

Systematic Characterization and Calibration of a Three Dimensional
Ultrasonic Positioning System for Use in Mobile Robots

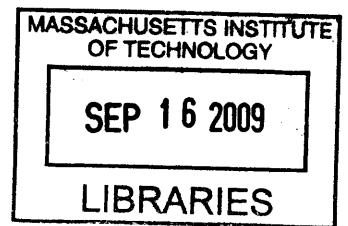
By

Yi Fei Wu

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

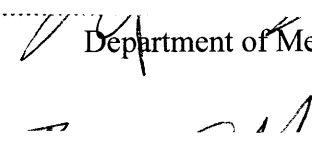
JUNE 2009



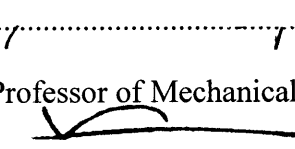
© 2009 Yi Fei Wu. All rights reserved

The author hereby grants to MIT permission to reproduce and to distribute publicly paper
and electronic copies of this thesis document in whole or in part in any medium now
known or hereafter created.

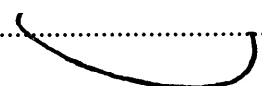
Signature of Author:.....


Department of Mechanical Engineering
May 8, 2009

Certified by:.....


Franz S. Hover
Assistant Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by:.....


J. Lienhard V
Collins Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

ARCHIVES

Systematic Characterization and Calibration of a Three Dimensional Ultrasonic Positioning System for Use in Mobile Robots

By

Yi Fei Wu

Submitted to the Department of Mechanical Engineering
on May 8, 2009 in partial fulfillment of the
requirements for the Degree of
Bachelor of Science in Mechanical Engineering

ABSTRACT

This thesis examined the characteristics of the Hexamite Hx11 ultrasonic sensor for the purpose of three dimensional positioning for mobile robots with multi-plane motions. The sensor system was studied and an experiment was conducted to locate a still transponder in the air using three transmitters on the ground. Due to the angular range of the transponder and the transmitters, data were not logged all of the time. An approximate lower z limit boundary surface was calculated, which predicted the behavior of 70% of the tests in the experiment. Sporadic errors were present in some of the data, with no apparent relation to the geometry of the experiment. If the erroneous parts were to be excluded, the accuracy and precision of the data obtained seemed to be quite high. It is concluded that further researches still needed to be done to determine the sources of the sporadic errors that appeared in some the data, and as of date, the Hexamite Hx11 ultrasonic positioning system cannot be reliably used for three dimensional sensing.

Thesis Supervisor: Franz S. Hover

Title: Assistant Professor of Mechanical and Ocean Engineering

Acknowledgements

To my friends and family for their support and encouragement.

Professor Hover for all of his help and guidance during 2.017, the class that started this project, and thesis preparation.

Matt Greytak for generous technical support and insights.

And Stephen Carl Licht, Josh Leighton, and Hector Vargas for helping me get this project off the ground during 2.017.

Table of Contents

Acknowledgements.....	3
Table of Contents.....	4
1. Introduction.....	5
2. Background.....	6
2.1 Ultrasonic Sensors.....	6
2.2 Hexamite Hx11 Ultrasonic Positioning System.....	8
2.3 Device ID.....	9
2.4 Serial Communication.....	9
2.5 Parameter Configuration.....	10
2.6 Sensor Operation.....	12
2.7 Output Data.....	14
3. Data Processing.....	14
3.1 Data Conversion.....	14
3.2 Calibration.....	15
3.3 The Code.....	16
3.4 Filter Data.....	17
3.5 Triangulation.....	18
4. Experimental Procedure.....	20
5. Results.....	21
6. Conclusion.....	31
7. References.....	31
A. Sensor Command and Data Processing Code.....	33
B. Lower Z Limit Boundary Surface Generation Code.....	37

1. Introduction

One fundamental problem in mobile robot applications deals with accurate vehicle positioning. This is a complicated problem because the environment, the sensors, and the robot are not perfect. The measured data may drift with time; the internal mechanisms of the robot may stall or slip [1]. Despite the wide range of systems commercially available, from odometer and inertial measurement unit for relative positioning to compass and GPS for absolute positioning, an elegant solution has yet to be found. Some of the positioning limitations can be removed if the robot's position is referenced to known landmark locations [2]. Ultrasonic sensors, due to high accuracy, low processing requirement, and relatively low cost, are often used for this application [3].

Most researches on using ultrasonic sensor systems for mobile robot positioning have concentrated on two dimensional sensing of the environment [4]. For robots that only move in one plane, a 2D view of the world may provide enough information. However, when the vehicle needs more degrees of freedom, such as in rescue and inspection robotics [5], it is useful to investigate three dimensional sensing.

This thesis examined the characteristics of the Hexamite Hx11 ultrasonic positioning system for the purpose of three dimensional sensing. An experiment was conducted to locate a still transponder in the air using three transmitters on the ground, as shown in Figure 1. By gathering the transponder position data at different locations and examining the data response with respect to time and geometry, the three dimensional sensing capabilities of the Hx11 system was determined.

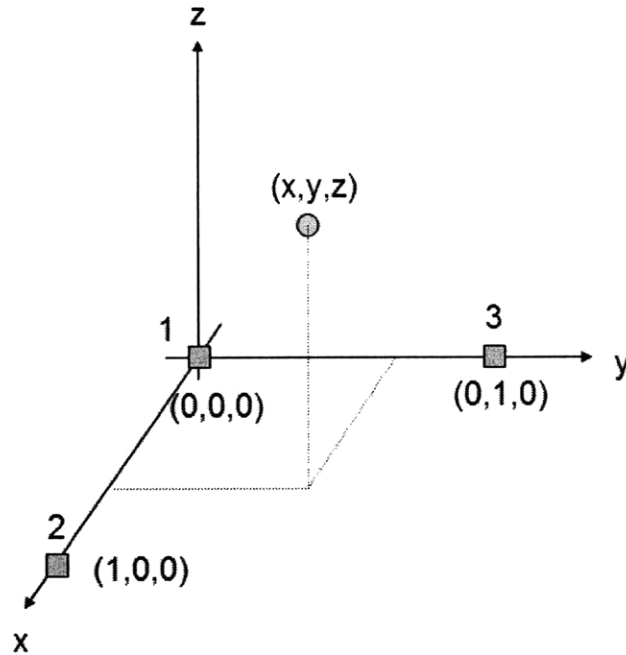


Figure 1: experiment set up with 3 transmitters and one transponder.

2. Background

2.1 Ultrasonic Sensors

The ultrasonic sensors utilize sound wave flight time to calculate distance. The sensor first converts electrical energy into high frequency sound wave, usually above 20,000 Hz, using piezoelectric crystals [6]. It then records the time interval between when the signal is generated and when the echo of the signal comes back. Since the speed of the ultrasonic signal is known, the distance traveled by the signal during the flight time can be calculated.

Ultrasonic sensors consist of two components: transmitter and receiver. The sensors that have both sending and receiving capabilities are called transducers. Sometimes, a sensor can receive a signal and then send a reply back to the original transmitter to enable a more accurate measurement of flight time. This type of sensors is called transponder, or receiver-transmitter [7].

Ultrasonic sensors are nondestructive and completely confined to the area in which it operates [3]. Because it uses sound rather than light, the sensor can work both indoors and outdoors with good accuracy. However, ultrasound can be absorbed and reflected. Environments constructed of strange geometry or absorbent materials may thus be more prone to erroneous measurements [8]. Ultrasound can also be blocked by objects. If the object is small, the ultrasound wave may go around it, leaving minimum damage to the signal. But if the object is large or has reflective surfaces, the signal may be damaged even if the object is not directly blocking the wave [9].

Furthermore, the ultrasonic sensors are only operational within the cone of signal propagation [3]. The probability of losing signals, therefore, increases as the operating angles and distance to the receiver become large. The angles that affect the operation of the ultrasonic sensors include the rotational angle of a transmitter, the shear angle between a receiver and a transmitter, and the propagation angle of the signal. Figures 2, 3, and 4 explain how the three angles are defined in a physical setup.

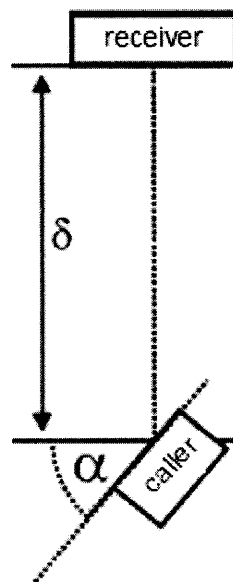


Figure 2: rotational angle of a transmitter [9].

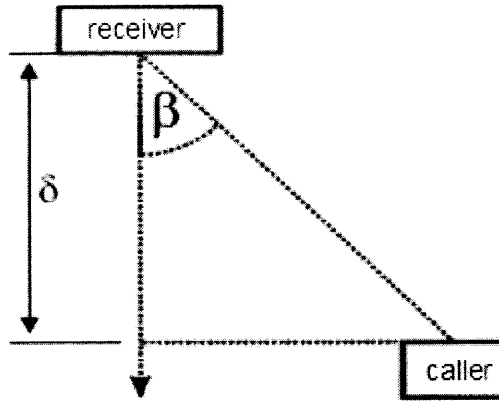


Figure 3: shear angle between a receiver and a transmitter [9].

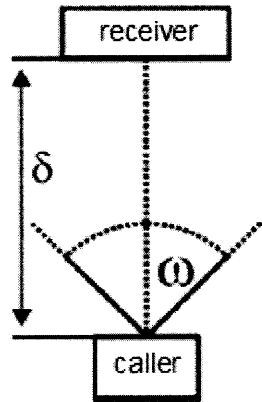


Figure 4: propagation angle of a signal [9].

2.2 Hexamite Hx11 Ultrasonic Positioning System

The Hexamite Hx11 is a high speed ultrasonic positioning system. This transducer module (Figure 5) is 80mm by 40mm by 20 mm and weighs about 40 g each. It requires a power supply of 7-16 VDC and has a current consumption of 40mA. The maximum operational range of the device is 8m, with a maximum rotational angle of +/-50 degrees, shear angle of +/- 40 degrees, and propagation angle of +/- 45 degrees. The Hx11 can be configured as a transmitter, a receiver, or a transponder [10]



Figure 5: Hexamite Hx11 ultrasonic positioning device module [10].

2.3 Device ID

As seen in Figure 5, each Hx11 module is pre-labeled with specific IDs. Take the ID in Figure 5 for example:

11526/283/83.

The ID is separated into three groups. The primary address or receiver ID is the first decimal number 11526. It is used for configuring device parameters. The second number, 283, is the hexadecimal transmitter ID, but it is usually not shown in logged data. Instead, a user defined device ID, the Caller ID, is used to show which transmitter the signal originates from. The third number, 83, is the transponder ID, and it is embedded in the data, along with the Caller ID, to distinguish among different transponders in a system [10].

2.4 Serial Communication

The required serial port configuration for the device to communicate with the computer is 19200 baud rate RS-232 with 8 data bits, 1 stop bit, no parity, and no handshaking [10]. Each of the Hx11 sensors can be connect to a RJ11/RJ12 plug and telephone cables. Figure 6a shows a RJ12 plug with 6 wire cable. The correct cable position, when looking from the side without the tab, is the leftmost pin (pin 1) attached to the white wire and the rightmost pin (pin 6) connected to the blue wire. After

connecting the device to a RJ12 plug, a DB9 plug is needed to connect to a computer's serial port. A DB9 female plug pin diagram is shown in Figure 6b. Table 1 explains the wiring configuration between a 6 wire phone cable in the RJ12 plug and a DB9 female plug to enable serial communication.

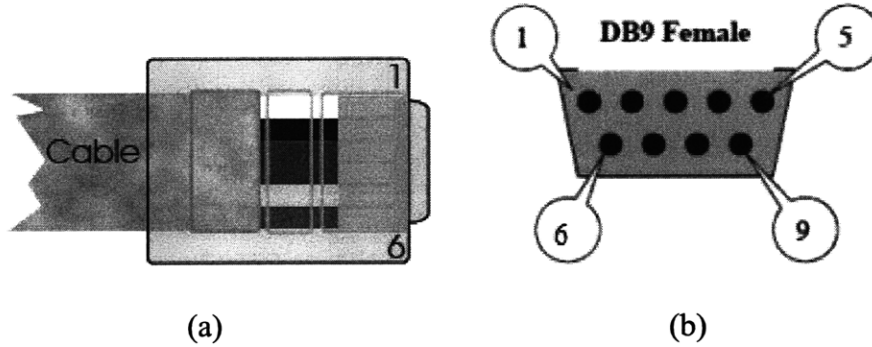


Figure 6: (a) RJ12 plug configuration; (b) DB9 female plug pin diagram [10].

RJ 12 Pin	DB9 Pin
1 (White) not used	Not connected
2 (Black, Ground)	5 (Ground)
3 (Red, +RS485) not used	Not connected
4 (Green, -RS485)	3 (TX) and 2 (RX)
5 (Yellow, Power 8-16VDC)	Not connected
6 (Blue) not used	Not connected

Table 1: corresponding RJ12 and DB9 connections for serial communication [10].

2.5 Parameter Configuration

There are several parameters that govern how the Hx11 device operates. The config.exe program is used to configure those parameters. Clicking “Find Device” enables the software’s automatic search for Hx11 modules that are connected to the computer via serial link. After about 20s, the primary IDs of all of the devices appear in the window, as shown in Figure 7. After clicking “Stop” to terminate the search, “Read EEPROM” can be used find out the current parameters on each of the devices. User can then modify the parameters and click “Write EEPROM” to transmit the modified parameters back to the devices [11]. Table 2 exhibits the parameters used for the experiment.

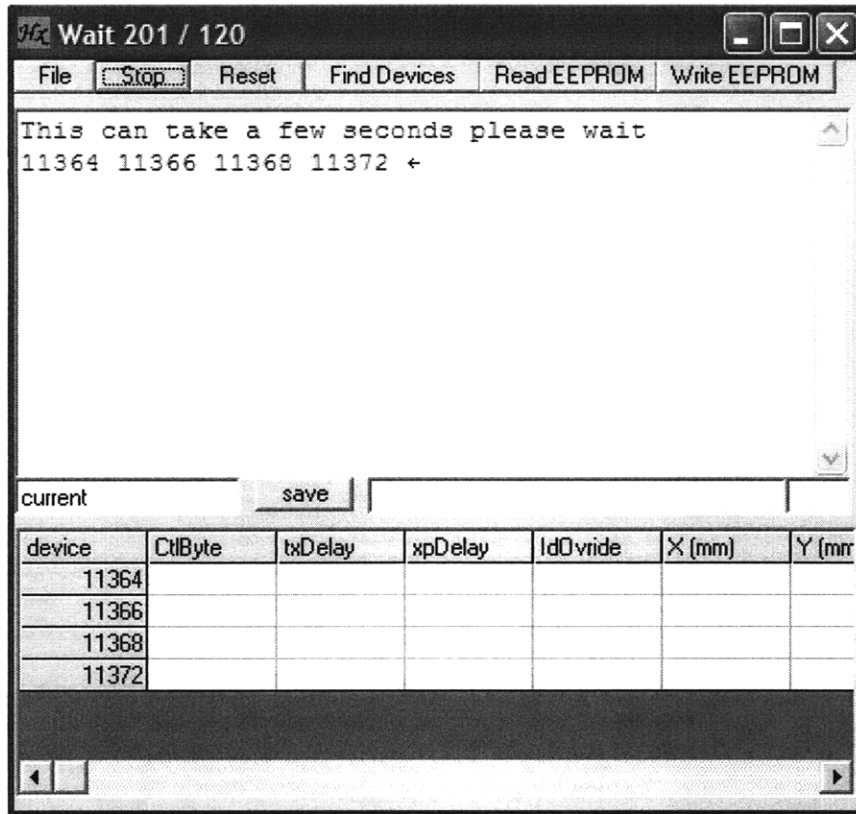


Figure 7: screen shot of config.exe, which is used to change device parameters.

Device	CtlByte	txDelay	xpDelay	IdOverride
11364 (32)	0	255	1	33
11366	3	220	1	13
11368	3	200	1	14
11372	3	240	1	12

Table 2: device parameters set up for the experiment.

The parameters are defined as the following:

CtlByte sets up the device function. In Table 2, devices 11366, 11368, and 11372 are configured as transmitters, while device 11364 is set up as a receiver.

IdOverride has two functions. If this value is less than 16, it becomes the Caller ID of the transmitter, which gets displayed in hexadecimal format in the logged data. If this value is 33, it enables transponding capabilities of a receiver. Thus in Table 2, device

11364 is set up as a transponder, instead of an ordinary receiver, and 11366, 11368, and 11372 are given Caller IDs C, D, and E respectively.

TxDelay sets the sample rate for the transmitter. It is in increments of 4.096ms, so a maximum of 255 increments translate to a sample rate of 1.044s, or a minimum update rate of 0.96Hz [11]. For the experiment, the three transmitters are set with different sampling rates, 0.901s, 0.819s, and 0.737s, to prevent the signal from different transmitters arriving at the transponder at the same time. The TxDelay has no meaning for the receivers or transponders.

XpDelay, also known as the transponder delay, is the delay between the instant the transponder receives a call and the instant the transponder responds to the call. It is used to prevent replies from different transponders arriving at the same transmitter at the same time. This is not an issue here since the experimental set up consists of only one transponder. The XpDelay is also calculated in units of 4.096ms [11], and it has no meaning for the transmitters.

2.6 Sensor Operation

This section explains how the Hx11 system operates. Assume that there is a set up of three devices. Device 11364 transmits a signal continuously at a preset rate of 100ms (txDelay), and devices 11366 and 11370 transpond to the call. Device 11366 is located 3m away from 11364, while device 11370 is placed 4m away from 11364. Figure 8 shows the events occurred in each of the devices during one transmitter signal.

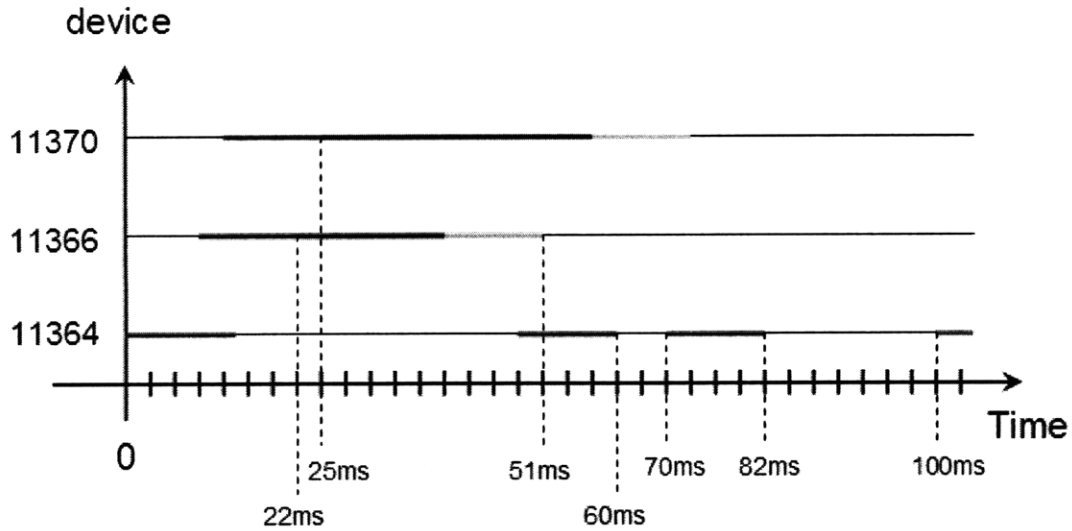


Figure 8: diagram showing the events occurred in ultrasonic devices during one transmitted signal. Red line is the time it takes to transmit a signal. Green line is the time it takes to receive a signal. Blue line is the pre-set transponder delay. Yellow line is the time it takes to encode the signal. Grey line is time it takes to send data over serial link.

At time zero, device 11364 transmits a call (red line). It takes some time for the signal to travel 3m to reach transponder 11366. The signal has to then travel one extra meter to get to transponder 11370. At 22ms, device 11366 finishes receiving the signal (green line) and logs the signal travel time into its ring buffers. At the same time, it starts the device transpond delay (blue line), which is defined by the user (xpDelay). At 25ms, device 11370 also logs the signal arrival time, and starts transpond delay. Since the transpond delays for 11366 and 11370 are set differently, there is a time differential between the two devices, in addition to the one extra meter of travel. At 51ms, device 11366 finishes encoding (yellow line) and sends a reply back to device 11364. At 60ms, device 11364 receives the replay and logs its time of arrival. By subtracting the reply time and the signal arrival time, the signal's time of flight between device 11364 and device 11366 can be calculated. At 70ms, device 11370 finishes encoding and sends back its replay. At 82ms, device 11364 receives the reply from 11370 and the same flight time calculation is done for 11370. At roughly 100ms, device 11364 transmits its ring buffer

content with both 11366 and 11370 flight time data through the serial port, clears the timer, and transmits a second call [9].

With the parameter configuration of the experiment, as described in Table 2, one transponder is set up with three transmitters. The sampling rates are set differently, so each transmitter sends data through the serial port at different time intervals. The transponder stores data from all three transmitters and replies to the signals in the order of arrival. Once a transmitter receives a reply, it sends the data over the serial link at its predetermined pace.

2.7 Output Data

To receive data from the Hx11 sensors, a “&” symbol needs to be first sent over the serial link to the system of devices [11]. The transponder then returns the data followed by a “#” sign and a carriage return at the pre-set sampling rate. If the ring buffer is empty, the response is “#” only. Below shows an example line of data:

034F05D339.

The first three characters are transponder ID, which means that the example data is replied by transponder 34. The fourth number is the transmitter’s caller ID in hexadecimal form. Here F means the signal comes from a transmitter with a pre-set Caller ID of F, or 15 in decimal number. The rest is the time data in hexadecimal form, or in this case 381753 in decimal.

3. Data Processing

3.1 Data Conversion

The raw data received from the ultrasonic sensors cannot be used directly. The various constants used to convert the logged data into distance data include:

$$K = 115464$$

$$M = 344 \text{ (speed of sound, m/s)}$$

$$\text{transpondDelay} = 262144.$$

The time of flight is calculated using Equation 1 and the distance can be found subsequently using Equation 2.

$$T = data - xpDelay \cdot transpondDelay - K \quad (\text{Eq. 1})$$

$$D = \frac{M \cdot T}{32000} \quad (\text{Eq. 2})$$

Therefore for the example data line shown in the section above, assuming that xpDelay is set to 1, the time of flight becomes

$$T = 381753 - 1 \cdot 262144 - 15464 = 4145,$$

and the distance becomes

$$D = 344 \cdot 4145 / 32000 = 44.6 \text{mm}.$$

3.2 Calibration

It is noticed that the measured data, calculated using Equation 1 and 2, does not match the actual distance. Tests were therefore conducted on one transponder and one transmitter, placed between 50mm to 600mm apart, to determine a new calibration value. Figure 9 shows that the measured data is linearly related to the actual distance. So with calculated offset and scale factor, the measured data can be calibrated to the correct size.

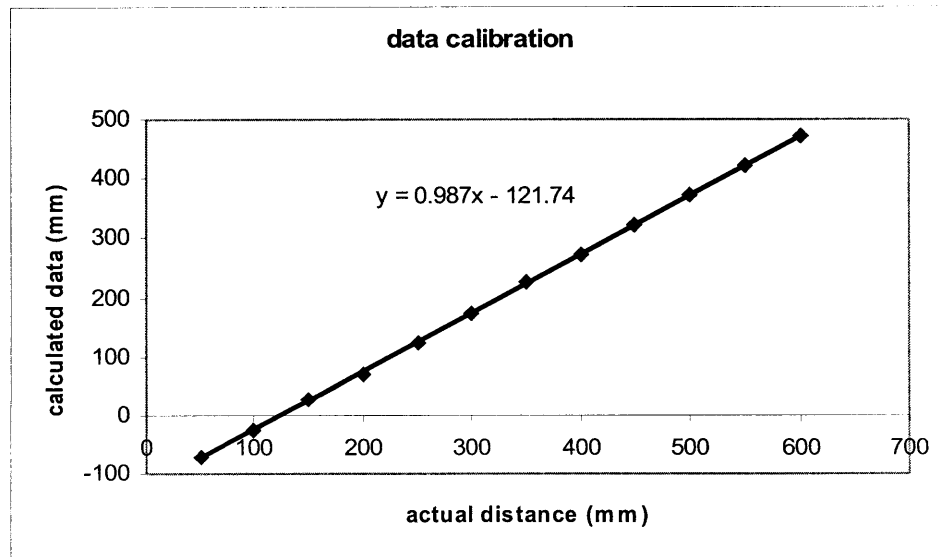


Figure 9: calculate data vs actual distance.

The new calibration constant K is calculated using Equation 3 for each of the measured distances. The average K for all of the data measured yields a new K of 103738. Figure 10 shows an almost one to one ratio between the actual distance and the

calibrated measured data. The newly calibrated data show an average error of 0.94%, and a maximum error of 9.13%, when compared with the actual distance.

$$K_{new} = data - xpDelay \cdot transpondDelay - \frac{D + 121.74}{0.987} \frac{32000}{M} \quad (\text{Eq. 3})$$

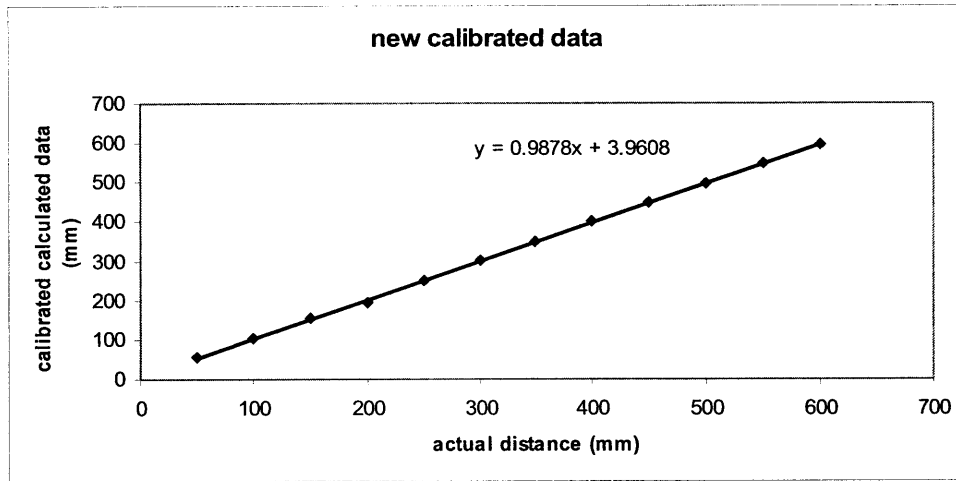


Figure 10: calculated data vs actual distance using the new calibration constant.

3.3 The Code

C code is used to operate the Hx11 ultrasonic positioning system, filter data, convert data into useful distances, triangulate transponder position, and record data. Figure 11 is the flow chart of the code used for the experiment described in Figure 1 and Table 2.

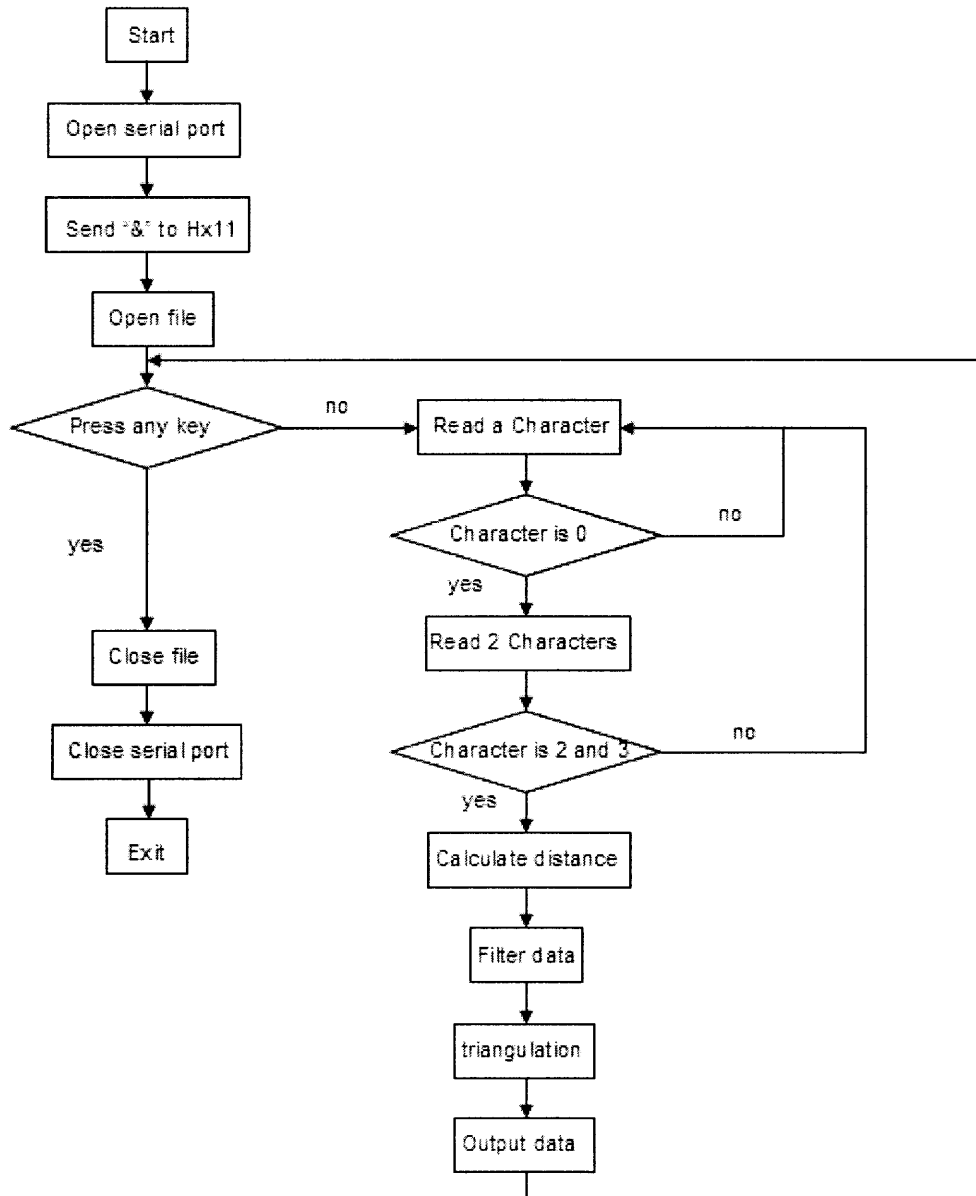


Figure 11: flow chart for ultrasonic sensor system operation code.

3.4 Filter Data

Noises from both inside of the device and the environment often accompany the raw data. Filters need to be used to increase the accuracy and precision of the measurements. An outlier filter is used to first keep the data within the appropriate range. Due to the particular set up of the experiment, the detected distances between a

transmitter and the transponder should be positive and less than 3m. Therefore, if a calculated distance data is bigger than 3000mm or less than 0mm, this data is ignored.

A second smooth filter is used to get rid of small noises in the data. A ring buffer of 5 spaces is created. The data first fills in the buffer until no empty spaces are available. Then the new data is compared with the average of the data already in the buffer. Since both the transponder and the transmitters are staying still, the average is stored in the buffer, taking the place of the oldest data, if the difference between the new data and the average is greater 5mm. Otherwise the new data is stored in the buffer instead.

These two filters decreased the standard deviation of the data received in a one transponder and one transmitter set up from 40mm to 4mm. Since the experiment consists of 3 transmitters and 1 transponder, the errors due to angular constraints are larger than that of the one to one set up. Hence, a larger standard deviation is expected.

3.5 Triangulation

The Hx11 system outputs only the distance between the transponder and the transmitter, thus triangulation is needed to find the Cartesian coordinates of the transponder from those distances. Figure 12 is the experimental configuration of Figure 1, where L is 1m, viewed in the xz -plane.

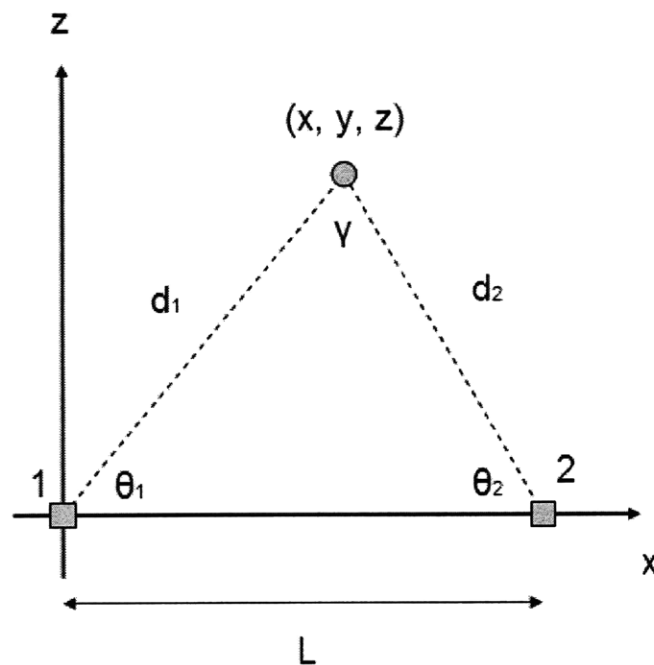


Figure 12: xz -plane view of the experimental set up showed in Figure 1.

Assuming that the angles to the x-axis, θ_1 and θ_2 , remain constant regardless of the transponder rotation around the x-axis, then based on trigonometry, the two angles can be defined using the Equations 4, 5, 6, and 7. Moreover, the angle γ is always the difference between π and the sum of the two angles, as shown in Equation 8.

$$\cos\theta_1 = \frac{x}{d_1} \quad (\text{Eq. 4})$$

$$\cos\theta_2 = \frac{L-x}{d_2} \quad (\text{Eq. 5})$$

$$\sin\theta_1 = \frac{\sqrt{d_1^2 - x^2}}{d_1} \quad (\text{Eq. 6})$$

$$\sin\theta_2 = \frac{\sqrt{d_2^2 - (L-x)^2}}{d_2} \quad (\text{Eq. 7})$$

$$\gamma = \pi - \theta_1 - \theta_2 \quad (\text{Eq. 8})$$

According to the Law of Sine,

$$\frac{\sin\theta_1}{d_2} = \frac{\sin\theta_2}{d_1} = \frac{\sin\gamma}{L}. \quad (\text{Eq. 9})$$

Then, Equations 6 and 7 can also be written in terms of $\sin\gamma$, or $\sin(\pi-\theta_1-\theta_2)$, or $\sin\theta_1\cos\theta_2+\cos\theta_1\sin\theta_2$. When combining Equations 6, 7, and 9, the relationship between the angles becomes

$$\frac{\sin\theta_1}{\sin\theta_1\cos\theta_2+\cos\theta_1\sin\theta_2} = \frac{d_2}{L} \quad (\text{Eq. 10})$$

$$\frac{\sin\theta_2}{\sin\theta_1\cos\theta_2+\cos\theta_1\sin\theta_2} = \frac{d_1}{L}. \quad (\text{Eq. 11})$$

By combining Equations 10 and 11 and plugging in Equations 4, 5, 6, and 7 for the sine and cosine values, the triangulated equation for the x coordinate is found.

$$x = \frac{d_1^2 - d_2^2 + L^2}{2L} \quad (\text{Eq. 12})$$

The y coordinate can be calculated using similar principles, as shown in Equation 13.

$$y = \frac{d_1^2 - d_3^2 + L^2}{2L} \quad (\text{Eq. 13})$$

To find the z coordinate, assume that the transponder is on the surface of a sphere that is centered at the origin with a radius of d_1 . The equation of the sphere is

$$x^2 + y^2 + z^2 = d_1^2 \quad (\text{Eq. 14})$$

which can be rearranged to get the z coordinate

$$z = \sqrt{d_1^2 - x^2 - y^2} . \quad (\text{Eq. 15})$$

4. Experimental Procedure

The experiment was set up with three transmitters placed 1m apart on the base, pointing up towards the ceiling. The transponder, facing down towards the base, was tied to a piece of string and dangled over a beam in the ceiling to keep its position steady, as shown in Figure 13. Caution was taken to make sure that the transmitters and transponder were completely leveled, i.e.: 0 degree rotational angle. When the transponder was deemed to be motionless, the computer connected in parallel to the network of transmitters sent a command over the serial link to start reading data from the Hx11 sensors. Each of the transmitters then sent out signals at its own predetermined sampling rate. The transponder stored all of the calls in its ring buffer and responded in the order the signals were received. Finally, the transmitter, after getting a reply from the transponder, sent the data back to the computer be logged in pre-determined files. The actual x, y, and z coordinates of the transponder were found using a measuring tape. Small currents or motion in the environment may alter the position of the transponder with respect to the base. Thus the precision of the actual coordinates were only around 0.5cm.



Figure 13: experimental set up with three transmitters and one transponder.

5. Results

The data recorded were plotted using a simple Matlab program. Figure 14 is an example plot of the data measured with respect to time, where the red points are in x direction, blue points in y direction, and green points in z direction. When plotting the three distances in Cartesian coordinates, the position of the transponder can be visualized in 3-D space, as shown in Figure 15. The positions of the transmitters are also shown in the figure to give better perspective of the experimental set up.

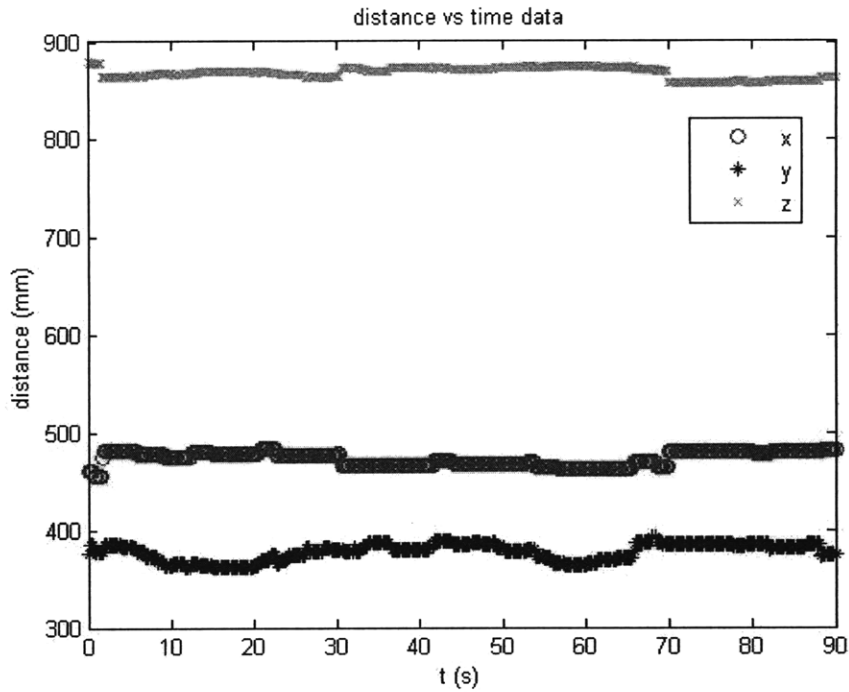


Figure 14: example x, y, and z ultrasonic data plotted against time.

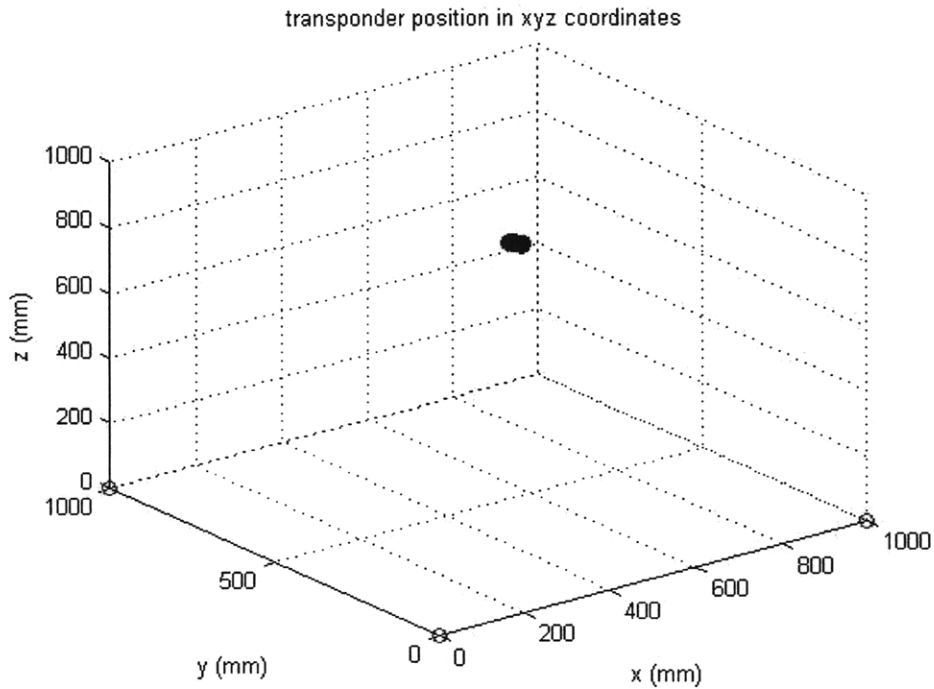


Figure 15: plotting x, y, and z data in Cartesian coordinate to visualize position of the transponder in 3-D space. The blue dot is the position of the transponder, and the red circles are the positions of the transmitters.

However, the data are not always as good as appeared in Figure 14 and 15. Sporadic errors can sometimes be introduced, as seen in Figure 16 around 40s. The filter damped out the effect of the sporadic data, but it still did not completely remove them. The errors do not appear to be related to the geometry of the experimental set up since they emerged in no particular patterns. The sources of the errors are still unknown. Disturbances from the environment and material properties of the walls, floor, and ceiling might contribute to some of the errors presented.

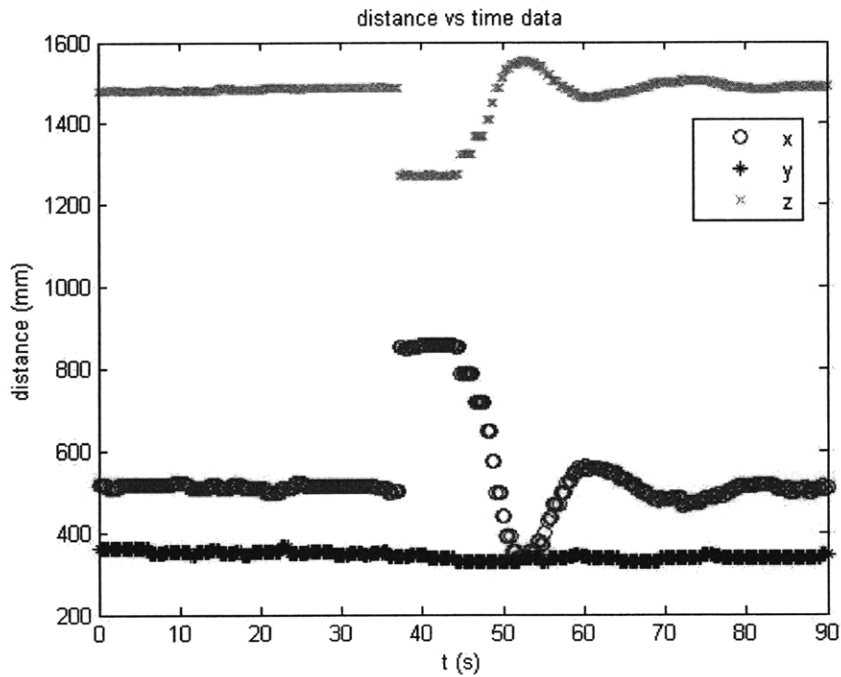


Figure 16: example x, y, and z ultrasonic data with errors plotted against time.

Furthermore, due to distance, rotational angle, and shear angle, as explained in section 2.1, data sometimes do not get logged. Figure 17 shows all of the tests that were done during the experiment. The blue circles represent data with no sporadic errors, the red Xs are data with errors as appeared in Figure 16, and the green triangles show positions where no data were logged. Out of all of the tests performed, 36% of the time the data was not logged, 38% of the time the data was good, and the remaining 26% of the time the data contained some errors.

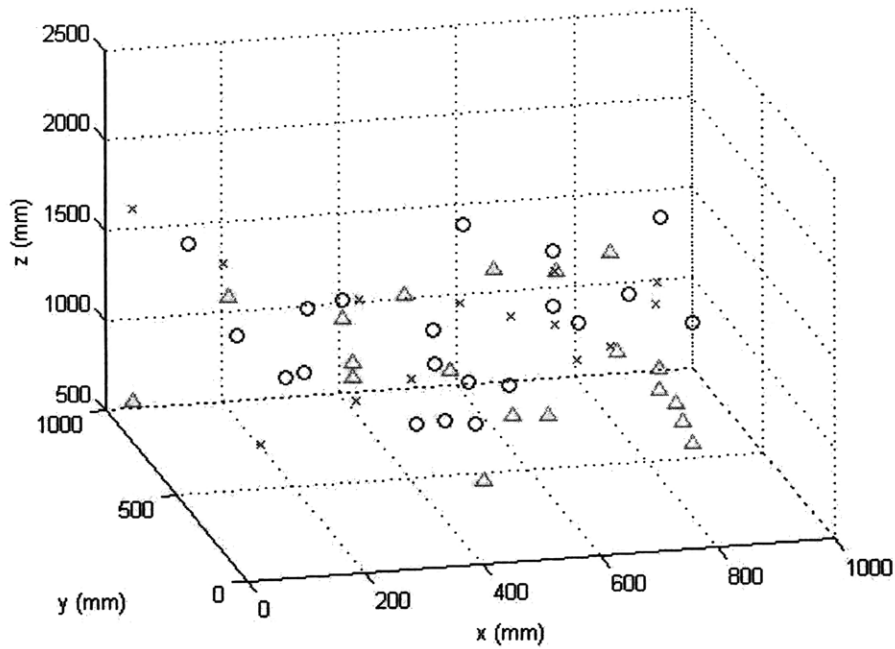


Figure 17: 3-D visualization of tests performed during the experiment. Blue circles are good data, red Xs are data with errors, and green triangles show no data were logged.

When inspecting Figure 17, there seems to be a clear curve above which the data start to get logged. To explain this phenomenon, the shear angle of the transponder, the propagation angle of the signal, and the geometry of the experimental set up were analyzed. Assume that the maximum shear angle of the transponder remains 40 degrees, or an operational cone of 80 degrees. Also assume that the transponder has to be within +/- 45 degrees of the clear line of sight from each of the transmitters, or an operational cone of 90 degrees, to get a signal. Then, data can only be logged when both the transmitters and the transponder are within the angular range of one another. For this particular experimental setup, the shear angle of the transponder is the limiting factor.

Figure 18 is a 3-D visualization of one of the shear angles between the transponder and a transmitter in the experiment. Since the maximum shear angle, β_2 , is assumed to be 40 degrees, the ratio between R_2 , the distance between the transponder and transmitter 2 in xy-plane, and z , the vertical distance, should be equal or less than $\tan(40\text{deg})$, or 0.8391, as described by Equation 16.

$$\frac{R_2}{z} = \frac{\sqrt{(1-x)^2 + y^2}}{z} \leq 0.8391 \quad (\text{Eq. 16})$$

Similar equations can be found for transmitter 1 and 3, as shown in Equations 17 and 18.

$$\frac{R_1}{z} = \frac{\sqrt{x^2 + y^2}}{z} \leq 0.8391 \quad (\text{Eq. 17})$$

$$\frac{R_3}{z} = \frac{\sqrt{x^2 + (1-y)^2}}{z} \leq 0.8391 \quad (\text{Eq. 18})$$

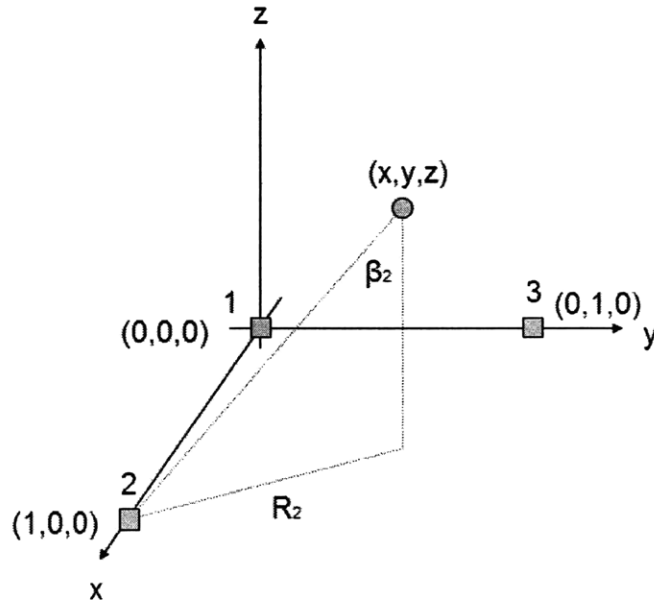


Figure 18: 3-D visualization of the shear angle between the transponder and the transmitter.

The vertical distance, z , is a function of R_1 , R_2 , and R_3 . If the maximum distance among the three fulfills the $\tan(40\text{deg})$ condition, then the other two must be within the operational angular range as well. A simple MATLAB code was written to find the maximum distance in xy -plane between the transponder and a transmitter, and use that distance to find the minimum vertical distance above which data can be logged. Figure 19 is the transponder lower z limit predicted by the MATLAB code. This means that if the transponder is placed below the surface, no data can be logged. The surface is only an approximation, thus it does not guarantee that all positions within the surface can receive data. When the lower limit surface is superimposed onto the test characteristics data, as

shown in Figure 20, the data seem to be following the general behavior of the surface. Out of the tests performed during the experiment, only 16% of the unlogged data came from positions above the lower limit surface, and 15% of the logged data were from positions below the lower limit surface.

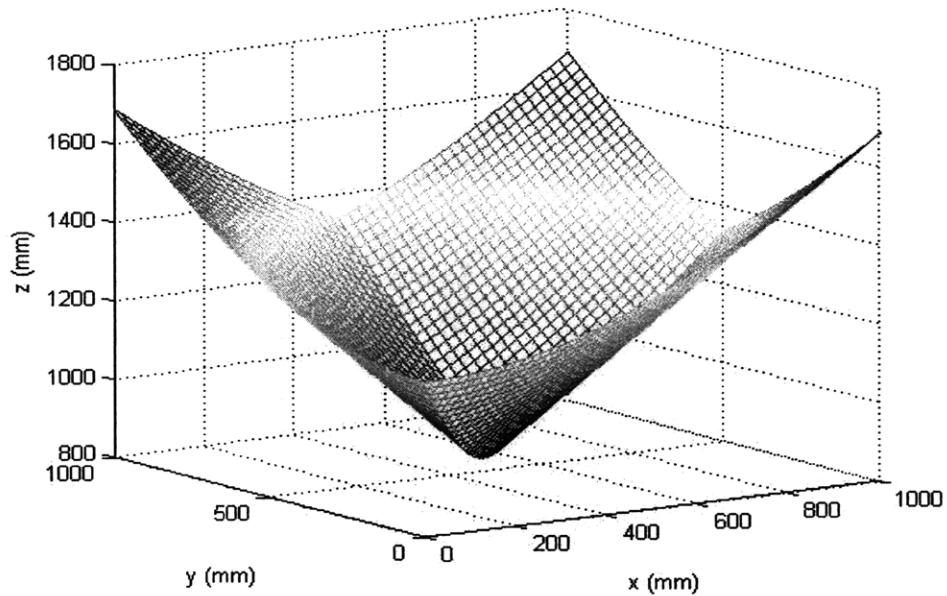


Figure 19: lower z limit surface the transponder must be placed on to properly log any distance data, assuming that the shear angle remains at a maximum of +/- 40 degrees, or an operational cone of 80 degrees.

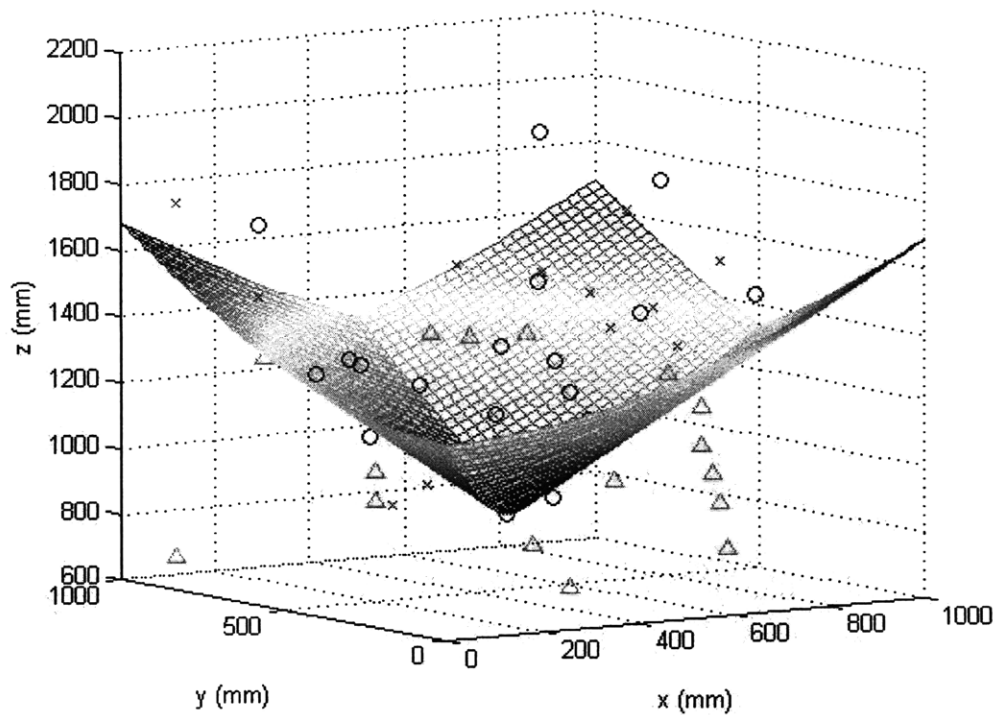


Figure 20: lower limit surface for transponders and experimental results.

To better visualize the data characteristics with respect to the lower limit surface, data with the same z , 920mm, were plotted in the xy -plane, as shown in Figure 21. All three types of data, good data, data with errors, and no data, are present even though z is constant. Since the data with errors contain a majority of good data, it can be viewed as good data within limited time. Then as the data is plotted on the lower limit surface contour, Figure 22, the data characteristics become more apparent. When the transponder is placed within the corresponding contour lines, the transponder is above the lower limit surface and the data are most likely to be logged. On the other hand, when the transponder is placed outside of minimum contour lines, the transponder is below the lower limit operational surface and the data do not get logged.

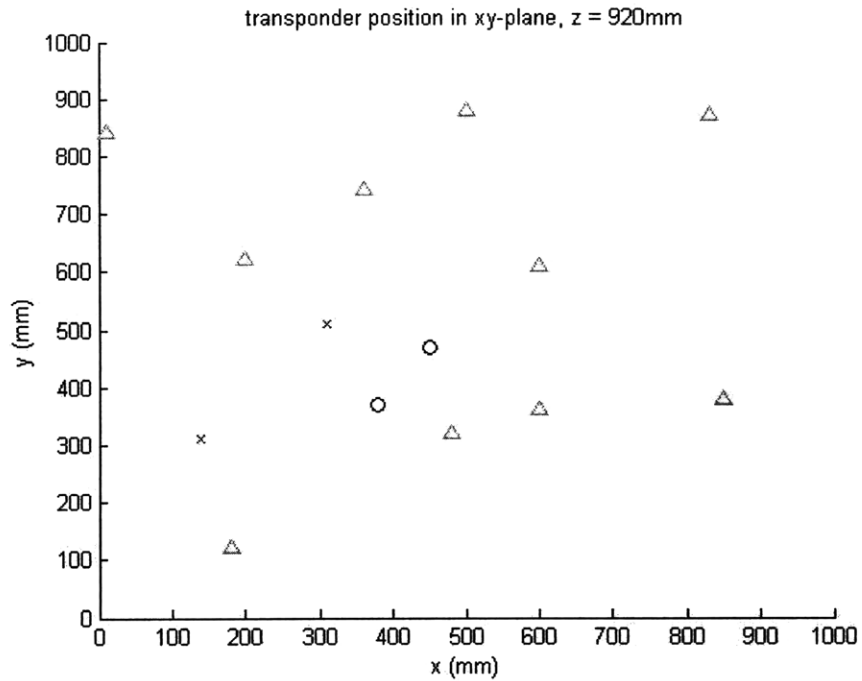


Figure 21: measured data in xy-plane with the same z. Blue circles are good data, red Xs are data with errors, and green triangles show no data were logged.

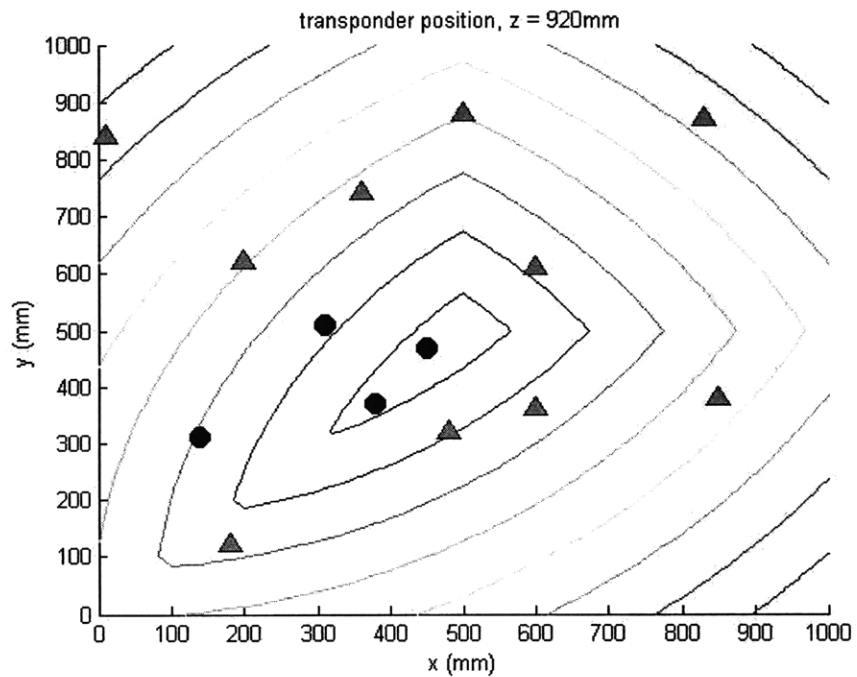


Figure 22: measured data in xy-plane with the same z, plotted along side the lower limit surface contours. Blue circles are good data (or good data within some time limit) and green triangles show no data were logged.

For the data that got logged, excluding the erroneous parts, the average accuracy of the measured data is within 6.29% for x, 5.04% for y, and 4.40% for z of the actual distance. Figures 23, 24, and 25 show the data's accuracy (blue dots) and precision in standard deviation comparing against the actual distance (red line). The average precision for the logged data is 11 mm in the x direction, 9 mm in the y direction, and 5 mm in the z direction.

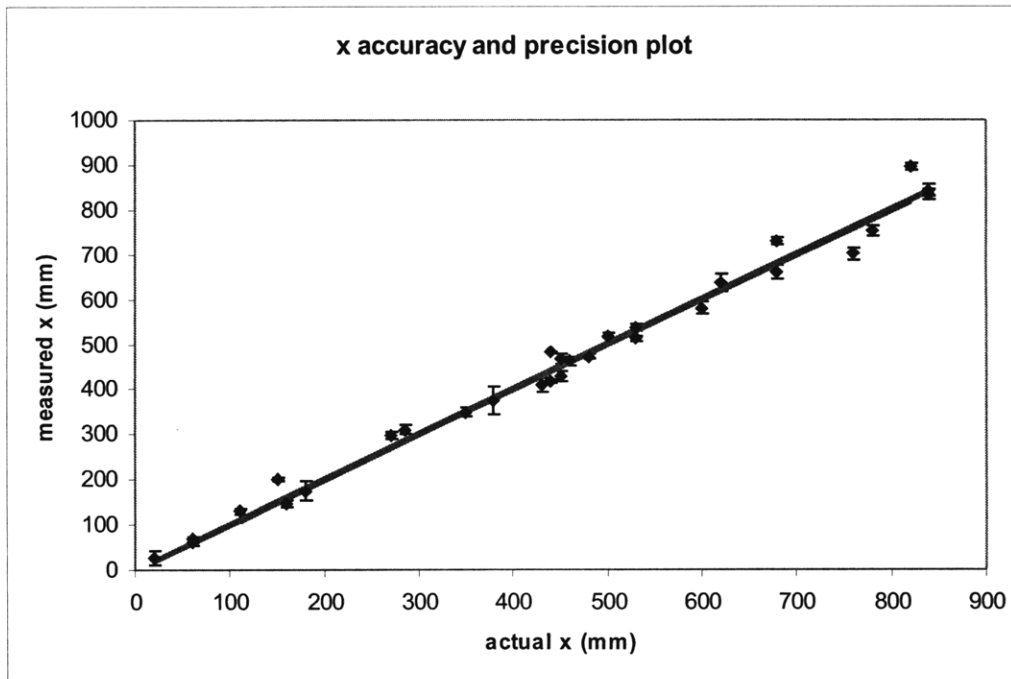


Figure 23: measured data vs actual distance in x direction.

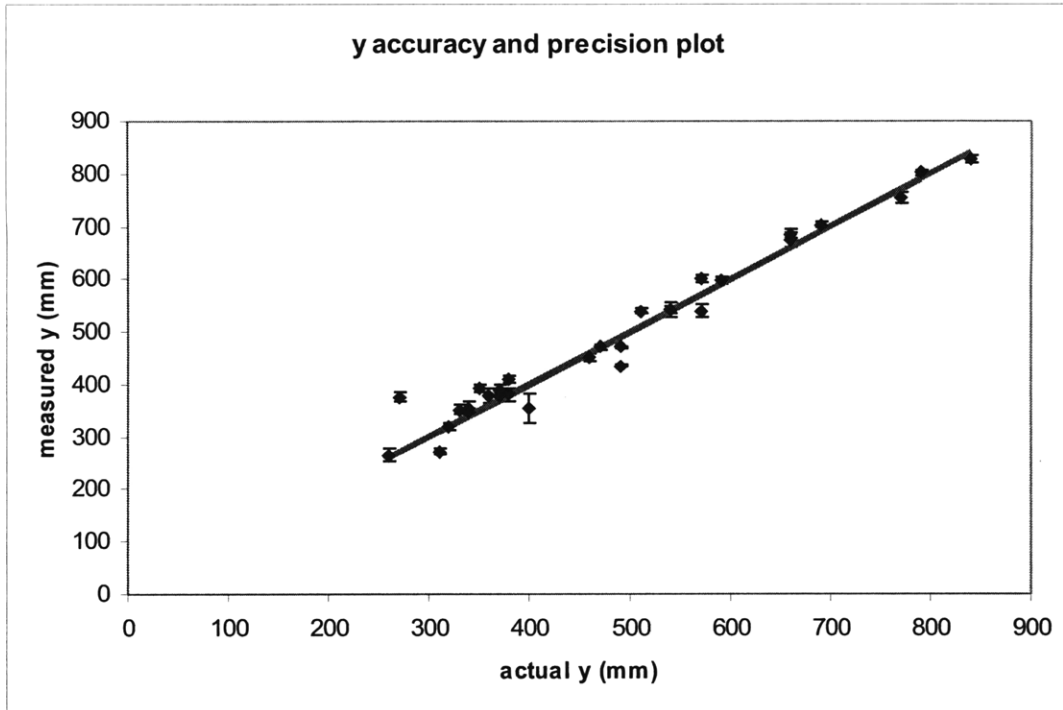


Figure 24: measured data vs actual distance in y direction.

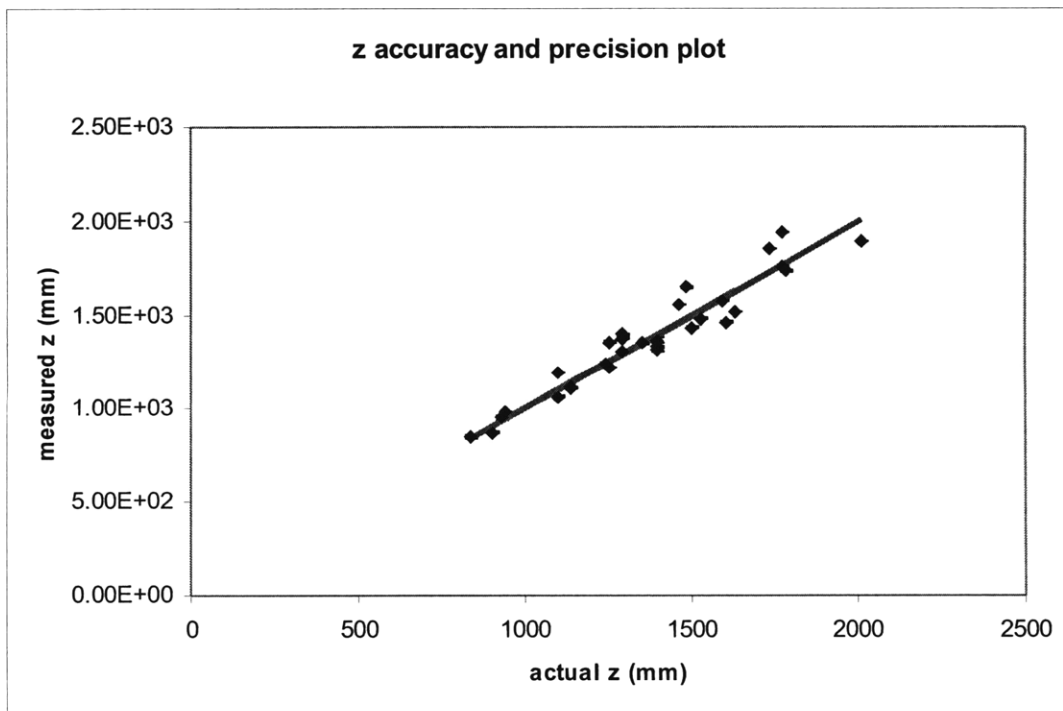


Figure 25: measured data vs actual distance in z direction.

6. Conclusion

Using the Hexamite Hx11 ultrasonic positioning system for three dimensional sensing was not completely reliable. Due to the angular range of the transponder and the transmitters, data did not get logged sometimes. An approximate lower z limit boundary surface was calculated, which correctly predicted the behavior of 70% of the tests in the experiment. Sporadic errors were present in some of the data, with no apparent relation to the geometry of the experiment. If the erroneous part of the data were to be excluded, the accuracy and precision of the data obtained seemed to be quite high. The average accuracy of the data that were logged was within 6.29% for x, 5.04% for y, and 4.40% for z of the actual distance; the average precision was 11mm in the x direction, 9 mm in the y direction, and 5 mm in the z direction.

Further researches need to be done to determine the sources of the sporadic errors that appeared in some the data. Possible causes for the errors may include the material properties and surface reflectivity of the walls and floor in the experimental setup. Ambient noise in the room, air conditioner, and electronic equipments may also contribute to the sensor errors. Better filters or different experimental setup might be able to limit some of the errors. Experiments can also be done in different environments to eliminate external errors. The Hexamite Hx11 ultrasonic positioning system could be useful for three dimensional sensing with its good accuracy and precision, but more reliable prediction of where a data can be logged is still required.

7. References

- [1] J. Borenstein and Y. Koren, "A mobile platform for nursing robots," IEEE Trans. on Ind. Electron., vol.32, no.2, pp. 158-165, 1985.
- [2] C. Cohen and F. Koss, "A comprehensive study of three of object triangulation," Proc. SPIE Conf. Mobile Robots, Boston, MA, pp. 95-106, 1992.
- [3] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning: Sensors and techniques," Journal of Robotic Systems, vol.14, no.4, pp. 231-249, 1997.

[4] H. Akbarally and L. Kleeman, "A sonar sensor for accurate 3D target localisation and classification," Robotics and Automation, 1995 IEEE International Conference, vol.3, pp. 3003-3008, May 21-27, 1995.

[5] H. Surmann, A. Nuchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," Robotics and Autonomous Systems, vol. 45, no. 3-4, pp 181-198, December 31, 2003.

[6] J. F. Tressler, S. Alkoy, and R. E. Newnham, "Piezoelectric Sensors and Sensor Materials," Journal of Electroceramics, vol. 2, no. 4, pp. 257-272, 1998.

[7] Telecommunications Glossary of Telecommunication Terms, National Communications System, Technology & Standards Division, fed-std-1037C, August 7, 1996.

[8] R. Braunstingl, J. Mujika, and J. P. Uribe, "A wall following robot with a fuzzy logic controller optimized by a genetic algorithm," Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference, vol.5, pp.77-82, Mar 20-24, 1995

[9] "Hx11 Sampling Rates and Specifications," <http://www.hexamite.com/hx11.htm>, last checked on: May 8, 2009.

[10] "Hx11tr Receiver-Transmitter," <http://www.hexamite.com/hx11.htm>, last checked on: May 8, 2009.

[11] "Hx11 Software Utilities," <http://www.hexamite.com/hx11.htm>, last checked on: May 8, 2009.

A. Sensor Command and Data Processing Code

```
//this file deals with communication with Hx11 system and data
processing.

#pragma once

#include <stdio.h>
#include "windows.h"
#include "time.h"
#include "conio.h"
#include "stdlib.h"
#include <iostream>
#include <fstream>
#include <math.h>

using namespace std;

static HANDLE _hCom;

void openComPort()
{
    DCB dcb = {0};
    COMMTIMEOUTS timeouts = {0};

    _hCom = CreateFile(TEXT("COM4"),
        GENERIC_READ | GENERIC_WRITE,
        0,
        0,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        0);

    if(_hCom == INVALID_HANDLE_VALUE)
    {
        _hCom = NULL;
        printf("error opening serial port\n");
    }

    /* set timeouts */
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    if(SetCommTimeouts(_hCom, &timeouts) == FALSE)
        printf("errors setting timeouts\n");

    /* set dcb parameters */
    dcb.DCBlength = sizeof(dcb);
    if(GetCommState(_hCom, &dcb) == FALSE)
        printf("error retrieving serial port state\n");

    dcb.BaudRate = CBR_19200;
```

```

    dcb.ByteSize = 8;
    dcb.StopBits = ONESTOPBIT;
    dcb.Parity = NOPARITY;

    if(SetCommState(_hCom, &dcb) == FALSE)
        printf("error setting serial port state\n");
}

void closeComPort()
{
    CloseHandle(_hCom);
}

void sendData()
{
    DWORD numberOfBytesWritten=0;

    WriteFile(_hCom,
              "&",
              1,
              &numberOfBytesWritten,
              NULL);
}

void receiveData ()
{
    float T = 0, D = 0, Dc = 0, Dd = 0, De = 0 x = 0, y = 0, z = 0;
    float Dcsum = 0, Ddsum = 0, Desum = 0;
    float Dcaverage = 0, Ddaverage = 0, Deaverage = 0;
    int K= 103738, M= 344, xpDelay = 1, transDelay = 262144;
    int L = 1000;
    char szBuff[8] = {0};
    char tempBuff[2] = {0};
    char tempBuffb[3] = {0};
    float DcBuff [5] = {0};
    float DdBuff [5] = {0};
    float DeBuff [5] = {0};
    int Dcnext = 0, Ddnext = 0, Denext = 0;
    char callerID;
    DWORD dwBytesRead = 0;
    long data=0;

    // open file to be recorded
    ofstream myfile ("C:\\Documents and Settings\\data\\test.txt");

    //looks for the 0 that indicates the beginning of a data, look
    for the appropriate transponder ID, and reads from there
    while(!_kbhit())
    {
        do {
            ReadFile(_hCom, tempBuff, 1, &dwBytesRead, NULL);
        }while (tempBuff[0] !='0');

        ReadFile(_hCom, tempBuffb, 2, &dwBytesRead, NULL);
    }
}

```

```

if (tempBuffb[0] == '3' && tempBuffb[1] == '2')
{
    ReadFile(_hCom, szBuff, 8, &dwBytesRead, NULL);
    sscanf(szBuff, "%c%x", &callerID, &data);
    T=data-xpDelay*transDelay-K;
    D=M*T/32000-14.322;

    if (D < 3000 && D > 0)
    {
        if (callerID == 'C')
        {
            if (Dcnext < 4 && DcBuff[Dcnext] == 0)
            {
                DcBuff[Dcnext] = D;
                Dcsum = Dcsum + DcBuff[Dcnext];
                Dc = DcBuff[Dcnext];
                Dcnext++;
            }
            else
            {
                Dcsum = Dcsum - DcBuff[Dcnext];
                DcBuff[Dcnext] = D;
                Dcsum = Dcsum + DcBuff[Dcnext];
                Dcaverage = Dcsum/5;
                if (abs(Dcaverage
                    - DcBuff[Dcnext]) > 5)
                    DcBuff[Dcnext] = Dcaverage;
                Dc = DcBuff[Dcnext];
                Dcnext = (Dcnext+1)%5;
            }
        }
        else if (callerID == 'D')
        {
            if (Ddnext < 4 && DdBuff[Ddnext] == 0)
            {
                DdBuff[Ddnext] = D;
                Ddsum = Ddsum + DdBuff[Ddnext];
                Dd = DdBuff[Ddnext];
                Ddnext++;
            }
            else
            {
                Ddsum = Ddsum - DdBuff[Ddnext];
                DdBuff[Ddnext] = D;
                Ddsum = Ddsum + DdBuff[Ddnext];
                Ddaverage = Ddsum/5;
                if (abs(Ddaverage
                    - DdBuff[Ddnext]) > 5)
                    DdBuff[Ddnext] = Ddaverage;
                Dd = DdBuff[Ddnext];
                Ddnext = (Ddnext+1)%5;
            }
        }
        else if (callerID == 'E')
        {
            if (Denext < 4 && DeBuff[Denext] == 0)

```

```

        {
            DeBuff[Denext] = D;
            Desum = Desum + DeBuff[Denext];
            De = DeBuff[Denext];
            Denext++;
        }
        else
        {
            Desum = Desum - DeBuff[Denext];
            DeBuff[Denext] = D;
            Desum = Desum + DeBuff[Denext];
            Deaverage = Desum/5;
            if (abs(Deaverage
                    - DeBuff[Denext]) > 5)
                DeBuff[Denext] = Deaverage;
            De = DeBuff[Denext];
            Denext = (Denext+1)%5;
        }
    }

    if (Dc > 0 && Dd > 0 && De > 0)
    {
        x = 1.0/(2.0*L)*(Dc*Dc-Dd*Dd+L*L);
        y = 1.0/(2.0*L)*(Dc*Dc-De*De+L*L);
        z = sqrt(Dc*Dc-x*x-y*y);

        myfile << x << "," << y << "," << z << endl;
    }

}

}

myfile.close();
}

int main()
{
    openComPort();
    sendData();
    receiveData();
    closeComPort();
    return 0;
}

```

B. Lower Z Limit Boundary Surface Generation Code

```
[x,y]=meshgrid(0:20:1000);

a=sqrt(x.^2+y.^2);
b=sqrt((1000-x).^2+y.^2);
c=sqrt(x.^2+(1000-y).^2);

for i=1:51
    for j=1:51
        m(i,j)=max(a(i,j),b(i,j));
        if m(i,j)<c(i,j)
            m(i,j)=c(i,j);
        end
    end
end

z = m./tan(40*pi/180);
mesh(x,y,z,'EdgeColor','black')
```