Jeremy Hunt and Christopher Buck
ELEC 424 Final Design Challenge
12-13-14

In our final lab assignment, we were to develop our quadcopter to hover within a 5 ft radius of the corner of two perpendicular walls between the heights of 4 and 6 ft. To achieve this, we could use the accelerometer, barometer, gyroscope, and an added on-board ultrasonic range finding sensors.

Changes for Our Solution

In our solution, we modified the autopilot routine in aicontroller.py to take off and hover for 45 seconds, which we determined to be sufficiently long for the competition. At 45 seconds, our limiting factor will be leaving the zone. Our autopilot primarily consisted of 3 major phases. First is the pre-takeoff calibration phase. In this phase, the motors are spun up to a speed that is insufficient to lift the quadcopter off the ground. This begins to acclimate the barometer to the moving air that causes error while flying. Second, during the takeoff phase, the thrust is increased to quickly bring the crazyflie to the hover height. AltHold mode is then enabled in hover mode, causing the AltHold functionality to sample the barometer and use the current altitude as a reference. The pre-takeoff thrust affects the barometer to make the readings more accurate relative to actual flying conditions. Additionally, we set a parameter “AltHoldTargetOffset” which specifies the height above the sampled reference the quadcopter should target. The yaw is set to a positive value to yaw the quadcopter during the takeoff and hover phases to promote better stability within the zone. In the third, hover phase, the positive yaw is maintained along with the AltHold settings. During this phase the quadcopter is intended to remain hovering, in a stable manner. Additional phases included landing and repeating the autopilot function, yet in practice they rarely ran.

The next bit change we made to aicontroller.py were the PID values. On connection, a callback was added to update a number of parameters from aicontroller.py in order to speed tuning of the system. This update system was compatible with the new parameters for the AltHold controller, as well as all of the existing parameters.

In the crazyflie’s firmware, the AltHold control code in stabilizer.c was re-written to include complementary filter based altitude estimation and better altitude controllers. Parameters were also added to this code to support these algorithms. For more information about them, see the sensor section of this report.

Lastly several bugs were fixed in the main crazyflie clients code, including a unprotected concurrent data structure access, causing radio failures in the cflib python library, and the lack of dead-zones on the pitch and roll control causing consistent slight drift in the crazyflie’s movements.

Jeremy Hunt and Christopher Buck
ELEC 424 Final Design Challenge
12-13-14

How To Obtain and Build the Code

While we have included source code, we recommend using our git repositories, which are in a known good state.

Modified Crazyflie Firmware

Clone the git repository with the althold2 branch with this command:

```
git clone -b althold2 git@github.com:greenlambda/crazyflie-firmware.git
```

Enter the directory created and run

```
make CLOAD=0
make CLOAD=0 flash
```

in order to compile, flash, and run the modified firmware. This firmware must be used with the modified Crazyflie client below.

Modified Crazyflie Client with AI Controller

Clone the Final Challenge git repository

```
git clone git@github.com:greenlambda/ELEC-424_Final_Challenge.git
```

and install using

```
sudo ./setup.sh
```

Then run the client using

```
bin/cfclient
```

The client should open and look normal.

Flying the Quadcopter

To fly the quadcopter with our code, on can use manual, AltHold or the completely automatic AI mode. Use the control sticks as normal to operate in manual control mode. Activate AltHold mode by toggling the AltHold button.Toggling the AltHold button again will resume manual control mode. In AltHold mode thrust is controlled automatically while pitch, yaw and roll are able to be used manually. Press the “exit” button on the controller to start AI mode. AI mode will block out all input from the controller other than pressing “exit” again to exit AI mode. AI mode causes the crazyflie to calibrate, takeoff, hover at a fixed height, then land.

Sensors

The crazyflie contains a number of sensors including: a three axis accelerometer, a three axis gyroscope, a three axis magnetometer, and a barometer. In addition, we were given a ultrasonic rangefinder to attach to the crazyflie. In our project, the ultrasonic range finder was not used due to concerns of stability with the added weight.

Data from the three axis gyroscope and three axis magnetometer was taken at 250 Hz, then fused together using Mayhony's Altitude and Heading Reference System Algorithm, which produces accurate, high speed, and stable information about the orientation of the crazyflie with respect to

gravity. This orientation information is used to control six PID loops which control the pitch, roll, and yaw of the crazyflie, making it stable and allowing it to fly. With these controllers, the yaw of the quadcopter is very stable, making integration with the magnetometer unnecessary. Because of this, the magnetometer is not used in our end device.

In addition to providing orientation information, the sensor fusion algorithm provides the vertical acceleration (with respect to gravity). By integrating this data twice, one can obtain an absolute height in space. This data however is extremely noisy and prone to large amounts of drift. In order to overcome this, we used a complementary filter with this data and the barometer. The barometer gives noisy, slow data, so it is low pass filtered and combined with high pass filtered data from the accelerometer fusion data. This is further improved by adding a bias cancelator to the accelerometer with an integral term based on the error in the barometer data and the full height estimation. In the process, this height estimation also produces an estimation of the vertical velocity of the crazyflie.

In order to produce the AltHold mode, we combined the readings from the vertical velocity and absolute position. These readings are independently fed into a two PID controllers, whose outputs are summed with a weight to produce the final thrust value of the system. The vertical velocity PID controller attempts to maintain the crazyflie's vertical velocity at zero the entire time, producing a relatively stable hover. However, due to error or drift in this sensor value, the controller cannot do a perfect job, causing the crazyflie to drift up or down. In order to combat this, the absolute position controller attempts to keep the crazyflie at a constant height, fighting against the output of the velocity controller. Two independent controllers are use because of the slow speed of the data used to fly the crazyflie versus the high speed nature of the control is required. The vertical velocity controller produces most the high speed control while the absolute position controller controls the slower overall goal.

Challenges

We faced several challenges in our progress to complete this lab. These challenges were varied in nature, and not all offered us many options. Possibly the most notable (and most noticeable) of these challenges was that the quadcopter drifts. In trying to get the quadcopter to hover at a specific altitude, the quadcopter would float up and down several feet. This posed a challenge in that this movement contributed to the overall instability of the system, cost additional battery life, and most importantly, sends the quadcopter out of the 4ft - 6ft scoring region for the competition. Tuning the PID for the AltHold control system assists in limiting some of this instability, however the real source of this issue was the inaccuracy of the data for computing this altitude. Our method for computing the altitude of the quadcopter consisted of accelerometer data to determine vertical velocities and the barometer to determine the absolute height. The accelerometer data would drift substantially and the barometer data was not quick enough to compensate. This all lead to inaccurate altitude estimations, contributing to an oscillating quadcopter.

In addition to vertical drift, the quadcopter would also drift horizontally. This drift wasn't necessarily the result of drift in onboard sensors, but rather the result of indoor air currents, motor biases, or natural error arising in the stabilizing functions of the quadcopter. This posed a challenge because this drift would take the quadcopter out of the scoring region or into the wall. During development, this horizontal movement put the quadcopter at risk of colliding with foreign objects causing damage or otherwise taking the quadcopter out of service.

In the discussion of the vertical drift, we noted the issue being difficulty in obtaining usable data from the involved sensors. The most substantial issues came from the barometer, which would oscillate on the order of tens of centimeters in a pattern consistent of very low frequency noise. The frequency of these oscillations made it very difficult to filter without losing all usable data, considering we were using the barometer readings as the reference for absolute altitude offset. This error affected the AltHold control system as described above.

To address these challenges, we developed a system of dualling PID controllers which would together control the thrust of the quadrotor. The first PID loop would aim to match the desired altitude, while the other would aim to maintain a constant vertical velocity of zero. This system aimed to combat the instability of solely relying on the altitude estimation functionality. When we rearranged the PID loops in a cascaded pattern rather than summing their outputs we faced issues in that the quadcopter could stay at accurate velocities, but with inadequate response times. There was not enough agility to account for errors. Ultimately, we addressed this challenge by reverting back to our solution with the outputs of the two PID loops summed rather than cascaded.

The next challenge associated with these PID systems is that the particular PID values for each needed to now be re-developed and re-tuned. To begin, the proportional values were increased until the quadcopter could reach a steady state with stability and compensate for error in an agile manner. Then the Integral values were increased to remove the offset. The Derivative values were set to small values to assist with overshoot issues. Each of the PID values were largely difficult to tune given the irreproducibility of the tests due to the randomness of each flight. Different air currents affected each flight differently, and testing in different locations would contribute to different responses to various PID values.

The next challenge is in regard to tracking the wall with the ultrasonic range finding sensor. Using the ultrasonic sensor would require an additional PID control system after lots of filtering on the input data. Given the difficulties simply having the quadcopter hover at a particular altitude, we decided that efforts to develop and use the ultrasonic sensor would serve little or no benefit. Additionally, using the ultrasonic sensor added weight to the quadcopter, decreasing battery life, offsetting the center of gravity, and harm the stability of the whole system. Due to the usability

issues and added complexity the sensor would add, we ultimately decided not to pursue implementing the sensor for our competition flights.

The final challenge we would like to note is that of replacing the hardware on the crazieflie. Testing the autopilot functionality requires putting the device in a state where it is vulnerable and not under direct human control. In this way it is in danger of crashing into different objects, walls, or ceilings. Motors are very prone to breaking when the crazieflie crashes. Unfortunately, replacing broken motors was very time and labor intensive. Ultimately we had to replace at least 6 motors prior to our completion of the competition. There was no permanent solution to this issue, so this challenge remained unaddressed.

Video

A video of our final design flying in the competition can be found here:
http://youtu.be/VB4XM5QM5es

Work Percentages

Jeremy Hunt (jrh6): 50%
Christopher Buck (cmb15): 50%