St. Olaf/Carleton Engineering Team Technical Design Paper

Brian Bettes
St. Olaf College
Thabiso Mabote
St. Olaf College
Kota Shibui
Carleton College
David Stone
St. Olaf College
Matt Thill
Carleton College
David White
Carleton College

ABSTRACT

Fully autonomous flight of drones is an emerging field in the aerospace industry which is becoming relevant for companies such as Amazon and important for the military. The issues involved in this field no longer focus on the ability of a single drone to operate independently, but instead, how a group of drones can operate as one body to complete a task in the most efficient manner. This paper details our attempt at producing four fully autonomous drones capable of completing tasks given by the voice commands of a human. We include details involving the physical structure of the drones and how they process a command.

INTRODUCTION

Statement of Problem

The problem posed for this year's competition revolves around the use of several drones to assist a human operator in reaching a designated area while avoiding an onslaught of enemy drones. The main objective is to successfully retrieve the four key components from the designated areas while avoiding the laser beam projectiles emitted by enemy, sentry drones. Success relies heavily on the ability for the human operator to interact seamlessly with her auxiliary drones to emit healing beams and shield the operator from enemy drones.

Conceptual Solution to Solve the Problem

Our solution to the problem focuses on the five specified points of emphasis: man-machine interaction, the fused sensory enhancement of a human operator by a fleet of aerial robots, swarm interaction, aerial target designation, and head-to-head interaction with opposing aerial robots. We solved the problem by using a combination of autonomous flight features and user commands activated via voice control. In this sense, our drones are autonomous, but are still able to interact and assist the human operator while attempting to successfully complete the mission.

Yearly Milestones

There were a number of yearly milestones each of our subgroups made throughout the year including our helmet detection algorithm, the building and completion of our drone, and the design of our obstacle detection algorithm.

AIR VEHICLE

Description of Configuration/Type

We have chosen to build each of our four drones using the same design so that each one is capable of performing all tasks if need be. This design is a generic quadcopter using the "X" formation for flight. The frame is an S500 which has been modified using 3D printed components in order to attach the necessary equipment. This frame in combination with the Hobbypower 2212 920 KV Brushless motors and a 2250mAh Lumenier battery is estimated to produce around 7 pounds of thrust at full throttle. With this amount of thrust and a small frame we have ensured that our drones remain highly responsive in flight. All components such as onboard computers and cameras have been mounted using 3D printed components in order to not interfere with the stability of the drones.

Flight Control System

Our drone's flight control system consists of an open source flight stack combined with hardware, sensors, and custom code. We used the PX4 flight stack, an open source project apart of Dronecode. The PX4 flight stack has been developed by countless people and companies around the world and is well tailored for autonomous flight with aerial vehicles. Our drone has the Pixhawk 2 flight controller as the hardware that runs the PX4 software. The Pixhawk 2 is a flight controller module that controls our flight using internal and external sensors. These include an accelerometer, gyroscope, magnetometer, and external optical flow camera. We chose the Pixhawk 2 because of its ability to incorporate a vast amount of sensors and connection to other systems. The last part of our flight control system is a computer that communicates with the Pixhawk 2 and computes our navigation (2). We use an onboard raspberry pi 3 board, which communicates via the MavLink protocol to the Pixhawk 2 flight controller.

Navigation

Navigation is computed by the onboard raspberry pi. All of the sensor data is first read by the Pixhawk 2 and then translated into useful estimates. The new mixed data is then sent to the raspberry pi for use in navigation. The raspberry pi also receives commands through a microphone of voice commands. The commands will be translated into the final movement of the drone. Based on the end result of the command received, the raspberry pi mixes data from the pixhawk 2 to calculate target velocities for the drone. These calculated velocities are then sent back to the pixhawk 2 to be executed. The raspberry pi also calculates the local position of the drone and keeps track of the changing velocities. All of the computation done on the raspberry pi is done within the framework of a ROS program. ROS is a set of tools developed specifically to help simplify the development of robots. A ROS package called MavROS combines all of our calculations and packages it in a MavLink message to send to the Pixhawk 2. In this way, our program calculates the velocity and action the drone should take, and the MavROS package takes care of the communication between the drone and raspberry pi.

Attitude

The Pixhawk 2 flight controller uses its accelerometer, gyroscope, and magnetometers to calculate the attitude of the drone automatically. The Pixhawk 2 has dual sensors for everything to ensure a stable flight and protect against sudden sensor failures. The pixhawk 2 receives messages from the raspberry pi and adjusts the attitude of the drone to reflect the desired velocity and movement. An optical flow sensor on the bottom of the drone is also used to adjust the attitude of the drone. We use the PX4Flow optical flow sensor and a lidar range sensor. The Pixhawk 2 mixes the optical flow data into its adjustment of the attitude in order to minimize drift.

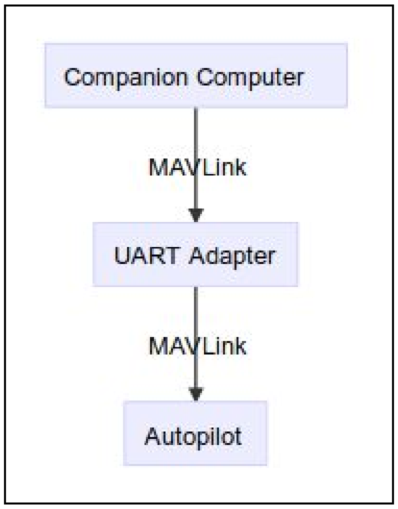

Figure 1. Control system architecture

Flight Termination System
We have equipped each of our drones with a communication line using a receiver and a transmitter. These transmitters have been programmed such that each drone can be shut down individually with a switch. The drones and transmitters have been color coordinated such that in the case of an emergency one drone can be shut down independent of the others. It should be noted however that this system does not remove power from the drone completely; instead, it halts the motors until given further commands. Leaving the drone in this state for more than 30 seconds will result in the disarming of the drone, however, if desired the switch can be reset and the drone can take flight again before the drone disarms. In addition to having this shutdown feature, we must also manually arm each drone individually for flight. This is a safety implementation that will help prevent errors within our code during the initial start-up phase.

MISSION PACKAGE

Perception System

Helmet Detection and Tracking
Our drones are capable of detecting the player's helmet and following it. A camera is mounted to the center of the bottom surface of the drone, facing down, which takes a video with the low fps of 5 - 10, as our drones will operate at a low speed. Our Helmet detection program determines if a helmet is present in an image by applying a procedure that uses the OpenCV library. The program filters the image for the color red, and then eliminates the noise consisting of small red pixels by erosion and dilation. Using the flood fill algorithm, the program calculates the area of each chunk of consecutive red pixels, and eliminates the chunks that are too oblong to be a helmet. Finally, the program applies a region of interest to the largest chunk, the one most likely to be the helmet. The program then returns the relative position of the helmet in the image to the helmet tracking program. The program tries to move the drone in order to center the helmet in the image, thus tracking the helmet. This program currently does not guarantee that the object of the interest is the helmet. However, voice commands allow a human to correct this situation, as well as reposition the drones to make them see the helmet.

QR code detection
We currently use a parallelogram based method for detecting QR codes. As the QR code can change, and at least one of the quadrants has none of the standard identifiers in such a code, we opted to detect the shape of the iPad itself using a detector based on the Hough transform. This detects lines in the image, which we then compare to find those that are likely parallelograms. From there, we verify a theoretical parallelogram against the actual pixels of the image, to check that it is not a false positive. Even with this, the algorithm is relatively unreliable in ideal circumstances and is the most significant source for improvement. Once the iPad has been identified, we use OpenCV's 3D reconstruction capabilities along with the camera constants to

unskew the screen of the iPad, and overlay all four fragments to read the full code. Although the overall algorithm is currently finicky, it only needs one clear, correct detection for each fragment, and it processes quickly enough that this can be achieved even with a low success rate.

Threat Assessment

The drones are equipped with relatively basic threat assessment in the form of 4 ultrasonic sensors. These allow it to react quickly to avoid a fast moving threat, either in the form of a drone, or a fast approaching wall. The values from the sensors are fed into a simple algorithm to remove noise and then turned into simplistic flight instructions to move to avoid crashing. The long term threat response is to be provided via voice instructions from a human, as long term threats are unlikely to cause a crash.

Communications System

The four drones use onboard raspberry pi computers to handle communication amongst themselves. Each drone is equipped with a raspberry pi that deals with the intense vision and navigation computations. In addition, the drones are able to send and receive messages from a ground computer. This ground computer will then communicate with each drone and coordinate their messaging. The ground computer will have a router that connects via wifi to each drone. In this way, each drone can communicate with any other drone through the ground computer. We also have a tablet with a microphone that sends our audio commands to the ground computer. The ground computer will then send these audio commands to each drone. So, we will be able to communicate with our drones through the tablet, ground computer, and onboard raspberry pi.

User Interface / Man-Machine Interface

The user interacts with our drone through a tablet and by being in the same physical space of the drone. We developed an Android application that receives a video feed from a raspberry pi camera. A raspberry pi camera is placed on the bottom of each drone to provide a video feed to the user. The application then displays this video feed to the user. The application also listens for audio input for our voice commands. The user speaks into the tablet, and the words are transmitted to the ground computer by the application. The tablet with the application allows the user to view what the drones are seeing and be able to transmit voice commands to them.

RISK REDUCTION

Vehicle Design

As required, we have encased the propellers of our quadcopters such that no human may be injured during operation. In addition, each drone requires individual activation to begin flight, thus there is little chance of startup error. The vehicles have been designed to be light in relation to their thrust output, therefore they are highly responsive in flight; this will help prevent accidental collisions.

Safety

Obstacle detection and avoidance

When our drones detect obstacles in a short range using the ultrasonic sensor and infrared sensor, they stop and wait for human commands. The drones constantly measure the distance to the nearest obstacle in four directions and slow down or stop depending on the distance to the obstacle. An ultrasonic sensor SRF05 is mounted to each side of the rectangular frame of the drone, thus four in total, in addition to the complimentary infrared sensor GP2Y0A710K0F (range 100-550cm).

Modeling and Simulation

For construction of the drones, we used Fusion360's 3D modeling, which allowed for accurate attachment of necessary components to our frame. In addition, Fusion360 provides the ability to do structural analysis of our drone to determine if there are any major issues when encountering outside stresses. These simulations showed that there are no major errors with our design.

We created an ubuntu desktop virtual machine using the Oracle VM VirtualBox Manager. For our robot software development, we installed the latest version of ROS called ROS Melodic. To communicate with the drone or the simulator in this case, we installed MAVROS. All the Python code we developed for our drone was run on MAVROS (1).

In order to see the drone hovering and to get a feel of what was going on without a distraction from the code, we decided to use PX4 and Gazebo. PX4 is an open source flight control software for drones and other unmanned vehicles. PX4 was chosen because it allows us to build and maintain the hardware and software in a scalable way. This excellent open source autopilot stack offers Software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulation. We can easily integrate with PX4's simulator using MAVROS.

Gazebo is a powerful 3D simulation environment for autonomous robots that is particularly suitable for testing object-avoidance and computer vision. The Iris quadcopter simulated in Gazebo/SITL permits us to control our simulation with MAVROS.

Physical Testing

There is only so much testing that the simulation can show, therefore we had to put our software on the drone itself. Due to the limited amount of space around St. Olaf College campus, we did a few tests in our lab space. This space proved to be somewhat adequate while using the software from the simulation. Later, when we had redefined our software and developed the specific settings for the flight testing, we did some tests outside in the open space.

When testing outside, we would first fly the drone in manual flight mode with RC control. We would then switch to whatever mode we were testing. To test the PX4Flow sensor for optical flow integration, the drone was changed into a position hold mode. We also tested offboard mode, where the drone would receive messages from an onboard computer to control its flight. The RC transmitter was present at all test flights and acted as a kill switch to immediately kill the drone to avoid any dangerous collisions.

CONCLUSION

In conclusion, we feel that we have put together drones capable of competing with the rest of the pool through our utilization of a reliable flight control system, an efficient communication method, and complete safety features. Our combination of autonomous flight and stabilization elements, including the necessary sensors and code associated with the design, and verbal communication system allow us to operate alongside our drone without having to take full control. Our intelligent helmet and QR code detection algorithms will allow our drones to fly alongside our operator while being able to decode the information necessary for the competition. With the combination of modeling and simulation, the physical testing performed, and obstacle detection and avoidance features of the drone, we feel confident that our drone is safe and ready for competition.

REFERENCES

1. Torres, Darien Martinez. PX4 SITL Example. *Darienmt.com.* 25 November 2018. https://darienmt.com/autonomous-flight/2018/11/25/px4-sitl-ros-example.html
2. Zihao. PX4 Research Log [12] – PX4 Off-board Control with MAVROS (1). *UAV-LAB @ Sydney.* 2019. http://uav-lab.org/2017/08/15/px4-research-log-12-mavros-off-board-control-1/