Autonomous flying of quadrotor
for 3D modeling and inspection of mineshaft.

Max Unander

Civilingenjör, Teknisk fysik
2018

Luleå tekniska universitet
Institutionen för system- och rymdteknik
Autonomous flying of quadrotor for 3D modeling and inspection of mineshaft.

Master Thesis in Engineering Physics and Electrical Engineering

Max Unander
maxuna-2@student.ltu.se

Academic supervisor: Per Lindgren
Project supervisor: Johan Eriksson

Dept. of Computer Science, Electrical and Space Engineering

2018-03-05
This thesis presents a purposed navigation and control solution for autonomous flying of a quadrotor in confined spaces where positioning with GPS is not feasible. The problem and related application considered is autonomous flying of a quadrotor in mineshafts for 3D modeling and inspection. Theory and test results are presented including positioning experiments with a camera combined with a UWB-ranging radio and a P-PI regulator for position control. It is argued that this is a viable and cost effective solution that can be used to determine the position of the copter and control its position to within 0.2 meters with stable controlled velocity.
ABBREVIATIONS

GPS ............ Global Positioning System
UWB ............ Ultra Wideband Radio
SLAM ............ Simultaneous localization and mapping
MCU ............ Microcontroller unit
LIDAR ............ Light detection and ranging
IMU ............ Inertial measurement unit
PID ............ A proportional-integral-derivative controller
P-PI ............ A proportional-proportional-integral controller
# Table of Contents

1 Introduction 1
   1.1 Background .................................................... 2
   1.2 Problem statement ............................................. 2
   1.3 Goal .......................................................... 2
       1.3.1 Scope .................................................... 2
   1.4 Motivation ...................................................... 3
   1.5 Previous work ................................................. 3
   1.6 Collaboration .................................................. 3
   1.7 Outline ........................................................ 4

2 Method 5
   2.1 Hardware ....................................................... 5
       2.1.1 Copter parts ............................................. 5
       2.1.2 Circuit boards ........................................... 7
   2.2 Positioning with camera and UWB .............................. 10
       2.2.1 Pixel positioning SimpleCV ............................ 15
   2.3 Kalman filter ................................................... 17
       2.3.1 Equations ................................................. 18
       2.3.2 Code generation with Matlab .......................... 20
   2.4 Positional control ............................................. 24
   2.5 Communication ................................................ 32

3 Results 33
   3.1 Positioning with camera and UWB ............................ 33
   3.2 Kalman filter .................................................. 34
   3.3 Positional control ............................................. 36

4 Discussion and Conclusion 39
   4.1 Positioning ..................................................... 39
       4.1.1 Kalman filter ............................................ 40
   4.2 Positional control ............................................. 40
   4.3 Conclusion ...................................................... 41
In recent times we have seen a huge interest in autonomous vehicles, you can see that many of the big car companies are developing self driving cars and some of them already have systems in place to let the car drive for you in certain situations like adaptive cruise control to the car in front, lane following and automatic braking to avoid collisions. Another area where autonomous vehicles are getting popular are unmanned aerial vehicles especially vertical take of and landing like multirotor copters. These have made things like aerial photography part of the mainstream that was only available to big productions before. Due to the characteristics of these copters they can takeoff and land in very confined spaces and fly accurately as long as it’s position is known, this makes them versatile and can be used for many applications.

This has sparked interested in companies that perform inspections or want to perform inspections in places that could be hard or dangerous for people to inspect.

In this project we will try to tackle one of these inspection problems, inspecting mineshafts inaccessible to people.
1.1 Background

Mining companies want to be able to detect damages in mine shafts that could cause cave-ins in nearby shafts that have people in them. The shafts they want to inspect are restricted to people because the risk of falling rocks so any solution must be operable from the outside. The way mining companies solve this now is that they drill holes from the nearest accessible points in the mine and inserting sensors through the hole. This method is expensive and time consuming.

1.2 Problem statement

The new proposed solution is to fly a quadcopter through the shaft with a laser scanner attached to get a 3D model of the shaft. The problem with autonomous flying with a quadcopter is that we are relying on integrated values for attitude which means that with any small error it will start to drift. This drift in angle is normally filtered out with an accelerometer bias but this only works if we don’t have any accelerations on the copter, which means we need a constant velocity or position. The most common way to solve this problem is to get position from GPS but this will not be available in the mines. Our solution for this is to combine a camera positioning system[1] with a ranging UWB-radio[2] to get position. By moving the positioning away from the copter we can drastically reduce the cost copter in comparison to other solutions like SLAM[3], which require a lot of computational power and high precision if the environment lacks distinct feature points.

1.3 Goal

The goal with this project is to see if it is feasible to develop a cost effective solution for autonomous flying of a quad rotor for scanning and inspecting mineshafts.

1.3.1 Scope

The scope for this project is to develop a prototype to test the functionality of the positioning and autonomous flaying. No test will be done in an actual mineshaft.
1.4 Motivation

The reason we are doing this is that we can see that there is a need for systems that could be implemented in places where the common forms of positioning like GPS are not available. The system proposed in this project could be implemented anywhere where can have line of sight to the copter from a base station.

1.5 Previous work

This master thesis is a continuation of a project that was started in a Master’s Project Course[4] conducted here at my university, I was part of this project and got the opportunity to continue with the project as my master thesis.

The work that was done that concerns this thesis were that the copter was built and tuned to fly properly. Communication between the MCU and flight controller was working and a simple collision avoidance was implemented using ultrasonic sensors.

1.6 Collaboration

This project was done in collaboration with Lars Jonsson, were I focused on positional control and Lars focused on scanning and building a 3D model. This means that the positioning part of our work overlap and the sections in this thesis concerning positioning, ”Positioning with camera and UWB”2.2 and ”Kalman filter”2.3.

Lars Jonsson thesis ,
"3D modeling of mineshaft using autonomous quad rotor”[5]
1.7 Outline

Chapter 1 will give an introduction to the project presented in this theses by going through background, problem, purposed solution, goals, motivation and previous work for the project to get an insight in why we are doing this.

Chapter 2 goes through the method and theory that was used in order to solve the problems presented in this thesis. These include choice and design of hardware 2.1, positioning with camera and UWB 2.2, Kalman filter theory and code generation 2.3 and positional control of the copter 2.4.

Chapter 3 presents important results that were gathered in this thesis.

Chapter 4 will discuss the advantages and disadvantages of the solutions presented in this thesis and give a final conclusion.
The problem with autonomous flying with a quadcopter is that we are relying on integrated values for attitude which means that with any small error it will start to drift. This drift in angle is normally filtered out with an accelerometer bias but this only works if we don’t have any accelerations on the copter, which means we need a constant velocity or position. The most common way to solve this problem is to get position from GPS but this will not be available in the mines. Our solution for this is to combine a camera and a ranging UWB-radio to get position.

2.1 Hardware

2.1.1 Copter parts

Figure 2.1: Picture of the copter.
Since we decided to base our quadcopter around the open source flight controller Flip32 that runs Cleanflight [6]. An open source flight controller was chosen because of the ability to customize your configuration and multiple ways of communicating with the flight controller, this also meant we had to build the copter. When starting to assemble a copter we first started by choosing a suitable frame to fit the equipment. A 65cm square frame was chosen to get space for the sensors between the rotors. Next step was to choose motors and propellers to get enough lift for our payload at a comfortable throttle setting. The NTM propdrive 28-30S motor with a TGS 12x6 propellers and a 4S Li-Po battery would suit our needs. This would give us enough lift for 3kg of payload at full throttle eq.(2.1 and 2.2), which means we should be able to fly with 1kg payload at a reasonable throttle setting. ESC’s where chosen to handle the battery voltage and amperage of the motors, Afro 30A fit these requirements and also runs the open source software SimonK[7] which allows a lot of settings. The last step was to choose battery size and discharge rating to handle flight time and current delivery. We wanted to at least five minutes of flight time so we chose a Zippy 8000mAh 30C battery which would give us around six minutes of flight at full throttle (eq.2.3) and more than enough current delivery.

\[
M_{\text{Copter}} = M_{\text{frame}} + 4 \cdot M_{\text{motors}} + 4 \cdot M_{\text{ESC}} + 4 \cdot M_{\text{prop}} + M_{\text{battery}} = 1750g \quad (2.1)
\]

\[
\text{Payload}_{\text{max}} = \text{Thrust}_{\text{tot}} - M_{\text{Copter}} = 4 \cdot 1200 - 1750 = 3050g \quad (2.2)
\]

\[
\text{Flight time} = \frac{\text{Battery capacity}}{\text{Max amp}} = \frac{8000mAh}{80A} \approx 6 \text{ minutes} \quad (2.3)
\]
2.1.2 Circuit boards

We decided to make a shield to distribute all connectors from the Nucleo board base. The reason we decided to make a shield was that we still wanted to use the Ethernet port on the Nucleo and since we lack any real analogue components keeping the short traces to reduce disturbances was not a priority. The Nucleo board can be seen in figure 2.2 and the shield in figure 2.3. The shield board contains connectors to all our peripherals except the LIDAR, because the LIDAR uses Ethernet communication which is already integrated on the Nucleo board. We used a dip socket for the IMU because of the convenience when setting up the chip with our breakout board and
then just moving the chip.

![Figure 2.3: Nucleo shield in Eagle](image)

We decided to use a switching regulator to make a five volt rail for peripherals, because of efficiency when stepping down from 15 volts. We also added a second switching regulator for driving the high power LED that will be used for positioning. A three volt rail for logic level was needed so a linear regulator from five volt was suitable there. All power rails and the stepper motor driver for the laser scanner were handled on a separate board to avoid disturbance seen in figure 2.4.
Figure 2.4: *Power distribution board in Eagle*
2.2 Positioning with camera and UWB

Our plan to solve the positioning problem is to have a camera and a UWB ranging radio, by combining the pixel position from the camera and combining it with range from the UWB it’s possible to calculate the position. To make it easier to identify were in the image the copter is, a bright light was attached to the copter.

To be able to calculate our distance y which is the perpendicular distance of the copter from the base station we need relationship between pixel position and angle $\theta$ since the UWB radio will give us the length of the hypotenuse.

To figure this out we are gonna look at it in two dimensions first see figure 2.5, we see that in order to calculate $\theta$ from Pixel $P_x$ we need to know $P_y$ (Eq.2.4).

$$\theta = \arctan\left(\frac{P_x}{P_y}\right)$$  \hspace{1cm} (2.4)
\[ \phi = \arctan \left( \frac{P_z}{P_y} \right) \]  

(2.5)

In order to calculate \( P_y \) we use Eq.2.6. We know that \( P_y \) is constant for all \( P_x \) and \( \theta \) because our camera gives a flat image without distortion.

\[ P_y = \frac{P_x}{\tan \theta} \]  

(2.6)

If we want to make this work with all dimensions we just need to work out the absolute distance in the picture plane with the Pythagorean theorem seen in Eq.2.7

\[ \alpha = \arctan \left( \frac{\sqrt{P_x^2 + P_z^2}}{P_y} \right) \]  

(2.7)

\[ P_y = \sqrt{P_x^2 + P_z^2} / \tan \alpha \]  

(2.8)

This means that the specifications from the manufacturer of Field of view angle (FOV) and resolution can be plugged in to Eq.2.8, with \( \alpha = \frac{FOV}{2} \) and \( P_x = \frac{(Horizontal\ resolution)}{2} \) and \( P_z = \frac{(Vertical\ resolution)}{2} \) (Eq.2.9). Important to use diagonal resolution because that’s the direction of which the field of view is defined on our camera.

\[ P_y = \frac{\sqrt{(Horizontal\ resolution)^2 + (Vertical\ resolution)^2}}{2 \tan \left( \frac{FOV}{2} \right)} \]  

(2.9)

Now that we are able to calculate \( \alpha \) we can simply use Eq.2.10 to calculate \( y \) position or when \( \alpha \) is small like for example when the copter is far away we can use small angle approximation and use Eq.2.11.

\[ y = UW B_{range} \cos \alpha \]  

(2.10)

\[ y \approx UW B_{range} \]  

(2.11)

Similarly we get \( x \) and \( z \) distance with,

\[ x = UW B_{range} \sin \theta \]  

(2.12)
\[ z = UWB_{\text{range}} \sin \phi \]  

(2.13)

Since we want the positional calculations to run as fast as possible, trigonometric functions are not optimal, since they are more computationally heavy than linear equations.

By using properties of similar triangles we can use Eq. 2.14 to calculate a linear equation for \( x \) if we know \( y \) and \( Px \) with Eq. 2.15 since \( Py \) is constant.

\[
\frac{Px}{Py} = \frac{x}{y} \tag{2.14}
\]

\[ x = y \cdot \frac{Px}{Py} \tag{2.15} \]

and similarly,

\[ z = y \cdot \frac{Pz}{Py} \tag{2.16} \]
To check the function for position in x, y and z due to pixel position we took pictures with the webcam and had an A4 paper for scale at multiple ranges. This way it’s possible to count how many pixels the paper is across and compare that to the size of the paper seen in figures 2.6 and 2.7.
Figure 2.7: Picture taken with webcam with A4 paper for scale at 16 meters to calculate distance in x and z direction due to pixel position.
2.2.1 Pixel positioning SimpleCV

To identify where in the picture the copter is a program called SimpleCV was used, this is a simplified version of the popular computer vision library OpenCV. SimpleCV has some useful tools like "blobs" which can find the pixel position of groups of the same color, but if we don’t do any image processing before you will find blobs everywhere.

![Picture of the copter with LED without image processing.](image)

**Figure 2.8:** Picture of the copter with LED without image processing.

The way we solve this is by taking a background picture before we start and because the camera doesn’t move and the background doesn’t change we can simply subtract every picture with our background. This way the bright LED on the copter will be the only blob detected. You can see the copter without image processing in figure 2.8 and with image processing in figure 2.9 where we can see a blob detected on the copter.
Figure 2.9: Screenshot of SimpleCV running image processing and blob detection for pixel position of the copter.
2.3 Kalman filter

The input to the flight controller runs at a refresh rate of 50 times per second, and the receiver for manual input also runs at 50Hz. Since we can’t guarantee that the raw positioning updates at that rate we need a way to predict position between measured positions.

When talking about predicting filters the most common and widely used are Kalman filters. Kalman filters are great for filtering out noise without introducing delay like a low pass filter would, but can also be run faster than input measurement by utilizing the predicting part of the filter. To get even more accurate between measurements when running the filter faster than the input you could also use sensor fusion. Sensor fusion is when you add more sensors to more accurately fill the gaps between measurement, a common solution is to add measurements from an inertial measurement unit. The measurements from this unit comes from integrated values of acceleration and angular velocity which make them unreliable but can greatly improve accuracy in the short time between measurements, one such filter can read about in ”Quaternion kinematics for the error-state Kalman filter, by Joan Sol’a” [8].

We decided to use a multi rate Kalman filter due to time restrictions. The difference between a multi rate and a conventional Kalman filter is that you run the prediction part at the update rate you want to have and only run the whole filter with estimation when you get an updated position. By running the Kalman filter faster than the measured position we also ensure a constant refresh rate which is desired when run the regulator which you can read about in section 2.4.
2.3.1 Equations

The input to our Kalman filter will be $p_m$ and it contains the measured positions $x_m, y_m, \text{ and } z_m$ seen in Eq.2.17

$$p_m = \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} \quad (2.17)$$

The $A$ in Eq.2.18 matrix is set up so that position is position and velocity is position times $\Delta t$ and $I_3$ is an identity matrix of size 3.

$$A = \begin{bmatrix} I_3 & I_3 \Delta t \\ 0_3 & I_3 \end{bmatrix} \quad (2.18)$$

The uncertainty of our states is modeled by the matrix $Q$ where all the uncertainty is in velocity due to the fact that we get velocity from position and can be tuned with $Q_v$.

$$Q = \begin{bmatrix} 0_3 & 0_3 \\ 0_3 & Q_v \end{bmatrix} \quad (2.19)$$

How much the noise in the measurement has is modeled white by noise $\sigma$. This noise is the same for $x$ and $z$ because they both come from a pixel position in the camera. The noise in $y$ comes from the UWB-radio. This is set up in matrix $R$ Eq.2.21

$$R_x = (\sigma_{cam})^2$$
$$R_y = (\sigma_{UWB})^2$$
$$R_z = (\sigma_{cam})^2$$

$$R = \begin{bmatrix} R_x & 0 & 0 \\ 0 & R_y & 0 \\ 0 & 0 & R_z \end{bmatrix} \quad (2.20)$$

$$R = \begin{bmatrix} R_x & 0 & 0 \\ 0 & R_y & 0 \\ 0 & 0 & R_z \end{bmatrix} \quad (2.21)$$

Prediction stage

Predicted state estimate where $\hat{x}_{k+1}$ contains predicted positions and velocities.

$$\hat{x}_{k+1} = A_k \hat{x}_k \quad (2.22)$$
Predicted estimate covariance

\[ P_{k+1} = A_k P_k A_k^T + Q_k \]  \hspace{1cm} (2.23)

To keep the covariance matrix non-singular

\[ P_k = \frac{P_k + P_k^T}{2} \]  \hspace{1cm} (2.24)

**Estimation stage**

Difference between Predicted and measured position,

\[ \hat{y}_k = p_{mk} - H_k \hat{x}_k \]  \hspace{1cm} (2.25)

Innovation or residual covariance

\[ S_k = H_k P_k H_k^T + R_k \]  \hspace{1cm} (2.26)

Optimal Kalman gain

\[ K_k = P_k H_k^T S_k^{-1} \]  \hspace{1cm} (2.27)

Updated state estimate

\[ \hat{x}_{k+1} = \hat{x}_k + K_k \hat{y}_k \]  \hspace{1cm} (2.28)

Updated estimate covariance,

\[ P_{k+1} = (I - K_k H_k) P_k (I - K_k H_k)^T + K_k R_k K_k^T \]  \hspace{1cm} (2.29)
2.3.2 Code generation with Matlab

Kalman filter will be calculated on the copter to avoid delay in sending and make it so that it still can predict position if some packets are lost when sending. This means that the Kalman filter has to be written in C-code to make compatible with code written for the MCU on the copter. We choose to generate this code with Matlab because of the poor support for vector and matrix operations in C.

To set up your Matlab-scripts for code generation you have to write it as a function with a line ”%#codegen” at the top so Matlab knows you want to generate code. Next step is to go to ”Apps” and click ”MATLAB Coder” seen in Figure 2.10.
Figure 2.11: Screenshot of first step of code generation where you choose function and numeric conversion for variables.

After you start "MATLAB Coder" you will choose the function you want to generate code for and choose numeric conversion. We want to use single precision to get float in the C code and click next seen in Figure 2.11
In the next step you will show "MATLAB Coder" how the function is run with appropriate input that will represent the position measurement $p_m$, this is done by writing it the same way you would write it when calling the function in a script or from the command line. We also choose numeric conversion and define orientation of the input vector seen in Figure 2.12.
The last step of the code generation is to define build type and code language. We choose “Source Code” and “C” to get the generator to output ”.c” and ”.h” files. This will generate all ”.c” and ”.h” files you need and will also generate an example ”main.c” file that shows how to call your function in C-code.
2.4 Positional control

Before we start setting up the regulator for controlling the system we need to establish how the copter is set up in our coordinate system. The coordinate system is set up so that Y is forward, X is left and right and Z is up and down. To move the copter we can use inputs "Roll, Pitch and Yaw" which can be seen in figure 2.14. For positional control it makes sense to line up the X-axis with pitch and Y-axis with roll, so that a positive input in roll will move the copter positive in X (Right) and a positive input in pitch will move the copter positive in Y (Forward), see figure 2.15.

**Figure 2.14:** *Picture of how Roll, Pitch and Yaw is defined on the copter. (Plane is used to make it easier to see direction)*
To make sure that the copter is lined up in the coordinate system so that the control signal will move the copter in the correct direction, a regulator that controls yaw angle is needed. A yaw controller is already integrated to the
flight controller that works well for controlling the angular velocity and keep yaw angle fairly consistent but not perfect. To remove imperfections in yaw angle a Proportional regulator with an external IMU was added this can be seen in figure 2.16.
The most common control loop system when you want to hold a setpoint is a PID-controller. A PID are good at holding position and easy to tune but if you want to change setpoint it’s not the best choice. In the case where you want to change setpoint P-PI controller is better because you are able to control velocity and position independently. The most common application for P-PI controllers are electrical servos, to achieve precise position and controlled rotation speed[9]. This characteristic is achieved by having a common PI-regulator that controls the velocity and then having a P-regulator for position that sends its output to the input of the regulator for velocity. This can be seen in figure 2.17. By not letting the position directly control the output we can achieve controlled velocity even with aggressive positional tuning. One of the biggest reasons we use this type of controller is that it can be tuned without a physical model of the system. Since every part of the system is computer controlled a physical model is difficult to derive.

Figure 2.17: Block diagram for the P-PI positioning controller reference is our set point, p and v are position and velocity from the Kalman filter and output is the control signal that is sent to the flight controller.

As you see in figure 2.17 it has three saturation blocks these are very important to limit the controller. If we start from the right with ”Saturation2” this limits the maximum output that the P-PI controller can send to the flight controller, it is important that this controller does not max out the input of the flight controller because we still need room for the collision avoidance to work and be able to control it manually if needed. Next saturation block ”Saturation1” is used as an anti wind up to limit integral part of the velocity controller from building up too much. In situations where the velocity is in one direction for a longer time can build up more than desired, an example where this could happen is when we change position. The last saturation block ”Saturation” is there to limit the input to the velocity controller from
the positional controller. Since the positional controllers output is an offset in velocity this saturation block limits the maximum speed the copter will move due to change in position. This means that can move our setpoint to move the copter far away and it will not overreact.

\[ \text{Figure 2.18: Block diagram for the P-PI With only the Kalman velocity and PI regulator for controlling the velocity. A manual input is added to introduce disturbances on the system for testing.} \]

The process of tuning the P-PI regulator starts by tuning the PI part that controls the velocity, the system is set up according to figure 2.18. The copter is flown with a manual input to introduce disturbances and velocity to the system. We start by tuning \( Kp \) and keeping the gain of the integrator to zero. The gain \( Kp \) controls the output to the flight controller in direct proportion the the measured velocity by the Kalman filter, we want this gain high enough to counteract the velocity on the system, but not so high to introduce a velocity in the opposite direction (this will cause the system to oscillate). Next step is to tune the gain of the integral part of the regulator. This part will build up a control signal and remove imperfections in the system like for example if the system has a tendency to drift in a certain direction the integral part will compensate for that. The integral part will also help keep the speed constant when moving the copter. When tuning the gain for the integral part it’s the same procedure as for the proportional gain, you want it to build up fast enough to remove imperfections, but not so fast as to introduce velocity in the opposite direction as it will cause oscillations. The last part of tuning is to set “Saturation1”, it needs to be large enough to be able to remove imperfections but no bigger than that so that the integral part doesn’t build up too much if the velocity is positive or negative for a longer time like for example when we are moving the copter.
Figure 2.19: Block diagram for the P-PI With only the Kalman position and P regulator for controlling the position. PI regulator for velocity is represented in one block.
When flying the copter we saw that we would get an offset in position, this is due to imbalance in the copter that would cause it to drift in a certain direction. To remove this offset an I regulator for position with a small gain was added to the output seen in figure 2.20. The reason for having small gain is that the offset is more or less constant and once it finds the correct value it will not have to change. Also by having such a small gain it will not impact the rest of the regulator. A saturation block was also added to prevent it from building up when moving the copter.
The last thing added was a simple collision avoidance which consisted of a P regulator that only kicks in when the sonar measures that the copter is close to a wall.

The complete system can be seen in figure 2.21.
2.5 Communication

Our system components and how they communicate can be seen in figure 2.22 here you can see how everything is connected to the MCU and what communication protocol they use. We focus on the parts that concern this thesis which is Position and positional control. We start at the base station which will be in charge of doing positioning of the copter with a camera and UWB-ranging, it will do all necessary calculations to calculate X, Y and Z position of the copter and then send it to the copter over Wi-Fi along with setpoint which is the position we want to copter to be at. These measured positions will then be run through the Kalman filter on the MCU before being sent to the P-PI regulator that will keep the copter at the setpoint. The MCU takes in PPM [10] signal from the receiver for manual control of the copter, PWM signal from the sonars for collision avoidance and UART from the external IMU. To communicate with the flight controller a UART protocol called Multiwii Serial Protocol [11] was used.

Figure 2.22: A picture of all parts in the system and how they communicate with each other
CHAPTER 3

Results

3.1 Positioning with camera and UWB

![Plot of the Width of a Pixel at Given Distances](image)

**Figure 3.1:** Plot of the width of a pixel given distance from the camera.

The result from our pixel size measurement in figures 2.6 and 2.7 can be seen in figure 3.1 where the size of one pixel is plotted over distance \( y \). Here we see that we get a linear behavior like we expected from Eq. 2.15.

We compare our measured with the theoretical which we can get by calculating \( P_y \) with Eq. 2.9 to 1975.83 and putting that in Eq. 2.15 with \( P_x = 1 \) we get a slope of \( 0.06E - 4 \). By getting the same result with measurement and theory we can be relatively certain that are calculations are correct. To get a reasonable refresh rate of 20 Hz on the position we decided to run the image processing at half the resolution of the camera this means that every pixel...
will be twice as big. This will give us a precision of ±10\,cm at a distance of 200 meters assuming the noise is one pixel.

### 3.2 Kalman filter

![Figure 3.2: Plots of measured position and Kalman filter position tested in a corridor up to 30 meters. Vertical axis is position in meters and horizontal axis is time in seconds.](image)

In figure 3.2 we see plots of the measured position that we get from the image processing with camera and UWB radio ranging and the Kalman position that is calculated on the copter. This test was performed by carrying the copter up and down a 30 meter long corridor. Here we see that the Kalman filter does a good job of calculating position, it follows the measured data well without over estimating and doesn’t cause filtered data to lag behind which could happen with an incorrectly tuned filter.
In figure 3.3 we can see a zoomed picture of the plot we saw in figure 3.2. In this picture we can see how well the Kalman filter handles noise on the measured positions. We can see that almost all noise is filtered out but still has some ripples, these ripples are in the magnitude of a couple of centimeters and will not greatly impact the ability to control the copter.
3.3 Positional control

We can see in figure 3.4 that with a well tuned P-PI regulator the copter maintains fairly constant position and changes position with a constant speed. We can also see that we get an offset from the setpoint, this is due to imbalance in the copter.

We can see in figure 3.5 that when once we added a trim function to compensate for the imbalance in the copter the problem with offset we had before is gone.

Figure 3.4: Plots of position setpoint and control signal when flying the copter with P-PI regulator without trim.

Figure 3.5: Plots of position setpoint and control signal when flying the copter with P-PI regulator without trim.
We can also see that the copter keeps a position that is about \( \pm 0.2 \) meters from the setpoint, this means that we could comfortably fly the copter in the specified mine shaft which has a diameter of 3 meters.

![Position and setpoint (X)](image1)

![Control signal [ROLL]](image2)

**Figure 3.6:** Plots of position setpoint and control signal when flying the copter with P-PI regulator with too high gain and saturation.

We take a look at figure 3.6 we can see an example of an improperly tuned regulator in this case the gain \( K_p \) that controls position is tuned too high which causes the copter to oscillate with increasing amplitude and does not keep position well.

![Position and setpoint (Y)](image3)

**Figure 3.7:** Plots of position setpoint to show how fast the copter moves when changing setpoint.

In figure 3.7 we can observe how the copter react when we change setpoint. We can see that moves towards the new setpoint with a constant speed we
no overcorrection, we can also see that it covers a distance of 1 meter in approximately 3 seconds which is in line with the $0.3m/s$ that the regulator was programmed to have as a top speed due to position in this test.

Figure 3.8: 3D plot of the setpoint and position of the copter flying in a square shape clockwise.

In figure 3.8 we see the result of the copter taking off, flying clockwise in a square pattern and landing. We can see that it follows the pattern of the setpoint fairly well with a maximum deviation of 0.3 meters from the set path. The reason the position shows has large fluctuations from the setpoint before it reaches the height of the square, is that it is too close to the ground for stable flight due to wind turbulence. We can also see a clear offset in position in the left leg of the square, this is because the trim function had not yet built up enough to compensate for the imbalance in the copter. We can see that the copter keeps position without offset, once the trim function has had more time to work.
4.1 Positioning

When discussing our choice of method for getting position on the copter I’m going to compare it to SLAM\[3\] and put forth the advantages and disadvantages of our solution, let’s start with with the advantages.

One of the biggest advantages of our solution is that it moves the computational power of calculating the position away from the copter which allows us to keep the price of the hardware cost low in comparison to SLAM and since this is designed to inspect areas were the risk of the copter being lost due to falling rocks or other outside hazards keeping the cost of the copter low is desirable. The second advantage of our system is that it does not rely on feature points like SLAM does which means the that it can be deployed without any work having to be done in the area that is going to be inspected as long the base station can have line of sight. Since SLAM relies on feature points for positioning which can be hard to find in a shaft that is basically a smooth pipe, this means that in some areas feature points like reflectors might have to be added in order for the SLAM to work correctly.

The biggest disadvantage of our system is that it relies on line of sight from the base station which means that the copter will not be able to fly outside the view of the base station, this is not a problem you get when using SLAM. Another disadvantage with calculating the position on a base station is that you will have to send that position wirelessly to the copter which will introduce delay.
4.1.1 Kalman filter

When discussing the performance of our multirate Kalman filter I’m going to compare it to a sensor fusion Kalman filter[8]. The advantage of using a sensor fusion Kalman filter compared to a multirate is that it actually gives input to the filter between measurements instead of just predicting the position, this means that the filter with sensor fusion can update its position and velocity between measurements while the multirate filter will only update its position with a predicted constant velocity. If we take figure 3.8 as an example we can see that the position deviates 0.3 meters from the setpoint when moving it in a square pattern but when flying it in a strait line we get a deviation of 0.2 meters this could be because the filter does not change the direction of its velocity fast enough when performing 90 degree turns, by using a sensor fusion filter the velocity would be able to change direction faster. The disadvantage of using a sensor fusion is that the filter becomes much more computationally heavy due to the increased size of the matrices.

4.2 Positional control

When discussing the positional control we must first address the choice of regulator. Since a PID is the most common and widely used regulator we will address first why we didn’t this type of regulator. A PID regulator works well for keeping position and would probably perform about the same as the P-PI regulator we used if the setpoint was stationary. Where the P-PI excels over the PID is when moving the setpoint because the position and velocity part is separated and the position does not directly control the output the maximum movement speed to position can be limited without impacting its ability to keep position. The next part we need to discuss is why not an optimal control regulator? An optimal control regulator is set up by using a physical model of the system, because this regulator is designed from the physical properties of the system it knows how the system will react to an input. The disadvantage of using this type of regulator is that you need to know the physical properties of the system you want to control and in our case a physical model is hard to derive because every part of our system is computer controlled.
4.3 Conclusion

In this thesis I was able to show that you are able to accurately determine the position of the copter using a normal webcam and a UWB ranging radio combined with a multirate Kalman filter. In testing we observed an estimated deviation of a few centimeters up to a distance of 30 meters and this accuracy would theoretically only increase to about 10 centimeters at a distance of 200 meters. We also showed that we are able to control the position of the copter using a P-PI regulator with deviation of 20 centimeters which would make it possible to fly in confined spaces, with a stable constant speed when moving the copter.

One thing that could have been done differently to improve the result of this project, is to base the project on a prebuilt copter. We had a lot of problems related to the flight performance of the copter, which is expected because we have no experience building quadrotors. If a prebuilt copter was chosen you would have to take the communication with the flight controller into account.

The things that are left to do in this project is, testing the position and positional control up to 200 meters, and for flying in a mine, weather proofing of all components is necessary.

My final conclusion is that this would be a viable low cost solution, for inspection of confined areas that are restricted to people. This solution could also be implemented in any application as long as line of sight is established to the base station.


[3] Visual 3-D SLAM from UAVs, Jorge Artieda, Jose M. Sebastian, Pascual Campoy, Juan F. Correa, Ivan F. Mondragon, Carol Martinez, Miguel Olivares http://www.disam.upm.es/campoy/Pascual_Campoy/publications_files/Visual3DSLAMfromUAVs.pdf


[9] Tuning P-PI and PI-PI controllers for electrical servos,  
    T. ZABINSKI and L. TRYBUS, 2010  
    http://bulletin.pan.pl/(58-1)51.pdf

[10] Pulse Position Modulation (PPM),  
    https://oscarliang.com/pwm-ppm-difference-conversion/

    http://www.stefanocottafavi.com/msp-the-multiwii-serial-protocol/