receiver and a transmitter is used to activate offboard mode or for manual control when testing. A second RC receiver is used for the kill switch transmitter.

Power Management System

Our system is powered by two battery packs containing 4 lithium-ion polymer cells each. The three main systems of our vehicle are powered this way, that is the propulsion system, the flight controller and sensors system, and the onboard computer. The propulsion system is powered through the kill switch, as mentioned before. The Pixhawk is powered by a 5 V regulator (LM2678 from Texas Instruments). The onboard computer is powered through a 12 V DC power regulator (Murata UWE-12/10-Q12P-C).

OPERATIONS

Flight Preparations

There are many software and hardware pre- and post-flight checks in place. However, manual verifications are still required to ensure performance and safety during flights, since elements such as mechanical integrity cannot be automatically verified.

Pre-Flight Checklist

The following is a checklist of elements and statuses that must be verified, at the very least, once before each extended flight session.

1. Verify that the overall mechanical structure is undamaged and that the payload is securely mounted
2. Verify that the propellers spin in the right directions and are properly tightened
3. Verify that the Li-Po batteries are sufficiently charged ($\approx$ 16.8 V) and well fastened
4. Verify that the battery alarm is connected and functional
5. Turn on and enable the kill switch transmitter
6. Turn on the manual control transmitter
7. Connect the batteries and power on the onboard computer, flight controller, and peripherals
8. Verify that radio calibration and presets are correct
9. Verify that the optical flow and computer vision cameras lenses are focused (and lens covers are removed)

10. Calibrate inertial sensors (magnetometers, gyroscopes, accelerometers)
11. Verify that telemetry and network communications are functional

Post-Flight Checklist

After each flight session or attempt, the following checks must be made.

1. Properly terminate every process and data acquisition/logging
2. Shut down the onboard computer and other peripherals
3. Disconnect the batteries
4. Turn off transmitters
5. Verify that the battery levels are over the safety threshold and physically intact
6. Verify that all component temperatures are within their normal operating temperatures
7. Verify that every mount, propeller and screw is properly tightened (especially after a hard landing or a crash)

Man/machine Interface

Prior to a flight, as mentioned in the *Target Detection* section, a remote calibration tool for the cameras is used to adjust to the environment's lighting.

In-flight, a SSH connection or a ROS remote connection allows us to monitor flight data over our Wi-Fi network. In case of emergency, the pilot's transmitter offboard switch permits manual control at all time, and the kill switch's dedicated transmitter cuts the power of all motors.

RISK REDUCTION

Vehicle Status

Shock/vibration Isolation

The sensors in the flight controller's inertial measurement unit (IMU) are vulnerable to vibrations. It introduces noise in the sensor data which can ultimately lead to crashes. In order to minimize the impact of vibrations, the Pixhawk is mounted on a gel material with vibration isolation properties.

As for shock absorption, we're using the same landing legs as the ones on the DJI Matrice 100. Since they contain both a spring and a damping mechanism, they're useful for absorbing shocks resulting from reasonable drops of up to about 1 m. The landing legs are fixed to the frame with 3D-printed symmetrical parts pressed against the legs and fastened with screws.

EMI/RFI Solutions

The EM noise generated by the propulsion system can be troublesome to the flight controller's internal compass. For this reason, we use an external magnetometer mounted on the very top, as far away from the power circuitry as possible.

Safety

The main physical safety feature is a 3D-printed prop guard. It is designed to counter shocks towards the inside of the quadcopter. Identical parts are fixed under every motor using the same mounting holes. Each part offers 90-degree protection around a motor. To get an all-around, complete 360-degree protection, we added rigid carbon fiber rods between prop guards. These rods are also useful for mounting sensors.

Modeling and Simulation

The SolidWorks software is used for the mechanical design of our custom parts. It's also useful for the integration of all other components and to optimize usable space for sensor mounting.

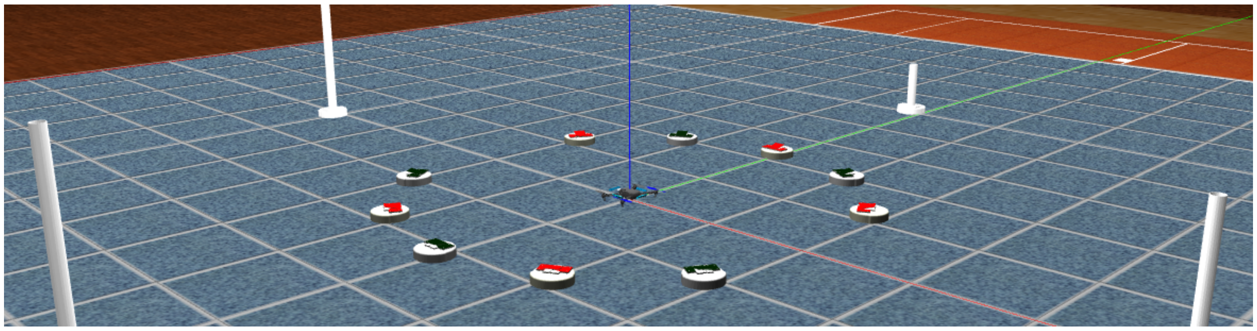

Figure 7. Mission 7 simulation in Gazebo.

As a part of our software development process, we use the Gazebo simulator [5] to reproduce the 7th mission's arena and to model the physics of a quadcopter controlled by a software in the loop (SITL) version of the Pixhawk controller. We added most of the quadcopter sensors, like the 2D and 3D cameras. Our simulation also includes the floor pattern used at the competition and the 3D models of the targets and obstacles as illustrated on figure 7.

Testing

Motion capture: Polytechnique's Electrical Engineering Department acquired a VICON Motion Capture, allowing us to fly indoor with an external navigation system. We were then able to test all of our other modules, including target identification and pursuit, high-level strategy, and threat avoidance in an environment similar to the competition's, without the localization module being finalized.

Competition environment: Last year, we acquired a 5x5 vinyl grid surface similar to the one used at the American Venue. This helped us reproduce the environment of the competition, allowing us to test our different modules more deeply than with the simulation.

Ground and obstacle robots: To test target detection and tracking, we have built ground and obstacle robots according to the reference instructions. However, for the control and programming of the Create 2 robots, we have decided not to use the reference Arduino setup. Instead, we mounted a Raspberry Pi 3 model B powered by a USB power bank on each robot.

It runs a ROS node for the ground and obstacle robots behavior implementation and uses a driver package for serial communication [7]. All robots are connected via Wi-Fi to our network. Therefore, we can control and send commands to the robots with the computer used for monitoring. We can also easily remotely control the robots with a game controller.

CONCLUSION

Elikos' fourth attempt at mission 7a of the IARC will be made with a much refined solution composed for the first time of all the required puzzle pieces. A consolidated detection module, an enhanced and configurable decision maker module, a fully integrated and adapted obstacle avoidance system, a custom and more reliable localization module, a brand new target tracking module, and a larger, more appropriate interaction landing gear for robots interaction have been united. With the combination of those, we are confident in our vehicle's ability to interact with ground robots, avoid obstacles, and navigate autonomously in the competition's environment.

REFERENCES

[1] John Canny. “A computational approach to edge detection”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698 (cit. on p. 4).

[2] Sachin Chitta, Ioan Sucan, and Steve Cousins. “MoveIt! [Ros topics]”. In: *IEEE robotics and automation magazine* 19.1 (2012), pp. 18–19 (cit. on p. 6).

[3] Lin Feng and Qi Fangchao. “Research on the Hardware Structure Characteristics and EKF Filtering Algorithm of the Autopilot PIXHAWK”. In: *Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 2016 Sixth International Conference on*. IEEE. 2016, pp. 228–231 (cit. on p. 7).

[4] Tully Foote. “tf: The transform library”. In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1–6 (cit. on p. 7).

[5] Nathan Koenig and Andrew Howard. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2004, pp. 2149–2154 (cit. on p. 11).

[6] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “Epnp: An accurate o (n) solution to the pnp problem”. In: *International journal of computer vision* 81.2 (2009), pp. 155–166 (cit. on p. 4).

[7] Jacob Perron. *create_autonomy ROS package*. 2017. URL: `http://wiki.ros.org/create_autonomy` (visited on 03/11/2017) (cit. on p. 12).

[8] Satoshi Suzuki and Keiichi Abe. “Topological structural analysis of digitized binary images by border following.” In: *Computer Vision, Graphics, and Image Processing* 30.1 (Sept. 4, 2006), pp. 32–46 (cit. on p. 5).

[9] HK Yuen et al. “Comparative study of Hough transform methods for circle finding”. In: *Image and vision computing* 8.1 (1990), pp. 71–77 (cit. on p. 5).