Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed
Computing

Path Tracking - Guiding Smartphones Based on Recorded Tracks

Group Thesis

Martin Ambühl
Nils Reinthaler

ambuehlm@student.ethz.ch
rnils@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zurich

Supervisors:
Tobias Langner, Jochen Seidel
Prof. Dr. Roger Wattenhofer

December 19, 2011

Abstract

The GPS signal can be used to determine the position of a navigation device in an outdoor setting. However, in buildings the GPS signal is often attenuated by the walls and the roof. Due to this, the signal strength is often too weak to obtain an accurate position or a position at all.
We are looking for an alternative method to guide a person in an indoor setting in this group thesis. For this purpose, we develop an Android application that is based on the information obtained from the accelerometer and the magnetic field sensor. The accelerometer measurements are used to detect and count steps and the magnetic field sensor serves as a compass. The aim of this thesis is to develop an application that doesn't rely on floor plans of buildings. In fact, with our application the user should be able to record routes and use already recorded routes to navigate to specific locations.

Acknowledgements

We would like to thank Prof. Dr. Roger Wattenhofer for offering us the opportunity to write this group thesis at the Distributed Computing Group.
Further we would like to thank our advisors Tobias Langer and Jochen Seidel for their support.

Contents

Abstract **i**

Acknowledgements **ii**

1 Introduction **1**

1.1 Related Work 1

2 Sensors **3**

2.1 Accelerometer 3

2.2 Magnetic Field Sensor 4

3 Processing Sensor Data **5**

3.1 Measuring Distances 5

3.1.1 Integration of the Accelerometer Data 5

3.1.2 Counting Steps 6

3.2 Determining the Orientation 14

3.2.1 Transforming Magnetic Field Sensor Measurements to Orientation Data 14

3.2.2 Relative vs. Absolute Azimuth Data 15

3.2.3 Detecting Turns and Calculating Their Sizes 16

4 From Distance and Orientation to a Route **19**

4.1 Recording a Route 19

4.2 Navigating Along a Prerecorded Route 20

5 Conclusion, Future Work and Outlook **21**

5.1 Conclusion 21

5.2 Future Work and Outlook 21

Bibliography **23**

A Application **A-1**

A.1 Code Structure A-1

A.2 User Interface A-3

Chapter 1

Introduction

Most modern cars are equipped with a navigation system that uses the global positioning system (GPS) to determine the position. These devices make it possible to find an arbitrary destination by providing route information to the driver of a car. However, as soon as the driver exits the car and enters a building, the GPS signal is often insufficient.
In this group thesis, we are looking for a method that does not rely on GPS signals to help a user to find a specific location inside a building. The aim is to develop a navigation solution, that is independent of maps and floor plans of buildings.
Modern smartphones include a multitude of different sensors. Two of those sensors, namely the accelerometer and the magnetic field sensor, are especially useful to determine the position of the user in an indoor setting. In this thesis, the information received from the accelerometer is used to detect steps made by the user. We then multiply the number of steps by the step length in order to get distances. But knowing only the distance that has been covered so far is not sufficient. Therefore, we use the magnetic field sensor in combination with the accelerometer to calculate the orientation of the smartphone.
To evaluate the quality of the aforementioned methods, we developed an Android application. In this application the distance measurements and the information about the orientation of the device are used to record routes and to navigate along prerecorded tracks.

1.1 Related Work

Several projects about indoor navigation already exist. Some of them rely on external map data. Ferial Shayeganfar and Amin Anjomshoaa and A Tjoa [1], for example, developed a smart indoor navigation solution that is based on Semantic Web technologies and a Building Information Model.
Other projects are based on wireless data networks. Suguna P. Subramanian, Jürgen Sommer, Stephen Schmitt and Wolfgang Rosenstiel [2] implemented a

system using bluetooth.
Furthermore Chin-Woo Tan and Sungsu Park [3] tried to build a navigation system making use of multiple accelerometers without the need of a gyroscope.
In contrast to the projects mentioned above, our goal is to implement an Android application for navigation that does not need GPS, external map information or dedicated hardware.
In our thesis, we first have a look at some of the different sensors a modern smartphone contains. Based on the sensor readings, we then develop a method to measure distances and determine the orientation of the phone without using the GPS. This information about the distance and the orientation is later on used to record routes and to navigate along prerecorded routes.

Chapter 2

Sensors

Modern smartphones include many different sensors. In order to determine the position of a smartphone in an indoor setting different sensors had to be taken into account, because no single sensor is accurate enough for this task. One of the sensors we analysed is the accelerometer.

2.1 Accelerometer

The accelerometer returns the actual acceleration of the smartphone splitted into the components x, y and z in the unit $\frac{m}{s^2}$. The x-axis points in the cross direction (from left to right) of the device, the y-axis points in the longitudinal direction and the z-axis is orthogonal to the display of the device (see figure 2.1).
The gravity measured by the smartphone points away from the Earth's center because the accelerometer experiences the same acceleration as if the device was

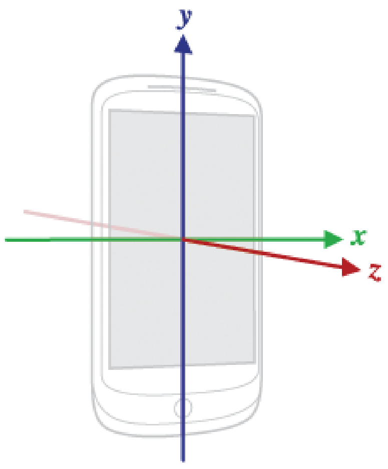

Figure 2.1: Device coordinate system
source: http://developer.android.com/reference/android/hardware/SensorEvent.html

accelerated towards the sky.
Technically, on our devices the acceleration is measured by an integrated circuit based on a spring-mass system. The springs and the mass are made of silicon. The mass forms a capacitor together with a reference point on the chip. The capacitance of this capacitor varies with the distance between the mass and the reference point and can be measured and converted to the acceleration.
The sampling rate of the accelerometer can be influenced by registering the listener with one of the following parameters: `SENSOR_DELAY_FASTEST`, `SENSOR_DELAY_GAME`, `SENSOR_DELAY_NORMAL` and `SENSOR_DELAY_UI`.
On the devices we used, `SENSOR_DELAY_FASTEST` corresponds to a sampling rate of about 50 ± 1 Hz. The other three parameters correspond to a sampling rate of about 25 ± 1 Hz.
Another sensor we used in our group thesis is the magnetic field sensor.

2.2 Magnetic Field Sensor

The magnetic field sensor measures the ambient magnetic field in the x-, y- and z-direction of the device (see figure 2.1) in μT. These measurements represent a vector that is tangential to the magnetic field line at the location of the device. The length of this vector represents the magnitude of the magnetic field at the specific location.
In order to measure the magnetic field in all directions, three orthogonal Hall sensors are used.
After this overview of the sensors, we describe how we process the sensor measurements in the next chapter.

Chapter 3

Processing Sensor Data

The raw sensor data mentioned above are not directly applicable for navigation. They need to be processed first in order to get measurements suitable for navigation. In this chapter we will discuss possible methods to process these sensor data. One quantity which has to be measured in order to determine a position is the covered distance.

3.1 Measuring Distances

To measure distances we use the accelerometer. The accelerometer provides raw acceleration data. These must be processed to obtain distance measures which then can be used for navigation. In the next subsection we would like to discuss the mathematical integration of the accelerometer data.

3.1.1 Integration of the Accelerometer Data

With an ideal accelerometer the speed ($\vec{v}$) of the device could be calculated by integrating the accelerometer data ($\vec{a}$) over time. Then the position ($\vec{x}$) of the device could be calculated by further integrating the speed over time.

$$\begin{aligned}\vec{v}(t) &= \int_0^t \vec{a}(\tau)d\tau \\ \vec{x}(t) &= \int_0^t \vec{v}(\tau')d\tau' \\ &= \int_0^t \int_0^{\tau'} \vec{a}(\tau)d\tau d\tau'\end{aligned}$$

But there are two problems with this approach:

1. The data of real accelerometers have small errors which are summed up when integrating. Subsequently, the already erroneous value of the speed must be integrated again to obtain the position, resulting in no longer usable data.
   For example, if the device is at rest, small errors are integrated resulting in a small value of speed. This small speed is interpreted as a change of position even if the device actually is not moving.

2. The problem is that the device can be arbitrarily oriented in space. However, for compensating the influence of gravity it is indispensable to know its direction. The direction of gravity cannot be determined exactly because the orientation of the device can only be calculated based upon accelerometer values. But only if the device is at rest, the accelerometer data can be used to determine the orientation of the device. As soon as the device is accelerated, the direction of gravity is not identifiable anymore.
   For example, if the device was positioned on a horizontal surface and accelerated parallel to it by $9.81\frac{m}{s^2}$, the x- and the z-axis would report an acceleration of $9.81\frac{m}{s^2}$.
   Given that constellation, it would not be possible to determine the orientation of the device, because the x- and the z-axes are ambiguous. The same measurements could be made if the device was laying on its side and was accelerated parallel to the z-axis.

Because of these two problems it is not possible to calculate the current position of the device by integrating the accelerometer values over time. With the use of a gyroscope the second problem could be solved, depending on its accuracy. However, the first problem would remain unchanged or even get worse because of the error imposed by the gyroscope. An error in the measurement of the orientation of the device results in an erroneous compensation of the influence of gravity. This further results in an increased error in the acceleration measurement. Regarding the first problem this increased error in the acceleration measurement results in an increased error of position.

3.1.2 Counting Steps

A more promising possibility for measuring distances is counting steps based on the readings of the accelerometer. By counting steps, distances cannot be measured exactly, but good estimates can be made.

Resultant Vector of the Accelerometer

Because the device can adopt an arbitrary pitch and roll, the individual axes are not bound to the world coordinate system (see figure 3.1). The gravity (g)

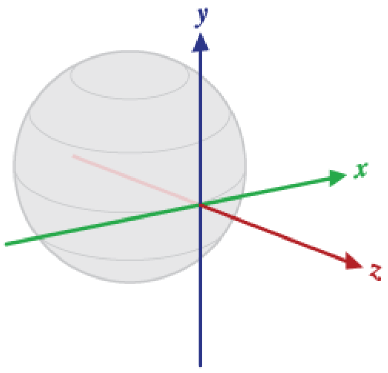

Figure 3.1: World coordinate system: x and y are both tangential to the ground at the device's current location. z is defined as the vector product of x and y. x points towards east. y points towards the magnetic north and z is perpendicular to the ground at the device's current location and points away from the center of the Earth.
source: http://developer.android.com/reference/android/hardware/SensorManager.html

could for example be mapped exclusively to the z-axis or as well to the y- and x-axis, depending on the alignment of the device. Because of this condition, steps cannot be measured based upon one particular component. We have to find a way to reduce the relevant acceleration to one quantity on which the calculations for counting steps could base upon. The relevant acceleration consists of the measured acceleration minus gravity.
The magnitude of the resultant vector is defined as: $|a_{res}| = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$ where a_x, a_y and a_z are the acceleration measurements in the x, y and z directions respectively. It would be most straightforward to use the magnitude of the resultant vector in order to reduce the three acceleration components to one quantity.
The main issue with this method is that the magnitude of the resultant vector primarily represents changes parallel to the direction of the gravity. Small changes in the acceleration ($\Delta a \ll g$) orthogonal to the gravity are only represented by a small percentage in the resultant vector (see figure 3.2) while changes in the acceleration parallel to the gravity are completely represented by the resultant vector (see figure 3.3). Furthermore, the influence of acceleration orthogonal to the gravity is nonlinearly depending on the magnitude of the orthogonal acceleration. This means that small Δa orthogonal to the gravity do not influence the magnitude of the resultant vector while for larger Δa the magnitude of the resultant vector is highly dependent on Δa. Therefore it is difficult to base the calculation on the magnitude of the resultant vector because the influence of acceleration orthogonal to the gravity is not known.
Another approach to eliminate the dependence of the direction of gravity we analyzed was to low pass filter the individual components. The three low pass

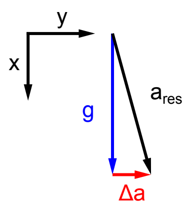

Figure 3.2: $|a_{res}|$ is not influenced by Δa for $\Delta a \ll g$. $|a_{res}| - |g| \approx 0$

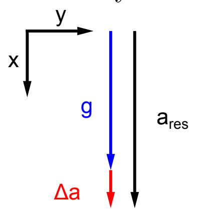

Figure 3.3: $|a_{res}|$ is directly influenced by Δa. $|a_{res}| - |g| = |\Delta a|$

filtered components represent gravity as long as the orientation of the device is not altered. As soon as the device is turned around an axis, the three low pass filtered components are delayed. The size of this delay can be influenced during the design of the low pass filter. If the delay is short enough it will not disturb the recognition of steps.
The three low pass filtered components can then be subtracted from the raw accelerometer readings. The outcome of this subtraction is a vector which represents the accelerometer readings without the gravity component.
If just the magnitude of the newly calculated vector is considered, a further problem is implicated, because the magnitude does not contain any information about the direction of the acceleration. This means, if the magnitude of the acceleration remained fixed at the same value but the direction of acceleration changed, this change could not be detected.
For example, if the device describes a circle in space without turning around the axes of the device (see figure 3.4), the vector of the measured acceleration describes a circle as well (see figure 3.5). However, this is not recognisable if only the magnitude of the resultant vector is observed, because the lengths of the vectors a_1 to a_{12} are the same. This means, that the device can accelerate in different directions without a change in the magnitude of the acceleration vector. Therefore, we need another method to reduce the three acceleration components to one quantity, without the loss of information about the direction of acceleration.

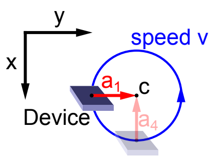

Figure 3.4: The device describes a circle around the centre c with speed v which implies an acceleration a towards c.

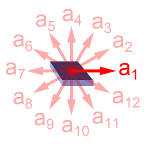

Figure 3.5: The acceleration experienced by the device describes a circle. $a(t_1) = a_1, a(t_2) = a_2, \ldots$

Turning the Coordinate System of the Device

As we can see in figure 3.6, steps are well-defined by the acceleration in the direction of the z-axis if the device is held horizontally. The x-axis shows the acceleration to the left and right side and the y-axis the acceleration to the front and back. The z-axis represents the acceleration upwards and downwards, which is the most important direction with regard to counting steps. The z-axis of the device corresponds to the z-axis of the world coordinate system if the device is held horizontally. If it is possible to determine the acceleration parallel to the z-axis of the world coordinate system (without influence of the horizontal acceleration) this would suffice to base the calculations for counting steps upon.

Thus our goal is to turn the z-axis of the device coordinate system so that it is parallel to the z-axis of the world coordinate system. On the Earths surface, gravity is always parallel to the direction of the z-axis of the world coordinate system. This fact can be used as a reference to the world coordinate system. To

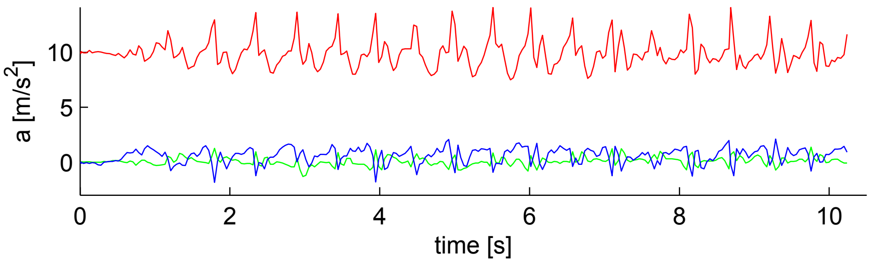

Figure 3.6: Acceleration components while walking. The device and the world coordinate system are congruent, because the device is held horizontally.
green: x-axis, blue: y-axis, red: z-axis.

obtain the gravity we filter the accelerometer readings componentwise with a low pass filter. The vector of these three low pass filtered components points in the same direction as the z-axis of the world coordinate system. We call this vector $\vec{g}$. With the rotation matrix $R_n(\phi)$ it is possible to turn the vector $\vec{a}$ around the axis represented by the vector $\vec{n}$ by the angle ϕ (see figure 3.7).

$$R_n(\phi) = \begin{pmatrix} \cos\phi + n_x^2 f(\phi) & n_x n_y f(\phi) - n_z \sin\phi & n_x n_z f(\phi) + n_y \sin\phi \\ n_x n_y f(\phi) + n_z \sin\phi & \cos\phi + n_y^2 f(\phi) & n_y n_z f(\phi) - n_x \sin\phi \\ n_x n_z f(\phi) - n_y \sin\phi & n_y n_z f(\phi) + n_x \sin\phi & \cos\phi + n_z^2 f(\phi) \end{pmatrix}$$

where

$$f(\phi) = (1 - \cos\phi)$$

We define

$$\vec{a}' := R_n(\phi) \cdot \vec{a},$$

where $\vec{a}'$ represents the vector $\vec{a}$ turned around the vector $\vec{n}$. The vector $\vec{n}$ must fulfill the condition $\|\vec{n}\| = 1$ for $\|\vec{a}\| = \|\vec{a}'\|$ to hold. Then, $\vec{n}$ can be calculated by

$$\vec{n} = \frac{\vec{g} \times \vec{e}_z}{\|\vec{g} \times \vec{e}_z\|}$$

where $\vec{e}_z$ is the unit vector of the z-axis of the world coordinate system. $\vec{g}$ represents the start of the turn and $\vec{e}_z$ the end of the turn. The turn angle is calculated by

$$\phi = \arccos\left(\frac{\vec{g} \cdot \vec{e}_z}{\|\vec{g}\| \cdot \|\vec{e}_z\|}\right).$$

Now the z-component of the turned acceleration measurement $\vec{a}'$ is parallel to the z-axis of the world coordinate system. From now on, we only need this z-component of the turned acceleration measurement called a'_z for counting steps. a'_z is not influenced by acceleration in the x- or y-direction of the world coordinate

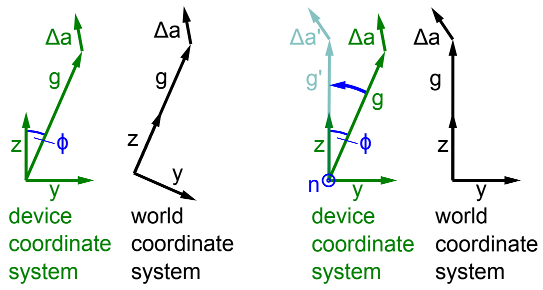

Figure 3.7: From the point of view of the device, the world coordinate system has an arbitrary rotation. The measured acceleration readings can be turned by ϕ so that the device coordinate system is congruent with the world coordinate system.

system. As can be seen in figure 3.6, the sampled data of a'_z are not yet suitable for step recognition. They first must be filtered. Because of the variety of the shape of steps it is not possible to apply a matched filter of one step. This means convolving the input data with a standard step is not useful, because the deviation of steps compared to a standard step can be too large.
Therefore we apply a band pass filter. A band pass filter consists of a passband and a lower and upper stopband (see figure 3.8). The lower stopband ranges from 0 Hz to F_{stop1}, the upper stopband ranges from F_{stop2} to $F_s/2$ and the passband ranges from F_{pass1} to F_{pass2}. F_s denotes the sampling rate of the signal. The Nyquist–Shannon sampling theorem states that the sampling rate has to be at least two times the maximal in the signal occurring frequency, in order to be able to reconstruct the sampled signal without loss of information. Therefore, only frequencies smaller than $F_s/2$ are present in a signal sampled with frequency F_s. A_{stop1} and A_{stop2} denote the attenuation of the lower and upper stopband respectively. A_{pass} denotes the attenuation of the passband. The lower stopband assures that low frequency influences (e.g. gravity) are filtered out while the upper stopband assures, that shaking and other disturbances are filtered out. The average frequency of steps is around 2 Hz. Therefore, 2 Hz must lay between F_{pass1} and F_{pass2}. To filter out most disturbances, the interval (F_{stop1}, F_{stop2}) should be as narrow as possible. However, if the passband was too small, steps would be filtered out as well. We did the fine-tuning by trial and error with the Filter Design & Analysis Tool (fdatool) of Matlab until the filtered $a'_z(t)$ was smooth enough to detect steps using a threshold.
A basic problem with filters is that the shorter the rising and falling edges are, the more coefficients the filter has. The more coefficients a filter has, the longer is the delay from a step to its recognition. To keep the number of filter coefficients as low as possible, we increase the interval of the rising and falling edges. Following

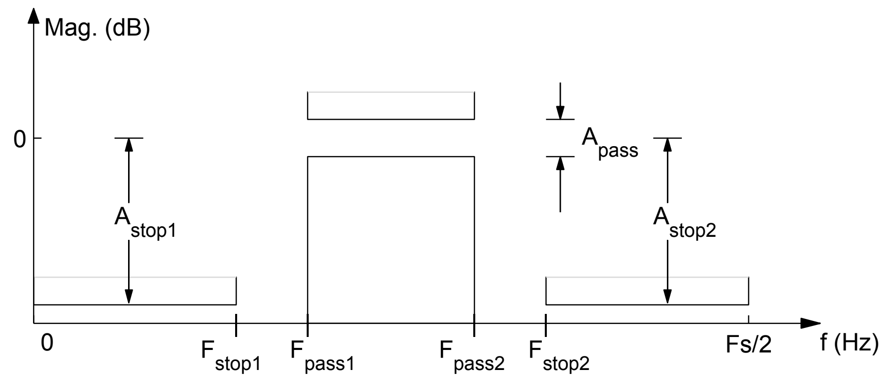

Figure 3.8: Explanation of the filter parameters.
source: Matlab Filter Design & Analysis Tool (fdatool)

filter parameters were found to fit our purpose best:

$$\begin{aligned} F_{stop1} &= 0.1 \text{ Hz} \\ F_{pass1} &= 2 \text{ Hz} \\ F_{pass2} &= 2 \text{ Hz} \\ F_{stop2} &= 4 \text{ Hz} \\ F_s &= 25 \text{ Hz} \\ A_{stop1} &= 80 \text{ dB} \\ A_{pass} &= 1 \text{ dB} \\ A_{stop2} &= 80 \text{ dB} \end{aligned}$$

These parameters result in the filter in figure 3.9.
Although the passband has size zero, steps can be recognised because the rising and falling edges are not abrupt. With this configuration we achieve a low number of filter coefficients.

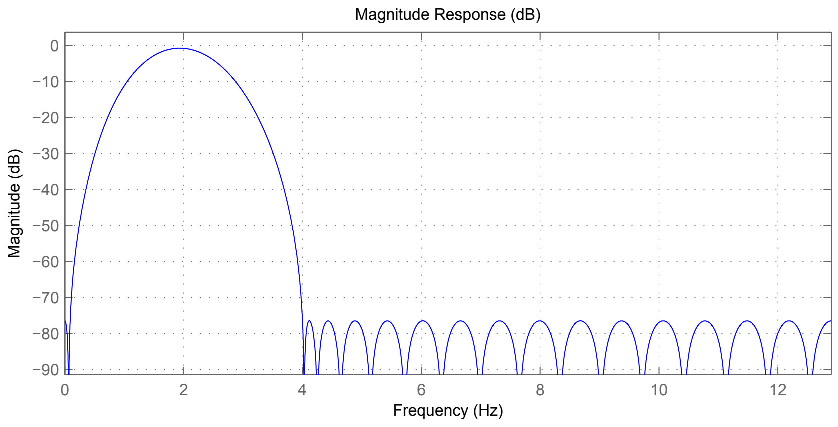

Figure 3.9: Band pass filter used for filtering the accelerometer data.

As the step frequency is not exceeding 3 Hz while walking normally, a sampling rate of about 6 Hz is sufficient according to the sampling theorem mentioned above.
Given that the three parameters `SENSOR_DELAY_GAME`, `SENSOR_DELAY_NORMAL` and `SENSOR_DELAY_UI` correspond on different devices to the same sampling rate of 25 Hz, we decided to register the listener of the accelerometer with the parameter `SENSOR_DELAY_GAME`. With this sampling rate, frequencies up to 12.5 Hz can be detected.
Filtering $a'_z(t)$ with the filter mentioned above results in the data in figure 3.10 where every sinusoidal wave represents a step. A step can be recognised with a threshold. Every time the threshold is crossed in positive direction a new step was made. To be immune to high-frequency shaking, we do not start counting steps until three steps are done within three seconds. If two steps are made more than two seconds apart, steps are no longer counted until three steps are made within three seconds again. This method to count steps is from now on called

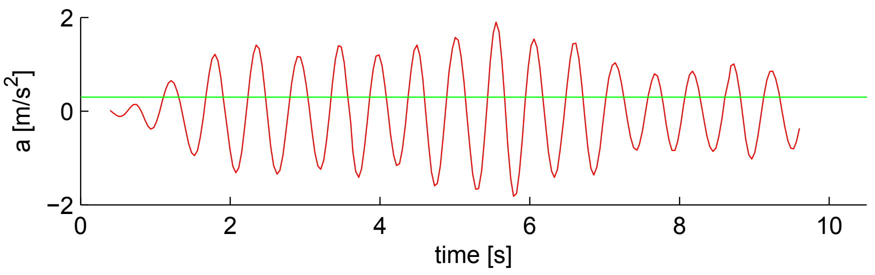

Figure 3.10: Accelerometer values shown in figure 3.6 (z-axis) filtered with the band pass filter shown in figure 3.9 with the threshold $\left(\text{in green at } 0.3\frac{m}{s^2}\right)$.

“Step Counter“.
To be able to calculate relative positions, not only the distance walked so far has to be known but also the direction in which this has been done. Therefore, the data received from the sensors have to be used to determine the orientation of the smartphone.

3.2 Determining the Orientation

The raw measurements of the magnetic field sensor are not yet suitable to derive the orientation of the device. First, the raw measurements of the magnetic field sensor have to be transformed in order to be independent of how the user is holding the device.

3.2.1 Transforming Magnetic Field Sensor Measurements to Orientation Data

The measurements from the accelerometer and the magnetic field sensor of the device are both returned in the coordinate system of the smartphone (see figure 2.1). A problem that arises with this is, that the sensor data depends on how the device is held in respect to the ground. We solve this problem by transforming the sensor data from the device coordinate system to the inverted world coordinate system (see figure 3.11).

In order to transform the sensor data, the raw data from the accelerometer and the magnetic field sensor are used to compute the rotation matrix and the inclination matrix. These matrices transform a vector from the device coordinate system to the world coordinate system. The rotation matrix is then used to calculate the device's orientation in respect to the inverted world coordinate system.

The device's orientation is defined by the following components: the azimuth (rotation of the device around the z axis), the pitch (rotation around the x axis) and the roll (rotation around the y axis). To determine the direction in which the user of the application is walking, the most important component is the azimuth

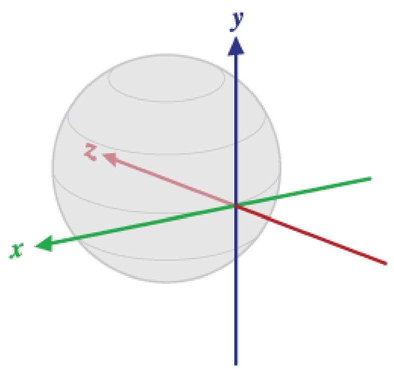

Figure 3.11: Inverted world coordinate system: x and y are both tangential to the ground at the device's current location. x is defined as the vector product of y and z. y points towards the magnetic north and z is perpendicular to the ground and points toward the center of the Earth.
source: http://developer.android.com/reference/android/hardware/SensorManager.html

due to the fact that it implements the same function as a normal compass.

3.2.2 Relative vs. Absolute Azimuth Data

There are two possibilities to determine the direction in which the user of the application is walking. The first possibility is to work with absolute azimuth data, i.e. the angle between the walking direction and magnetic north. The second possibility is to process relative azimuth data, i.e. the angle between the direction in which the user is currently walking and the direction in which he walked before the last turn. We will now give some rationale why we ultimately made the decision to process azimuth data in a relative manner.
As discussed in subsection 3.2.1 the magnetic field sensor measurements are transformed into orientation data. The most important component of the orientation data, in order to determine the orientation of the device, is the azimuth value which ranges from $-180°$ to $180°$ (see figure 3.12). It indicates the rotation around the z-axis of the inverted world coordinate system (with zero degree pointing in the direction of the magnetic north).
However, due to this $-180°$ to $180°$ discontinuity there would be a fundamental problem if only the azimuth value was considered to determine changes in the orientation of the device. If e.g. the user walked in the direction corresponding to an azimuth value of $170°$ and turned right $30°$ the new azimuth value would read $-160°$ (see figure 3.12) instead of $170° + 30° = 200°$, because of the $-180°$ to $180°$ discontinuity which was crossed in clockwise direction. This means, that the sense of direction would get lost (the data could be interpreted as if a $170° + 160° = 330°$ turn in the counter-clockwise direction was made).
We solve this problem by considering relative azimuth data. In the example above this would imply that instead of the direction before (170°) and after the

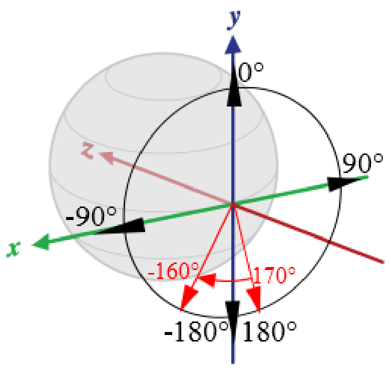

Figure 3.12: Boundary crossing example. The depicted coordinate system is the inverted world coordinate system. The circle represents the azimuth.
source: http://developer.android.com/reference/android/hardware/SensorManager.html
edited by: Nils Reinthaler

turn ($-160°$), only the size of the turn (in this case $+30°$) would be of interest. To calculate the size of a turn, it is important to be able to detect $-180°$ to $180°$ discontinuity crossings. This can be done by calculating the difference of the two newest azimuth samples. If the difference is greater than $180°$, the $-180°$ to $180°$ discontinuity must have been crossed.

Once a discontinuity crossing has been detected, the newest and all the following azimuth samples can be shifted by $360°$. Depending on the direction (clockwise/counter-clockwise) in which the discontinuity was crossed, the azimuth samples are shifted by $+360°$ or $-360°$ respectively. The effect of this shift can be seen in figure 3.13. Plot 1 shows the raw azimuth values plotted over time. To record this plot, the device has been oriented in the direction corresponding to an azimuth value of $85°$ for about the first 3 seconds. Then it has been turned $160°$ in clockwise direction and at second 10 it has been turned back to $85°$. The abrupt shifts from $+180°$ to $-180°$ and back due to the discontinuity crossing can be seen at second 4 and 11 respectively.

Plot 2 in figure 3.13 shows the azimuth data that have been shifted as described above. The shifted azimuth takes values that are greater than $180°$ and therefore cannot be used to determine absolute orientation values anymore. However, to calculate the size of a turn, which will be described in more detail in the following subsection, the shifted azimuth values are ideal.

3.2.3 Detecting Turns and Calculating Their Sizes

After solving the problem with the $-180°$ to $180°$ discontinuity, the shifted azimuth data can be used to detect turns and to calculate their sizes.

First, we lowpass filter the data in order to reduce small disturbances by calculating the average over the last 20 shifted azimuth samples. The result of this can be seen in plot 3 in figure 3.13. To detect turns and calculate their sizes, only the lowpassfiltered shifted azimuth data, from now on called "filtered samples", were considered.

If a user starts walking, his initial walking direction is saved as the direction, in which he last walked straight (see figure 3.14). All the following filtered samples are compared to the last straight direction. If one sample differs by more than $30°$ from the last straight direction, the beginning of a turn is detected. To determine the end of a turn, for each new sample, the difference between the newest and the last sample is calculated. If two samples lie in between a previously specified boundary of constant size, the end of a turn is detected (see figure 3.15). Subsequently, the size of the turn can be found by calculating the difference between the direction right after the turn (which is also the new straight direction of the user) and the last straight direction. If the user turned to the right, this difference is positive, otherwise it is negative.

Therefore, all the information about the orientation of the device is ultimately simplified to the moment, the size and the direction of turns. This can be seen on plot 4 in figure 3.13. The first peak in the plot represents a turn to the right of about 160° (illustrated by the peak height) and the second peak a turn back to the initial direction.

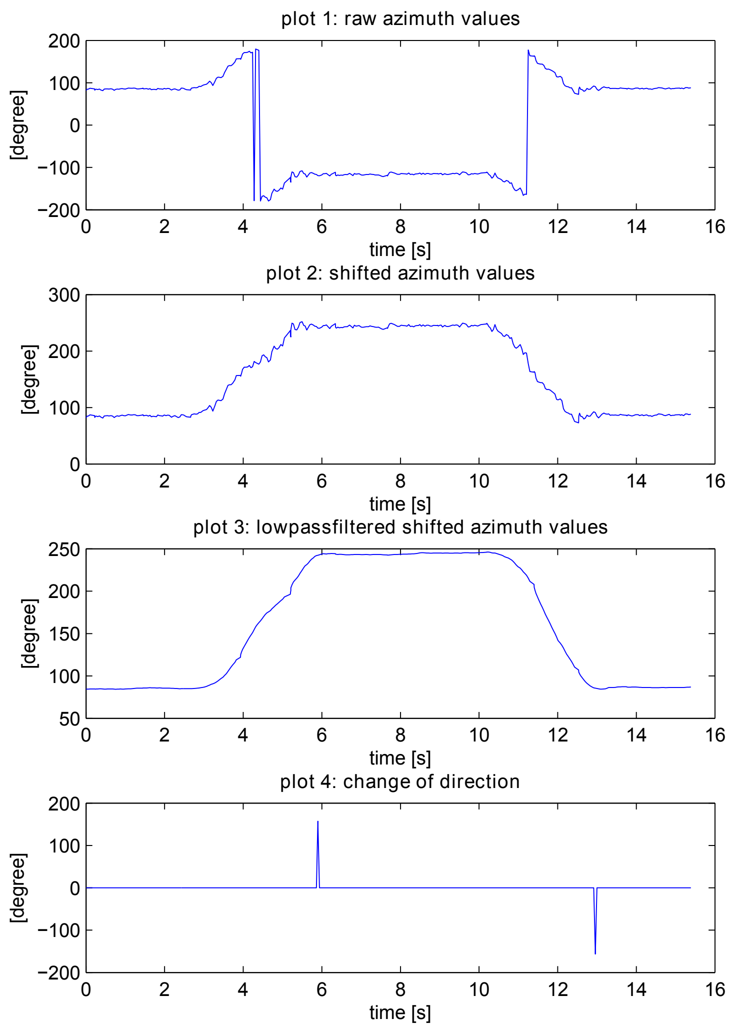

Figure 3.13: Different plots of azimuth data and direction change

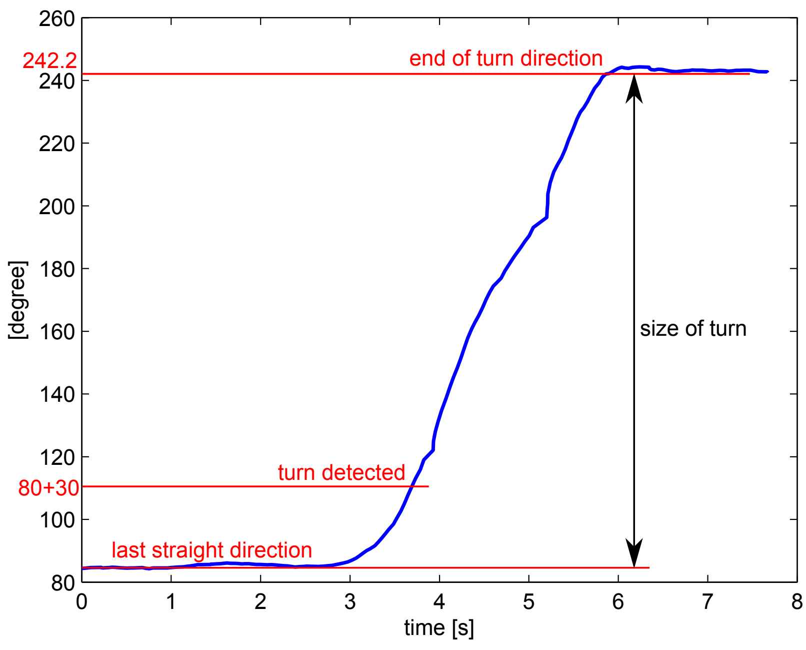

Figure 3.14: Example of a right turn (filtered shifted azimuth data)

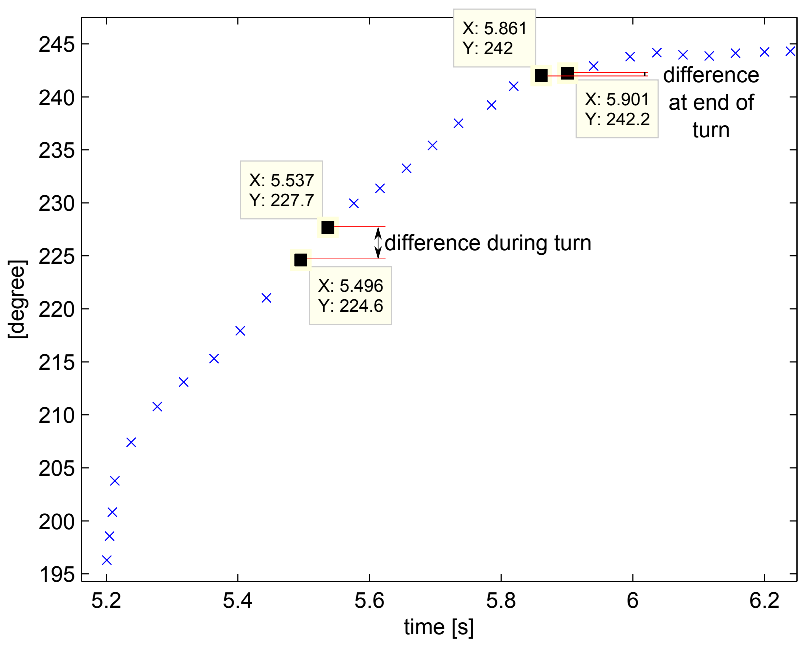

Figure 3.15: End of turn example (same data as in figure 3.14 with data points)

Chapter 4

From Distance and Orientation to a Route

With the information about the walked distance and the change of direction we can now derive the position of the user. By knowing the position of the user at every instant we are able to record a route.

4.1 Recording a Route

A route is subdivided into straight parts and turns. Each straight part followed by a turn is handled as a 2-tuple whereas the straight part is measured in meters (m) and the turn in degrees ($^\circ$). The length of a straight part is the distance between the end of the last turn and the beginning of the next turn.
We decided to measure the distance between two turns in this way because we do not know the exact form of a turn. He could stop walking and then turn or he could as well turn while walking. With the first possibility the user turns at a well-defined position while with the second possibility the way described by the user is a curve. The difference in the distance is for small curves small (less than one step). Larger curves are subdivided into several turns because the turn speed is too small and therefore an end of turn is detected while turning.
During the process of recording a route, these tuples are saved in a file on the SD-card of the smartphone. A route is completely represented by a collection of tuples.

distance [m]	turn size [$^\circ$]
4.0	75.0
6.4	-34.4
3.2	120.6
8.0	60.0

Table 4.1: Example of a route

Now that we are able to record routes, these routes can be used to give directions to the user.

4.2 Navigating Along a Prerecorded Route

To navigate along a prerecorded route the file containing the route information has to be interpreted.
A prerecorded route is imported and processed tuple after tuple. Once the user starts navigating the first tuple serves as instruction set whereas the distance is divided by the step length of the navigating user. Furthermore, positive angles corresponds to right turns and negative angles to left turns. This means that e.g. the tuple $(6.4, -34.4)$ would be translated into the instruction set "8 steps straight, then turn left". This instruction is updated for each step of the user. If the user starts turning the covered distance from the end of the last turn is compared to the distance entry of the tuple. If the difference between the two values is smaller than a fixed distance tolerance the moment of turn is accepted. After the turn is completed and if the moment of turn was accepted the size and direction of the turn are compared to the turn entry of the tuple. If the difference between these two values is smaller than a fixed turn tolerance and the direction of the turn is equal to the entry of the tuple, the turn is accepted and the next tuple is considered. Otherwise the user has to return to the start.

Chapter 5

Conclusion, Future Work and Outlook

5.1 Conclusion

During this group thesis, a reliable way to count steps has been found. Combined with the information about the size and moment of turns the basics are covered in order to be able to record and navigate along a route.
Due to the fact that the measurement of distances is relative and the application is based on turn sizes instead of absolute azimuth data, a well-defined departure point and bearing are required. At the current state of the application, there is no way to instruct the user on how to return to the route if the prerecorded route was left. While it is possible to navigate on a single floor of a building and on stairs, stairs are not recognised as such.
A major issue that was discovered during development is that the magnetic field sensor measurements are heavily influenced by external disturbances such as power lines or elevators. We tried to compensate for these local disturbances of the magnetic field by using the accelerometer data.
The acceleration measurements in the direction of the x-axis of the device (see figure 2.1) have been analyzed to check if this information could be used to detect if changes in the magnetic field sensor measurements are due to external disturbances or due to actual changes in the orientation of the device.
Unfortunately, the changes in the acceleration data measurements during an actual turn of the device were too small to be detectable.

5.2 Future Work and Outlook

To improve the performance of our application, a gyroscope could be used to compensate disturbances of the magnetic field sensor. Gyroscopes are as of today only available in a few but increasing number of smartphones.
Furthermore, to increase the user-friendliness, a new temporary route could be

recorded if the prerecorded route was left, in order to give instructions to the user on how to return. To further simplify the handling, spoken directions could be used. Additionally, to offer navigation over different floors, recognition of stairs could be implemented or the user interface could be adapted to allow the user to add additional information about a route. This additional information could involve stairs and the identification of the current floor. Ultimately, it may be possible to use our findings to compensate for weak GPS signal reception and therefore improve navigation system performance.

Bibliography

[1] Shayeganfar, F., Anjomshoaa, A., Tjoa, A.: A smart indoor navigation solution based on building information model and google android. In: Computers Helping People with Special Needs. (2008)

[2] Subramanian, S.P., Sommer, J., Schmitt, S., Rosenstiel, W.: Sbil: Scalable indoor localization and navigation service. (December 2007)

[3] Tan, C.W., Park, S.: Design of accelerometer-based inertial navigation systems. (December 2005)

Appendix A

Application

A.1 Code Structure

The only activity of our application is called “PathFinder“. This class manages the user interface and handles recording and navigation. “PathFinder“ is based on the classes “Orientation“, “StepCounter“, “ManageFile“ and “RouteDrawer“. (see figure A.1)
The class “Orientation“ calculates turn information needed by “PathFinder“ based on the magnetic field sensor and the accelerometer. For the communication with the activity “PathFinder“, “Orientation“ implements a first listener which is called every time the start of a turn is detected and a second listener which is called every time the end of a turn is detected.
The class “StepCounter“ recognises steps based on accelerometer readings. To determine the direction of gravity and to turn the coordinate system accordingly, “StepCounter“ needs the classes “GExtractor“ and “RotateVector“ respectively. To handle filter coefficients used to filter the accelerometer readings, the class “Filter“ is used. To apply the filter to the accelerometer readings “StepCounter“ uses the class “InnerProduct“. The class “GExtractor“ applies a low pass filter to the accelerometer readings in order to determine the direction of gravity. The class “RotateVector“ turns a passed vector according to the descriptions in section 3.1.2 (Turning the Coordinate System of the Device). The class “Filter“ simplifies the handling of filter coefficients and the class “InnerProduct“ calculates the inner product of two passed mathematical functions.
The class “ManageFile“ manages the file access of files used to store route information.
The class “RouteDrawer“ uses the methods provided by “DrawView“ to draw the current route. The class “DrawView“ extends the class “View“ from the package “android.view“ and provides methods to draw on the display of the device.

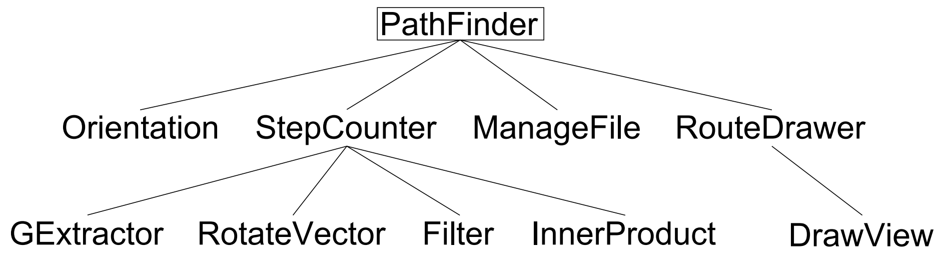

Figure A.1: Structure of the “Path Finder“ application.

A.2 User Interface

The user interface consists of three buttons, the text and graphic information for navigation and the menu. (see figure A.2)
By pressing the “record button“ the recording of a new route will be started. By pressing this button again, the walked route will be saved on the SD-card. With the button “select route“ a window can be opened where the user can select a route for navigation. If a route is selected, the caption of this button shows the name of this route and the route will be drawn in the “graphical route information“ section. The navigation can then be started by pressing the “navigate button“. While navigating, the user is guided by information in the “text output“ and the “graphical route information“, where the blue triangle represents the current position.
By pressing the menu button of the phone the “menu“ shows up. A route can then be deleted by pressing the button “Delete Route“. To calibrate the step length the user can press the button “Calibrate Step Length“. A window with the information for calibration of the step length is displayed subsequently. The calibration of the step length can then be accepted or canceled.

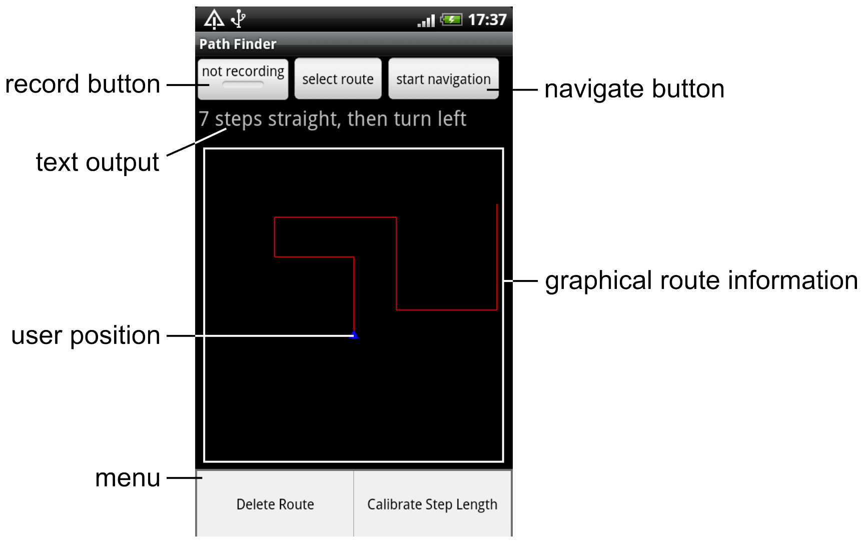

Figure A.2: User interface of the “Path Finder“ application.