Flying multiple drones from 1 remote controller

Stephen Lazzaro
University of Wisconsin-Madison
Madison, WI
slazzaro@cs.wisc.edu
Supervised by: Michael Coen

Abstract

In this paper, we present our methodology of modifying the firmware for the Pixhawk flight controller in order to enable the safe flight of multiple UAVs with 1 remote controller. In order to accomplish this, we focus on implementing a low level behavior of distance maintenance, because as similarly described in Braitenburg's novel Vehicles, this would set the stage for higher level and more elegant multi-copter behaviors to be added in the future. The implementation of this distance maintenance behavior involves two simple principles listed below.

1. *If the copters are too far from each other, they must autonomously move closer.*
2. *If the copters are too close to each other, they must autonomously move farther from one another.*

Copters send each other GPS coordinates in order to be aware of the positioning of surrounding copters and maintain a particular distance. Having this behavior allows one pilot to control multiple copters in real time while using just 1 remote controller. This research has many applications, some of which include agriculture as well as search and rescue missions. We take an important first step towards the useful control of multiple drones from 1 remote, but there is still more work to be done as future work should focus on adding in more complex behaviors on top of the low level distance maintenance behavior.

1. Introduction

In the start of the 21st century, the development of drones has become increasingly popular [5]. [1] Drones can now be bought commercially at various sizes, types, and prices. Pilots of drones will typically fly one drone at a time where they can control the actions of the drone with a remote controller (Figure 1), and monitor the state of the drone through a ground station (Figure 2). However, what if a pilot wanted the capability of flying multiple drones at the same time? This could have a variety of benefits, with many of those centered around the increase of efficiency for tasks that are currently being tackled by one drone. In order to enable a pilot to control multiple drones at once, there must be some modifications to the firmware of the copters' autopilots, where an autopilot can be simply described as the brain of a copter.

Along with an increase in the popularity of drones, there has also been vast improvements in the autopilots for drones. Autopilots for hobbyist UAVs come with a variety of sensors and features, many of which are described by Chao et al [8]. All of these built in sensors already provide many uses to autopilots, and with the growth of open source firmware, it has become much simpler to leverage these sensors in order to add new capabilities to autopilots. Therefore, we did just that as we used the Pixhawk autopilot (Figure 3) along with the open source ArduCopter firmware in order to enable the control of multiple copters from 1 remote controller [9, 12]. We modified the ArduCopter firmware to add a key behavior of distance maintenance between airborne copters. Building this low level behavior of distance maintenance set the stage for higher level and more elegant multi-copter behaviors to be added in the future.

[1] Throughout the paper, we will use drone, copter, and UAV interchangeably.

Figure 1. An example of a remote control for a drone.

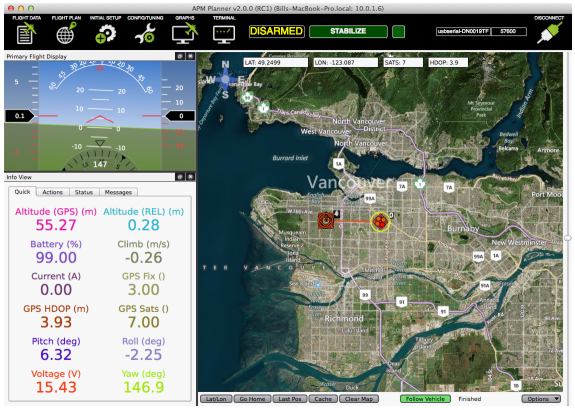

Figure 2. An image of the APM Planner ground station interface.

Figure 3. The Pixhawk autopilot that we use in this research.

In order to implement this behavior, a few pieces of hardware needed to be added to each drone. First, an Arduino Uno was connected to the Pixhawk, which communicated via UART. The ArduCopter firmware was modified so that the Pixhawk would periodically (4 Hz) send data including GPS coordinates to the Arduino. Then, a 2.4 GHz transceiver was connected to the Arduino so that the data could be wirelessly transmitted to neighboring copters. Each copter would use other airborne copters' GPS coordinates to assure it was maintaining a specified distance from neighboring drones. If the calculated distance to a neighboring copter fell outside the bounds of the desired distance, copters would autonomously alter their pitch and roll speed to correct the difference.

A large benefit of this research is that it can be replicated and utilized with much ease as well as minimal cost since the only hardware added was an Arduino Uno and the 2.4 GHz transceivers. Additionally, this research has many applications as it can help increase the efficiency of tasks normally completed with one pilot. Enabling one pilot the control over multiple copters allows a faster search over areas of interest since more copters can canvass an area quicker than one copter. The research presented here provides an important building block for the control of multiple copters from 1 remote, but there is still much more to be done. Many higher level behaviors have yet to be added on top of the distance maintenance behavior we've built, so this can certainly be an interesting focus of future research.

The paper will proceed as follows. Section 2 will describe previous research in the area, Section 3 will present the methods used, Section 4 will discuss the results, Section 5 will discuss implications as well as potential future work, and Section 6 will provide references.

2. Previous Research

There has been a considerable amount of research that has focused on flying swarms of drones. Burke et al. presented a platform for the flight of a swarm of drones [7]. At Georgia Tech, great results have been achieved in having multiple copters autonomously flying in formation [10]. However, this research, along with most other research on swarms of drones, has achieved its goals in a different format. The drones in these cases seemed to have flown in pre-programmed flight paths. Previous research on swarm drones has discovered extremely useful techniques for autonomous applications, but what if a pilot wanted maximal control of multiple drones at once? There are many situations where real time control over many drones by one pilot can be of great use.

Therefore, we explored this possibility of giving an individual pilot the control over multiple drones. Although it may seem that this would require an extremely skilled pilot and very complex controls, this is not necessarily the

case. Based on Braitenburg's novel Vehicles, we followed the main principle of enabling complex behaviors through the creation of simple building blocks [6]. In our case, the simple building block behavior is distance maintenance between copters. By building this functionality into each copter, it allowed us to fly multiple copters from one remote just as before, with the exception that two copters were now in the air instead of one.

3. Methods

The methodology for flying multiple copters from 1 remote revolves around the implementation of one key low level behavior: distance maintenance. Each UAV contains a user modifiable parameter, which determines the distance each copter must maintain from other copters. The implementation of this distance maintenance behavior involves two simple principles listed below.

1. *If the copters are too far from each other, they must autonomously move closer.*

2. *If the copters are too close to each other, they must autonomously move farther from one another.*

With this distance maintenance behavior built into the copters, more complex and elegant behaviors will easily follow. Some examples of more complex behaviors will be discussed in more detail later on.

Multiple options were considered for implementing this behavior. RSSI was thought to be a possibility for estimating distance between copters, but we decided against this as RSSI is quite noisy. Sonar is another potential option where each copter could produce a map of nearby objects and their distances. However, the maximum ranges of affordable sonar devices are not sufficient for this application with them commonly approximating between 5 and 11 meters [2]. Computer vision techniques can be utilized, in which each copter can survey for surrounding copters with an on-board camera(s). However, this may become a bit complex and pricey because we would need cameras to capture all directions from each copter.

Instead, the technique utilized here centered around GPS coordinates. In particular, each copter is aware of neighboring copters' positioning by having knowledge of their GPS coordinates. This does not come without its own issues, with one of them being that GPS is not always the most reliable piece of technology. For this reason, before each flight, it's crucial to verify that the calculations of distance between copters is accurate. [2] As long as this verification is performed successfully, the methods discussed below have proven to be quite effective.

The modifications/additions made to the firmware and hardware can be separated into two sections: adding a way for multiple copters to communicate, and designing how the copters should utilize the data they obtain to maintain a consistent distance from one another.

3.1. Copter Communication

The copters communicated with one another via 2.4 GHz wireless transceivers [1].[3] One transceiver along with an Arduino was added to each copter, with the Arduino acting as the communication bridge between the Pixhawk autopilot and the transceiver [3].

At 4 Hz, the Pixhawk sends data to an Arduino via UART.[4] The Arduino takes the information it receives, and sends it to the other copter via the 2.4 GHz transceiver.[5] On the other end, once the other copter's Arduino receives this information, the Arduino stores the data and waits until its Pixhawk is ready to read it. A diagram of this process is provided in Figure 4.

3.2. Distance Maintenance

The data that is transmitted to and from neighboring copters typically includes the following information:

1. *GPS Latitude*

2. *GPS Longitude*

3. *GPS Altitude*

4. *GPS HDOP*[6]

5. *Compass Orientation*[7]

[2] In this case, a small LCD screen was added to one of the copters, which allowed us to quickly verify accurate distance calculation before flight. This LCD screen was connected and fed information from an Arduino Uno that we will discuss further later on.

[3] These transceivers have a maximum range of 80 meters in ideal conditions, so depending on the range required between copters, better quality transceivers may be necessary.

[4] The Serial 4/5 port is used on the Pixhawk.

[5] The RF24 library is used for communication between the Arduino and the 2.4 GHz transceivers [4].

[6] HDOP is the approximate accuracy for the GPS coordinates of the copter. If an HDOP of 1 meter is received, it means there is a *high likelihood* that the copter is within a 1 meter radius circle of the GPS coordinates given. It's important to note that HDOP does not give 100% likelihood which is why it's extremely important to test the accuracy of the calculated distance in the field before flying multiple copters using the methods discussed in this paper.

[7] This falls in a range of 0-360 degrees where 0 degrees indicates the copter is pointing north.

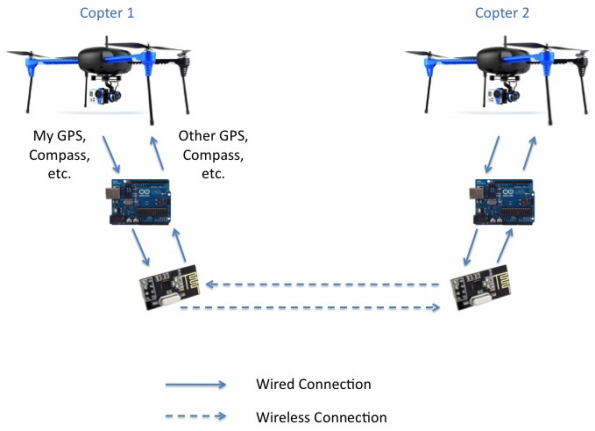

Figure 4. Depiction of how the copters communicate with one another. Data is first transferred from a copter to its Arduino via UART. Then, the data is transmitted via a 2.4 GHz transceiver to the other copter's transceiver.

Although compass orientation hasn't been discussed much, it is a crucial part of enabling copters to be flown from one remote controller. Airborne copters must remain aligned with one another, because for instance when the pilot increases the copters' pitch, the copters should move in the same general direction. Therefore, compass orientation is always sent along with location data so that the copters will automatically align their compasses if necessary. In this auto-alignment procedure, one copter acts as the leader so any copter that is a non-leader will auto-fix its alignment to match that of the leader.[8]

Apart from compass information, the other crucial part of the distance maintenance behavior is GPS.[9] It's important to note that retrieved GPS data is only considered valid if the HDOP is below 1.5 meters.[10] If valid GPS data is retrieved from another copter, then that GPS and HDOP information are stored along with their five most recent respective values. These values are repeatedly used along with the current copter's GPS coordinates to assure that the airborne copters remain within a specified distance. In calculating the distance between copters, an average of the most recent GPS coordinates is used.

If the copters are too far from one another, their roll and pitch speed are altered so that they move closer together. If the copters are too close to each other, the copters' roll and pitch speed are altered so they will move farther apart. It's important to note that as HDOP indicates an error bound around a copter's GPS coordinates, no corrections to the copters' positions will be made unless the difference between the calculated distance and desired distance is greater than the sum of the copters' HDOPs. For instance, suppose the desired distance between copters is 10 meters and the copters each have an HDOP of 1 meter. In this case, no corrections to either copter's position should be made unless the calculated distance between them is less than 8 meters or greater than 12 meters (10 +/- 2 * HDOP). Additionally, the amount of correction added to the copters' position is not linear with respect to their offset from the desired distance. Instead, a square function is used so that the copters are penalized more for greater offsets from the desired distance.

4. Results

The copters used were two 3D Robotics Iris + copters as shown in Figure 5 with modified firmware as discussed in the methods section. In order to incrementally test the modified firmware, we performed a series of experiments with successful outcomes that are described below.[11] Throughout the description of the experiments, the two copters will be referred to as Copter 1 and Copter 2. Additionally, the copters were configured to maintain a distance of 10 meters, which was also their starting distance for each experiment.

Figure 5. The 3D Robotics Iris + copter used for testing.

4.1. Experiment 1: Compass alignment with 1 copter stationary on the ground

Copter 1 was stationary on the ground while a pilot controlled Copter 2. The compasses of the two copters were pointed in different directions. The pilot elevated Copter 2

[8]The copter which is chosen as leader is determined at random in order to make the modified firmware more user friendly, because users do not have to worry about correctly setting some sort of leader parameter.

[9]At times, the GPS coordinates sent will not contain the copter's current location, but instead its home location. The home GPS coordinates is the location where the copter took off.

[10]The ArduCopter firmware documentation mentions that an HDOP below 1.5 meters is the best range for HDOP [9].

[11]As previously mentioned, before each flight, it's crucial to verify that the calculations of distance between copters is accurate i.e. GPS is reliable at your location.

off the ground. Once Copter 2 was at least 4 meters above the ground, it automatically yawed to align its compass to Copter 1.

4.2. Experiment 2: Distance maintenance with 1 copter stationary on the ground

Copter 1 was stationary on the ground while a pilot controlled Copter 2. The pilot elevated Copter 2 off the ground. The pilot first moved Copter 2 farther from the other copter so that it would auto correct itself and fly back within the configured distance range. The pilot then moved Copter 2 closer to the other copter so that it would auto correct and fly back within the configured distance range.

4.3. Experiment 3: Distance maintenance and compass alignment with 2 pilots

Before attempting to control both copters from 1 remote, the next experiment involved testing the modified firmware with 2 pilots, one controlling each copter. The compasses of the copters were not aligned as they started out on the ground. Then, the pilots elevated both copters off the ground. Once the copters rose to at least 4 meters above the ground, Copter 1 automatically yawed to the same orientation as Copter 2. When the second pilot would change the yaw of Copter 2 (i.e. change the compass orientation), Copter 1 would automatically follow to match the updated compass orientation. After this, the first pilot moved Copter 1 farther from Copter 2 to confirm that the two copters would autocorrect and move towards one another. Then, the copters were moved closer together to confirm that the copters would autocorrect by appearing to repel each other.

4.4. Experiment 4: Distance maintenance and compass alignment with 1 pilot

This experiment replicates the desired goal, where one pilot controls multiple copters from 1 remote control. Both Copter 1 and Copter 2 were on the ground and the remote controller of the 1 pilot was bound to both copters so they would both respond to commands from the 1 pilot. The compasses of the copters were not aligned as they started out on the ground. The pilot elevated both copters off the ground. Once the copters rose to at least 4 meters above the ground, Copter 1 automatically yawed to the same orientation as Copter 2. As the pilot updated the yaw for the copters, they would rotate their orientations together as they were receiving the same command. If there was some error to cause one copter's yaw to shift more than the other, the copters would auto-align as described in the previous experiments. The copters were then flown around to confirm that they would remain within the specified distance as they moved.

[12] If the answers don't match, the copters start over and recalculate.

4.5. GroupRTL

In addition to the adding the low level distance maintenance behavior, we also added 1 higher level more elegant behavior/flight mode on top of it called Group Return to Land (GroupRTL). In order to understand the GroupRTL functionality, it's necessary to understand normal Return to Land (RTL) when flying one copter. When the flight mode of the copter is changed to normal RTL, the copter flies back and lands at its home position. In GroupRTL mode, this functionality has been modified a bit. Rather than each copter flying back to its home position, the copters first calculate which airborne copter is closest to any of the home positions (while hovering). After this, the copters confirm (via the 2.4 GHz transceivers) that they came up with the same answer. If their answers match, the copter closest to any home position lands at that point, while the other copter waits by hovering.[12] Once the first copter has landed, it sends confirmation to the other copter, which will land at the remaining unoccupied home position.

One potential issue came up with this strategy: What if the copters were so far from their home positions that once the first copter landed, it was too far to communicate with the other copters (as the 2.4 GHz transceivers have limited range)? Well, this turned out to be a non-issue due to the low level behavior of distance maintenance being built in. As the first copter would travel towards its landing point, the other copter would follow in order to maintain the desired distance. This illustrates the power of building in this low level behavior, as it simplifies problems that may arise with more complex behaviors.

5. Discussion

This work makes important contributions to the research on swarm drones. Although there has been previous research that involved multiple copters flying together, these copters seemed to be flying along pre-programmed paths. The work done here enables the flying of multiple copters from 1 remote, giving a pilot much more control over multiple copters in real time. An additional benefit of this research is that it can be replicated and utilized with much ease as well as minimal cost since the only hardware added was an Arduino Uno and the 2.4 GHz transceivers.

This research has many applications with one being agriculture, where large fields can be canvassed without the need to replace batteries and perform multiple flights, as multiple drones can be flown simultaneously. Another application is search and rescue, where particular areas can be searched orders of magnitude faster with multiple drones flying together. Being that the Pixhawk flight controller is also being used by some military forces [11], this also could have potential impacts in that area.

An important implication of this research falls straight from Braitenburg's novel Vehicles [6]. With the low level behavior of distance maintenance built into the copters, many more complex and useful behaviors easily follow. Future work should explore these behaviors and build them on top of the ideas presented here.

One important feature that would seem to be useful for pilots of multiple copters is the capability for copters to fly in different formations. For instance, the pilot should have the ability to tell the copters to fly in a line, or a triangle, etc. Once copters can be commanded to fly in different formations, a new necessary feature arises, which we will describe as formation yaw. Normal yaw involves the rotation of copters about their own z axis as displayed in Figure 6. However, for proper control of copters in a formation, there must be some way to rotate the formation as displayed in Figure 7. Rather than rotation about each copter's own z axis like normal yaw, formation yaw would involve rotating about 1 of the airborne copters.

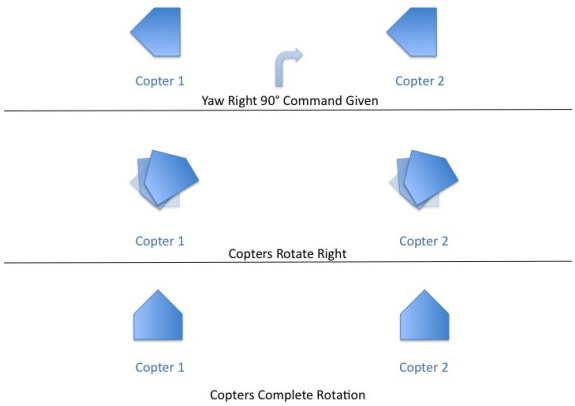

Figure 6. Depiction of normal yaw to the right by 90 degrees.

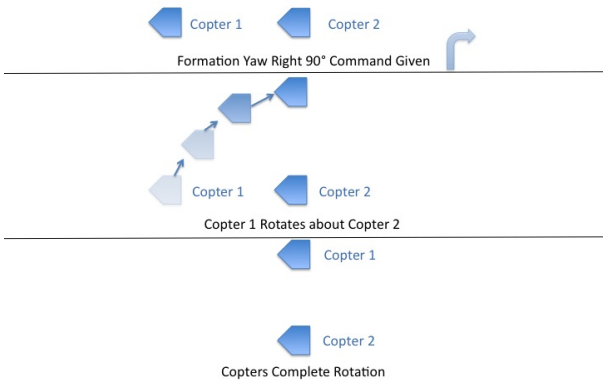

Figure 7. Depiction of **formation yaw** to the right by 90 degrees.

Another potential area of future research could focus on adding alternative methods that can be used beside GPS for distance maintenance. As mentioned in the methods section, we utilized GPS so that each copter could be aware of surrounding copters. However, as GPS is not always reliable, it would be quite useful to add in alternative techniques such as computer vision and sonar, just in case GPS is not working well at the location where the drones are in use.

The research presented here provides an important building block for the control of multiple copters from 1 remote, but there is still much more to be done. The most important idea to take away from this research is the importance of the low level distance maintenance behavior. All other interesting features of multiple copters in flight depend on an effective distance maintenance behavior. Therefore, the initial focus of future research should be ensuring that this distance maintenance behavior is sufficient for all conditions, specifically when GPS becomes unreliable.

6. References

1. 2pcs Addicore nRF24L01+ Wireless Transceiver Modules, http://www.addicore.com/2pcs-Addicore-nRF24L01-Wireless-Transceiver-p/112.htm

2. Analog Sonar (AC3.1) — Copter, http://copter.ardupilot.com/wiki/common-optional-hardware/common-rangefinder-landingpage/sonar/

3. Arduino - Home, http://www.arduino.cc/

4. Arduino Playground - Nrf24L01, http://playground.arduino.cc/Interfacing WithHardware/Nrf24L01

5. Barron, James. Drones Outpacing Rules as Popularity Soars in New York. The New York Times, August 3, 2014. http://www.nytimes.com/2014/08/04/nyregion/drones-outpacing-rules-as-popularity-soars-in-new-york.html

6. Braitenberg, Valentino. Vehicles: Experiments in Synthetic Psychology. MIT Press, 1986.

7. Burkle, Axel, Florian Segor, and Matthias Kollmann. Towards Autonomous Micro UAV Swarms. Journal of Intelligent & Robotic Systems 61, no. 14 (October 27, 2010): 33953.

8. Chao, HaiYang, YongCan Cao, and YangQuan Chen. Autopilots for Small Unmanned Aerial Vehicles: A Survey. International Journal of Control, Automation and Systems 8, no. 1 (February 17, 2010): 3644.

9. Copter — Multirotor UAV, http://copter.ardupilot.com/

10. GT — News Center?:: Multiple UAVs Perform Autonomous Formation Flight, http://www.news.gatech.edu/2014/08/07/multiple-uavs-perform-autonomous-formation-flight

11. In Ukraine, Tomorrows Drone War Is Alive Today. Defense One, http://www.defenseone.com/technology/2015/03/ukraine-tomorrows-drone-war-alive-today/107085/

12. Pixhawk Autopilot - PX4 Autopilot Project, https://pixhawk.org/modules/pixhawk

7. Resources

The code can be retrieved on request by emailing *slazzaro@cs.wisc.edu*