Development of Aerial Robot Swarm for Interaction with People and Other Robots

Damian Owerko, Joshua Chen, Andrew Butt
Ariana Harvey, Ben Kramer, Owen Simon, Edward Atter

University of Pennsylvania

June 1st, 2019

Abstract

In this paper we present Penn Aerial Robotics's solution to the 2019 IARC Mission 8. In this challenge a human player works together with several autonomous helper robots in a scavenger hunt, while under attack from sentry robots. The helper robots work together to find four storage bins and read four parts of a QR code. The QR code is reconstructed and decoded to provide a 3 digit code to open pad locks on the storage bins and retrieve objects from inside them. The human player can only issue vocal or gesture commands. Thus, the challenge requires integration of different machine perception systems for obstacle avoidance, object detection and speech recognition. We propose a solution which combines statistical and machine learning system. We use semi-direct visual odometry for indoor localization, which is fused with IMU data. A forward facing Intel RealSense camera provides a depth map which is used for obstacle avoidance. Finally a downward facing camera allows tracking of the player and bin localization.

INTRODUCTION

Statement of the Problem

Mission 8 of the International Aerial Robotics Competition (IARC) requires teams to build four drones that are capable of communicating between themselves and responding to a human player's body gestures and/or vocal commands (without electronic signals) to decipher a passcode given four separated sections of a QR code, heal the human player with healing lasers, and avoid collision with playing-field sentry robots. The human player must use the passcode to retrieve a "critical component" from four bins while commanding the helping drones and avoiding harmful laser beams from the sentry robots. The playing field is about the size of a basketball court and is situated indoors. The floor is of unknown pattern and there are various obstacles that may be used as cover from sentry robots. The goal is to complete all of the above as efficiently as possible within the time limit of eight minutes.

Conceptual Solution to Solve the Problem

In addition to full autonomy, obstacle avoidance, and tracking, drones for IARC Mission 8 also require human gesture recognition, swarm interaction (multiple drones communicating together), and head-to-head interaction with opposing drones among other capabilities. The increased complications from Mission 7 demands that a drone has a precise perception of both its own location as well as the locations of obstacles and moving targets. Hence, we split the perception system into three distinct projects: human detection with color vision tracking, storage bin detection with a convolutional neural network, and a localization system with Semi-Direct Visual Odometry (SVO) [1]. Some parts of this perception system are too computationally intensive to be performed via the UASs onboard Raspberry Pi. Instead they are carried out on a base station. A laptop will be utilized for this purpose – providing an ideal mix between portability and power.

Additionally, a mechanical subsection of the team was established for the assembly of the laser, camera, and microphone, as well as electrical components for mounting on the drones. The laser system is mounted on a gyroscope, allowing tilting and panning, facilitating target aquisition.

During the mission the unmanned aerial systems (UASs) perform two roles. The guardian follows the contestant, providing healing if necessary and listening for voice commands. The rest of the UASs are scouts. Their objective is to reach the four bins and read the QR codes and/or scan the area with their cameras. If the contestant is hit, they can issue a voice command to heal. Upon this, the guardian will swap roles with one of the scouts.

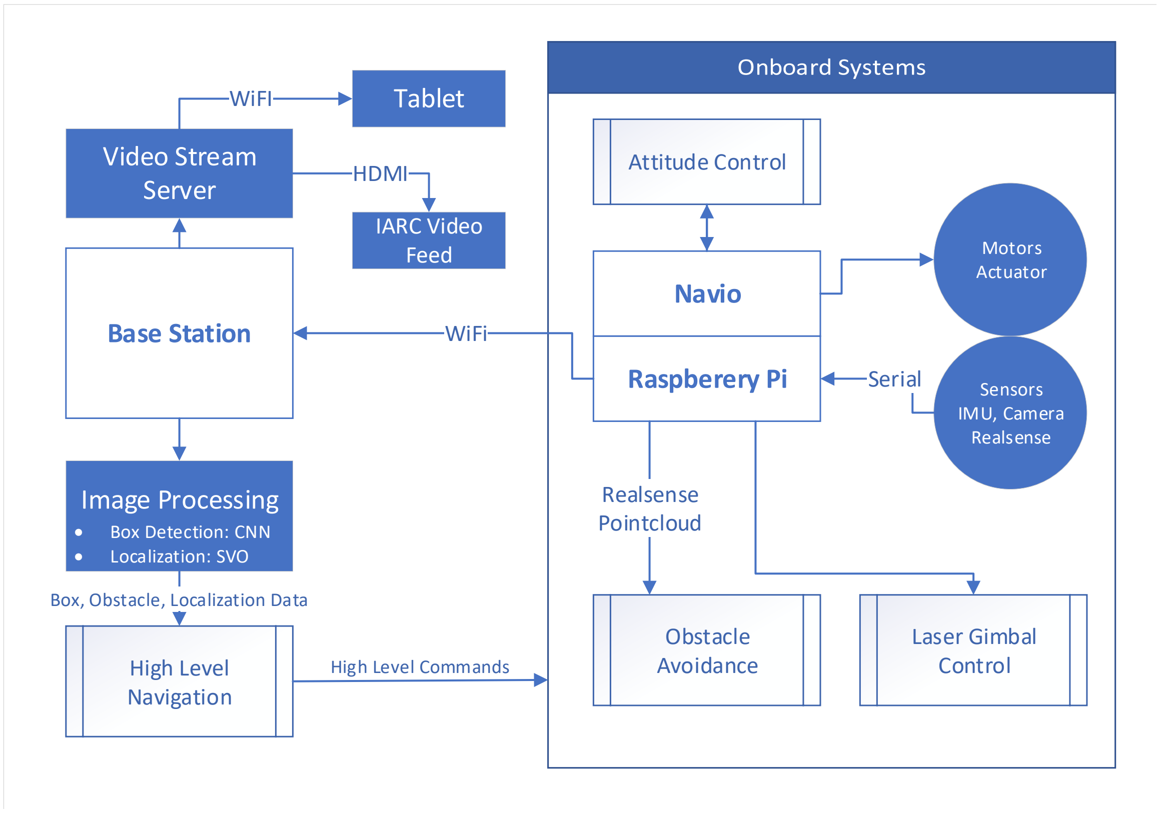

Figure 1: Overall System Architecture

Yearly Milestones

Spring 2014	– Penn Aerial Robotics is founded at the University of Pennsylvania
September 2015	– Club first pursues the IARC Competition
November 2015	– Quadcopter for IARC is designed and assembled
August 2016	– 2016 IARC Competition attended
December 2018	– Planning begins for the 2019 IARC Competition
January 2019	– Development begins for Perception System using a convolutional network
	– Development begins for drone Localization System
	– Development begins for human detection system
February 2019	– Hardware for laser and camera placement designed and tested
April 2019	– Tests begin for Perception and Localization Systems
June 2019	– Competition drone assembly completed for further testing

AIR VEHICLE

Airframe and Propulsion

The quadcopters used for this competition are based on the DJI F450 frame. They use Sunnysky X2212 980KV brushless motors controlled by HP SimonK 30A electronic speed controllers (ESC). The motors spin 10×4.5 inch propellers secured by a positively locking nut. This is powered by a 4S 5200mAh 10C MultiStar battery. The battery was chosen for its high-capacity, trading off the discharge rate (12C). This is lower than most Li-Po battery packs, but allows currents of up to 62.4A continuous. This is sufficient since the hardware kill switch is designed for 60A continuous. Therefore, it was the best choice as it has the highest capacity per weight of all comparable batteries.

The F450 airframe was chosen because it balances ease of assembly with performance. The F450 frame can be sourced from a variety of manufacturers, and uses standardized metric fasteners. Simultaneously, it provides ample space for payload. Performance testing showed that the system could perform aggressive maneuvers of up to 60 degree banks (maximum limit deemed safe to test) and could reach a top speed of 53 mph.

In order to make the propulsion system man-safe, a specialized cage was designed around the propellers. The cage frame was 3D printed and a wire was threaded

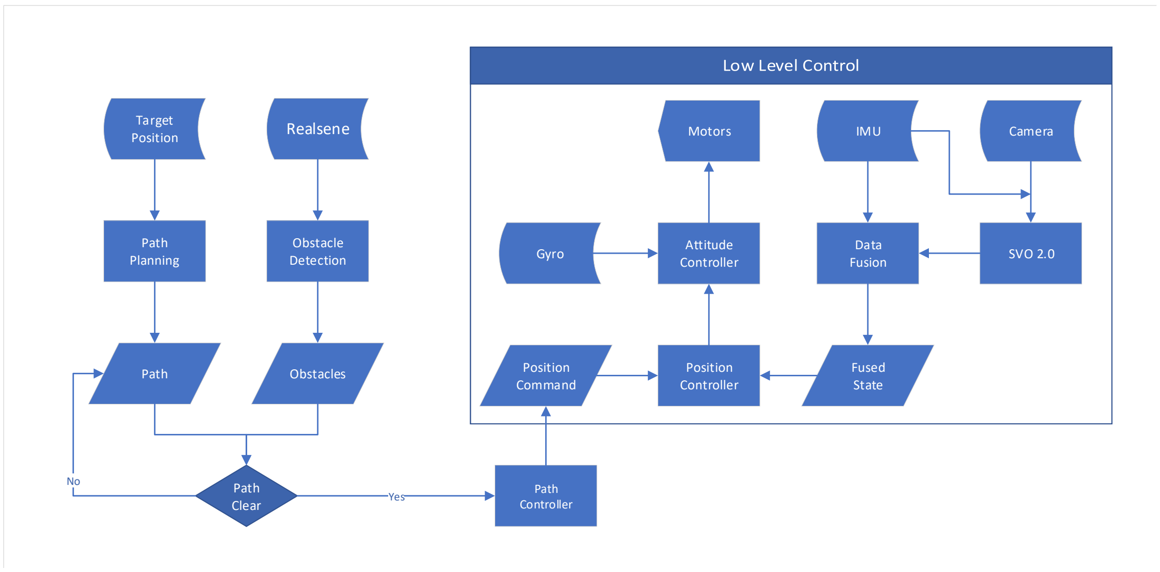

Figure 2: Control system architecture

around it. The size of the cage grid was designed to be as light as possible, while preventing any object larger than a finger from entering.

Flight Control System

Navigation/State Estimation System

For state estimation we are using the Navio's built-in IMU in combination with visual odometry to reduce drift. The visual odometry data is published to ROS [3] and taken as an input by the PX4 flight stack [2]. The visual odometry data is produced by the SVO 2.0 algorithm (Semi-Direct Visual Odometry) using images from a Raspberry Pi camera. The quadcopter is then able to navigate just as it would with only an IMU, but with drastically less drift.

Attitude/Position Control System

Top-level guidance, navigation, and control is performed on a Raspberry Pi running ROS and the PX4 flight stack. The Raspberry Pi issues commands to the Navio flight controller, which performs low level control functions and implements PID controllers for flight stabilization. High level code is written in Python. Most of it is contained within an open source ROS abstraction layer: Pennair2 - a library developed by Penn Aerial Robotics.

Flight Termination System

There are three ways in which a flight can be terminated. First, in the event of a non-critical failure of the system, the mission can be interrupted from the command line on the ground station. This leads to a safe return to the start position for the mission. Second, if the flight control system is still in control, but there is a mission failure such as loss of communication to the ground station, the system will initialize a safe landing at current position. Finally, if the event of a critical failure of the flight control system, such as loss of control or departure from the mission area, power can be cut using the independent kill switch system. Activating the kill switch cuts power from the flight controller and the motors, immediately terminating the flight. The kill switch can be remotely activated by the judges during the competition. A loss of signal to the remote activation system will also cause kill switch activation.

MISSION PACKAGE

Perception System

Target Identification

On receiving the “search” command, the quadrotors will initate a search for the four bins. The four bins are located on the end of the arena opposite the starting point, so the quadcopters will move to this end of the arena and begin their search there. Then, the imagery from the downward camera is analyzed using a Fully Convolutional Network (FCN) trained with a model based on VGG-16 [4] to identify the bins. Once the bins are identified, the drones then adjust their position until they are centered over the bins and descend. Figure 3 shows the FCN output: a mask over the original image, showing where one of the bins is.

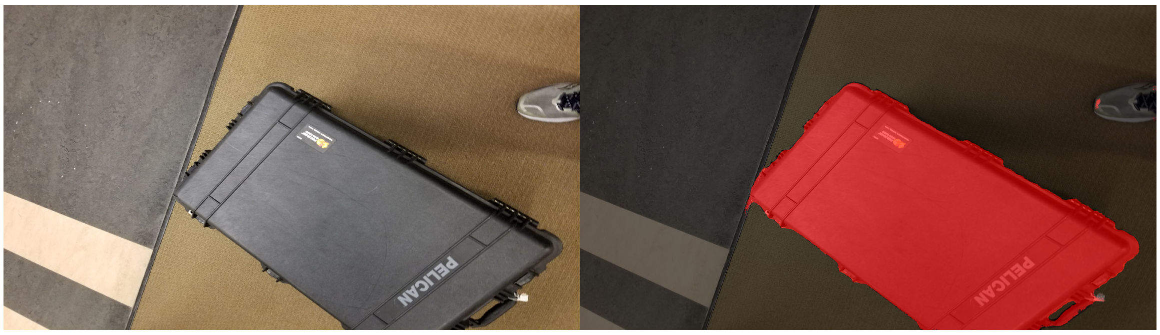

Figure 3: Example input to and corresponding output from the FCN. Red pixels denote show where the estimated location of the bin.

Command	**Description**
Heal	UASs locates the player and heals him/her
Scan	UASs patrol the arena to provide live video of the field
Search	UASs locate the bins and identify the information contained in the split QR code

Table 1: Available voice commands and corresponding actions

Obstacle Identification

The core of our obstacle identification system lies within the Intel RealSense Camera. RealSense allows for depth recognition, which allows for our drones to interpret their proximities to potential obstacles. The RealSense camera is forward facing. Therefore, the UASs' heading will always be in the direction of their travel in order to use the depth data for obstacle detection. The camera provides a depth map of objects in front of an UAS. If there is an obstacle in the path of the UAS, the bug algorithim is used to move around the obstacle perimeter.

Voice Identification

Due to the noise produced by the propulsion system, the voice recognition system needs to be very robust. To achieve this we develop a real-time keyword detection system using convolutional neural networks. The system is able to classify 5 second audio samples with pre-trained keywords such as "search", "heal" and "scan". In order to achieve real-time detection, the system buffers live data in a moving 5-second window and increments the window start time by one second every second. Available voice commands and their actions are listed in table 1.

Player Identification

We implement a naive system for player identification. Our contestant wears a brightly colored vest. A color mask is used to detect the position of the vest in camera video. Specifically, the hue saturation value (HSV) of each pixel in the image is compared against a predetermined ground-truth reference of the vest color. The vest's physical dimensions, combined with the knowledge of its pixel location, can be combined to crudely estimate the position of the player. The laser is aimed by yawing until the identified person is in the center of the video frame.

Communications System

Communication between the quadcopters and ground station is done through one main channel: the long-distance Ubiquiti airMAX WiFi; connecting the base station with the onboard computer (the Raspberry Pi has a built-in WiFi module). This way, we can treat the quadcopter as if it were on the local network. This allows us to connect via SSH. Applications requiring a GUI may use either the X11 forwarding feature provided by OpenSSH or a separate VNC connection. The mission can be aborted by using the appropriate command at the base station. Our system uses ROS [3] to mananage communications between different UASs and the ground station. ROS also facilitates inter-process communication onboard both types of systems.

Unlike in Mission 7, it is unrealistic that our safety pilot can take manual control of all four UASs. Therefore, the only other connection to the UASs is the individually keyed emergency kill switch given to the judges at the start of competition.

User Interface / Man-Machine Interface

The human operator is able to view the camera feeds from the friendly quadcopters on a website accessed on a handheld, Wifi-enabled tablet. The live video streams from each quadcopter are displayed in a grid on the tablet's screen. Also, overlaid on each video stream is data about the quadcopter's state, including its current task and position. This stream is view-only: the human operator can see data from the friendly drones on the site, but commands to the drones are issued via voice.

The video streaming is accomplished using a web server ROS node on the ground station, which receives image and state data from topics that each of the quadcopters publishes on the ROS network. The web server node then aggregates the images and superimposes the data on the images. Meanwhile, the server listens on a specified port for incoming requests, and when a client (i.e. the operator's tablet) tries to connect, a TCP connection is established and the data is transmitted. The frame rate at which the server reports is configurable at launch from 5-30 fps, limited by the maximum framerate of the drone cameras and the WiFi bandwidth.

RISK REDUCTION

Vehicle Design

EMI/RFI Solutions

To reduce EMI, an LC power filter is installed between the battery and components, which requires 5 to 12 volts. To reduce RFI, signal wires are twisted and isolated from each other—rather than bundling wires together into groups, the wires are instead run along different areas of the vehicle.

Shock/Vibration Solutions

The most critical and expensive sensors are located onboard the Raspberry Pi and the Navio, which is a hat bolted on top of the Raspberry Pi. The two are vibrationally isolated from the rest of the UAS using dampening balls. The balls are mounted using 3D printed parts, as recommended by the Navio manufacturer.

Safety

Risk reduction consists of both onboard and human-interactive measures to ensure safe operation of the vehicles in flight.

The guidance system attempts to take rather safe trajectories, opting largely to perform a series of straight-line, bang-bang trajectories. The guidance system also imposes a limit on velocity during the mission to 5 m/s. If a quadcopter attempts to fly outside of the arena, the guidance system indicates failure and returns to land. Alternatively, the base station operator can initiate a return to land or immediate landing at current position.

The quadcopters fly at a relatively high altitude to avoid collision with obstacles in the arena. Intel RealSense technology is utilized to detect nearby drones and adjust their path to avoid collisions. Additionally, the quadcopters attempt to fly at slightly different altitudes from each other to further reduce the likelihood of a collision.

Each packet of data from the IMU and cameras is carefully monitored to ensure every component is functioning properly. Data differing by excessive margins between packets or frames is considered indicative of failure of the equipment. In the event of a single component failure, less-reliable methods of position and velocity estimation are used. In the event that the vehicle can no longer determine its position and velocity, the vehicle will immediately attempt to make a vertical landing.

The PX4 flight stack also requires a 2Hz heartbeat through MAVROS. In the event that the high-level controller on the Raspberry Pi crashes and this is interrupted, the PX4 firmware enters a failsafe mode, and performs a safe landing using altitude control.

In the case of total system failure or in the event that a vehicle takes a dangerous path outside the arena, a remote kill switch is triggered to cut off power to the system. There is a unique kill switch for each of the vehicles.

Modeling and Simulation

The vehicle is simulated in Gazebo. A model of the quadcopter was constructed, and its flight physics were programmed. The simulation interacts with ROS just as the real quadcopter would (known as Software in the Loop SITL); commands can be issued to the PX4 firmware, and it provides 6-DOF IMU, and 640x480 camera data. The custom gazebo world, simulates a closed environment with simulated obstacles and bins. It is used to verify localization and obstacle detection algorithms.

Additionally, we implement a testing environment for QR code detection. It generates sample images of a QR code split into four overlapping sections, as described by the Mission 8 specification. It allows for testing of algorithms for QR code assembly and detection in 3D space.

Physical Testing

The vehicle was tested at each stage of development, from motor control to initial libraries issuing control commands to the PX4 stack. Once proven in simulation the software was tested for autonomous flight with GPS data. After integrating the SVO algorithm, indoor localization was tested by moving the system around without spinning motors. Finally, autonomous flight was tested inside a netted test area.

Functions such as QR code detection and box detection are independently tested. As development progresses iteratively, each change to the software is tested on previously collected test data. Periodically these systems are also tested on the target hardware in real-time.

Physical tests will continue until the competition date. They will focus on ensuring smooth integration between system components and multi-system flight.

Integration Testing Pipeline

1. Bench tests of motors and sensors.
2. Chassis takeoff/land and area circling test: use IMU and altitude data fed into ROS to control flight using basic libraries in simulation and physical netted area.
3. SVO velocity/position estimate and feature tracking testing.
4. SVO optical flow and feature tracking testing in netted area
5. Vehicle area circling, pattern flying, and waypoint testing using fused SVO, IMU and barometer data.
6. Mission execution

CONCLUSIONS

The design we will showcase features several technical innovations that will provide a competitive advantage. In particular, our perception system with a convolutional neural networks and localization system using semi-direct visual odometry, including our voice command recognition and efficient mechanical design will allow us to perform effectively in the arena. Likewise, our extensive use of computer modelling and simulation allows us to rapidly test many scenarios and tune our algorithms.

Nevertheless there are several improvements that could be made to the device. Currently we use the Intel Realsense Camera's stereo image for SVO. However, the Raspbery Pi does not have a USB3.0 port, which significantly limits the resolution. In previous years we used an Intel NUC, which provided a lot more computational power and a USB3.0 interface. Unfortunately, the cost of a NUC was too high for developing several UASs for Misison 8. In the future we want to look into a middle ground solution that keeps system costs down, while providing slightly more power and a modern I/O.

ACKNOWLEDGEMENTS

We'd like to thank University of Pennsylvania faculty for continued support of our club in preparation for the competition. A special thanks to our faculty advisor, Dean Vijay Kumar. We'd also like to thank Prof. Jnaneshwar Das who provided us with continued support since the conception of Penn Aerial Robotics.

References

[1] C. Forster et al. "SVO: Semidirect Visual Odometry for Monocular and Multi-camera Systems". In: *IEEE Transactions on Robotics* 33.2 (Apr. 2017), pp. 249–265. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2623335.

[2] L. Meier, D. Honegger, and M. Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

[3] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: vol. 3. Jan. 2009.

[4] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.1556.