ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Localizing your smart phone

Bachelor Thesis

Patrick Tremp

June 30, 2011

Advisor: Johannes Schneider
Supervisor: Prof. Dr. Roger Wattenhofer
Department of Computer Science, ETH Zürich

Abstract

Not just since the growing distribution of smart phones, tablets, navigation systems and similar, localization turns out to be more and more an important topic. For an outdoor environment the global positioning system, or often just called GPS, is probably best known and used by a large share of the population. Nevertheless with an accuracy of about one meter at best and almost no help for indoor environments there is still room for improvement. In the thesis described within this document we wanted to lay our focus to a more fine grained localization in a more restricted area.

By using smart phones with built in magnetic field sensors and the help of external magnets, we wanted to localize the phone in a small area with accuracy in the range of centimetres. The results lead to an implementation of a small and simple game. With a hit rate of more than 90 per cent and an accuracy of less than three centimetres the game works fine.

Additionally we want to examine another form of indoor localization using sound. Measuring the round trip time of a sound signal to determine distances is well known. However, less work has been done using multiple distance measurements as well as the orientation to localize the phone. When a phone is rotated, it should automatically determine the environment and its position within. Using a chirp signal as sound source we could measure the distance with an accuracy of approximately ten centimetres. With a measurement to all walls in a room we were finally able to draw the surroundings of the room on the phone.

Finally we briefly evaluated the possibility of locating obstacles between two mobile phones using signal strength and Bluetooth. Doing so, we could identify obstacles in a range of about five meters.

Contents

Chapter 1

Introduction

This Bachelor thesis focus on three different mechanisms for indoor localization using a smart phone. So far different strategies have been studied such as localization using GSM [6] or via received signal strength values (RSSI) of wireless networks (see e.g., [8]). All these approaches realize an accuracy of about 1 - 5 meters.

We like to present another idea of indoor localization where less work has been done yet. Using sound for distance measuring and additionally the orientation in order to localize the phone within a room. We developed an application which uses the round trip time of sound signals combined with the orientation to draw the room the phone is currently in. The proposed algorithm could then be further used to orient oneself in a given room.

Detecting obstacles between two phones or a sensor and a phone is another field of interest which we cover in this thesis. We experiment with Bluetooth and signal strength values in order to find an algorithm which is able to detect such obstacles.

The general idea of indoor localization has an accuracy of room level or the range of one meter at best. We wanted to go to an even smaller area. Therefore this thesis also discusses algorithms and experiments with magnetic fields using the magnetic field sensor built in many smart phones these days. We present around device interaction (ADI) as well as localization in a magnetic field in the plane with a range of 30 to 40 centimetres and accuracy of less than five centimetres. For demonstration we developed two small games one for ADI and one for the magnetic field.

In this paragraph we will discuss the variety of tools we used. We were working with the Android software development kit (SDK)[1] provided by Google with Android 2.2. As for testing and developing we used the developer phone Nexus One[2]. If for an experiment a second phone was needed, we worked with a HTC Desire phone.[3] Since the application programming interfaces (API's) for Android are all written in the programming language Java, we as well used this language for all our applications. For development we were working with the integrated development environment (IDE) Eclipse in the version 3.6.1 with the Android development tools (ADT) plug-in installed. (For a detailed description on how to install the Java development kit, Eclipse, the SDK and the ADT please consider the developer site developer.android.com.) Screen shots of the mobile phone were also taken using the ADT. Pictures of the experiments were either taken by the Nexus One or the HTC Desire built in camera. For illustrating the results and plotting values we used the numerical computing environment Matlab by MathWorks.[4] Finally to write this thesis we used LaTeX and worked with the IDE TeXnicCenter[5].

After having briefly introduced what this thesis is all about Chapter 2 will discuss in detail our work on the magnetic field localization. Chapter 3 then describes the algorithm we have developed and experiments done for the localization using sound. The research on the obstacle recognition using Bluetooth and signal strength is explained in Chapter 4. Each of these chapters describe the applications which we developed, experiments and evaluations as well as a conclusion and possible future work. In Chapter 5 we share our experiences on working with the Android platform and the different problems which we encountered during the thesis. Finally in the appendix we put additional images, maps of our experiments and code snippets.

[1]For tutorials, class descriptions and other useful tips and tricks about the Android SDK visit the developer site: developer.android.com

[2]See the description on Google phone gallery: www.google.com

[3]The official website of HTC www.htc.com.

[4]The official website of Matlab: www.mathworls.com

[5]TeXnicCenter: www.texniccenter.org

Chapter 2

Using the Magnetic Field Sensor for Localization

2.1 Introduction

This chapter discusses the studies made with the magnetic field sensor of the Nexus One phone and external magnets. As smart phones with a lot of sensors are more and more common, additional input possibilities for application and games are in demand. Around device interaction (ADI) using magnets and the compass sensor is already known[4]. During our studies we were able to use ADI for input with an external magnet too. We used ADI in order to adapt the game 'Snake' taken from the Android sample code [2] in a way that one can control the game with a magnet.

The more we laid our focus on localization aspects in magnetic fields. We worked on the problem of discovering the phone in a magnetic field generated by two magnets in a plane. The magnetic field sensor in the Android phones is usually used as a compass measuring the magnetic field of the earth. Values are mostly in the range of 30 to 60 micro Tesla (μT). The magnets used for this study create much stronger fields and may go as high as the limits of the sensor (i.e. $2000\mu T$). All magnets used are bar magnets. Such a magnet creates a magnetic field which looks like the one in Figure 2.1.[1] The magnetic field of a bar magnet is approximated by the formula

$$B = \frac{B_0 A L}{(2\pi r^3)}$$

Here B is the magnetic field, L the length of the magnet, B_0 the field at a pole, A the area and r the diagonal distance to the closer end. We arranged two magnets on the table in a 90 degree angle between them in order to create a magnetic field in a plane. We expect the resulting magnetic field

[1]Image taken from Wikipedia: http://en.wikipedia.org/wiki/File:Magnet0873.png

to be the superposition of the fields generated by two magnets according to the above formula. Since the magnets used for this research are rather weak we expected to measure the generated field in a range of up to around 35 - 50 centimetres. The measurable values are expected to be quite high close to the magnets - especially in the corner between the two magnets - and to decrease fast with increasing distance to the magnets. Using stronger magnets was not an option because the limits of the built in sensors are to low and we already reach them with the weak magnets.

In order to present our results for the magnetic field studies we developed a small Tic Tac Toe game. This game is completely controlled by the position of the phone within a three times three game board lying in the magnetic field. For the localization algorithm two approaches are discussed in the according section.

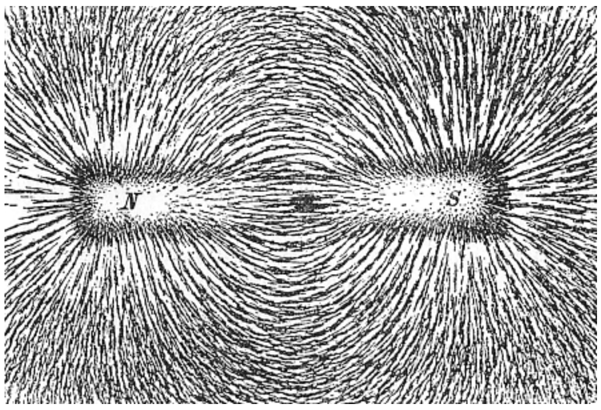
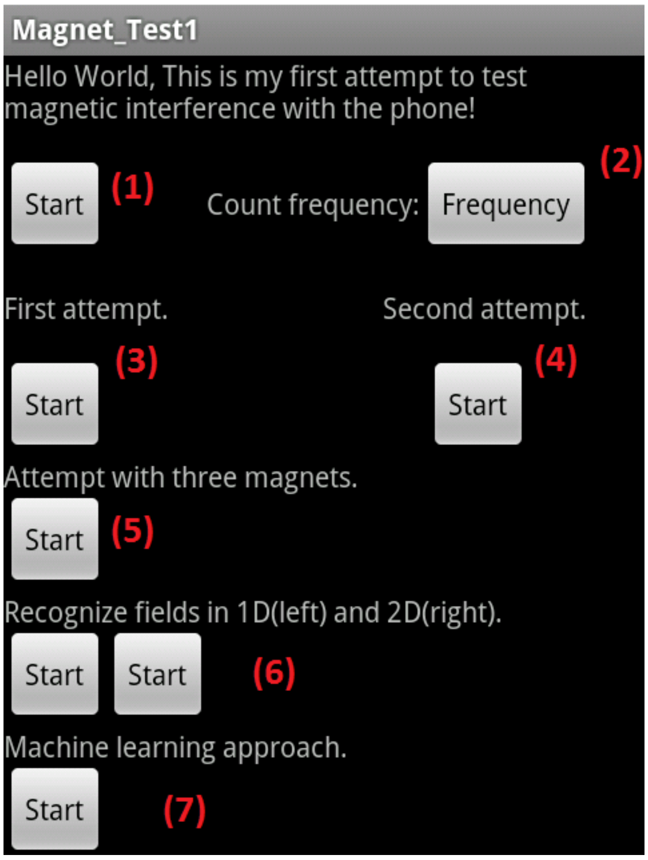


Figure 2.1: Iron filings that have oriented in the magnetic field produced by a bar magnet

2.2 Applications

2.2.1 Testing Application

In order to get some information about the strength of the magnetic field measured by the phone we developed a testing application. This rather big application includes different smaller sub applications. These are used for example to print the measured values to a Matlab file. This file then can be further processed by the Matlab software to produce a graphical output. Another sub application displays the measured values on the display of the phone. Furthermore there are sub applications which are used to recognize different movements with the magnets around the device as well as movements with the phone in the magnetic field spanned by the two magnets. In Figure 2.2 an overview of the main view is given in order to orient the reader with the different small applications.


Magnet_Test1

Hello World, This is my first attempt to test magnetic interference with the phone!

Count frequency:

First attempt.

Second attempt.

Attempt with three magnets.

Recognize fields in 1D(left) and 2D(right).

Machine learning approach.

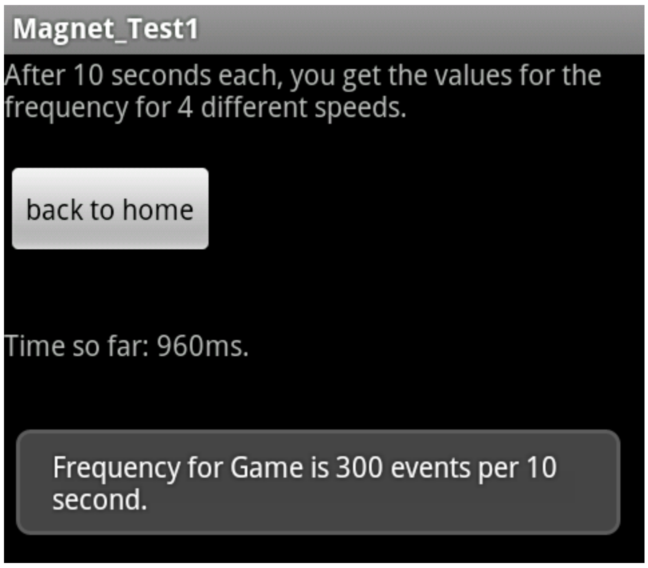
Figure 2.2: Screen shot of the main view of the Magnet_Test application. (1) Record sensor values, (2) calculate frequency of different modes, (3) + (4) attempts to read and interpret ADI input, (5) + (6) attempts to locate phone in magnetic field with different amount of magnets, (7) test of machine learning approach for locating the phone in a magnetic field spanned by two magnets.

View and Record Readings

In a first sub application (number 1 in the figure) we display x-, y- and z-axis values read by the magnetic field sensor of the phone. Besides the values of the three axes we also display the number of significant changes. Such a change is defined by a threshold depending on the event one is looking for. We used 2.0 as a standard value. These numbers helped us to identify special events such as an interaction with an external magnet. Since heavy listings of sensor values consume a lot of battery life we introduced buttons to pause and resume measuring. By pressing the save button we store the readings for each axis in a text file in the download folder of the SD card.

Count Frequency

The count frequency defines the sampling frequency. It can be one of four values defined in the SensorManager class. To measure the impact for different values we created another sub application. For a defined time (we used ten seconds as a standard value) the number of reading events by the magnetic field sensor get counted for each delay rate. The results are displayed on a pop up message as shown in Figure 2.3. After the fourth run has finished, automatically the main view gets displayed again.


Magnet_Test1

After 10 seconds each, you get the values for the frequency for 4 different speeds.

Time so far: 960ms.

Frequency for Game is 300 events per 10 second.

Figure 2.3: Screen shot of a pop up message in the frequency application for the SENSOR_DELAY_GAME delay rate.

Recognize ADI Input

The two sub applications highlighted with the numbers 3 and 4 are two different attempts to recognize ADI input with an external magnet. The goal of these two applications is to recognize the following movements: Movements from left to right and vice versa, movements from top to bottom and vice versa as well as a 'clicking' movement from close to the phone to further apart. In the application 3 we tried the approach to consider always the last one hundred values collected. For all the five movements described above we check whether the current values match the reference values of an event. Because the measured values have different peaks depending on the position to the phone we can recognize where the magnet currently is. If e.g. we observe the magnet first on the left hand side of the phone and later on the right hand side, we can report such a move. Then again in application 4 we immediately set a boolean value for the position of the magnet when a special position is detected. The algorithm then further checks on each sensor event if e.g. the boolean value for right and left is true. Each position of the magnet has not only a boolean value as described but as well a time stamp of the moment it was set. Therefore we can detect if the movement was in our example from left to right or vice versa. When there is no special event detected for a given time a timer goes off and the boolean values are reset. Using this approach we were even able to detect diagonal movements additionally to the already described ones. For both approaches the interface looks the same. The around device interaction worked out really nice and we were able to adapt the Snake game described in the next sub chapter. Therefore we wanted to lead our studies to something different were not much work has been done yet. We switched from moving magnet and a more or less inactive phone to the opposite. The following four applications hence describe the interaction of the phone in a magnetic field spanned by a different number of magnets.

Movements in Magnetic Fields in One to Three Dimensions

In order to test movements in a magnetic field spanned by three magnets we developed the small application marked with the number 5 in Figure 2.2. The interface looks the same as in the previous two applications. Here we tried to identify movements in all the three axis. In other words movements to the left or right, forth or back as well as up or down. The algorithm remembers the last couple of values as well as values at a given time. Whenever a timer runs off we check, if there has been a significant change on the values in any direction from the current values to values from the last timeout. If so, we again report the direction of the move. The values are saved and we wait for the next timer to expire. The reason for the timer is to guarantee that the user interface (UI) is always responsive. All in all this approach did not work out proper and neither did a similar algorithm a the one described for application 3.

Since three dimensions did not work out as desired we then first focused on a setting with just one magnet. Therefore we developed a small application (number 6 on the left) which tries to identify where a phone is located in front of a single magnet. Figure 2.4 illustrates the setting for this test where just the magnet on top in the figure was used. In the UI we again display the current sensor readings. Whenever a user presses a so called 'Hit me' button the algorithm determines in which field the phone is in and displays this information. The algorithm is quite simple and uses four boolean flags for each of the single fields. On every sensor reading we check with a few if-statements whether we have found values that are typical for a certain field. If so we set all flags to false except for the detected one. Doing so we just have to check which flag is set when the user hits the according button. Simple as that this solution seemed to work out very well.

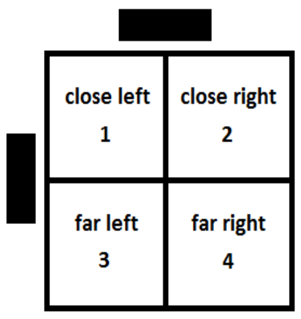


Figure 2.4: The setting for the localization of the phone using one or two magnet. For test using one magnet, it is positioned on top of the field. Tests with two magnets have them left and on top of the field. Four different zones are used for testing.

Machine Learning and Nearest Neighbor Approach

With this small success we finally worked on two dimensions and solved the problem of locating the phone in a magnetic field spanned by two magnets. The setting for the application on the right hand side of number 6 is described in Figure 2.4. Here we use both magnets the one on top and the one left to the board. We used a similar algorithm than in the case with just one algorithm. For each field we set a boolean flag indicating whether the phone is currently in that field or not. By pressing the 'Hit me' button we just check which flag is set. On each sensor reading we check if the actual reading is within a polygon defined by some predefined values for the four fields. If so we set the right flag. Unfortunately this approach worked not so well. Hence we decided to start over and implement yet another one.

The machine learning approach defined in the application with the number 7 in Figure 2.2 is based on user input at first for calibration.[2] There are two versions supported at the moment, one for four fields and one for six fields. If future work in this field is to be done this application can easily be enlarged to nine fields or even more. As for both versions the same algorithm is applied, we focus in the following on discussing the one with four fields. On starting the application we get to the first phase of calibrating. Here we see four buttons which stand for the four fields and a counter for the number of collected points so far. The user is asked to position the phone in one of the four fields and hit the according button. For the algorithm to work proper it is best to collect at least two points in each field. After having collected a couple of data points the user hits the finish button and is lead to another view. In this second view the algorithm does his work. Whenever one hits the test button in the middle of the screen the algorithm calculates the field the phone is in. This is done by scanning through the data points

[2]For a definition of machine learning please consider Wikipedia.

and calculating the nearest neighbor. The data is organized in different clusters or collections according to the corresponding fields. So whenever we found a nearest neighbor, we can just look up the field it belongs to by identifying the cluster the point lies in. After the algorithm finishes it presents the calculated field on the screen together with a simple question if the given answer is correct. If so we just acknowledge and proceed with the next test. If the algorithm calculated a wrong field we can now tell it the correct value. This additional information gets saved along the others in order to improve the results further on. As we do not want to calibrate and save the initial values all the time, we also added the functionality to save and load collected data to a file on the SD card. An approach like this turned out to work really well with four and six fields. Therefore we used this algorithm for the tic tac toe game.

2.2.2 Snake Game

In order to show the possibilities of the around device interaction (ADI) using the magnetic field sensor of the phone we developed a small game. In the Android sample code [2] applications there is an implementation for a 'Snake' game. We took this application and made some small adaptations of the input methods. Now one can play the game not only by using the track ball of the Nexus One but as well by moving a magnet around the phone. The 'Snake' game, as known from former mobile phones, contains of a snake crawling over a game board. The player needs to conduct it to some fruits to allay its hunger. With eating fruits the snake gets longer and longer. The games ends when the player conducts the snake either to the an edge of the board or to the snake itself. Goal of the game is to eat as many fruits as possible. The gestures necessary to control the games are the following. Movements in the x-axis from left to right or vice versa let the snake move right or left respectively. Movements in the y-axis from top to bottom or vice versa let the snake move down or up. The move from top to bottom is also applied to start a new game.

The implementation might not be perfect and some gestures may not be recognized but still this application shows the possibilities that ADI offers. In this case we only make use of four different gestures in the two dimensional space. With some more effort we could also make use of gestures in the vertical direction and hence make use of all the three different dimension. These gestures might be used for zooming in a gallery view or similar.

2.2.3 TicTacToe Game

In order to show the results of the studies on localizing the phone in a magnetic field in a plane, we developed a TicTacToe game. The game is designed for two players. Alternately one player has to set crosses to the fields and the other circles. When someone manages to have three of his symbols in a row (no matter if horizontally, vertically or diagonal), then that player wins. As already described in the introduction the game board contains of three fields times three fields. The game board has on the left and on the top bar magnets which together span the desired magnetic field. To achieve best results the board should be about 15 centimetres by 15 centimetres whereas each field is five centimetres by five centimetres. A picture of the board used for our experiments and testing of the game is attached in the appendix.

For detecting the phone in the field we used two different approaches which both need some user feedback at first. We will discuss them separately in this section.

Machine Learning and Nearest Neighbor

The first approach to the problem is to solve it by means of machine learning as described in the last application of the testing application. When the user starts to play for the first time he is asked to give some feedback. The collected data in this step is saved for further games. The user may change it anytime by recalibrating the system. Using a clustering approach the player first needs to tell the system in which field the phone currently is in. This has to be done for each field at least two times. In a second step of the calibration the player may lay the phone somewhere on the board and ask the algorithm about the location. If all fields are recognized correctly we are good, can save the data and proceed to the game. If some locations are falsely estimated the player can correct it immediately. The algorithm itself works like this. We save the x, y and z values of the magnetic field sensor data obtained together with the field number (1 to 9) the user told us. After the calibration, we have a set of about 20 to 50 or even more data points all over the board. Localization than uses a nearest neighbor search (also known as proximity or closest point search)[3] where the vector of the readings of the actual position is compared to the given data. The given data point which is closest to the reading is chosen and its field is selected as output. The feedback needed from the players in this approach is quite high. For a public game this may be a killer. Still the algorithm gives good results and the game can be played fluently with very few false inputs.

Four Corners

The second approach then tried to use less feedback from the user. Here the calibration is reduced to just four readings at each corner of the board. The values measured by the sensor decrease rapidly at first and little later on the bigger the distance gets between the magnets and the phone. Therefore we

[3]See nearest neighbor search on Wikipedia

can estimate the field by trying to fit the measured values in the according curves generated by the four collected values. Since we have just four points to compare to this approach is not as accurate as the machine learning one.

Game Modes

There are three different game modes to play. Two are local on the phone. The player can either play against the computer or against another human player on the same phone in an alternative manner. For the third mode two phones connect to one another via Bluetooth and may play on two different game boards. Here of course calibration needs to be done on both phones separately and the data points are not synchronized because sensor readings may differ from phone to phone.

2.3 Experiments

In order to get results for both around device interaction (ADI) as well as the localization in magnetic fields we made many small experiments. As described in the test application section, we wrote applications to display the values read by the sensor and to produce Matlab files. On testing in the magnetic field we especially made some recordings of moves along all three axes as well as turning the phone 360° in a circle. The magnets used for these experiments were built in white board erasers. (We put a few pictures of the erasers in the Appendix.) The magnets lay still and the phone was moved in the magnetic field generated by them. Depending on the particular experiment one, two or three magnets were used. If multiple magnets were used, we arranged them in a way that each magnet covered one dimension. Hence two magnets were arranged in a ninety degree angle to each other. A possible third one was put below the two in order to span the third dimension. For ADI we usually held the phone in one hand or put it on a table. Then we moved a magnet with the other hand around the phone. Again movements from left to right and vice versa as well as from top to down and vice versa were tested. Additionally we made a few experiments with diagonal movements over the phone as well as movements in the z axis closer and further away from the phone. All in all the entire procedure of developing the algorithms needed for the games and the study where heavily based on countless small experiments. A description of each individual experiment would blast this thesis and is therefore omitted.

2.4 Evaluation

Some small evaluations of the different applications were already presented in the testing application subsection. In this section we like to present the outcome of a few small experiments in order to give the reader an insight to the Matlab files. These files and the resulting graphs helped a lot to fine-tune the algorithms and to set the right values.

First we like to present the reader the outcome of some ADI tests made. In Figure 2.5 we can see a movement from the right of the phone to the left. We observe a sinusoidal like curve for the x-axis values. Figure 2.6 shows a movement from the bottom of the phone to the top. Here we can observe the y-axis making a huge jump from negative values to positive. Combining these two observations as well as results from other movements and plots lead to the different thresholds for our around device interaction applications.

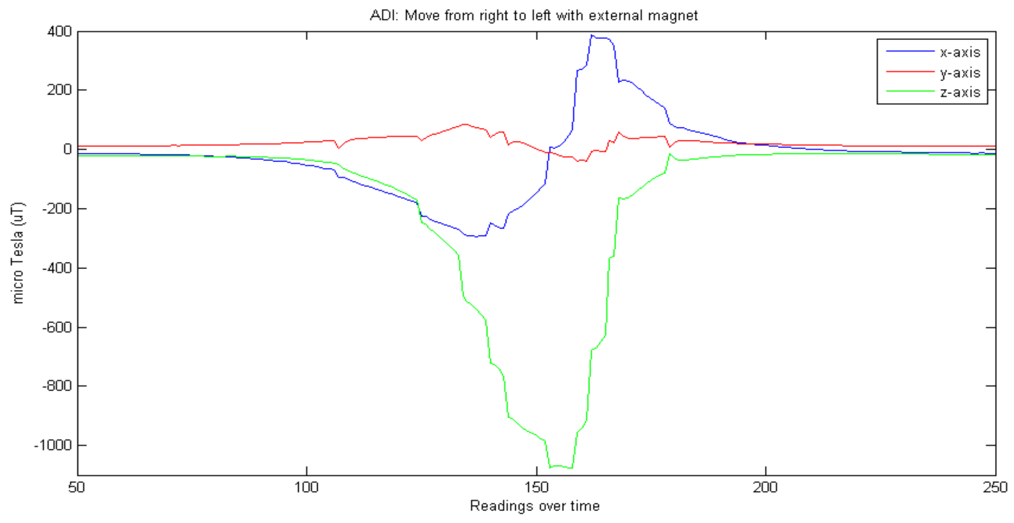


Figure 2.5: Matlab plot of an ADI move from the right of the phone to the left.

The outcome of the tests made for the phone in a magnetic field was quite similar. Figures 2.7 and 2.8 show the results for a 360° turn within the field as well as a move from the right hand side into the field and back out again. In Figure 2.7 we can observe that the z-axis values always stay more or less on the same level. Since we just have magnets to span two axes, this makes perfectly sense. The more we see that as long as the phone moves on the same plane as the magnets are positioned, we may omit the z-axis for the algorithm. The curve of the x-axis first goes to negative values and then smoothly increases to its maximum before it falls again to the starting values. A similar curve can be observed by following the y-axis. These smooth curves helped us a lot to define some thresholds and how a certain movement should look like. Figure 2.8 shows one such movement namely a movement with the phone from right to left. Since in comparison to the 360°

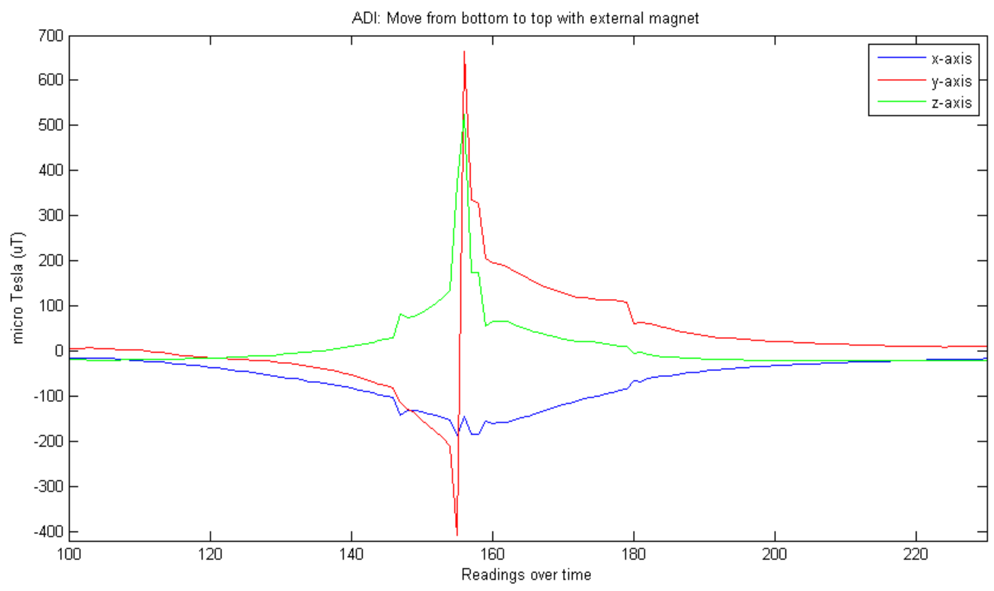


Figure 2.6: Matlab plot of an ADI move from the bottom of the phone to the top.

turn we are initially outside the magnetic field, the z-axis has a somehow big curve in this figure too. Regarding the x-axis curve we can observe that the values first grow when we come into the field. As soon as we get close to the left magnet, they drop to very high negative values. This behavior was utilized for the definition of the movements and certain thresholds as well. By closely examining these two and many other similar graphs we were able to adjust our algorithms to produce good results.

For the tic tac toe game we made different tests in order to get the nearest neighbor problem to work. We heavily used the calibration tool where we gave the algorithm a few fix points and then analyzed whether it determines new values correctly. For every measurement we had direct feedback and could adjust our algorithm very well. In the end we were able to determine more than 90% of the positions correctly. In other words, this means that we could detect the position of a phone on a board in a range of less than five centimetres. This is quite an improvement compared to GPS and certain other indoor localization approaches. Nevertheless such an approach requires a magnetic field and a lot of user input which does not make it very beneficial for real-life applications. Still in an environment where using magnetic fields is not a problem this approach would work just fine.

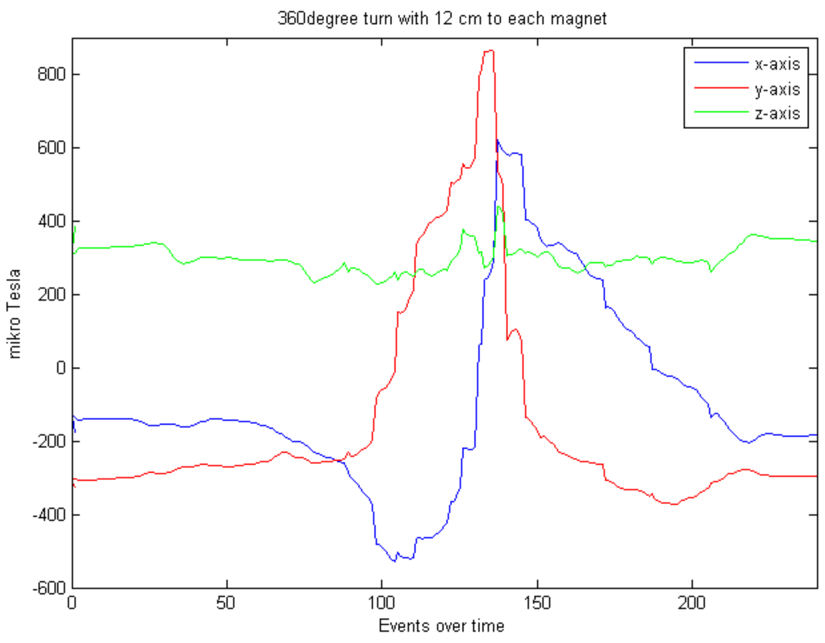


Figure 2.7: Matlab plot of a 360° turn of the phone in a magnetic field spanned by two magnets.

2.5 Conclusion

The studies on the magnetic field sensor showed that these are not just useful for a compass and orientation but may also be used for additional input methods as well as localizing the phone in a close area. Experimenting with around device interactions using the magnetic field sensor confirmed the results of Katebdar[4] and showed what might be possible with future studies. With localization using magnetic field sensor we were able to be quite accurate in the range of less than five centimetres. Hence such an approach is a very interesting possibility for indoor localization. The algorithm based on machine learning in a testing phase and a nearest neighbor approach for position detection was very accurate. Unfortunately it forces a user to go through a long calibration process. So in order to make games and other applications interesting for all the impatient people out there possibly needs some improvements here.

Certainly not everything just worked out as it should. One of the biggest problems that we encountered during the studies was the recalibration of the magnetic field sensor. This happens when values larger than the maximum value of ±2000 were detected. This leads to incorrect localization. Also for ADI the sensor looked as if it would recalibrate quite often which turned given thresholds unusable.

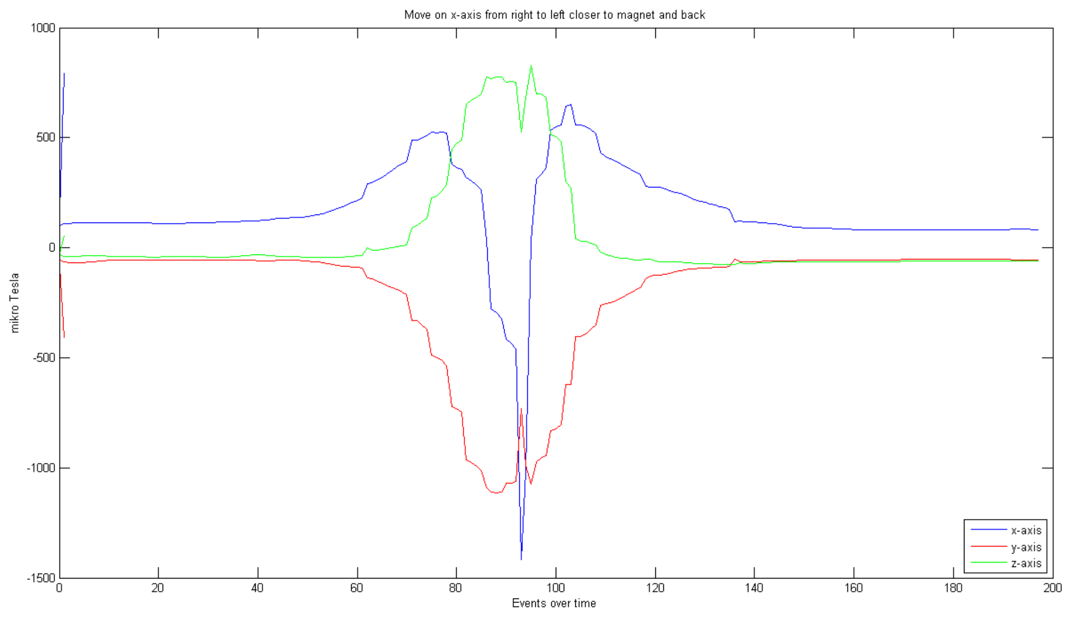


Figure 2.8: Matlab plot of the phone moving from right into the magnetic field close to the left magnet and back out of the field again.

As for future work we like to experiment with stronger magnetic field sensors to work on a bigger scale. The localization within magnetic fields could not only be used for smart phones and the like but as well for other applications. HTC will soon publish their new tablet called Flyer with an additional pen for different input possibilities[3]. With some more work in the field of ADI using an enhanced magnetic field sensor we can imagine that soon devices might be published which come with magnetic rings or similar and allow the user to use the entire space around the phone for input. Another interesting research topic is the usage of multiple magnetic field sensors to detect multiple magnets as input in a more accurate manner. This may lead to similar possibilities as multi touch gestures on the screen but in the three dimensional space.

Chapter 3

Localization using Sound

3.1 Introduction

This chapter discusses our efforts to localize a smart phone in a room by means of measuring sound echoes which return from a wall of the room. Since simply measuring the distance from the phone to a wall has already been developed for the IPhone[5] we wanted to focus on something different. Hence we went one step further and use the distance measuring to find the position of the phone in a room. To figure out the position we make a 360 degree turn while continuously measuring the distance to the wall. Using this data, an algorithm will generate a map of the room and the position of the phone therein. Such an application may not only help people to take the measurements of a room but as well assist other algorithms for exact indoor localization.

For the sound output we used a chirp signal. A chirp is in our case a signal with a linearly increasing frequency over time. As we use a linear chirp the frequency varies like this: $f(t) = f_0 + k \cdot t$ with t being the time, k the chirp rate or frequency increase rate and f_0 the starting frequency.[1] The corresponding time-domain, sinusoidal function is:

$$x(t) = sin\left[2 \cdot \pi \cdot (f_0 + \frac{k}{2} \cdot t) \cdot t\right].$$

Figure 3.1 is taken from Wikipedia[2] and shows such a chirp.

In order to analyze the recorded data and to calculate distances we used the fast Fourier transform (FFT). The FFT is an efficient way to calculate the

[1]This is taken from Wikipedia. For more information please consider the entry about chirps. www.wikipedia.org

[2]www.wikipedia.org

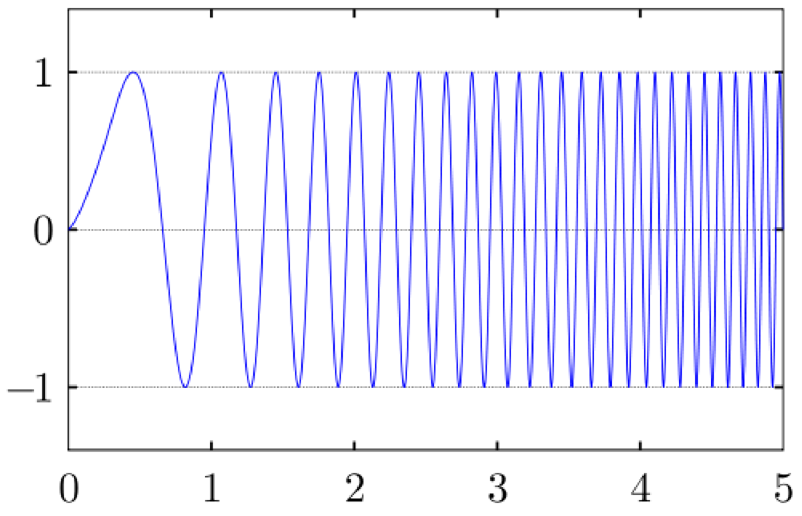


Figure 3.1: A chirp signal. Image taken from Wikipedia.

discrete Fourier transform (DFT) as well as its inverse. [3] This DFT breaks down the data into components of different frequencies. For $x_0 \dots x_{N-1}$ being complex numbers the DFT is calculated like this:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i \cdot 2\pi \cdot k \frac{n}{N}}, \quad k = 0, \dots, N-1$$

This usually takes as an upper bound $O(N^2)$ operations. Typical FFT algorithms like, the one used in this application, usually have an upper bound of $O(N \cdot \log N)$ operations.

Another crucial thing for measuring distance using sound is the speed of sound. Depending on the temperature, humidity, air pressure and density of the gas the speed of sound may vary. For our purposes we just consider the temperature and set the speed of sound according to this formula:

$$c_{air} = (331.3 + (0.606°C^{-1} \cdot \vartheta))m \cdot s^{-1}$$

In this formula c_{air} describes the speed of sound and ϑ the current temperature.

We knew about the success of the SonarRuler[5] application for the IPhone. Therefore we wanted to achieve an accuracy of around ten centimetres. The application shall work accordingly in most of the measurements and the resulting drawing shall look similar to the real room.

[3]This information is mainly taken from www.wikipedia.org

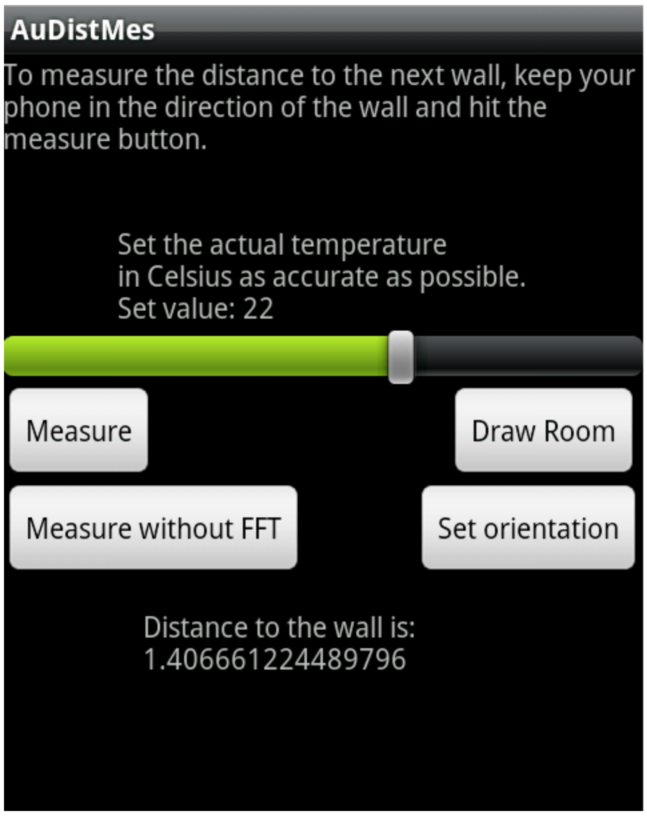
3.2 Application

The application which we developed lets the user calculate the distance from the phone to the wall. This is possible by means of the time it takes for listening to a sound output and its echo from the wall. In order to draw a picture of the room the phone is in, all it takes is to measure all directions of the room. Eventually one can hit the draw room button and a new view will try to predict the size and shape of the room. Figure 3.2 shows the main functionality of the application. There is a slide bar where one can enter the current temperature of the room to vary the speed of sound. The initial value is set to 23 degree Celsius. There are two buttons to measure the distance to the wall, one for using the fast Fourier transform and another to measure the distance without FFT. Because of the heavy calculations the measurement can take a while, so try to keep the phone still for at least two seconds. Furthermore the calculated distance is displayed as well as possible other information. When measuring the distance, we need to hold our phone in an upright position. This leads to rather weak results from the orientation sensor. Therefore we added another button to the view. The user now has to measure first holding the phone upright. Then he or she needs to turn it to a horizontal position and press the set orientation button. Best results were achieved when holding the phone horizontally and pointing the microphone directly to the wall. In order to direct the sound output to the wall as well we used a cardboard as a cover. This is illustrated in Figures 3.3 and 3.4.

In a second application the user can estimate the distance to the wall and give some upper and lower bounds. This helps the algorithm to find the correct echo and measure the distance with a better precision. To accomplish this, the input for the temperature in this view is done by a text field. Two slide bars let the user define the lower and upper bounds. The rest of the application is the same as with the original one described above. In this second view we just work without FFT.

In order to do accurate measurements a user has to make sure that as many as possible of the following facts hold.

- The item to measure the distance to has to be at least as big as one square metre and consist of a strong and solid area. Carpets, curtains and drapery of any kind potentially absorb sound too much.
- The phone has to point directly to the object to measure in a ninety degree angle.
- Try to hold the phone so that the microphone faces directly the object to measure.
- Use a cover (a cardboard or similar) to direct to sound output from the


AuDistMes

To measure the distance to the next wall, keep your phone in the direction of the wall and hit the measure button.

Set the actual temperature in Celsius as accurate as possible.
Set value: 22

Distance to the wall is:
1.406661224489796

Figure 3.2: A screen shot of the AuDistMes application.

speaker to the object to measure.

- There are as few disturbing noises as possible in the surroundings.

3.2.1 The Algorithm

The algorithm works as follows. First we need to produce a sound output that is strong enough to generate echoes from the wall. After a few experiments with alternating, edgy, strong and low signals we finally decided to use chirps as our sound output. These have the advantage that they are some sort of a sine curve. Therefore we do not have huge jumps from a low to a high value. In addition a chirp has an increasing frequency. Hence we do not just use one frequency in the signal, but multiple, which increases the quality of the echo. Once we have a constant sound output, we can now try to record the echoes. We can do so by accessing the built in microphone of the Nexus One and using the AudioRecord [4] class of Android. After initializing the recorder with the same frequency and audio formats as on the sound output, we are able to read one sample at a time. Depending on a

[4]See AudioRecord on developer.android.com

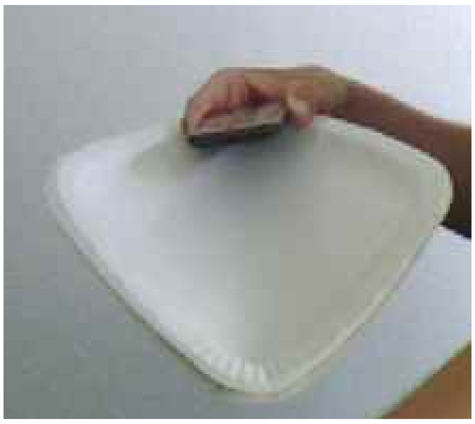


Figure 3.3: Set up for measuring the distance to an object front view .

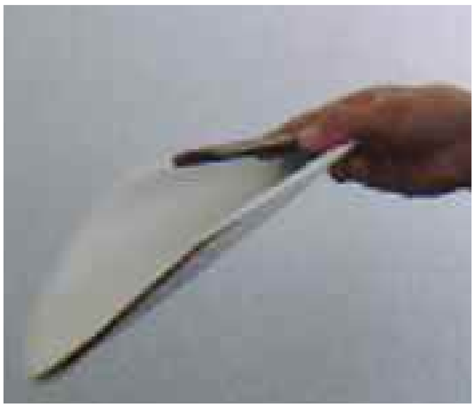


Figure 3.4: Set up for measuring the distance to an object side view.

flag in the code we can print the read sample to a file on the SD card for further edition on Matlab. Once we have the data from a record we have to determine the sent chirp and the echoes from different walls. Identifying these turned out to be the most difficult part of the entire application. We used several different approaches from simply calculating maximums in the given data up to running the fast Fourier transformation on the product of the chirp and the record.

Depending on which button the user presses, the current algorithm takes the readings and does some pre-processing. In case of using the fast Fourier transform, first the FFT of the sound output is calculated. Then we calculate the FFT of the listening. Finally the inverse FFT of the conjugate of the their product is calculated. For the fast Fourier transform we used an algorithm from the book "Introduction To Programming in Java"[9], which uses a radix two Cooley-Turkey algorithm. This is probably not the most efficient algorithm for memory usage, but it is solid and was easy to integrate to our project. The last step in pre-processing the data is, independent of using FFT, to filter the data by taking the absolute values of the entries.

The next step is to find the largest peak, recorded immediately when the sound is played on the phone. In other words we look for the peak coming without any reflections from the speaker directly to the microphone. If we found such a peak, which is high enough according to a threshold, we know that there have to be some echoes in the listening. We do search these echoes by maximizing the height of a newly peak minus the minimum value found in the data after the initial peak. Again this procedure is independent of using FFT on pre-processing.

When finally we have identified the chirp and its echoes in the data, we use the following simple function in order to calculate the distance from the phone to the wall. Here ϑ stands for the current temperature.

$$distance = \frac{(Peak\ of\ echo) - (Peak\ of\ chirp)}{(Frequency\ used)} s \ * \ (331.3 + (\vartheta * 0.606)) \frac{m}{s}$$

Finally, all we need to do is displaying this value to the screen and keep it together with the current compass value in mind for further processing. The angle between the North Pole and the current direction the phone is pointing to is received by a sensor listener on the orientation sensor. As described above the orientation is set by a new button separate to the actual measuring. When there have been enough distances collected, we can finally draw the room. The view then takes the calculated distances and angles and tries to estimate the shape of the room. To do so we calculate the direction of each point, take the distance into account and anticipate the walls. In the end all the calculated lines get drawn on the screen.

For the second view, where users have to estimate a lower and upper bound of the distance, we changed to above algorithm slightly. Pre-processing of the data and finding the initial peak of the sound played directly from the speaker to the microphone has been done exactly the same way as described above. For finding the echoes we just looped over the estimated region of the data instead of searching in the entire data. This reduced the execution time and increases the accuracy. For finding the echoes, again the same algorithm as described above was applied.

3.3 Experiments and Evaluation

During the development of the application countless experiments have been accomplished. Most of them were simple recordings of a sound output and its echo with the application itself. We measured the distance from the wall to a certain point in the room by a measuring tape and tried to find the according echo in the data. To do so, we let the application save the readings to files on the SD card. After loading those files to our laptops, we examined the curve using Matlab. Many different experiments have been done including studying different sound outputs, different approaches for reading the data, different types of calculating the distance between the sound output peak and the echo peak, using the fast Fourier transform and so on. The rich amount of variant experiments and data allowed us to fine tune our algorithm and to quickly identify falsely parts within it. As described before, the most difficult part of the algorithm was to determine which echo to use and to get the distance. Since not all readings actually contained any kind of peak, this was a hard task to do by hand just having the Matlab plots too. Figure 3.5 shows a Matlab plot where a echo is actually visible. An FFT output for another reading is shown in Figure 3.6.

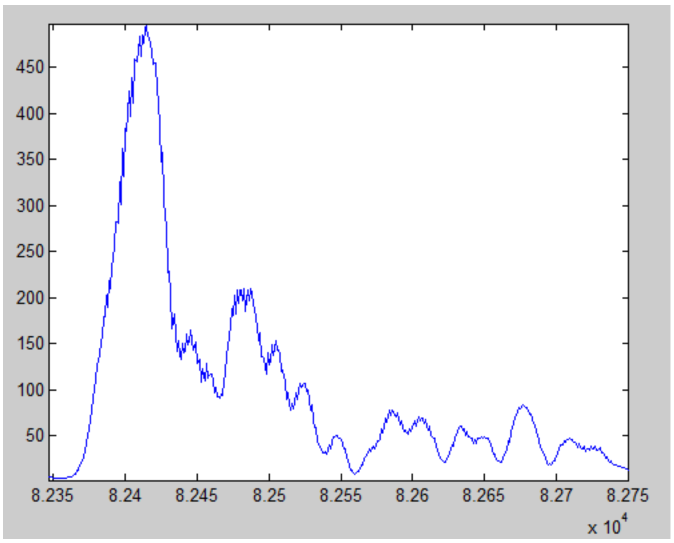


Figure 3.5: An output of a reading visualized. The strength of the signal is indicated on the y axis, the time on the x axis.

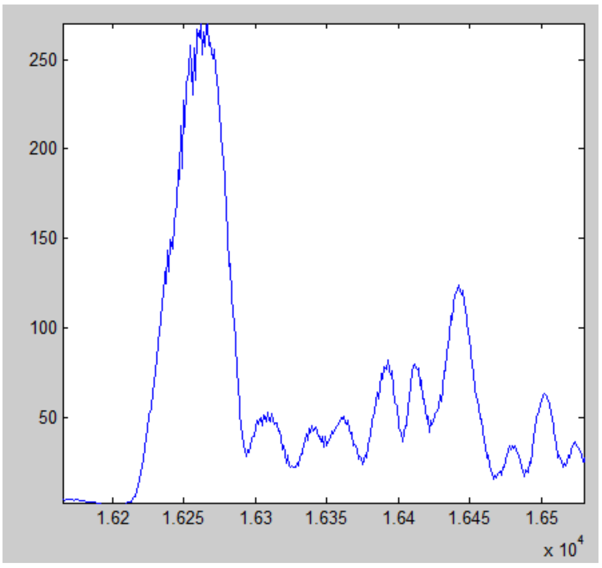


Figure 3.6: An output of a reading with FFT turned on. The strength of the signal is indicated on the y axis, the time on the x axis.

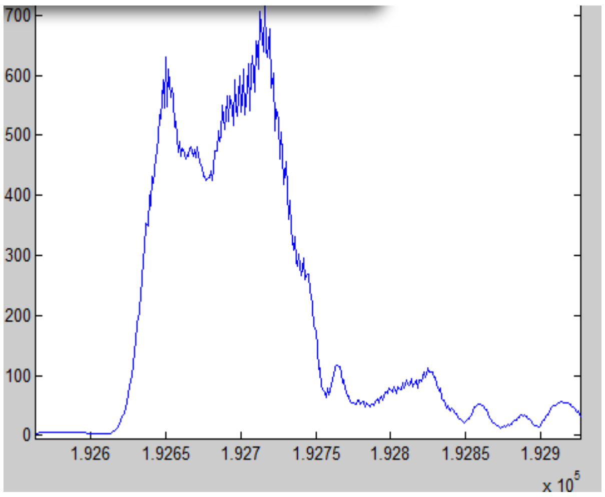


Figure 3.7: An output of a reading where peak and echo merge. The strength of the signal is indicated on the y axis, the time on the x axis.

Finally we have done two experiments described in the following. For both experiments we did measurements without FFT. This is, because FFT needs a lot of time for calculation and the results are not much better if at all.

In the first experiment we were facing a wall directly. Next to us we put a cord to physically measure the distance. Then we used our phone to measure using sound. If we are closer to the wall then approximately 50 centimetres, the phone has a hard time to measure the distance. The peaks of the direct sound output and the echo merge and hence the phone can barely identify any echoes. Such an output is shown in Figure 3.7. This is the reason for the big amount of ”Could not find an echo. Please try again” error messages during our tests. In the range of 50 - 200 centimetres we were able to achieve in 80% of the successful measurements within accuracy of 5 - 12 centimetres. For the other 20% the accuracy dropped to around 15 - 20 centimetres. In this region the error message hardly ever occurred. When we measure further apart from the wall than the two meters, the accuracy decreases as well. And again the phone is more often not able to find a sufficient enough echo.

During the second experiment we were in an office and measured the distance to each of the four walls from a given point with a cord. Then we used our application to do the same. All of the four to eight measurements in each run had an accuracy between 3 - 12 centimetres. One of the generated drawings for the room is shown on Figure 3.8. Figure 3.9 shows the original floor plan of the room. We can see that they look almost the same. There are edges on the left and the right hand side of Figure 3.8 close to the measurements. They exist because our algorithm tries to detect the walls and edges

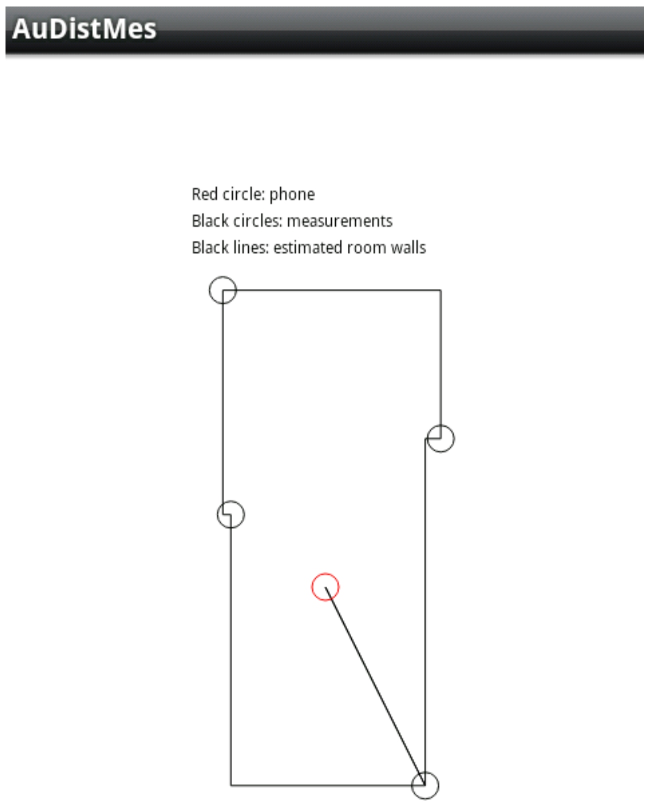

AuDistMes
Red circle: phone
Black circles: measurements
Black lines: estimated room walls

Figure 3.8: A screen shot of the drawn room by measuring all four walls of an office.

of a room instead of just connecting the measurements with a direct line.

According to our experiments we could measure the distance to the wall with an accuracy of about ten centimetres in around 80% of the measurements. The drawing of the room and hence the localization of the phone in the room did work but there is still room for improvement. The reason for this possibly is the fact, that the speaker of the Nexus One is fitted on the rear of the device, whereas the microphone is at the bottom. In order to play the sound output directly to the wall, we need to hold the phone in an upright position. This might lead to some confusions of the orientation sensor. This problem led to the separation of the measurement and the orientation, which is not really user friendly. Using a cover and pointing the microphone directly to the wall was one approach to solve the problem. But a user might not always have something in range to use as a cover. So again this solution is not very user friendly. The physical configuration of the speaker and the microphone might also be a source of errors in the measurements. We cannot point the speaker and the phone both to the wall at the same time. This

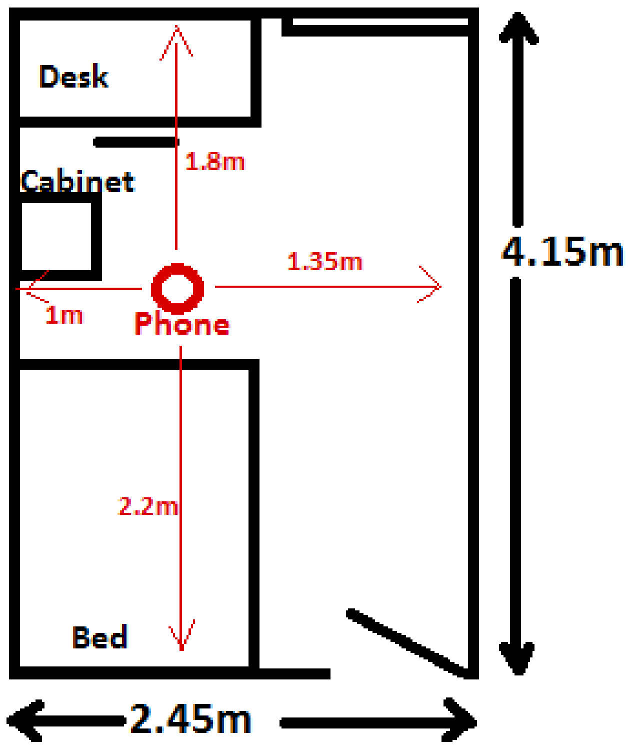


Figure 3.9: The floor plan of the room.

might be one reason, why our algorithm seems to perform worse than the three centimetres accuracy promised by the developers of the SonarRuler[5] IPhone application.

3.4 Conclusion

We knew right from beginning, thanks to the Sonar Ruler application for the IPhone[5], that measuring the distance using sound and echoes is possible, but whether perfectly accurate nor easy. Therefore we have laid our focus localization in a room. In the end we were able to deliver an algorithm that measures the distance to the wall with the desired accuracy. Distance measurements in the range of 50 centimetres to two meters are as precise as about ten centimetres. Closer to the wall and further apart it is hard for the algorithm to find the right echo to measure due to peak merging and noise. With the additional button for the orientation and a patient handling of the application the drawing of the room worked out as well. We had big issues on this part for a long time because of the physical circumstances on the phone.

Regarding indoor localization we can state that using sound is a real alternative. Nevertheless loud rooms lead to big noise signals and make measurements very hard. Moreover a user has to do quite an effort to get the localization data (do multiple measurements for sound and orientation, turn around, etc.). By improving the algorithm we can get a quite accurate localization method.

For future work we would be highly interested in possibilities for improving the accuracy of the distance measuring by sound and echo. Trying to filter out noise or improving on the method using the fast Fourier transform would be reasonable approaches. The more using a different approach for the sound output instead of clicking noises and chirp signals as we used them would be another interesting topic.

Chapter 4

WiFi/RSSI with Bluetooth for Detection of Obstacles

4.1 Introduction

In this chapter we describe how to detect obstacles between two phones using Bluetooth RSSI values. There are many applications, which may make use of an obstacle detection algorithm. To demonstrate this, we chose to implement a small shooter game. In this game two or more participants play against each other. The goal is to walk around in- or outdoor and try to shoot all the other players (In some shooter games this is called a deathmatch). When a player hits the shoot button the position of all the players is known by GPS or some indoor localization algorithms. In addition, the orientation sensor of the phones will provide us with the direction the player is pointing at. But how can we detect if the shot will hit the opponent or if there is a wall or another obstacle between the two participants? One idea is to use predefined maps. But are they up to date and accurate enough? What about moving obstacles? Another possibility is to use the built in camera. This approach is used for example to help blind people to detect abnormalities such as steps and obstacles on the floor.[7] Using such an approach the participants would have to hold there smart phone camera in front of them all the time. Like this, finding the direction with the orientation sensor gets quite hard. Therefore we try to come up with a solution based on Bluetooth RSSI values.

In order to obtain results, a small application called WIFI Shooter (The name is related to the planed shooter game) was developed. Using this application we made two experiments in different locations and using different obstacles.

In the experiments and implementation we seek for the Bluetooth received signal strength indicator (RSSI) value which indicates the power present in a received radio signal. We expect the range of the Bluetooth connection between the phones described below to be up to 20 meters. In this range we expect the signal to decay with about $\frac{1}{x^a}$ where we assume a to be in the set $\{2, 3, 4\}$. At the same time we expect the signal to noise ratio [1] to increase in the same range. Hence we hope to detect obstacles in at least a range of about ten meters between the two phones. This is needed in order to be able to play the game.

4.2 Application WIFI Shooter

As mentioned before, we used a Nexus one device running Android 2.2.2 for testing purposes. The application was developed for Android 2.2 (Build 8).

The WIFI Shooter application contains four different small views that are accessible through buttons in the main view.

- The so called WIFI_TEST, where one is able to search and compare all the WIFI networks in range.
- The similar BLUETOOTH_TEST view, where one can search and compare Bluetooth devices in range.
- The record view allows the user to record the RSSI values obtained from Bluetooth devices in range. The recorder values are saved to a text file on the SD card of the phone and can (if the phone has an active Internet connection) be displayed as a Google chart.
- The GSM_TEST is a small view that just displays the signal strength of the closest GSM antenna.

All the different views are closer described in the following subsections. For the evaluation and the detection of obstacles using Bluetooth RSSI values only the BLUETOOTH_TEST and record view was used. The two others were just used for comparison between WIFI and Bluetooth or GSM and Bluetooth respectively. The reason for this is that in order to detect obstacles between two phones the GSM signal strength is not usable. This is because there is no direct connection between the two phones. In the WIFI case we do have the problem that tethering is in fact possible with the Nexus one, but it is impossible to listen to another WIFI network while tethering. Possible we can detect obstacles from a phone to the tethering one. Assuming symmetric values it is also possible to send the obtained values back to the tethering phone. Hence we may find obstacles between a tethering phone and one which is connected to it. But this just works for two participants. As soon as we have more phones, which probably will be the case in our game example,

[1] For more details consider Wikipedia/Signal-to-noise-ratio

we would need to take another approach. As far as we know for Android 2.2 one would have to use the Android native API to better control tethering. Because the implementation for Bluetooth was way easier we decided to leave WIFI aside.

4.2.1 WIFI_TEST

On opening the WIFI_TEST view the users gets a list of the names of all WIFI networks in range. Additionally the user gets the signal strength RSSI values to each network. A pop-up message displays the name of the strongest network in range. For the scanning of the networks the android.net.wifi.WifiManager class is used. After initializing the manager and starting the scan we can obtain all the networks. Then all the data such as the SSID name of the network, the RSSI level in dBm and frequencies as well as capacities can be read using the ScanResult. The following listing shows the usage of the WIFIManger in the view.

```
 WifiManager wm = (WifiManager) getSystemService(WIFI_SERVICE);
wm.startScan();
 List<ScanResult> list = wm.getScanResults();
 // loop over all list items. list.get(i)
 // is the i-th network in the list
 list.get(i).SSID;
 list.get(i).level;
```

A refresh button gives the user the possibility to repeat the scan and to measure the RSSI values in different locations. The list gets cleared and rebuilt with each refresh.

4.2.2 BLUETOOTH_TEST

This view is very similar to the WIFI_TEST. On staring this view, the user is asked to allow the phone to activate Bluetooth and to make it visible for other phones for 300 seconds. The mobile phone keeps a list of paired devices, which can easily be accessed without scanning. Hence those can be displayed first using their name and MAC address. Unfortunately this does not provide us the RSSI value that we seek for. Hence we need to do a scan similar to the WIFI case. Doing so is a little bit more complex though and involves starting an Intent with an *ACTION_REQUEST_ENABLE* and registering a BroadcastReceiver for the IntentFilter *ACTION_FOUND* that signals a discovered remote device. Once this has been done, the actual discovery can be started. Whenever a new device is discovered the BroadcastReceiver adds the name and the RSSI value of the device to the list. The following listings show the code construct for the set up and the BroadcastReceiver.

```java
private BluetoothAdapter b_adapter =  BluetoothAdapter.
    getDefaultAdapter();
private IntentFilter filter = new IntentFilter(BluetoothDevice.
    ACTION_FOUND);
// enable Bluetooth
if(!b_adapter.isEnabled()){
    Intent bt_intent = new Intent(BluetoothAdapter.
        ACTION_REQUEST_ENABLE);
    startActivity(bt_intent);
}
// get paired devices
Set<BluetoothDevice> pairedDevices = b_adapter.getBondedDevices();
// Register the BroadcastReceiver
registerReceiver(mReceiver, filter);
if(b_adapter.startDiscovery()){
    // do something
}
b_adapter.cancelDiscovery();
```

```java
// Create a BroadcastReceiver for ACTION_FOUND
private final BroadcastReceiver mReceiver = new BroadcastReceiver()
    {
      public void onReceive(Context context, Intent intent) {
          String action = intent.getAction();
          // When discovery finds a device
          if (BluetoothDevice.ACTION_FOUND.equals(action)) {
              // Get the BluetoothDevice object from the Intent
              BluetoothDevice device = intent.getParcelableExtra(
                  BluetoothDevice.EXTRA_DEVICE);
              // Get the RSSI of the device
              short rssi=0;
              if(intent.hasExtra(BluetoothDevice.EXTRA_RSSI)){
                 rssi = intent.getShortExtra(BluetoothDevice.
                     EXTRA_RSSI, Short.MIN_VALUE);
              }
              // add it to the list ...
          }
      }
};
```

4.2.3 Record

The record view is the most important of the application since we use it the most. The functionality is, as its name suggests, to record obtained Bluetooth RSSI values to a text file for further evaluation purposes. The more, after a sequence has been recorded, it has the possibility to show the results in a chart using the Google Visualization API [2]. To start and stop recording one can just use the according buttons. The files get saved with a

[2]Google Visualization API

unique, increasing number in the DownloadsBluetooth_Recording directory of the SD card in the phone. From there one can load the files to a computer. Using Matlab one can produce a graph with the RSSI values on the y-axis and the time on the x-axis.

To obtain the Bluetooth RSSI values turned out to be way more complex than in the WIFI case. With the Android Bluetooth API it is only possible to obtain the RSSI value of a Bluetooth connection upon discovery of a device (at least this was the case with version 2.2 that we used all over the thesis). We had to start a new thread when the user starts recording. This thread will then repeatedly start a discovery, discover the devices and finally cancel the discovery again. The BroadcastReceiver saves the obtained RSSI values for each discovered device in a list. Finally when the user stops the recording the thread gets killed and the data gets saved to file. In the file the date, time and duration of the recording is saved together with, for each device found, the name, address and the collected RSSI values with a time stamp.

To get rid of the repeatedly scanning there would be another way to access the RSSI value of a connected device. But this is out of the SDK and involves native API as well as to write JNI wrapper code to our function in order to access the Bluez API the underlying Bluetooth framework in Android (and almost any Linux system). This information is taken from a blog on *stockoverflow.com*.[3] As we found a reasonable way to get the information needed, even though it is not a pretty one, we decided not to leave the Android SDK.

4.2.4 GSM_TEST

This view displays the signal strength one receives at any moment from your Carrier provider as a pop up message. Most of the code of this view was taken from a tutorial on [1] as mentioned in the code. The Telephony-Manager class and a PhoneStateListener are used to obtain the necessary information needed.

4.3 Experiments

For the evaluation and to get some observations we made two experiments. The first one was executed in an exercise room and tested the influence of a blackboard being between two mobile phones. In the second experiment we placed one phone on a stool in the middle of the living room and walked away through the corridor with the second phone. In both experiments we made several runs with different obstacles. The phones used in the experiments were a HTC Desire that was not moving and a Nexus One which

[3]Consider this website.

we moved around. Both phones were running Android 2.2, had Bluetooth enabled and were visible to one other. In the following we will describe the two experiments in detail.

4.3.1 Blackboard Experiment

As described above we made this experiment in an exercise room at the ETH.[4] We put the Desire phone close to the wall (approximately ten centimetres away) in the front of the room. On a first run we walked with the Nexus One from close to the Desire to the rear of the room with no obstacles in between. To record the Bluetooth RSSI values of the Desire on the Nexus One the record view was used. For the results obtained by this experiment we need to consider that, because the phone is close to the wall, many echoes are probably produced that may influence the results. In a second run we put a blackboard between the phone and the rest of the room. Since the blackboard is around 20 centimetres away from the wall, on top there is no obstacle. Nevertheless the direct line between the two phones contains an obstacle. We walked the same way back to the rear of the room, as in the first run. Finally a second blackboard was installed between the phone and the rest of the room. The second blackboard is about 30 centimetres away from the wall and the same problem as before occurs. Again we walked the same way as before. All three runs were made two times to avoid some possible special events to influence the results. The results of each run were saved on the SD card of the Nexus One as text files.

4.3.2 Living Room Experiment

For this experiment we put the Desire phone on a stool in the middle of the living room.[5] Then we walked out of the door and along the corridor of the flat (about 12 meters of length) with the recording Nexus One. All doors to other rooms as well as all windows where closed. In the following we describe the door between the living room and the corridor as door. Whenever the door is described as closed, we started the recording outside the door which makes the walk approximately two meters shorter. This time six runs were made with different obstacles[6]:

- Door is open. There are no obstacles and hence direct view between the two phones.
- Door is closed. There are no other obstacles introduced.
- Door is closed. There is a metal basket around the Desire phone as an additional obstacle.

[4]A map of the exercise room is available in the Appendix
[5]A map of the flat and a picture of the setting is available in the Appendix
[6]Pictures of the obstacles are available in the Appendix

- Door is open. There is a metal basket around the Desire phone.
- Door is closed. There is a plastic bowl around the Desire phone as an additional obstacle.
- Door is open. There is a plastic bowl around the Desire phone.

4.4 Evaluation

The results of the first experiment are shown in Figure 4.1. The figure shows all the runs done. The blue lines (record 5 & 6) show the result for direct view and no obstacles and the red and green ones (record 7 - 10) show the results for the case where one or two blackboards where present as obstacles. On the x-axis of the graph we see the time. Since we walked approximately 15 meters in about 30 seconds, this means that after 10 seconds (the 1 on the x-axis) the two phones were about five meters apart of one another and after 20 seconds (2 on the scale) until the end the two phones were approximately 10 - 12 meters apart.

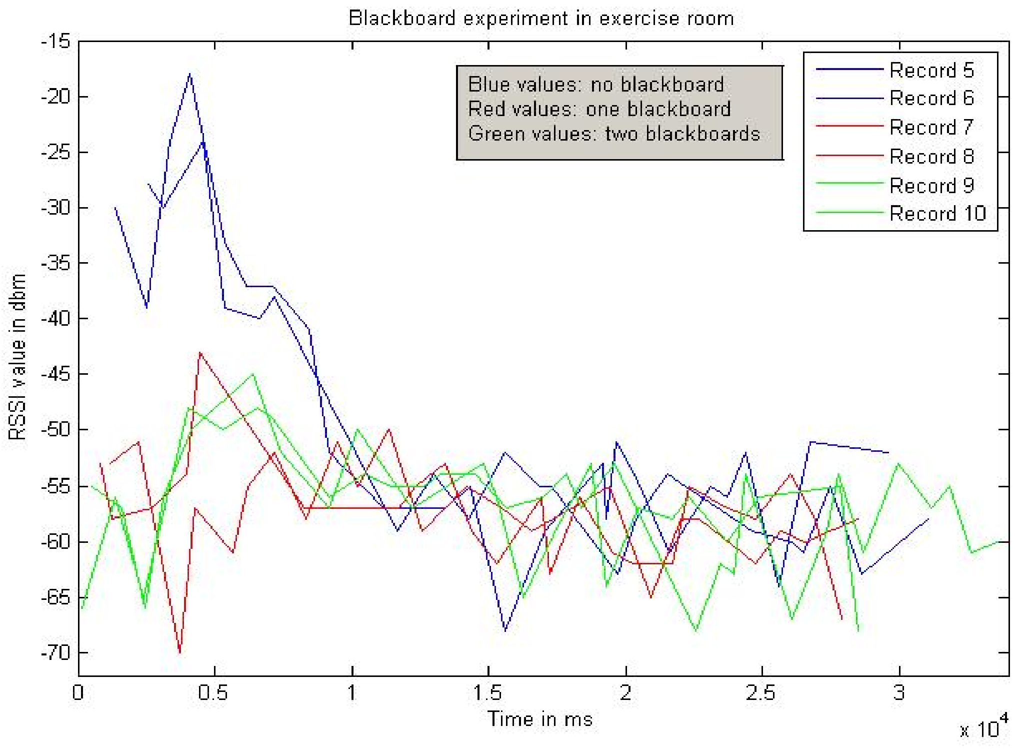


Figure 4.1: The evaluation of the first experiment for Bluetooth RSSI values

In this graph we can see that at first, within a range of approximately five meters, the obstacle can be clearly identified. If one considers the lower of the red curves, then there is even a small difference observable of having one or two blackboards as an obstacle. As soon as we walked further apart with the second phone the curves almost converge. The blue line stays highest during the entire recording period as expected, but the difference is quite small to the end.

For the second experiment in the flat the results are shown in Figure 4.2. Again the figure shows all the 12 runs made for the experiment. In the legend of the figure 'open' and 'closed' describe the state of the door between the living room and the corridor. 'No' means that there were no additional obstacles between the two phones, 'MB' describes that the metal basket was on top of the Desire phone and 'PB' that the plastic bowl was covering the Desire phone.

Looking at this graph, we can clearly see that the main influence in the curves is due to the door being open or closed. If we consider the case where no additional obstacles were introduced (the blue and the green curves), we see a difference of about 20 dBm. This would definitely be recognizable by a good algorithm. Furthermore we can observe that the plastic bowl was not at all recognized as an obstacle. The yellow curve compared to the blue one has almost stronger results. On the other side the metal basked had a bigger influence to the values in the case, where the door was open, at least when the distance between the phones was small. But if we compare the magenta curve with the green one, we can observe that in the case of a closed door, there is almost no difference at all. Here the closed door or wall between the two devices clearly has a higher impact on the results. If we consider the results, where the phones were further apart of one other, we again observe that the curves all in all tend to converge. Still the difference of the state of the door is recognizable if we compare the red and blue curve with the green and magenta one. This experiment shows that we can detect walls between the two phones even in distances between the phones of about 10 - 15 meters. But other obstacles such as the metal basket or a plastic bowl can just be detected in a range of around five meters.

4.5 Conclusion

The obtained results were not quite what we expected, since we were able to detect obstacles only in a range of about five meters distance between the two phones instead of the expected ten meters. When going further apart the values are more or less in the same range and the signal to noise ratio increases. This makes it very hard, if not to say impossible, to detect an obstacle with an algorithm. Since we initially wanted to demonstrate this detection of obstacles in a game with two or more moving participants, we have bigger distances most of the time and using Bluetooth seems to be inapplicable. WIFI definitely has a wider range. As we discussed in the WIFI_TEST application section without using the Android native API an implementation just works for two participants. Hence in our case also WIFI was not a good option. Still we claim that with some more research on the topic and possibly some other information about the location of the phone, one could realize an algorithm that detects, if two phones are in the same room or have any obstacles in between them.

For future work in this field we suggest to use a combination of WIFI RSSI values and the Bluetooth data to get the best results. Using some WIFI fingerprinting may help the algorithm to localize the phones within room accuracy. In a room the five meters for detecting obstacles may probably be sufficient. On the other hand an algorithm working with the Bluetooth RSSI values may also support other algorithms for indoor localization.

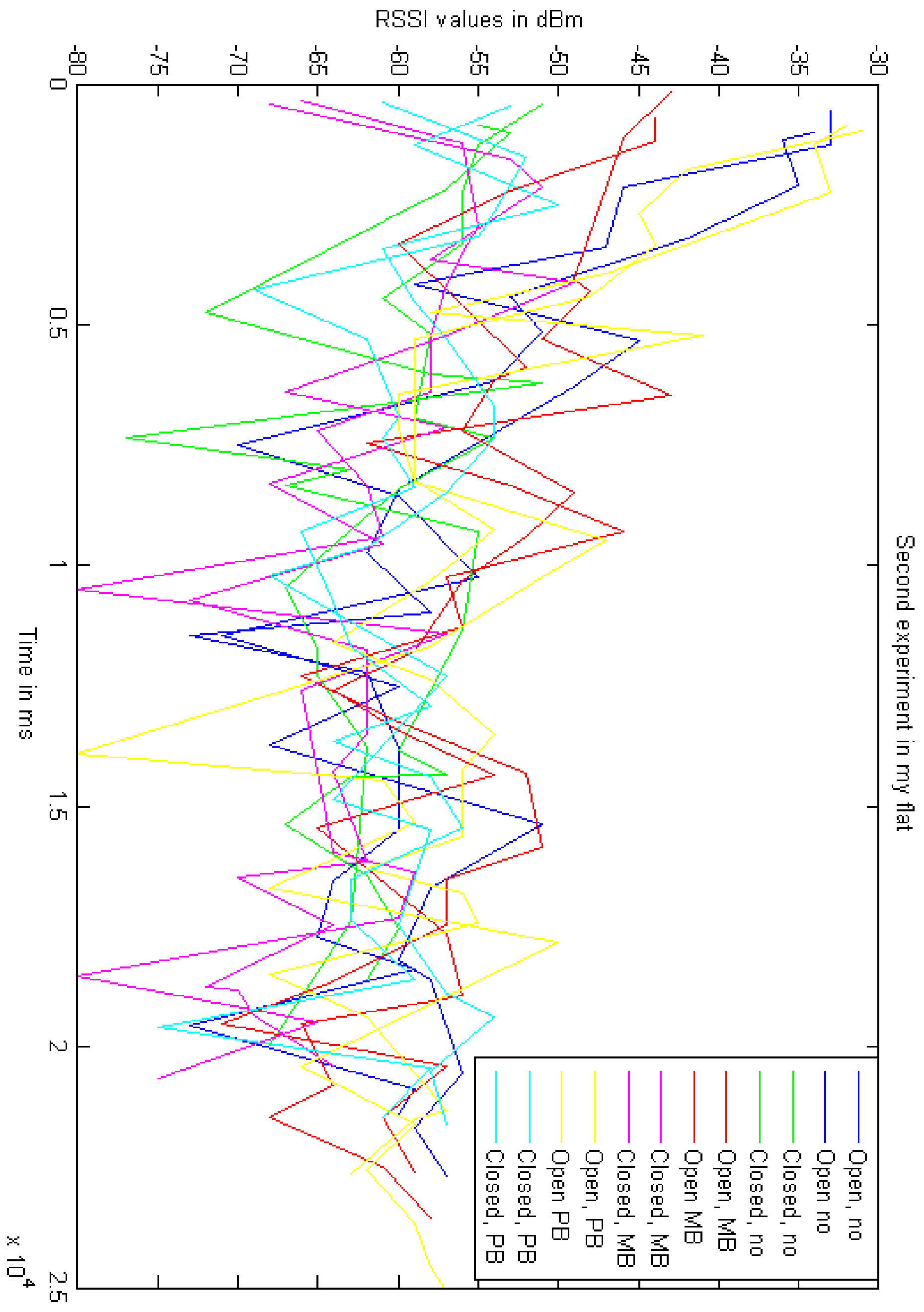


Figure 4.2: The evaluation of the second experiment for Bluetooth RSSI values

Chapter 5

Summary and Conclusion

In this thesis we have seen that indoor localization can be as accurate as less than five centimetres working in a magnetic field. Using test data and a nearest neighbor algorithm we can develop algorithms that fulfill exactly these requirements. But still there is a need for other algorithms to be able to better use such an approach in real-life applications. The more we approved the results of Ketabdar[4] and demonstrated as well, that around device interaction with magnets is possible even with today's hardware.

Using sound to measure the distance between a phone and a wall turned out to be harder as expected with the current hardware in the Nexus One. The different physical location of the speaker and the microphone makes it hard to get very accurate results. Nevertheless we were able to have an accuracy of about ten centimetres for around 80% of the measurements. The more we were able to draw a map of the room, the phone is in.

Finally we discovered that finding obstacles between two phones using Bluetooth signal strength values is possible in a range of about five meters. Going further apart it gets really hard to distinguish between an obstacle and just being far away from each other.

All together we can say that indoor localization is absolutely possible. There are many different ways for localizing one self within a room including WIFI/RSSI, magnetic field measurements, sound, Bluetooth/RSSI and many more. But the choice for the best method to use is highly dependent on the accuracy one is seeking for. Whereas magnetic field localization turned out to be very accurate up to a range of less than five centimetres, they are not really applicable in a large scale. Localization using sound is less accurate but can be used independent of other equipment such as WIFI routers or magnets. WIFI/RSSI fingerprinting is used a lot nowadays. Here an accuracy of meters is certainly possible but depending on the building and the amount of deployed WIFI routers the accuracy may go down to room level or below. The same goes for Bluetooth/RSSI. If one really needs a good and solid indoor localization algorithm we suppose that a combination of multiple methods would fit best.

Working with Android and the Nexus One had a lot of pros but also some cons. We really liked the fact that Android uses the Java programming language and provides tools for integration in Eclipse. This made development easy and smooth since Java is well known and there are plenty of tutorials for almost every problem that might occur. Google also provides a lot of demo material for starting to program with Android. As well on developer conferences like the Google IO new features get shared with the developers immediately and make their lives easier. Another fact, which we loved on working with Android, is the easy way to test applications either with the emulator on the PC or by just plug and play to the Nexus One. The more the debugging mode turned to be our biggest friend in finding bugs. This is also the reason for the many log messages in our code. The openness of Android really makes it easy to save and load any kind of files onto the SD card which we used for the Matlab plots and the like.

After praising Android that much we also like to mention some problems which we encountered during writing this thesis. We often missed a detailed description of some Android classes which turned out to be important for our applications. The lack of those documentations made it really hard to guess the right attributes sometimes. Let us make a small example which was not particularly important for our applications but shows perfectly our point. For the sound output using the MediaPlayer there is a method called setVolume(float leftVolume, float rightVolume). The documentation tells about balancing the output but there is no description of minimum and maximum values, some standard values and the like. So there is still a lot of space for improvements on the APIs. Then there was the problem described in the Bluetooth chapter. With the API for Android 2.2 it was not possible to continually get the RSSI values of a Bluetooth connection. We can say that for some details which might be quite important for developers there is still a lack on useful methods in some APIs.

We did really like the Nexus One device and are astonished on what is already possible on such small devices. Nevertheless we came quickly to its limits in the matter of CPU power as well as the limits of the built in sensors. It is still a phone and not a computer. As a programmer we should always be aware of this fact and try to make our programs faster and more efficient in terms of storage and CPU power.

Appendix A

Appendix

In this section we provide additional images that did not fit well into the text in the according chapter.

A.0.1 Map of exercise room

Figure A.1 shows a map of the exercise room that we used for our first Bluetooth experiment with the blackboards to determine obstacles between two mobile phones. The Desire phone that was not moving throughout the experiment is marked with a yellow box close to the right wall.

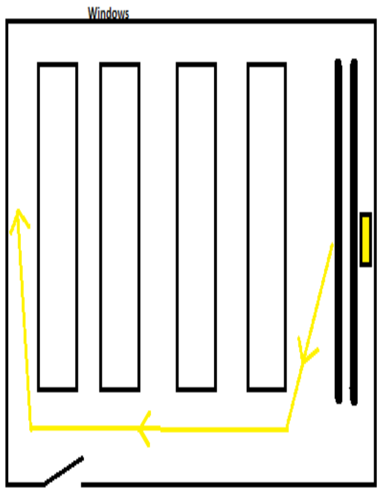


Figure A.1: The map of the first experiment in the exercise room

A.0.2 Map and images of the living room experiment

On Figure A.2 there is a map of one of our flats and Figures A.3 - A.5 show pictures of the obstacles used in the second experiment for the detection of obstacles using Bluetooth RSSI values.

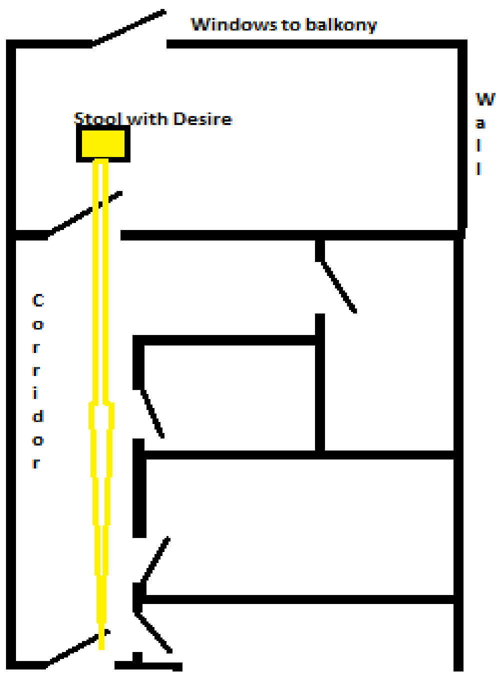


Figure A.2: The map of the flat used for the experiment.

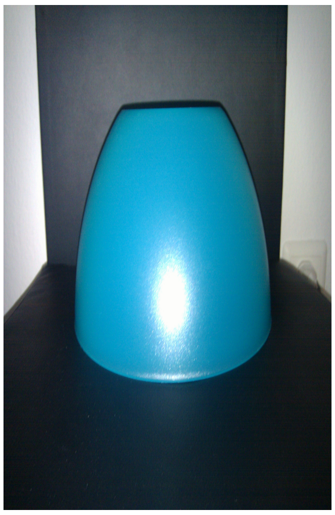


Figure A.3: The plastic bowl used as obstacle

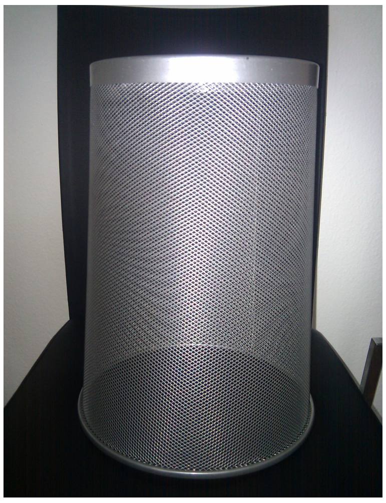


Figure A.4: The metal basked used as obstacle

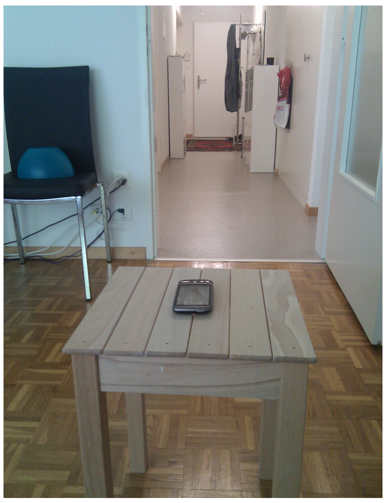


Figure A.5: The setting used for the experiment

A.0.3 Magnets in White-Board Eraser

Figures A.6 - A.7 show the magnets built in white-board erasers used for the magnetic field studies.

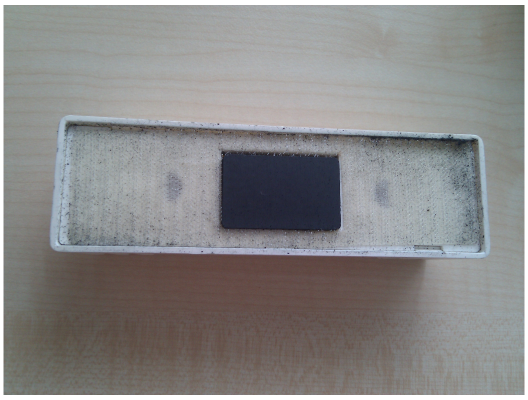


Figure A.6: LegaMaster white-board eraser.

Figure A.7: LegaMaster white-board eraser. Picture taken from www.legamaster.com

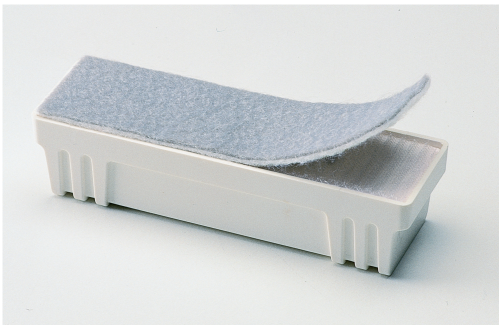


Figure A.8: LegaMaster white-board eraser. Picture taken from www.legamaster.com

A.0.4 Set-Up for the TicTacToe Game

Figure A.9 shows the set up for the tic tac toe game developed for the magnetic field sensor studies. On the left and on top one can see the two bar magnets built in white board erasers. The game board contains of nine fields ordered in a three by three manner. They are numbered from one to nine starting at the top left corner. The distance between the magnets and the board should be around 5 - 10 centimetres in order for the game to work properly.

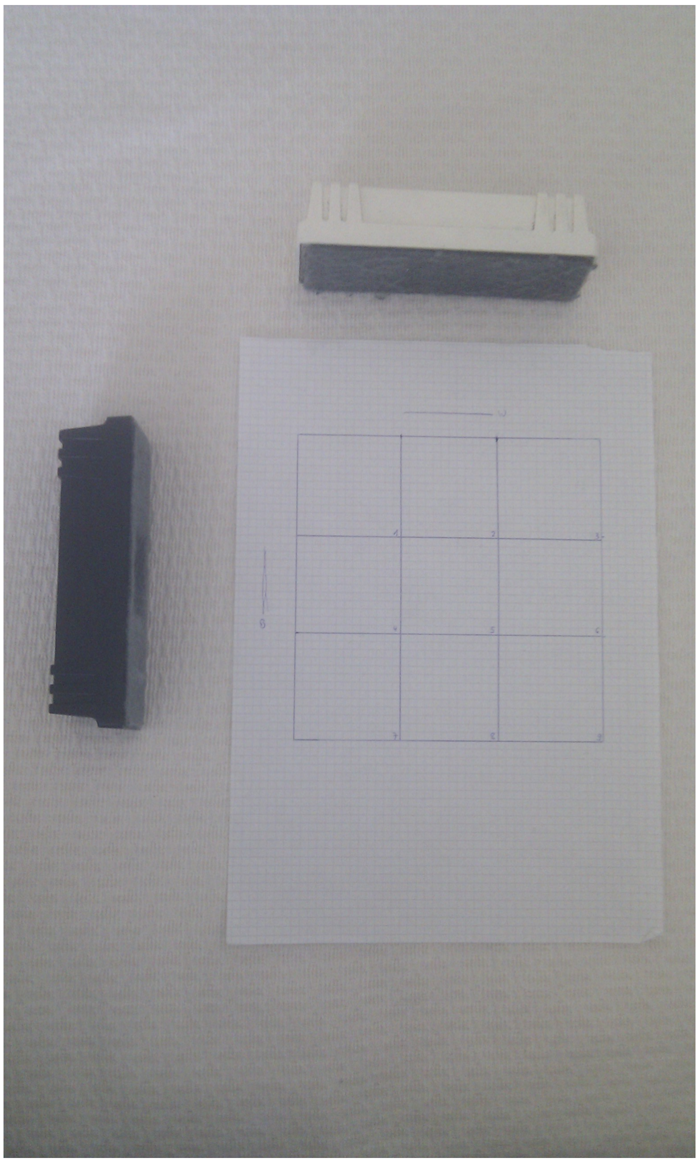


Figure A.9: The set up for the TicTacToe game

Bibliography

[1] George from Firstdroid tutorial. Get GSM Signal Strength Android Tutorials. http://www.firstdroid.com/2010/05/12/get-provider-gsm-signal-strength/, 2010. [Online; accessed 17-March-2011].

[2] Google. Android Sample Code. http://developer.android.com/resources/browser.html?tag=sample, 2007. [Online; last accessed 11-May-2011].

[3] HTC. HTC Flyer tablet. http://www.htc.com/www/product/flyer/specification.html, 2011. [Online; last accessed 11-May-2011].

[4] Hamed Ketabdar, Kamer Ali Yüksel, and Mehran Roshandel. *MagiTact: interaction with mobile devices based on compass (magnetic) sensor*. IUI '10. ACM, New York, NY, USA, 2010.

[5] LAANLABS. Sonar Ruler. http://labs.laan.com/wp/2009/08/sonar-ruler-iphone-app-measure-with-sound/, 2009. [Online; last accessed 29-June-2011].

[6] Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal de Lara. *Accurate GSM Indoor Localization*, volume 3660 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005.

[7] En Peng, Patrick Peursum, Ling Li, and Svetha Venkatesh. *A Smartphone-Based Obstacle Sensor for the Visually Impaired*, volume 6406 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010.

[8] F. Reichenbach and D. Timmermann. Indoor localization with low complexity in wireless sensor networks. In *Industrial Informatics, 2006 IEEE International Conference on*, pages 1018 –1023, aug. 2006.

[9] Kevin Wayne Robert Sedgewick. *9.7 Data Analysis*, volume 1. July 2007.