8-2017

Min Kao Drone Tour

Ethan Black
University of Tennessee, Knoxville, eblack4@vols.utk.edu

Holden Coppock
University of Tennessee, Knoxville, hcoppock@vols.utk.edu

Victoria Florence
University of Tennessee, Knoxville, vflorenc@vols.utk.edu

Elliot Greenlee
University of Tennessee, Knoxville, egreenle@vols.utk.edu

Caleb Mennen
University of Tennessee, Knoxville, cmennen@vols.utk.edu

See next page for additional authors

Author
Ethan Black, Holden Coppock, Victoria Florence, Elliot Greenlee, Caleb Mennen, and Jacob Pollack

Min Kao Drone Tour:

Final Project Report

Team 2:
Ethan Black, Holden Coppock, Victoria Florence,
Elliot Greenlee, Caleb Mennen, Jacob Pollack

Customer:
Dr. Hairong Qi

November 29, 2016

Table of Contents

Executive Summary

Need and Goals

Our project, Min Kao Drone tour, had the lofty goal of an automated drone capable of performing a tour of Min Kao. Since that goal was so complicated, we spoke with our customer and came to a clearer understanding that our goal would be to build the foundation for the drone tour project to be continued after this semester. Working with Dr. Qi, we confirmed our three main goals for the project. The first goal was to develop a drone capable of following a tour guide, based on either a tracker, identifying piece of apparel, or facial recognition. The second goal was to develop a way for the drone to navigate through the hallways of Min Kao based solely on image processing for navigation. The last goal was to develop a way to manually control drone without a computer to interface directly with. Our task was essentially to progress as far as possible towards the completion of these three modules and also to try and integrate them into a tour simulation. By comparing those goals and the accomplishments our group has made by the end of the semester, we will be able to get a more concrete idea of our own success.

Results

Our manual control module is perfectly finished. We can use the Wii remote through either a Raspberry Pi 3 or through our computers to completely remotely control the drone. While the controller for the object following module could certainly be improved, this mode also works, and can safely follow a person wearing a distinct, brightly colored vest of any color. Line following is still in the beginning stages; the drone can successfully recognize a line, a cross, or a corner and move along a straight line, but the controller for this movement is fairly unstable.

Challenges

One of our main challenges during this project was having enough time to reach our goals. A primary cause of this challenge was the DIY drone. From the outset of the project, we had the option to create a drone from scratch or use a prebuilt Parrot drone. We spent the first two months of the project developing code for the Parrot drone and developing the hardware of the DIY drone in hopes of transferring our software from the Parrot drone to the DIY drone. Eventually we decided that the benefits gained from building our own drone were not worth the cost of continuing work. Since ceasing work on the DIY drone, progress on the Parrot drone improved immensely, because we were not trying to split out development efforts, or find solutions that matched both drones. Additionally, our team was challenged to keep every team member with something to do. Over the course of the semester this project became more and more software oriented. Our electrical engineering team members kept busy by assisting with documentation, design, and peripherals as well as the automatic controller tuning for our project.

Requirements

1. Drone
    1.1. Chassis
        1.1.1. The drone will be a quadcopter.
        1.1.2. The drone will travel aerially.
        1.1.3. The drone will be capable of surviving the impact of normal landing.
            1.1.3.1. Normal landing will be on tile floors of Min Kao.
            1.1.3.2. Normal landing will occur from up to 7’ in the air.
            1.1.3.3. The chassis must protect all internal parts.
            1.1.3.4. The chassis itself must not be damaged during landing.
    1.2. Cameras
        1.2.1. The drone will have at least one camera
            1.2.1.1. The camera will provide at least 30 frames/second.
            1.2.1.2. The camera will support at least 720p video.
            1.2.1.3. The camera will be mounted to the front of the drone.
    1.3. Flight Control System
        1.3.1. The drone will have an on-board Raspberry Pi 3.
            1.3.1.1. The Raspberry Pi 3 will run Ubuntu 14.04 LTS.
            1.3.1.2. The Raspberry Pi 3 will be powered by its own Flight pack 2100 battery.
            1.3.1.3. The Raspberry Pi 3 will perform image processing.
                1.3.1.3.1. OpenCV will be used for this
            1.3.1.4. The Raspberry Pi 3 will calculate and perform the desired movements of the drone.
        1.3.2. The drone will have on-board autopilot hardware.
            1.3.2.1. The autopilot hardware will solely control the motors of all four propellers.
        1.3.3. The Raspberry Pi 3 and autopilot hardware will communicate to control the drone.
            1.3.3.1. The Raspberry Pi 3 will send movement commands to the autopilot hardware.
            1.3.3.2. The autopilot hardware will execute movement commands sent by the Raspberry Pi 3.
    1.4. Power System
        1.4.1. The drone will have two batteries
            1.4.1.1. The motors and autopilot hardware will be powered by one 11.1 V Flight Pack 2100 battery.
            1.4.1.2. The Raspberry Pi 3 will be powered by 5 V.

1.4.2. The power system must actively power the drone for at least 5 continuous minutes.

1.5. Safety

1.5.1. The drone can be landed using a wireless emergency stop button.

2. Movement Control Code

2.1. Stability

2.1.1. The drone should be able to hover in an 8 cubic foot area.

2.2. Teleoperation

2.2.1. The drone commands will be oriented relative to the drone nose, where the camera is mounted.

2.2.2. The drone should have a command for forward movement.

2.2.3. The drone should have a command for backwards movement.

2.2.4. The drone should have a command for leftwards movement.

2.2.5. The drone should have a command for rightwards movement.

2.2.6. The drone should have a command to increase elevation.

2.2.7. The drone should have a command to reduce elevation.

2.2.8. The drone should have a command to rotate yaw clockwise.

2.2.9. The drone should have a command to rotate yaw anti-clockwise.

2.3. Automation

2.3.1. Preprogrammed Commands

2.3.1.1. The drone will take off upwards without sideways motion of more than 1 foot.

2.3.1.2. The drone will hover within an 8 cubic foot area.

2.3.1.3. The drone will land without without sideways motion of more than 1 foot.

2.3.2. Following a line

2.3.2.1. The line

2.3.2.1.1. The line should be placed immovably along the ground.

2.3.2.1.2. The line should be colored green.

2.3.2.1.3. The line should be at least 0.75 inches wide.

2.3.2.2. The drone should be able to orient itself using a colored line of tape at least 3 feet long.

2.3.2.3. The drone should be able to move forwards and backwards relative to the camera along a colored straight line of tape at least 3 feet long.

2.3.2.4. The drone should be able to navigate turns in the tape.

2.3.3. Navigate a Floor Using No Landmarks

2.3.3.1. The Floor

2.3.3.1.1. The floor should be in Min Kao.
2.3.3.1.2. The floor will be either the 3rd or 5th floor.
2.3.3.2. The drone should be able to traverse a hallway without colliding with the walls.
2.3.3.3. The drone should be able to make stable 90 degree turns at hallway intersections.
2.3.3.3.1. Turns may be done with the combination of both translational and rotational movements.
2.3.4. Following a Person
2.3.4.1. The person
2.3.4.1.1. The person will be wearing a yellow hat.
2.3.4.1.2. The person will maintain an average walking speed of less than three miles per hour.
2.3.4.2. The drone should be able to recognize the hat while facing it from ten feet away.
2.3.4.3. The drone should stay at least two feet away from the hat.
2.3.4.4. The drone should stay at most five feet away from the hat.

3. Vision Processing Code
3.1. Line recognition
3.1.1. The vision processing should be able to recognize a green line as described above.
3.2. Object detection
3.2.1. The vision processing should be able to recognize an object of a given color.
3.2.2. The vision processing should be able to give a distance from a known object within 10 feet to 1 ft accuracy.
3.3. Communication
3.3.1. The code should communicate the object's position to the movement control code.
3.4. Collision avoidance
3.4.1. The vision processing should use the forward facing camera to detect objects in front of it that could potentially collide with the drone.
3.4.2. The drone should attempt to avoid collisions.

Change Log

The following changes were made to the Drone Tour of Min Kao Requirements and Specifications Document after discussion within the team as well as feedback from the graduate teaching assistants and professors on how to best move the project forward. The changes are as follows:

Added	Removed	Date	Author
2.3.4.1.1 The person will be wearing a colored shirt.	2.3.4.1.1 The person will be wearing a yellow hat.	09/21/16	Elliot Greenlee
1.2.1 The drone will have two cameras.	1.2.1 The drone will have at least one camera	09/28/16	Elliot Greenlee
1.2.1.3 One camera will be forward facing.	1.2.1.3 The camera will be mounted to the front of the drone.	09/28/16	Elliot Greenlee
1.2.1.4 One camera will be downward facing.	1.2.1.3 The camera will be mounted to the front of the drone.	09/28/16	Elliot Greenlee
1.3.1 The drone will have a separate Raspberry Pi 3.	1.3.1 The drone will have an on-board Raspberry Pi 3.	09/28/16	Elliot Greenlee
1.3.1.1 The Raspberry Pi 3 will run Ubuntu 16.04 LTS.	1.3.1.1 The Raspberry Pi 3 will run Ubuntu 14.04 LTS.	09/28/16	Elliot Greenlee
1.3.1.2 The Raspberry Pi 3 will be powered by an external rechargeable phone battery.	1.3.1.2 The Raspberry Pi 3 will be powered by its own Flight pack 2100 battery.	09/28/16	Elliot Greenlee
1.4.1 The drone will use one manufactured battery.	1.4.1 The drone will have two batteries.	09/28/16	Elliot Greenlee
1.4.2 The power system will actively power the drone for at least 10 continuous minutes.	1.4.2 The power system must actively power the drone for at least 5 continuous minutes.	10/25/16	Elliot Greenlee

2.3.4.2 The drone should be able to recognize the shirt while facing it from fifteen feet away.	2.3.4.2 The drone should be able to recognize the hat while facing it from ten feet away.	10/25/16	Elliot Greenlee
2.3.4.3 The drone should stay at least one foot away from the shirt.	2.3.4.3 The drone should stay at least two feet away from the hat.	10/25/16	Elliot Greenlee
2.3.4.4 The drone should hover around four feet from the shirt	2.3.4.4 The drone should stay at most five feet away from the hat.	10/25/16	Elliot Greenlee
	2.3.3 Navigate a Floor Using No Landmarks (and all sub requirements)	11/10/16	Victoria Florence
2.3.2.4 The drone should be able to orient itself using tape intersections.	2.3.2.4 The drone should be able to navigate turns in the tape.	11/15/16	Caleb Mennen
	3.4.2 The drone should attempt to avoid collisions.	11/15/16	Elliot Greenlee

Design Process

Decomposition

The team agreed to separate the project into three different modules that corresponded to the three main goals that Dr. Qi recommended for us: object following, line following, and manual control. In the object following mode, the drone can follow a person wearing an orange vest at a safe distance. In line following mode, the drone can fly along a route set by green tape on the ground. This goal was separated into three underlying goals: hovering in place, following a straight line, and completing a turn. In manual control mode, the user can manually control the drone using directional keys on a keyboard or the directional pad on the integrated Wii remote. By separating the project into these modules, our group was able to divide the work between group members and make it manageable to complete in the time frame that we were given.

Research

One of the largest questions that we had to explore with this project was what the best method of control would be for a best approximation of the drone giving a tour of Min Kao. While we had three possible approaches, manual control was quickly determined to be of least desirability, as it added little value to the drone usage. That left us to determine whether object following or line following would be best. In addition, our group had little to no experience with drone flight, computer vision, or automatic control. During our research, we needed to figure out how the drone could be sent commands and controlled using the developer sdk. We also spent a lot of time exploring computer vision and learning how we could perceive the environment through the drone's camera. Lastly, we pursued the topic of automatic control. Methods of implementing commonly used automatic control systems within our specific project and the creation of custom automatic control systems were two of our top concerns when performing the initial research for this project.

Alternatives

The major alternative we explored throughout the first half of the semester while working on the project was possibly using a Do It Yourself kit drone instead of the Parrot AR Drone. Initially we were planning on using the DIY drone, as it was lighter and could have a longer battery life, as well as the customizability allowing us to add more sensors that could in theory grant us many more sensor inputs and options to control the drone. However, since we essentially had to construct the DIY drone from scratch, and since we were supplied with a smaller sized chassis for the drone than the one intended initially for the motors and propellers, there were many issues with it's construction. Unfortunately, due to the timeframe of the project being so limited

and the experience of most of our group leaning away from hardware, it became readily apparent that our plan to develop code for the AR Drone and then adapt it to the DIY drone was becoming increasingly unrealistic. We then decided to simply focus on preparing the Parrot AR drone as well as we could in our remaining time. The parrot AR drone had sufficient battery life for an entire tour to be given, and it had forward and downward mounted cameras with sufficient resolution and refresh speed for our purposes. This did mean however, that we would be unable to mount any sort of mobile computer (such as the Pi we had been planning to implement to function as a mobile controller for the drone) on the drone, so we had to change our requirements to reflect that the tour guide would carry the Pi with them on a velcro strap.

On the software side of the project, Robot Operating System (ROS) was considered as a software framework for our project. We believed that ROS would help keep our project modular and transferable from the Parrot drone to the DIY drone once its construction was complete; however, when work on the DIY drone was halted, we pulled away from the use of ROS. Since the code from the Parrot drone no longer needed to be transferred, we created a library structure that interfaced directly with the ARdrone library as well as OpenCV.

Selected Solution

We chose to use the AR Parrot drone for our solution. With the drone's two cameras and limited sensor availability, we needed to navigate using only video processing. The use of the Parrot drone also required that we have a device nearby which controlled it through a Wi-Fi connection.

In order to meet those needs we are running our vision processing code on a battery-powered Raspberry Pi 3 and have the user carry it. This Raspberry Pi provides both the mobile computation and tour functionality for our solution. The user is able to wirelessly interface with the Raspberry Pi by using a Wii remote, and can control the Raspberry Pi in three modes as seen in Figure 1. Mode 1 is manual control, in which the drone remains stable while allowing the user to move in 3D space. Mode 2 is object following, in which the drone tracks and follows a colored object at a safe distance. Mode 3 is is line following, in which the drone traverses the length of a line. Each of these modes allows the drone to navigate indoors in a different way.

The manual mode for drone control is fully operational. It allows user control via a keyboard or Wii remote, using either a computer or a Raspberry Pi 3. The drone remains stable between user actions, and has a scaling speed to allow for different movement needs. Users can control velocities among any axis in three dimensions, as well as rotating along the vertical axis.

The object following portion of the code is based off of work done by user Puk0X. The code uses minimum and maximum HSV values and selects pixels within this range that should be

recognized as the target color. These HSV threshold values can be altered during program operation using a learning mode which learns based on the centermost pixel of the frame. Out of all of the pixels determined to be within the HSV threshold, the code selects the largest group of adjacent selected pixels and calls this group the target. It then creates a rectangle that encloses the entire group of pixels, and calculates the rectangle's area. This rectangle's area is compared to a goal area that is set as a constant in the code. Based on the desired change in area and the location of the target, our control code sets the x, y, z, and rotational velocities of the drone in order to maintain the drone's approximately constant bearing relative to the target. It uses a p controller to set the velocity towards and away from the target in order to prevent oscillation from overshoot when corrections are made at full velocity. The object following code can reliably follow a target down the hallway and around corners. Following distance and target color are adjustable to match any change in circumstances.

With line following, we decided that our best solution was to use the Hough transform to extract lines from the drone's bottom camera feed. In order to make this possible, several pre-processing steps needed to be completed. First we determined that to simplify and reduce the search space for line detection, we would convert the image feed to a HSV color space, and then within that space create a binary mask, where only the correctly colored area (where the tape is) would be white, and every other pixel is black. We found that this helped to isolate the tape from the image, and kept other lines in the image from being detected. We also found that we got more desirable results when passing the image through a Canny edge detector, as the outputted image with detected edges generated more consistent and stable lines after the Hough transformation was applied.

Through the course of detecting lines, we also found that multiple similar lines would be detected instead of one "super-line", and so we also came up with an algorithm to take the lines from the Hough transformation, and simply down the found lines into a consistent average of the similar lines. With the tape lines on the floor being properly detected, we were able to calculate what point the drone should try to center itself to, and how that compared to the point directly under the drone. For example, we might be able to determine that the drone is .35 meters above the line and .12 meters to the left, while also being at an angle of 15 degrees to the line. With this information, we were able to begin to direct the drone in order to decrease all values to zero (vertical offset, horizontal offset, rotational offset).

In order to achieve this, we investigated the use of a PID controller. The goal was to come up with a tuned system that would move the drone to the desired point, but not overshoot or oscillate above the tape line. While much time was spent investigating this, we have yet to adequately tune the drone to align itself to the line without great error.

Figure 1. Our main control loop operates in three modes, each of which flies the drone.

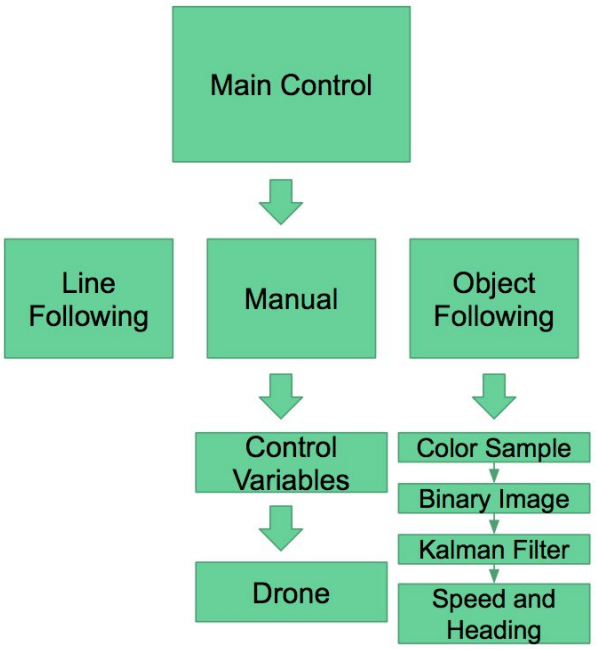
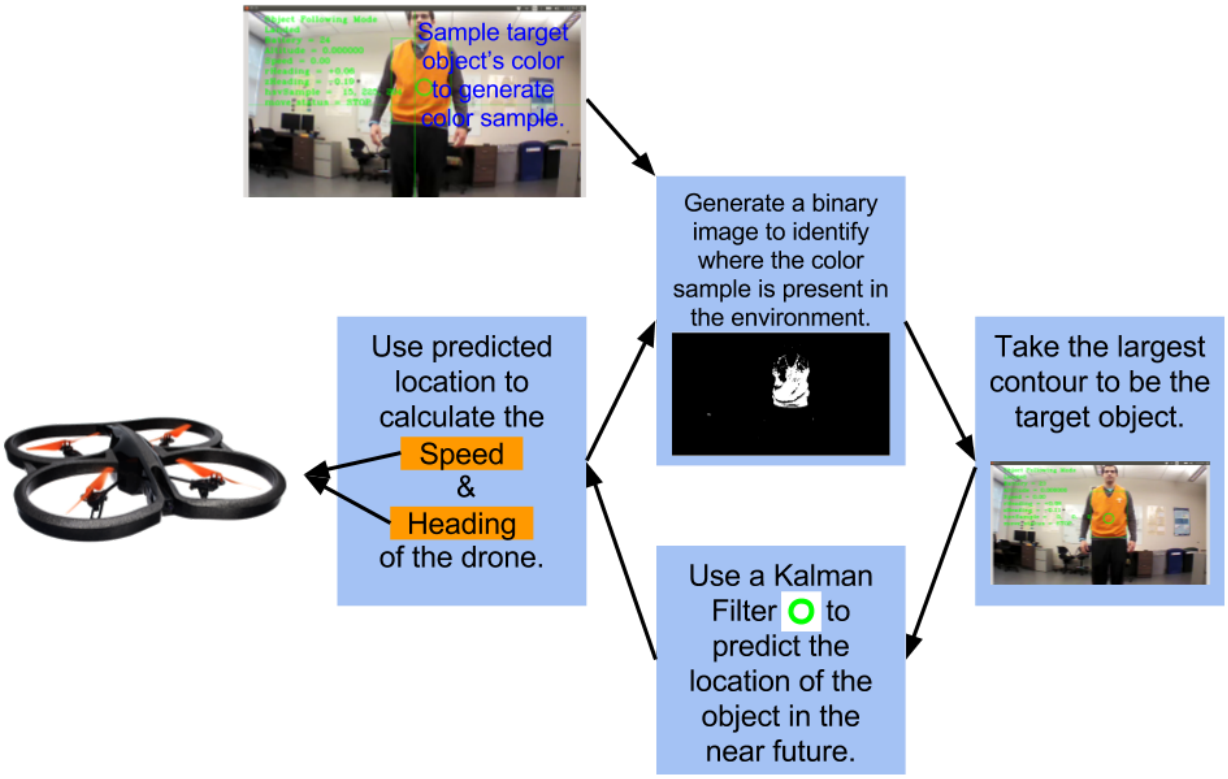


Figure 2. Vision processing explanation for our line following and object following modes.



Requirement Evaluation

The simplification of our project into three modes of operation and a couple of peripheral tasks such as the Wii Remote and tour audio creation allowed us to maintain individual responsibility for the requirements that applied to each member's tasks. Through reference to the requirement document and discussion among ourselves regarding progress and challenges at least twice per week, our group was able to keep track of the requirements being met. Even when we were unsatisfied with our progress, we were constantly aware of our next task thanks in part to Elliot and Ethan's management of the overall scope of the project as well as the assignments that were due.

Phase Evaluation

The requirements document was tested against the specification documents that we collected in our research of our hardware components during the research phase. It was also compared to the knowledge we had gained of computer vision and drone control.

During the design phase, we ensured that our design matched up with the goals we had set out in the requirements document; furthermore, in implementing our designs, we made sure to follow through on each element of our agreed upon requirements.

In the testing phase, it became more difficult to verify our work against the requirements as we fell into a rapid prototyping workflow; however, we kept track of the expected abilities of the drone and tried our best to achieve these abilities. When evaluating our results, it was necessary to change the requirements document in order to match what we had learned was reasonably possible throughout the project.

Outcomes

The Raspberry Pi currently works as a desktop computer interface to the drone. It performs the same functionalities as a laptop computer and can successfully launch the drone application on startup. One remaining issue is ensuring that the drone is connected via Wi-Fi to the Pi without a display monitor. A second issue is ensuring that the Wii remote is connected to the Pi via bluetooth without a display monitor.

The Wii remote control works extremely consistently. The remote connects to the computer on which the drone code is running via bluetooth, and it functions through a separate application that runs simultaneously. This program converts the Wii button presses into keyboard presses, so the Wii remote functions as a limited keyboard that can be used to control the drone just as a keyboard would be.

The manual control mode of drone operation works fairly reliably. All errors in key recognition have been eliminated; however, the drone sometimes experiences a delay in responding to key presses or the end of key presses.

The object following mode of drone operation is in general reliable, but is currently hardcoded for the vest size and other parameters we are using for demonstration. We did not have time to

add an element of object avoidance to the drone, but we believe the crash and emergency landing safety measures we have implemented make up for this.

With line following, we decided that our best computer vision solution was to use the Hough transform to extract lines from the drone's bottom camera feed. In order to make this possible, several pre-processing steps needed to be completed. First we determined that to simplify and reduce the search space for line detection, we would convert the image feed to a HSV color space, and then within that space create a binary mask, where only the correctly colored area (where the tape is) would be white, and every other pixel is black. We found that this helped to isolate the tape from the image, and kept other lines in the image from being detected. We also found that we got more desirable results when passing the image through a Canny edge detector, as the outputted image with detected edges generated more consistent and stable lines after the Hough transformation was applied.

Through the course of detecting lines, we also found that multiple similar lines would be detected instead of one "super-line", and so we also came up with an algorithm to take the lines from the Hough transformation, and simply down the found lines into a consistent average of the similar lines. With the tape lines on the floor being properly detected, we were able to calculate what point the drone should try to center itself on and how that compared to the point directly under the drone. For example, we might be able to determine that the drone is .35 meters above the line and .12 meters to the left, while also being at an angle of 15 degrees to the line. With this information, we were able to begin to direct the drone in order to decrease all values to zero (vertical offset, horizontal offset, rotational offset).

In order to achieve this, we investigated the use of a PID controller. The goal was to come up with a tuned system that would move the drone to the desired point, but not overshoot or oscillate above the tape line. While the electrical engineering members of the group had limited experience with PID controllers in electrical systems, the application of this controller to drone flight turned out to be much more difficult. One issue that we faced with the drone was having it drift away from its current position; this problem was amplified when the propellers became bent or damaged throughout the year. When the drone drifts more than approximately one foot, it is difficult to keep the tape line within sight of the downward facing camera. Although we tried various techniques for combatting the issue, it continues to hinder our progress in tuning the PID controller. While much time was spent investigating this controller, we have yet to adequately tune the drone to align itself to the line without great error.

In order to tackle the challenge of making sharp turns that are necessary in a hallway, we set out to recognize when a right-hand or left-hand turn needed to be made. Our solution to this problem was an additional line that was at a 45 degree angle extending from the inner corner of the turn.

By comparing the angles of the lines at a corner, we are able to categorize the lines as vertical, horizontal, and corner then determine the direction which the line is turning. Given the ability to maintain its position over the corner for a couple of seconds, this corner recognition will allow the drone to correctly pivot on a corner rather than needing to make a sweeping turn.

Lessons Learned

Unexpected Events

A large part of our team's time at the beginning of the semester was spent on building and developing the DIY drone, while evaluating the progress we were making relative to the cost. After over a month of work, we found this solution to be untenable, in that little progress had been made in exchange for both a high workload to build the drone, and a slowing down of software development. This slowdown was caused by the need to investigate and learn about two different flying systems at once, while also attempting to create a processing solution for both. After reviewing with the team and our customer, we made the decision to set the DIY drone aside, and focus on the AR Parrot drone. This resulted in immediate progress in that we could begin seeing the results of our code act on the drone. Additionally, some tools we had set aside because they would not work on the DIY drone were brought back to the forefront and became the base for our initial effort.

Although initially our team was hoping to perform hallway navigation in a clean hallway, a comparison of progress against our investigations of difficulty, especially with line following, led us to believe that this is much more of an area for future work. However, this challenge was one of the more fun aspects of our project, as we were able to grow our vision processing skills while adapting solutions to finish this goal. In the end, our team discussed the difficulties of hallway navigation with our customer, which she agreed were beyond the scope of undergraduate research. In the future, we hope to combine our work and experience with line following and object following into a solution that can navigate without hallway markings.

Our team fought against the safety challenge of operating the drone throughout the semester. Initially our team spent time trying to make the DIY drone safe to operate once it became flyable, looking into large PVC pipe and net enclosures and styrofoam bumpers that could be added to the drone. These hardware challenges were solved by switching to the AR Parrot drone. This drone already comes with a sturdy bumper, and has onboard software which will react properly to crashes, and will land in the event of a disconnect. However, our team did suffer a few nicks from the propeller blades, a problem we solved by using small square pieces of plastic to guard against the drone. We were very grateful to have our own lab to operate the drone in this semester.

All of our team is from the EECS college, which means we had little experience with mechanical or aerospace engineering projects or concepts. This lack of knowledge revealed itself in small ways throughout the project, such as when we discovered the incredible improvement to drone performance adding new quadcopter blades could bring. We had been attempting to correct

problems in code which were simply a result of hardware issues. This challenge was met mostly by small incremental experience shared amongst our group members.

Future Projects

It is highly likely that both Elliot Greenlee and Caleb Mennen will continue work on this project as an independent study and graduate work over the next semester and year. In light of this, there are multiple organizational changes we think could be made to improve this project.

Because much of this project involved research into libraries, setup on multiple operation systems, and co-opting of prior code bases, it became very difficult to keep up on the software development side. To this end, in the future we will spend time making decisions about and creating builds for the correct development operating system, IDE, and runtime environment. Additionally, we will make alterations to code base in order to better understand the underlying libraries, and to organize them with our own documentation in a way that can be carried forward. Finally, we will begin using GitFlow as our primary development paradigm in order to keep demos, features, tests, release, and our main development branch separate and organized. Although some team members have prior experience with this technology, we felt that it would only slow development and require a high learning curve for the team members who had not worked with it before.

On the actual development side, it would be helpful to spend time reducing the image processing functionality created into specific abstracted functions that can be used across modules. Much of our work was focused on rapid prototyping and drone progress, rather than sustainability across solutions. In light of this, there are many optimizations that can be made to improve single functions. For example, we found that both the PID controller code and HSV selection code were needed in multiple locations. Writing a set of functions that abstract and improve these solutions would give us and other developers the confidence needed to use these functions again and again without modification.

Although each of our group members performed testing on incremental code improvements they deployed, the testing side of this project should be improved in future projects. As individual functions for image processing and flying control are developed, it would be helpful to write unit tests for each one. Our team struggled because we only had one drone for six team members to test code on; creating a separate environment for testing of vision processing, as well as a drone simulator for control code, would be great ways to improve development times without purchasing a second drone.

Team Member Contributions

Elliot Greenlee - Computer Science

Elliot acted as team lead for the duration of the project. To fulfill his responsibilities, he organized team meetings, wrote the management by objective reports, kept team members up to speed on current progress, and acted as liaison for classwork. This included turning in assignments and meeting with teaching assistants about our project. He also acted as the primary customer contact, keeping Dr. Qi up to date on the project throughout the semester. Elliot created the outward facing documentation for the project, including a website, YouTube channel, and GitHub Readme. On the code side, Elliot implemented the main control loop and full manual control for the drone. He also managed overall software engineering decisions to maintain the three modes and their interface needs with the drone. For each of his responsibilities, he added documentation in the form of comments. Additionally, he contributed to object following improvements including the distance p controller, height tracking, color variance targeting, and emergency landing. Elliot tested his individual software components, and evaluated overall drone performance in order to match Dr. Qi's requirements.

Caleb Mennen - Computer Science

Caleb focused primarily on the line following section of the project after the move away from investigating ROS as a potential platform. Within this capacity, Caleb was responsible for all of the components of the line following code base, such as the line detection, and flight logic. He made the core code that drives the line simplification algorithm, although Victoria contributed to this later in the project's lifespan. Caleb was also responsible for the initial implementation of the PID controller for the flight logic in line following mode, and the current tuning of the PID controller. Caleb is also responsible for tuning the parameters for many of the OpenCV functions, as well as determining the color masking range.

Victoria Florence - Computer Engineering

Victoria contributed early on the development of the DIY drone kit. She worked on soldering connectors for the DIY drone power system and batteries and learned about the software packages available for interaction with the DIY drone. Victoria installed the Ubuntu Mate operating system, ROS, and ROS libraries on the Raspberry Pi 3. During this installation, she found it necessary to troubleshoot and alter the ROS libraries so that they would run on the new

ARM processor of Raspberry Pi 3. After halting work with ROS and the DIY drone, Victoria switched to the new code library that was created by Elliot and ensured that it worked correctly on the Raspberry Pi 3. She also altered the library's class structure in order to make drone control code more efficient and readable. At the end of this project, Victoria worked with Caleb to improve the performance of the line following code. Particularly, Victoria helped Caleb correct by correcting the normalization of the lines returned by HoughLines function. She implemented the line categorization code for corner recognition and tested a method of corner recognition used symbol matching that ended up being less reliable. She also took part it the hours of learning and testing that went into the disappointingly unsuccessful, but improved PI controller for line following. Outside of her direct contributions to the project, Victoria helped brainstorm ideas for the object following code and researched methods of clean hallway navigation using OpenCV.

Jacob Pollack - Computer Science

Jacob immediately was focusing on implementing and creating the object following module. He implemented and modified a project originally developed by Github user Puk0X. The project uses OpenCV libraries and a drone communication interface to control the drone's movement. Modifications to the original project were substantial and included changes to allow the drone to follow a tracked object at a safe distance and provided a much more stable takeoff procedure. Jacob also provided some help in testing the capabilities of the Raspberry Pi and the battery pack used to power the Raspberry Pi computer. During the last weeks he helped implementing audio integration to let the environment cue the drone to play audio files for the tour infrastructure. Additionally, Jacob worked with the team to brainstorm solutions and debug to some of the various problems encountered throughout the project.

Ethan Black - Electrical Engineering

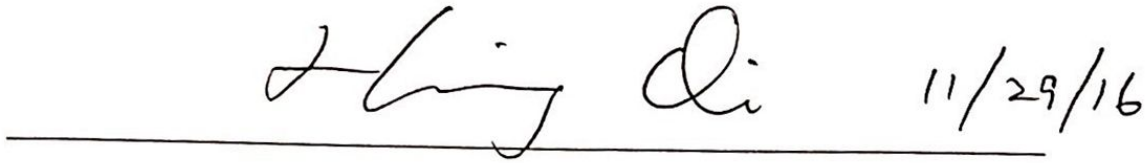
Early on, Ethan was assigned to work on the development and construction of the DIY drone kit. Due to design constraints and shipping times, we eventually decided to not pursue the DIY drone solution. After the shift to the Parrot AR drone, he focused on working on the P and PI controllers setup for the line following and object following code. He also assisted on finding a wireless controller and functioned as a document lead for the team, as well as setting up possible infrastructure for the tour.

Holden Coppock - Electrical Engineering

Holden was assigned to the DIY drone as well, and did work on that by selecting and soldering connectors to supply it power. He also was assigned to identify and order all of the necessary parts for this solution. After the pivot, he was assigned to work with Victoria on implementing the Raspberry Pi and interfacing it to the drone. He also worked on the P and PI controllers for the object following and line following modules. In addition, Holden was assigned to make the Wii remote to work as a wireless controller, including programming and implementing a control scheme on it for the different control modes of the drone.

Customer & Team Member Agreement

I, Dr. Hairong Qi, have read the above Final Project Report today, November 29, 2016, and I have approved that plan.



We, the students in group 2, approve this final project report