In-door Localization based on Ultra Wide Band Distance Sensor

Mathias Koch, s123950
Nicolai Sahl-Tjørnholm, s123860

DTU, 11. December 2015

Mathias Koch

Nicolai Sahl-Tjørnholm

Abstract

To navigate an autonomous robot safely in its surroundings, the position of it must be known to some extent. One way of doing this is by having an external system capable of tracking the device of interest.
In this report a robust positioning system is introduced and tested. It consists of four anchors and one tag, and the position of the tag is estimated by trilateration of the distances from anchors to the tag. The final system is using a non-linear least squares algorithm to perform the trilateration.
Different test scenarios are set up and the results are compared. Tracking of a moving tag was showing very precise estimations, but with the same errors for every repetition of the same test. Tests with a non-moving tag showed estimation errors of ± 5 cm.

Contents

1 Introduction

To navigate a device through an environment it is crucial to know its position compared to the surroundings. It does not become less important when dealing with autonomous devices like a driving robot or an autonomous drone. To deal with this problem, a variety of sensors can be placed on-board the devices, to estimate its current position compared to its starting position, by for instance sensing acceleration in different directions. The problem with these techniques is that there will always occur some sort of drift in the measurements, meaning that as time goes on the error in the estimated position will increase. To correct this drifting of measurements, a system that estimates the device position based on distances to other devices, placed in the surroundings, can be used. These estimates can be fused with on-board sensors, to combine the best of both worlds.

Today the most commonly used and well known system of this type, is the Global Positioning System (GPS). It computes distances between satellites with known positions, to a GPS-device on earth and by utilizing geometry, it estimates the position of the GPS-device. This method has some quite unfortunate disadvantages such as low accuracy and extremely poor performance when used indoors. When it comes to navigation of robots, it is often required to have a high accuracy and it should be able to be used indoors.

Alternatives that are actually made for indoor positioning purposes is *local* positioning systems. These types of systems work somewhat like the GPS. Distances from *anchors* with known positions, are measured to a *tag* and a relative position is estimated. The way the distances are measured in these systems are by either by loss in signal strength or delay from transmission time till reception of a signal. The signal strength method is enough to estimate which room a device is in, but not much more than that. The signal delay method is more precise, but it is influenced by noise sources that uses the same RF bands as the transmitted signals. To overcome this issue, devices with Ultra Wide Band (UWB) transceivers has been developed. It is this technology that is explained throughout this report and used to obtain a high precision positioning system.

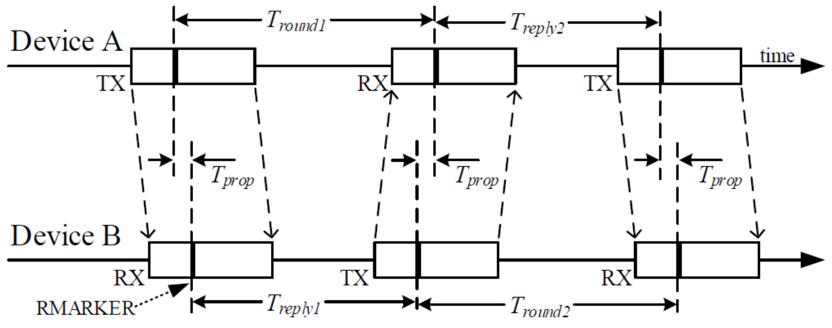
2 System Overview

The system consists of several devices that communicates and performs distance measurements to each other with UWB transceivers. The transceivers used are DW1000, a low power, single chip CMOS radio transceiver IC produced by decaWave. They span 6 RF bands from 3.5 GHz to 6.5 GHz and support data rates of 110 kbps, 850 kbps and 6.8 Mbps[1]. Due to the wide span on the RF bands, the performance of the transceivers are not affected by noise, in the same way a narrow banded system would be. This is due to the interferences often coming from narrow banded systems, like bluetooth or WiFi, thus having lower impact on a wide banded system. The manufacturer states that the operating range of the system may vary from 60 m line-of-sight (LOS) at 6.8 Mbps data rate to 250 m at the 110 kbps data rate and the accuracy is stated to be +/- 10 cm.
For this report a setup was made from four anchor nodes and one tag. It is possible, and recommended, to expand the system so that it consists of more anchors, since this will result in higher precision tracking and be more fault tolerant.

2.1 Distance measurements

In order to do distance estimation, a ranging algorithm based on time of flight is being used. This means that every communication package between nodes are precisely timestamped at time of transmission and again at time of reception, in order to calculate the propagation time. As the speed of light is constant, the distance between the transmitter and the receiver can be calculated as a direct scalar of the propagation time. In theory this alone should give a very accurate distance measurement, but this is not the case in reality. This is due to the fact that there is a lot of different delays in both the transmitting unit and the receiving unit, delays that are not always constant. The delays could depend on things like package size, processor type, electronics layout, transmission data rate and so forth. Furthermore the clocks of the two nodes has to be synchronized to perfection in order to have comparable timestamps, which is not possible as the crystals has some drift.

The solution to some of these issues, is to use what is called a two-way ranging scheme, allowing the system to compensate for some of the drift in the clock synchronization and some of the variable delays in the system. In a basic single-sided two-way ranging scheme you measure the time it takes for a message to propagate from device A to device B and reverse, subtracting the processing time on device B and dividing by two to get the average propagation time. In this case, an extension of the normal single-sided two-way ranging scheme is used called *Double-sided Two-way ranging.* The extension of the basic single-sided two-way ranging, means that instead of measuring the propagation time from one node to the other and back again, the process is repeated in reverse, meaning that the round-trip of device A to device B and back again is measured, and the round-trip time of device B to device A and back again is also measured. This makes it possible to compensate, in case of any difference in the processing time, internal delays etc. between the two devices. The process can be optimized, by using the reply from the first round-trip, as an initiator for the second round-trip, as seen in figure 1. It is however worth noting that even though this ranging scheme allows a good range estimate in just three packages, the last package ends up at the opposite device, of who initiated the



Figur 1: Double-sided Two-way ranging, using the reply of the first round-trip as initiator for the second round-trip.

ranging. This means that a fourth package is required to report the last T_{round2} back to the initiator for distance calculation.

The resultant time-of-flight estimate T_{prop} can now be calculated as

$$\hat{T}_{prop} = \frac{T_{round1}T_{round2} - T_{reply1}T_{reply2}}{T_{round1} + T_{round2} + T_{reply1} + T_{reply2}} \tag{1}$$

And the distance can be calculated as

$$r_i = T_{prop}C \tag{2}$$

Where C is the speed of light, $3 \cdot 10^8$ m/s.

3 Position Estimation

The problem of estimating the position of an object with unknown position, based on distance measurements between the object and multiple reference points, with known positions, are known as trilateration [2]. In theory trilateration is as simple, as solving a system of equations in the form of

$$(\mathbf{p}_i - \mathbf{p}_0)^T(\mathbf{p}_i - \mathbf{p}_0) = r_i^2 \tag{3}$$

where $\mathbf{p}_0$ denotes the position estimate of the object, $\mathbf{p}_i$ the known position of the i'th reference point and r_i the measured distance between the object and the i'th reference point. When structuring the trilateration problem in this way, equation (3) is equivalent to the equation of a sphere, with center point in $\mathbf{p}_i$ and radius of r_i. The solution to a system like this, is essentially equivalent to finding the intersection point of the N spheres contained in the system, this intersection point will be the unknown position, $\mathbf{p}_0$.

This can be solved in a number of ways, ranging in complexity from simple hyperbolic interpolation and least squared methods to Extended Kalman filters and Particle filters. In this report we have chosen to focus on the least squares methods.

3.1 Linear Least Squares

The problem of estimating a position in 3 dimensions, from N distance measurements to known positions, can be solved as a least squares optimization problem. To do this the problem can be simplified into a linear system of N equations with 3 unknown variables in the form $\mathbf{Ax} = \mathbf{b}$, with

$$\mathbf{A} = \begin{bmatrix} \mathbf{P}_{2,x} - \mathbf{P}_{1,x} & \mathbf{P}_{2,y} - \mathbf{P}_{1,y} & \mathbf{P}_{2,z} - \mathbf{P}_{1,z} \\ \mathbf{P}_{3,x} - \mathbf{P}_{1,x} & \mathbf{P}_{3,y} - \mathbf{P}_{1,y} & \mathbf{P}_{3,z} - \mathbf{P}_{1,z} \\ \vdots & \vdots & \vdots \\ \mathbf{P}_{n,x} - \mathbf{P}_{1,x} & \mathbf{P}_{n,y} - \mathbf{P}_{1,y} & \mathbf{P}_{n,z} - \mathbf{P}_{1,z} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{P}_{0,x} - \mathbf{P}_{1,x} \\ \mathbf{P}_{0,y} - \mathbf{P}_{1,y} \\ \mathbf{P}_{0,z} - \mathbf{P}_{1,z} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{n1} \end{bmatrix} \tag{4}$$

where

$$(\mathbf{P}_{0,x} - \mathbf{P}_{1,x})(\mathbf{P}_{n,x} - \mathbf{P}_{1,x}) + (\mathbf{P}_{0,y} - \mathbf{P}_{1,y})(\mathbf{P}_{n,y} - \mathbf{P}_{1,y}) + (\mathbf{P}_{0,z} - \mathbf{P}_{1,z})(\mathbf{P}_{n,z} - \mathbf{P}_{1,z}) = b_{n1} \tag{5}$$
$$\frac{1}{2}[r_1^2 - r_n^2 + d_{n1}^2] = b_{n1}$$

and d_{ij} being the distance between the i'th and the j'th node, given as

$$d_{ij} = \sqrt{(\mathbf{P}_{i,x} - \mathbf{P}_{j,x})^2 + (\mathbf{P}_{i,y} - \mathbf{P}_{j,y})^2 + (\mathbf{P}_{i,z} - \mathbf{P}_{j,z})^2} \tag{6}$$

The full derivation of these equation can be found in Appendix A.
With this given, a new problem arises, as the distance measurements contains noise and is therefore only distance approximations with some uncertainty. This is where the least squares approximation comes into play, as this uncertainty essentially means, that we are searching for a solution $\mathbf{x}$, such that $\mathbf{Ax} \approx \mathbf{b}$. By minimizing the sum of the squares of the residuals,

$$S = \mathbf{r}^T\mathbf{r} = (\mathbf{b} - \mathbf{Ax})^T(\mathbf{b} - \mathbf{Ax}), \tag{7}$$

the following equation can be obtained [3]

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b} \tag{8}$$

From here, there are several possible methods to solve this equation, based on how $\mathbf{A}^T\mathbf{A}$ is conditioned. In this report we will only investigate the case where $\mathbf{A}^T\mathbf{A}$ is non-singular and well-conditioned, leading to the simple solution given in equation (9).

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{9}$$

Now the position estimate of the robot $\mathbf{p}_0$, can be found as

$$\mathbf{p}_0 = \mathbf{x} + \mathbf{p}_1 \tag{10}$$

In the cases where $\mathbf{A}^T\mathbf{A}$ is either singular or poorly conditioned, there is as mentioned before, still ways to solve the linear system of equations. These involves either normalized orthogonal decomposition or singular value decomposition. These cases will however not be investigated in this report, as the results of the best case, where $\mathbf{A}^T\mathbf{A}$ is non-singular and well-conditioned deemed unsatisfactory and neither of the two extended methods will increase the accuracy. The unsatisfactory solution, is due to the nonlinear nature of equation (3) and the errors existing in both the fixed reference positions, $\mathbf{p}_i$ and the measurements, r_i, making it impossible for the linear least squares method to approximate the position with higher accuracy.

3.2 Nonlinear Least Squares

To overcome some of the issues with the linear least squares implementation, an algorithm based on a nonlinear least squares formulation is implemented instead. The *efficient least squares trilateration algorithm* [4]. The Efficient Least Squares Trilateration Algorithm for Mobile Robot Localization, used has the major advantage, that it is built solely on simple algebraic matrix instructions, meaning no matrix inversion is involved in the algorithm. This makes for an extremely robust algorithm, in the sense that it can handle singular matrices e.g. in the case of all anchors being positioned in the same single plane. Furthermore it means that the algorithm is extremely fast to process, which is an important factor, as the processing power might be very limited in real life applications.
Essentially the algorithm proposes a way to obtain the optimal position estimate of the robot, as an minimization problem in the form:

$$\mathbf{p}_{0,opt} = \underset{p_0}{\operatorname{argmin}}\, S(\mathbf{p}_0) \tag{11}$$

Where

$$S(\mathbf{p}_0) = \sum_{i=1}^{N} [(\mathbf{p}_i - \mathbf{p}_0)^T(\mathbf{p}_i - \mathbf{p}_0) - r^2]^2 \tag{12}$$

With $\mathbf{p}_0$ denoting the estimated position of the robot, $\mathbf{p}_i$ the position of the i'th reference point, N the number of reference points used and r_i the measured distance between $\mathbf{p}_0$ and $\mathbf{p}_i$.

Using a least squares algorithm, we relieve ourselves from the constraint that is usually associated with equation systems with noise present, being that as soon as more equations are available than unknown variables, the system generally does not have one exact solution. This means that courtesy of using least squares approximations we are not limited to the case where $N = 3$, but this solution can be used in the general case where $N \geq 3, \quad N \in \mathbb{N}$.

It has been chosen to include only the final equations required to calculate the position estimate in the report, the full derivation of said equations can be found in Appendix B.

The algorithm can be described in 7 steps:

1. Calculate **a**,**B**,**c**,**f**,**f'**,**H**,**H'**,**Q** and **U**, from equations (13) - (20).
2. Calculate $\mathbf{q}^T\mathbf{q}$, from equation (21).
3. Calculate q_3, from equation (24).
4. Calculate q_1 and q_2, from equation (25).
5. Calculate $\mathbf{p}_0$, from equation (26).
6. Choose one of the two candidates of $\mathbf{p}_0$.
7. Return $\mathbf{p}_0$.

$$\mathbf{a} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i\mathbf{p}_i^T\mathbf{p}_i - r_i^2\mathbf{p}_i)\,, \qquad \mathbf{a} \in \mathbf{M}_{3x1}(\mathbb{R}) \tag{13}$$

$$\mathbf{B} = \frac{1}{N}\sum_{i=1}^{N}[-2\mathbf{p}_i\mathbf{p}_i^T - (\mathbf{p}_i^T\mathbf{p}_i)\mathbf{I} + r_i^2\mathbf{I}]\,, \qquad \mathbf{B} \in \mathbf{M}_{3x3}(\mathbb{R}) \tag{14}$$

$$\mathbf{c} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i\,, \qquad \mathbf{c} \in \mathbf{M}_{3x1}(\mathbb{R}) \tag{15}$$

$$\mathbf{f} = \mathbf{a} + \mathbf{B}\mathbf{c} + 2\mathbf{c}\mathbf{c}^T\mathbf{c}\,, \qquad \mathbf{f} \in \mathbf{M}_{3x1}(\mathbb{R}) \tag{16}$$

$$\mathbf{f'} = [f_1 - f_3, \quad f_2 - f_3]^T\,, \qquad \mathbf{f'} \in \mathbf{M}_{2x1}(\mathbb{R}) \tag{17}$$

$$\mathbf{H} = -\frac{2}{N}\sum_{i=1}^{N}\mathbf{p}_i\mathbf{p}_i^T + 2\mathbf{c}\mathbf{c}^T\,, \qquad \mathbf{H} \in \mathbf{M}_{3x3}(\mathbb{R}) \tag{18}$$

$$\mathbf{H'} = [H_1 - H_3, \quad H_2 - H_3]^T\,, \qquad \mathbf{H'} \in \mathbf{M}_{2x3}(\mathbb{R}) \tag{19}$$

The two matrices **Q** and **U**, can be obtained by orthogonal decomposition [5] of **H'**, with the result given by the relation

$$\mathbf{H'} = \mathbf{Q}\mathbf{U} \tag{20}$$

Where **Q** is a 2x2 orthogonal matrix, and **U** is a 2x3 upper diagonal matrix.

$$\mathbf{q}^T\mathbf{q} = \frac{1}{N}\sum_{i=1}^{N}(r_i^2 - \mathbf{p}_i^T\mathbf{p}_i) + \mathbf{c}^T\mathbf{c}\,, \qquad \mathbf{q}^t\mathbf{q} \in \mathbb{R} \tag{21}$$

In the following, u_{kj} references the element in the k'th row and the j'th column of **U**, v_k references the k'th element of the vector $\mathbf{Q}^T\mathbf{f'}$ and q_k references the k'th element of **q**. To make matters a little easier to read, we introduce 2 helping variables here given as

$$g_1 = \left(\frac{u_{12}v_2}{u_{11}u_{22}} - \frac{v_1}{u_{11}}\right) \tag{22}$$

$$g_2 = \left(\frac{u_{12}u_{23}}{u_{11}u_{22}} - \frac{u_{13}}{u_{11}}\right) \tag{23}$$

q_3 can now be found as

$$q_3 = \frac{\pm 1}{g_2^2 + \left(\frac{u_{23}}{u_{22}}\right)^2 + 1}\Bigg[g_1g_2 + \left(\frac{v_2}{u_{22}}\frac{u_{23}}{u_{22}}\right) + \Bigg(- g_1^2\left(\frac{u_{23}}{u_{22}}\right)^2 - g_1^2 + 2g_1g_2\left(\frac{v_2}{u_{22}}\frac{u_{23}}{u_{22}}\right) \\ - g_2^2\left(\frac{v_2}{u_{22}}\right)^2 + \mathbf{q}^T\mathbf{q}g_2^2 - \left(\frac{v_2}{u_{22}}\right)^2 + \mathbf{q}^T\mathbf{q}\left(\frac{u_{23}}{u_{22}}\right)^2 + \mathbf{q}^T\mathbf{q}\Bigg)^{1/2}\Bigg] \tag{24}$$

This will give rise to two solutions, due to the ± 1, with only one of the two candidates resulting in the true $\mathbf{p}_0$. The true **q** can be chosen based on a number of criterion's based on the application, e.g. it could be known that the true $\mathbf{p}_0$ is always on a specific side of the base plane, or that the current $\mathbf{p}_0$ cannot move far from the previous estimate of $\mathbf{p}_0$.

Now that we know the height of the $\mathbf{p}_0$ point, we can substitute into the two equations

$$q_1 = g_1 + g_2 q_3 \qquad\qquad q_2 = \left(-\frac{v_2}{u_{22}} - \frac{u_{23}}{u_{22}}\right) q_3 \tag{25}$$

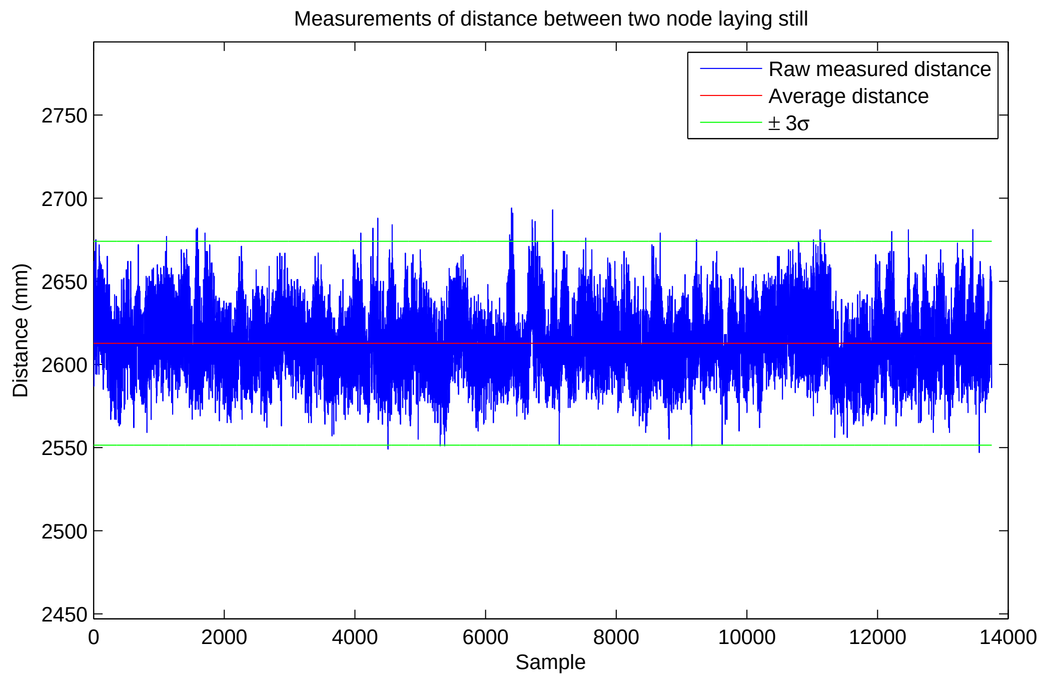
Lastly the estimated position can be obtained by

$$\mathbf{p}_0 = \mathbf{q} + \mathbf{c} \tag{26}$$

4 Results

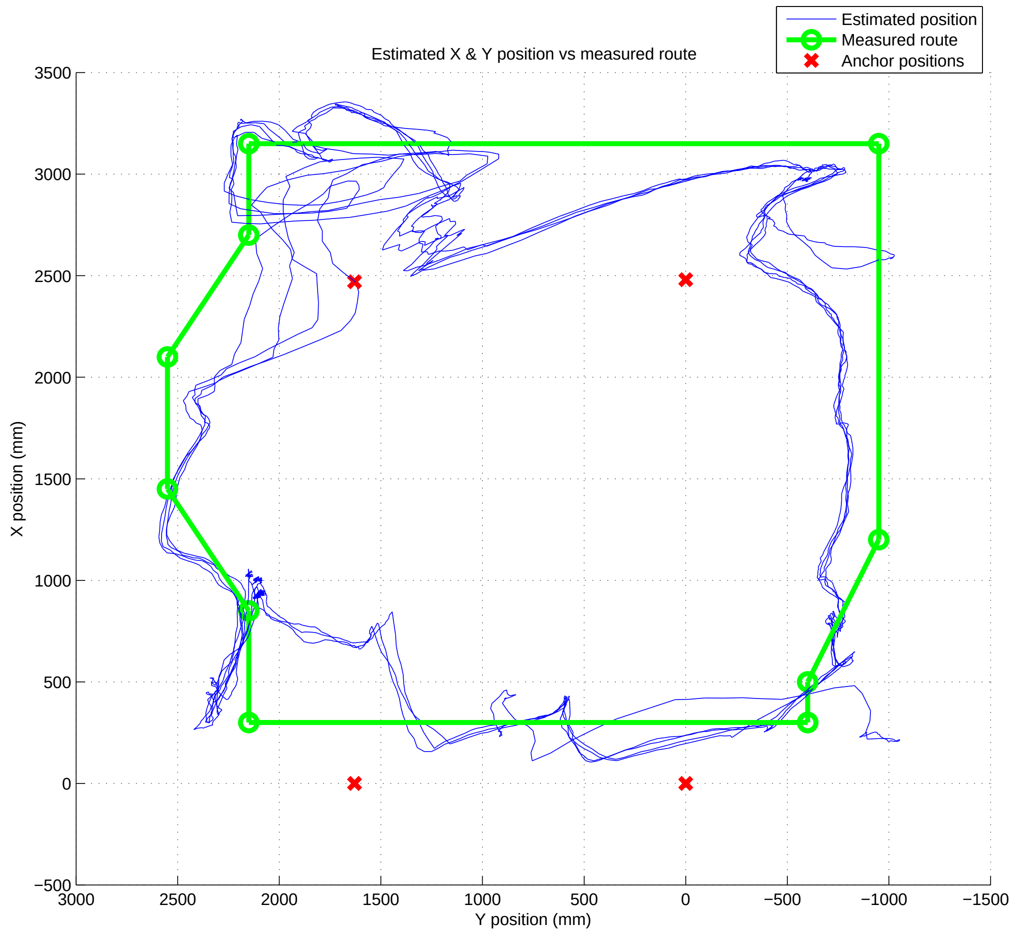
During the project various techniques and methods have been tested, in order to achieve the best possible position estimate. This includes multiple time-of-flight range calculation schemes, different methods for solving the trilateration problem and eventually simple hardware alterations to reduce reflections.

Looking at just a single series of distance measurements, before processing the data will give the best overall assessment of what the system might be capable of, assuming the data is processed right. Such a measurement series can be seen in figure 2, together with the mean value of the series shown and $\pm 3\sigma$ highlighted as a confidentiality interval. As it can be seen from the figure, nearly all of the measurements are within ±5 cm, way better than the specified ±10 cm from the manufacturer. These results are consistent across all of the tests conducted with non-moving nodes. The obtained precision is consistent with moving modes as well, as long as the nodes are in line of sight and does not come to close to the other tag, as they seem to require a minimum distance between each other of 10-15 cm.
With raw distance measurements within ±5 cm of each other, the system should be capable of delivering quite accurate three dimensional position estimates. There is however a couple of other things in play, when calculating position from distance measurements. First of all there is an overdetermined system of equations to be solved, as soon as more than three reference points are introduced and with the addition of measurement noise, these are generally prone to have no real solution. This means that the best position estimate obtainable is the position that makes for the least error, making it an optimization problem. This has been solved by the least squares method, leading to accurate position estimates in the cases where there is line of sight between the tag and all the anchors, there is no motion of any of the nodes and the nodes are oriented and positioned in a way that minimizes the reflection impact on the measurements.



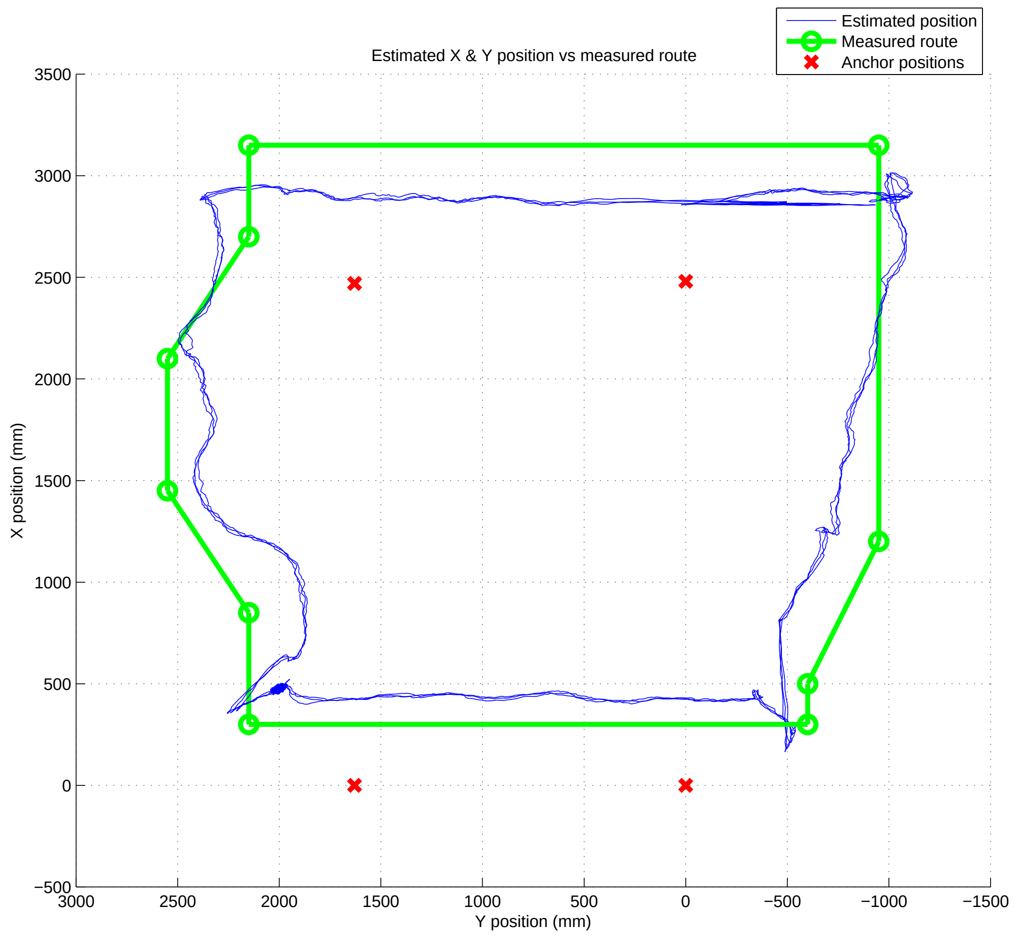
Figur 2: Measurement series of distance between two nodes laying still.

In figure 3, the X and Y position of a small mobile robot, equipped with a tag and driving a preprogrammed route, can be seen before orienting and positioning the nodes to minimize reflection impact. It can be seen how the measurements are somewhat along the expected route, but with a lot of jitter, offset and unexplainable behaviour. As it turns out, this behaviour is explainable though, as it is cause by wrong distance measurements as a consequence of the reflections from walls containing higher power at reception than the direct route. This can be explained by looking at the antenna profile of the modules, as this antenna has so called *dead-spots* where the reception gains are extremely low. This means that in the case where the direct line of sight route hits a dead-spot and the reflections hits a non-dead-spot on the antenna, the reflections will seem like the superior signal to the system.



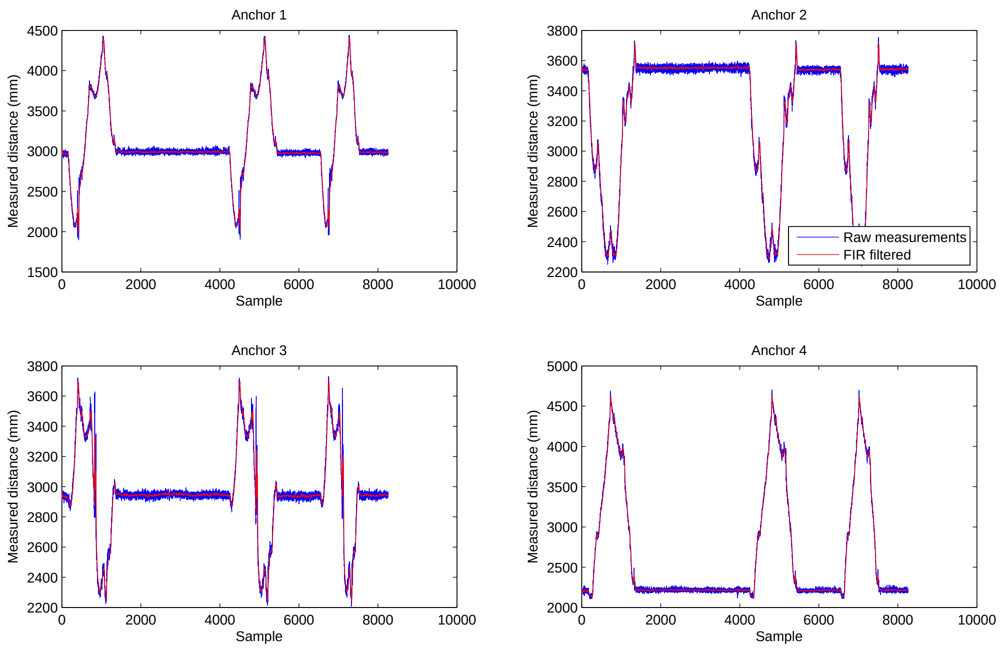
Figur 3: Position estimate of a small mobile robot, driving a preprogrammed route. The data is logged with no hardware alterations and the anchors are placed at a height of 3 meters.

By correcting for these antenna dead-spots as much as possible, it was possible to obtain much better results. In figure 4, the same small mobile robot, driving the same preprogrammed route can be seen, but this time the results are much better. The estimated route is still not the correct route, but there is a much better coherence between the estimated and true route.



Figur 4: Position estimate of a small mobile robot, driving a preprogrammed route. The data is logged after doing hardware alterations to minimize reflections.

By looking at the distance measurement series individually, it can be seen how the system measures the exact same distances throughout the route, every round.



Figur 5: Distance measurements from each reference point to a small mobile robot, driving the preprogrammed route seen in figure 4. On the figure the raw distance measurements can be seen, as well as the signal after an FIR filter of 15 samples. The data is logged after doing hardware alterations to minimize reflections.

5 Conclusion

From the results of the performed tests of the system, it can be concluded that the system is capable of tracking an object with a tag mounted on it. The position estimate of the tag is within ±5 cm when the tag is not moved, which is twice as good as the promised ±10 cm by the manufacturer. When it comes to a moving object which is turning the antenna of the tag while moving around, the precision drops and the position error reaches *approximately* 20 cm. This is based on the calculated position compared to a hand-measured true position, thus not completely accurate.
The tests of the moving tag showed a very high precision during operation (positions were estimated with maximum ±5 cm differences between all rounds driven), but the accuracy was significantly lower when the tag was moving.

6 Future Works

As seen there are plenty of possibilities to improve on the system. This section will try to dive into some of the most obvious places to improve, expand and work with the system in the future.

One thing to improve on the current system, is to add a high sensitivity pressure sensor to each node, both the anchors and the tag. This should be added in order to obtain a relative pressure difference between each known position, the anchors, and the unknown tag position. With these N relative pressure differences it would be possible to calculate a relative height difference between each anchor and the tag, in order to estimate the height of the tag, even before measuring distances. In combination with a modified version of the implemented nonlinear least squares, this might lead to a better height estimation, leading to a better position estimate all together.

Another place to improve the current system, would be to alternate the current hardware of the anchors, in order to obtain a better antenna leading to less impact of reflections. Also the hardware needs to be able to charge the onboard battery through the usb port.

The last obvious place to expand the system, would be to implement the ability for the system to range in a mesh, meaning that all nodes obtains ranges to all other nodes in range. This would allow the system to auto obtain the fixed reference point positions and thus eliminate the need to measure the reference positions by hand.

7 References

[1] DecaWave. *DW1000 User Manual.* URL: `http://www.decawave.com/support/download/file/nojs/692`.

[2] F Thomas og L Ros. "Revisiting trilateration for robot localization". und. I: *Ieee Transactions on Robotics, Ieee Trans. Robot, Ieee T Robo, Ieee T Robot, Ieee Trans Robot, Ieee Trans. Rob* 21.1 (2005), s. 93–101. ISSN: 19410468, 15523098. DOI: `10.1109/tro.2004.833793`.

[3] William S. Murphy Jr. og Willy Hereman. "Determination of a position in three dimensions using trilateration and approximate distances". I: (1999).

[4] Yu Zhou. "An Efficient Least-Squares Trilateration Algorithm for Mobile Robot Localization". und. I: *2009 Ieee-rsj International Conference on Intelligent Robots and Systems* (2009), s. 3474–3479.

[5] G.H. Golub og C.F. van Loan. *Matrix computations.* The Johns Hopkins University Press, 1996, 694 s. ISBN: 0801854148, 9780801854149.

8 Appendix

Appendix A - Linear Least Squares Derivation

In a thee dimensional space, trilateration is equivalent to finding the intersection between N spheres of the form

$$(x-x_i)^2+(y-y_i)^2+(z-z_i)^2=r_i^2 \qquad (i=1,2,...,N) \tag{27}$$

To linearize this equation, x_j, y_j and z_j can be added and subtracted in (27), to give

$$(x-x_j+x_j-x_i)^2+(y-y_j+y_j-y_i)^2+(z-z_j+z_j-z_i)^2=r_i^2 \tag{28}$$

By expanding (28) and regrouping the terms, leads to

$$\begin{aligned}&(x-x_j)(x_i-x_j)+(y-y_j)(y_i-y_j)+(z-z_j)(z_i-z_j)\\&=\frac{1}{2}[(x-x_j)^2+(y-y_j)^2+(z-z_j)^2-r_i^2+(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2]\\&=\frac{1}{2}[r_j^2-r_i^2+d_{ij}^2]=b_{ij}\end{aligned} \tag{29}$$

With d_{ij}, being the distance between the i'th and the j'th node, given as

$$d_{ij}=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2} \tag{30}$$

By selecting the first reference point as the linearizing tool ($j=1$), this leads to a linear system of $(n-1)$ equations in 3 unknowns of the form $\mathbf{Ax}=\mathbf{b}$, given as

$$(x-x_1)(x_2-x_1)+(y-y_1)(y_2-y_1)+(z-z_1)(z_2-z_1)=\frac{1}{2}[r_1^2-r_2^2+d_{21}^2]=b_{21} \tag{31}$$

$$(x-x_1)(x_3-x_1)+(y-y_1)(y_3-y_1)+(z-z_1)(z_3-z_1)=\frac{1}{2}[r_1^2-r_2^2+d_{31}^2]=b_{31} \tag{32}$$

$$\vdots \tag{33}$$

$$(x-x_1)(x_n-x_1)+(y-y_1)(y_n-y_1)+(z-z_1)(z_n-z_1)=\frac{1}{2}[r_1^2-r_2^2+d_{n1}^2]=b_{n1} \tag{34}$$

Where

$$\mathbf{A}=\begin{bmatrix}x_2-x_1 & y_2-y_1 & z_2-z_1\\ x_3-x_1 & y_3-y_1 & z_3-z_1\\ \vdots & \vdots & \vdots\\ x_n-x_1 & y_n-y_1 & z_n-z_1\end{bmatrix}, \quad \mathbf{x}=\begin{bmatrix}x-x_1\\ y-y_1\\ z-z_1\end{bmatrix}, \quad \mathbf{b}=\begin{bmatrix}b_{21}\\ b_{31}\\ \vdots\\ b_{n1}\end{bmatrix} \tag{35}$$

Appendix B - Nonlinear Least Squares Derivation

The problem is essentially the optimization problem given by

$$\mathbf{p}_{0opt}=argmin\cdot S(\mathbf{p}_0 \tag{36}$$

where

$$S(\mathbf{p}_0 = \sum_{i=1}^{N}[(\mathbf{p}_i - \mathbf{p}_0)^T \cdot (\mathbf{p}_i - \mathbf{p}_0) - r_i^2]^2 \tag{37}$$

$\mathbf{p}_0$ denotes the estimate of the tag position , $\mathbf{p}_i$ the pre-mapped reference positions of the i'th anchor, r_i the measured distance between $\mathbf{p}_0$ and $\mathbf{p}_i$ and N is the number of reference points used to determine $\mathbf{p}_0$.
Given $\mathbf{p}_i$ and r_i, solving equation (36) is equivalent to solving

$$\frac{\partial S(\mathbf{p}_0)}{\partial \mathbf{p}_0} = \mathbf{a} + \mathbf{B}\mathbf{p}_0 + [2\mathbf{p}_0\mathbf{p}_0^T + (\mathbf{p}_0^T\mathbf{p}_0)\mathbf{I}]\mathbf{c} - \mathbf{p}_0\mathbf{p}_0^T\mathbf{p}_0 = 0 \tag{38}$$

where

$$\mathbf{a} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i\mathbf{p}_i^T\mathbf{p}_i - r_i^2\mathbf{p}_i) \tag{39}$$

$$\mathbf{B} = \frac{1}{N}\sum_{i=1}^{N}[-2\mathbf{p}_i\mathbf{p}_i^T - (\mathbf{p}_i^T\mathbf{p}_i)\mathbf{I} + r_i^2\mathbf{I}] \tag{40}$$

and

$$\mathbf{c} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i \tag{41}$$

To simplify equation (38), we introduce a linear transform

$$\mathbf{p}_0 = \mathbf{q} + \mathbf{c} \tag{42}$$

and obtain an equation containing no quadratic term of $\mathbf{q}$

$$(\mathbf{a} + \mathbf{B}\mathbf{c} + 2\mathbf{c}\mathbf{c}^T\mathbf{c}) + \{\mathbf{B}\mathbf{q} + [2\mathbf{c}\mathbf{c}^T + (\mathbf{c}^T\mathbf{c})\mathbf{I}]\mathbf{q}\} - \mathbf{q}\mathbf{q}^T\mathbf{q} = 0 \tag{43}$$

Defining

$$\mathbf{f} = \mathbf{a} + \mathbf{B}\mathbf{c} + 2\mathbf{c}\mathbf{c}^T\mathbf{c} \tag{44}$$

$$\mathbf{D} = \mathbf{B} + 2\mathbf{c}\mathbf{c}^T + (\mathbf{c}^T\mathbf{c})\mathbf{I} \tag{45}$$

we rewrite equation (43) as

$$\mathbf{f} + [\mathbf{D}\text{ - }(\mathbf{q}^T\mathbf{q})\ \mathbf{I}]\mathbf{q} = 0 \tag{46}$$

It is noticed that

$$\mathbf{D}\text{ - }(\mathbf{q}^T\mathbf{q})\ \mathbf{I} = -\frac{2}{N}\sum_{i=1}^{N}\mathbf{p}_i\mathbf{p}_i^T + 2\mathbf{c}\mathbf{c}^T \tag{47}$$

which is an $n \times n$ symmetric matrix and does not contain $\mathbf{q}$. By defining

$$\mathbf{H} = \mathbf{D}\text{-}(\mathbf{q}^T\mathbf{q})\mathbf{I} \tag{48}$$

we obtain from equation (46)

$$\mathbf{F} = \mathbf{H}\mathbf{q} = 0 \tag{49}$$

This equation is a linear system of n equations of the unknown n-dimensional vector $\mathbf{q}$. If $\mathbf{H}$ is full rank, $\mathbf{q}$ can be calculated as $\mathbf{q} = \textbf{-}\mathbf{H}^{-1}\mathbf{f}$.
In case $\mathbf{H}$ is not full rank, which would be the case if all anchors were placed in the same height, equation (49) does not represent a system of n independent linear equations and we cannot uniquely determine $\mathbf{q}$ from equation (49).
To make a solution for the cases where $\mathbf{H}$ is not full rank, we first denote the k'th component of $\mathbf{f}$ as f_k and the k'th row of $\mathbf{H}$ as $\mathbf{h}_k^T$. We construct a n-1 dimensional vector

$$\mathbf{f} = [f_{1}\text{-}\ f_n,\ ...,\ f_{n-1}\text{-}\ f_n]^T \tag{50}$$

and a $(n-1) \times n$ matrix

$$\mathbf{H'} = [\mathbf{h}_1 - \mathbf{h}_n,\ ...,\ \mathbf{h}_{n-1} - \mathbf{h}_n]^T \tag{51}$$

We can then obtain from equation (49)

$$\mathbf{f'} + \mathbf{H'q} = 0 \tag{52}$$

Then by using orthogonal decomposition [5], we obtain

$$\mathbf{H'} = \mathbf{QU} \tag{53}$$

where $\mathbf{Q}$ is a $(n-1) \times (n-1)$ orthogonal matrix and $\mathbf{U}$ is a $(n-1) \times n$ upper diagonal matrix. By multiplying both sides of equation 52 by $\mathbf{Q}^T$, we get

$$\mathbf{Q}^T\mathbf{f'} + \mathbf{Uq} = 0 \tag{54}$$

Rewriting equation (54) in its scalar form, we have for the 3D case

$$\begin{cases} v_1 + u_{11}q_1 + u_{12}q_2 + u_{13}q_3 = 0 \\ v_2 + u_{22}q_2 + u_{23}q_3 = 0 \end{cases} \tag{55}$$

where v_k denotes the k'th component of $\mathbf{Q}^T\mathbf{f'}$, u_{kj} the (k, j) entry of $\mathbf{U}$ and q_k the k'th component of $\mathbf{q}$. From equation (55) we obtain

$$\begin{cases} q_1 = (\frac{u_{12}v2}{u_{11}u_{22}} - \frac{v_1}{u_{11}}) + (\frac{u_{12}u_{23}}{u_{11}u22} - \frac{u_{13}}{u_{11}})q_3 \\ q_2 = -\frac{v_2}{u_{22}} - \frac{u_{23}}{u_{22}}q_3 \end{cases} \tag{56}$$

Now we have 3 unknowns but 2 equations, and therefore need one more independent equation to solve for $\mathbf{q}$. One valid constraint is

$$q_1^2 + q_2^2 + q_3^2 = \mathbf{q}^T\mathbf{q} \tag{57}$$

where $\mathbf{q}^T\mathbf{q}$ can be obtained from $\mathbf{D} - \mathbf{q}^T\mathbf{q} = \mathbf{H}$ as

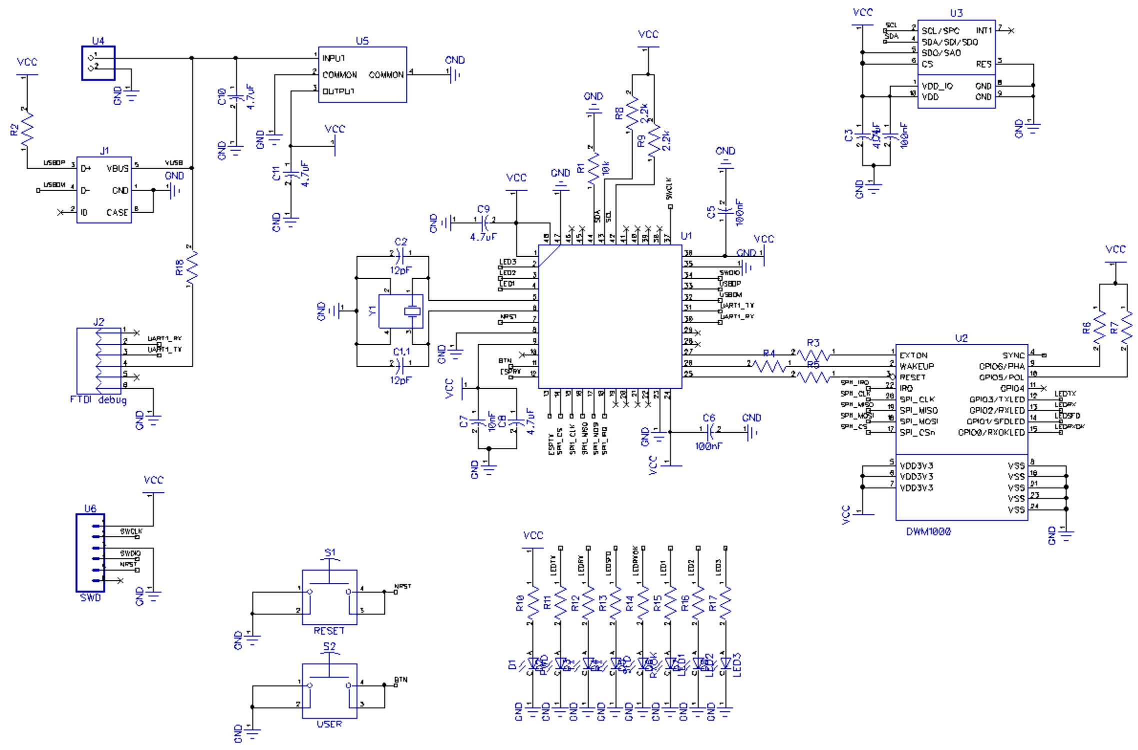
$$\mathbf{q}^T\mathbf{q} = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i^T\mathbf{p}_i + \frac{1}{N}\sum_{i=1}^{N}r_i^2 + \mathbf{c}^T\mathbf{c} \tag{58}$$

substituting equation (56) into (57), we obtain

$$[(\frac{u_{12}v_2}{u_{11}u_{22}} - \frac{v_1}{u_{11}}) + (\frac{u_{12}u_{23}}{u_{11}u_{22}} - \frac{u_{13}}{u_{11}})q_3]^2 + [\frac{v_2}{u_{22}} + \frac{u_{23}}{u_{22}}q_3]^2 + q_3^2 = \mathbf{q}^T\mathbf{q} \tag{59}$$

which is a quadratic equation of q_3 and can be solved in closed form. Substituting the resulting q_3 into equation (56), we can obtain q_1 and q_2.

Appendix C - Hardware Schematic



Figur 6: Schematics of the hardware.

Appendix D - MATLAB Scripts

Appendix D.1 Script

```
clear all
close all


```

```matlab
log.anchors = [0      0     2960      % Anchor 1
               2480   0     2960      % Anchor 2
               2470   1630  2960      % Anchor 3
               0      1630  2960]';   % Anchor 4      % [mm]


log.ref =      [0      0    0       % Point 1
               200    0    0       % Point 2
               900    -350 0       % Point 3
               2850   -350 0
               2850   2750 0
               2400   2750 0
               1800   3150 0
               1150   3150 0
               550    2750 0 ]';

log.ref = [];

log.AVG = 15;


delete(instrfindall);
s = serial('/dev/ttyUSB0');
set(s,'BaudRate', 115200);
set(s,'DataBits', 8);
set(s,'StopBits', 1);
fopen(s);
s.ReadAsyncMode = 'continuous';
readasync(s);
while(s.BytesAvailable <= 0)
    %
end

log.time = [];
log.mean = [];
log.xyz_nl = [];
log.xyz_l = [];
log.elapsed = [];
i = 1;
init = 0;
escape = 1;
while(escape == 1)
    sSerialData = fscanf(s);
    flushinput(s);
    t = strsplit(sSerialData,'\t');
    anc = strsplit(t{1},':');
    if anc{1} == 'N'
        N = str2double(anc(2));
        if size(t,2)-1 == N
            if init == 0
                Fs = 0.01*N;

                % Raw measurements plot
                figure('Name','RAW','units','normalized','outerposition',[0 0.5 0.5 0.5],'KeyPressFcn','escape = 0;')
                hold on
                for n = 1:N
                    h(n) = plot(0);
                    mn(n) = plot(0,'r');
                end
                ylabel('Distance (mm)')
                xlabel('Time (s)')

                % XYZ subplots
                figure('Name','XYZ','units','normalized','outerposition',[0 0 0.5 0.5],'KeyPressFcn','escape = 0;')
                subplot(3,1,1)
                xp = plot(0);
                ylabel('X')
                subplot(3,1,2)
                yp = plot(0);
                ylabel('Y')
                subplot(3,1,3)
                zp = plot(0);
                ylabel('Z')
                xlabel('Time (s)')


                % XY plot
                figure('Name','XY','units','normalized','outerposition',[0.5 0 0.5 1],'KeyPressFcn','escape = 0;')
                hold on
                %xlim([-1500 4000])
                %ylim([-1500 4000])
                ylabel('Y (mm)')
                xlabel('X (mm)')

                %for n = 1:max([size(log.ref,1) N])
                %     if n<=N
                %         plot(log.anchors(1,n),log.anchors(2,n),'rx','LineWidth',2,'MarkerSize',10);
                %
                %     end
                %     if n<=size(log.ref,1)
                %         plot(log.ref(1,n),log.ref(2,n),'gx','LineWidth',2,'MarkerSize',10);
                %     end
                %end
                xy = plot3(0, 0,0);
                daspect([1 1 1]);
                %legend('Anchors','References')

                nl_struct = initiate_nonlinear(log.anchors);
                l_struct = initiate_linear(log.anchors);

                init = 1;
            end

            tstart = tic;
            log.time = [log.time i*Fs];

            for n = 1:N
                log.raw(i,n) = str2double(t(n+1));
                set(h(n),'XData',log.time(1:i),'YData',log.raw(:,n));
            end
            ax = get(h(1),'Parent');
            set(ax,'XLim', [(i-400)*Fs (i+100)*Fs]);


            if(i > log.AVG+1)
                mean_val = sum(log.raw(i-(log.AVG-1):i,:))/log.AVG;
                log.mean = [log.mean; mean_val];
                for n = 1:N
                    set(mn(n),'XData',log.time(1:i),'YData',log.mean(:,n)');
                end

                P0 = calculate_nonlinear(nl_struct, log.anchors, mean_val);
                P0_l = calculate_linear(l_struct, log.anchors, mean_val);


                log.xyz_nl = [log.xyz_nl P0];
```

```
                    log.xyz_l = [log.xyz_l P0_l];

                    set(xy,'XData',log.xyz_nl(1,:),'YData',log.xyz_nl
                        (2,:),'ZData',log.xyz_nl(3,:));

                    set(xp,'XData',log.time(log.AVG+2:i),'YData',log.
                        xyz_nl(1,:));
                    set(yp,'XData',log.time(log.AVG+2:i),'YData',log.
                        xyz_nl(2,:));
                    set(zp,'XData',log.time(log.AVG+2:i),'YData',log.
                        xyz_nl(3,:));

                    drawnow
                else
                    log.mean = [log.mean; log.raw(i,:)];
                end

                i = i + 1;
                log.elapsed = [log.elapsed toc(tstart)];
            end
        end
end
save('log.mat','log');
fclose(s);
delete(instrfindall);
```

Appendix D.2 Initiate Linear

```
function [l_struct] = initiate_linear(anchor_pos)
    N = size(anchor_pos,2);
    A = zeros(N-1,3);
    for n = 1:(N-1)
        A(n,:) = anchor_pos(:,n+1)-anchor_pos(:,1);
    end
    l_struct.A = A;
    l_struct.M = (transpose(A)*A)^-1*transpose(A);
    l_struct.dij = A(:,1).^2 + A(:,2).^2 + A(:,3).^2;
end
```

Appendix D.3 Calculate Linear

```
function [xyz] = calculate_linear(l_struct, anchor_pos,
    measurements)
    N = size(anchor_pos,1);
    for n = 1:N-1
        B(n) = .5 * (measurements(1)^2 - measurements(n+1)^2 +
            l_struct.dij(n));
    end
    xyz = l_struct.M*transpose(B) + anchor_pos(:,1);
end
```

Appendix D.4 Initiate Nonlinear

```
function [nl_struct] = initiate_nonlinear(anchor_pos)
    N = size(anchor_pos,1);
    nl_struct.c = 0;
    nl_struct.H = 0;
    for n = 1:N
        nl_struct.c = nl_struct.c + anchor_pos(:,n);
        nl_struct.H = nl_struct.H + (anchor_pos(:,n)*transpose(
           anchor_pos(:,n)));
    end
    nl_struct.c = nl_struct.c/N;
    nl_struct.H = -2/N * nl_struct.H + 2*nl_struct.c*transpose(
       nl_struct.c);

    for n = 1:2
        nl_struct.H_m(:,n) = nl_struct.H(:,n)-nl_struct.H(:,3);
    end
    nl_struct.H_m = transpose(nl_struct.H_m);
    [nl_struct.Q, nl_struct.U] = qr(nl_struct.H_m);

end
```

Appendix D.5 Calculate Nonlinear

```
function [xyz] = calculate_nonlinear(nl_struct, anchor_pos,
   measurements)
    a = 0;
    B = 0;
    N = size(anchor_pos,2);
    for n = 1:N
        a = a + (anchor_pos(:,n)*transpose(anchor_pos(:,n))*
           anchor_pos(:,n) - measurements(n)^2*anchor_pos(:,n));
        B = B + (-2*anchor_pos(:,n)*transpose(anchor_pos(:,n)) - (
           transpose(anchor_pos(:,n))*anchor_pos(:,n))*eye(3) +
           measurements(n)^2*eye(3));
    end
    a = a/N;
    B = B/N;

    f = a + B*nl_struct.c + 2*nl_struct.c*transpose(nl_struct.c)*
       nl_struct.c;

    for n = 1:2
        f_m(n) = f(n)-f(3);
    end
    f_m = transpose(f_m);


    v = transpose(nl_struct.Q)*f_m;

    qtq = 0;
    for n = 1:N
        qtq = qtq + (measurements(n)^2 - transpose(anchor_pos(:,n))*anchor_pos(:,n));
    end
    qtq = qtq/N + transpose(nl_struct.c)*nl_struct.c;


    g1 = (nl_struct.U(1,2)*v(2))/(nl_struct.U(1,1)*nl_struct.U(2,2)) - v(1)/nl_struct.U(1,1);

    g2 = (nl_struct.U(1,2)*nl_struct.U(2,3))/(nl_struct.U(1,1)*nl_struct.U(2,2)) - nl_struct.U(1,3)/nl_struct.U(1,1);

    g31 = v(2)/nl_struct.U(2,2);

    g32 = nl_struct.U(2,3)/nl_struct.U(2,2);

    q(3,1) = -(g1*g2 + g31*g32 + (- g1^2*g32^2 - g1^2 + 2*g1*g2*g31*g32 - g2^2*g31^2 + qtq*g2^2 - g31^2 + qtq*g32^2 + qtq)^(1/2))/(g2^2 + g32^2 + 1);
    q(3,2) = -q(3,1);

    q(1,1) = g1+g2*q(3,1);
    q(1,2) = g1+g2*q(3,2);

    q(2,1) = -g31-g32*q(3,1);
    q(2,2) = -g31-g32*q(3,2);

    qo = zeros(2,3);

    qo(1,:) = q(:,1) + nl_struct.c;
    qo(2,:) = q(:,2) + nl_struct.c;

    xyz = qo(1,:);
end
```