Context Resolving using accelerometer and non-GPS location data

Peter Frøkjær Strand, DTU Informatics

December 1st, 2010

Abstract

An mobile application is designed to detect what the person carrying a device is doing - walking, running or on board a moving vehicle. It is shown that basic gait detection can be performed using Fourier transformation on data collected from accelerometers, without putting any restrictions on how the device is carried or who is carrying the device. Also, it is shown that it is - in some cases - possible to determine how fast a user is moving without using the GPS signal. Finally, a method to map accelerometer and location data to a generic movement context is described.

Contents

1 Introduction

1.1 Motivation and goal

The general motivation for this project is to explore what a modern mobile device can tell about its user, simply from using the on-device sensor systems. Today, a typical, modern smartphone will have at least the following sensor capabilities

- Camera
- microphone
- proximity sensor
- accelerometer
- magnet field sensor
- orientation sensor (compass)
- light sensor
- GPS

In addition to the sensors, a wide range of API's to query the phone state are typically available.

- Call state (is the user talking on the phone)
- incoming messages (SMS, bluetooth, mail)
- location services (location using WIFI and cell tower triangulation)
- WIFI and cell information
- etc.

It is clear that these sensors and services can provide detailed information about a users actions, intentions and whereabouts. For this project, it has been chosen to focus on gait detection - ie. is the bearer of the device idle, walking or running. In addition, it will also explored if it possible to detect if a user is biking or on board a moving vehicle. As it will be described, gait detection can be achieved by using data from the accelerometers, while vehicle detection require location information. Thus, focus will be on analyzing the data from the accelerometers and from the location services.

A primary goal for this project is to develop a working mobile application, and analyze its output. This application should work without user interaction (ie. as a background service) and should track the user whenever he is using the device. The two major requirements are

- There are no restrictions on how the device is carried.
  - the application should work while the device is placed in a backpack or in a pocket.
- No calibration is required
  - the application should work regardless of who the user is or how the user is moving

1.2 Related work

This field is currently seeing significant interest, both from the scientific communities and commercial entities. Nokia has recently announced Project Jigsaw [3], and Intel is also pushing the idea of context aware devices [4].

Gait detection is often used with video input, which is not relevant for this project. A project based on accelerometers is described in [5], where a device is strapped to a specific point to the leg. This accelerometer data is then analyzed with the intent of being able to identify the person by her gait. [6] is a brief overview of the different techniques for analyzing accelerometer data.

Details on how the currently available location services work is sparse, most likely due to their commercial nature. Wikipedia has a article on the subject [7].

1.3 Tools and Hardware

It has been chosen to use Android as a platform for this project. This platform has the advantage of being capable of running background services, which is - from a usability perspective - very handy. Also, the author is familiar with this platform and has access to a relatively modern device. For mapping and graphs, Google Maps and Google Chart is being used. In terms of hardware used, a Nexus One running Android 2.2 has been used. Accelerometer and location data has been collected by this device only.

2 Sensor analysis

2.1 Accelerometers

A perfect accelerometer would enable us to measure speed, as the speed is a function of acceleration. Unfortunately, the accelerometers used in consumer devices are not perfect, and the quality of the data they supply is low. As described in [2], calculation speed from accelerometer data while sitting in an starting and stopping train is not feasible.

Furthermore, as the requirement for this application is that the device can be used and carried without restrictions while measuring, additional acceleration noise must be expected. One example could be that the user, while in transit, takes the phone up from the pocket, types a text message, and places the phone in a backpack when done. Measuring accelerations from the train he would be sitting in would require very accurate sensors and a lot of data analysis to filter out the movements not related to the trains acceleration.

An accelerometer will measure the device acceleration relative to free fall, meaning that the gravity of the earth will cause a resting accelerometer to indicate 1G upwards. Values are given for x, y and z, so to calculate actual acceleration the gravity effect must be included in the calculations. For the purpose of this project, the individual axes are not relevant, as the exact placement and handling of the device is unknown and can change during measurement. Instead, the total change of force affecting the device at a given time can be calculated as shown in figure 1.

$$energy_t = |x_t - x_{t-1}| + |y_t - y_{t-1}| + |z_t - z_{t-1}|$$

where x_t, y_t and z_t are the raw sensor values at time t

Figure 1: *energy summation*

2.1.1 Data sampling

Running and walking An example of an adult person walking with the device loosely placed in a chest pocket is shown in figure 2. The sample rate is 256 samples pr. 14 seconds, ie. around 18hz, which has proved sufficient for this purpose.

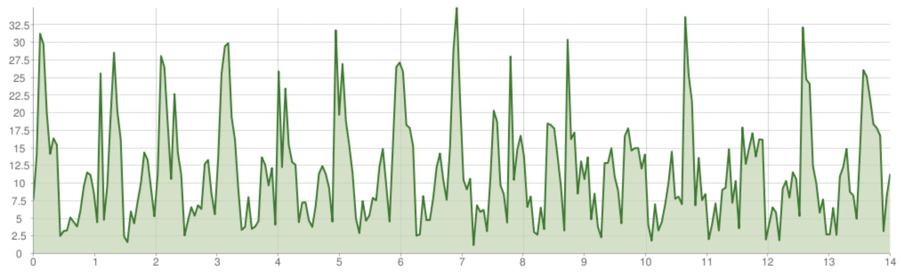

Figure 2: *Energy graph, walking (x-axis is seconds, y-axis is energy)*

Another set of data shows an adult person running - shown in figure 3. As expected, the frequency of the spikes - which represent a step - is higher. Surprisingly, the energy values are generally lower than the values for walking in figure 2 - one would expect that running would cause more force but this is due to how the device is being carried. The person that was running carried the device in the pocket of a tight running jacket, whereas the walking person carried the device in a loose pocket. This detail can affect how much force the device is subject to.

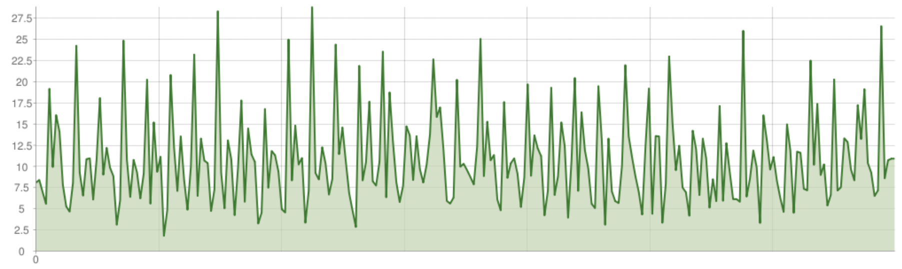

Figure 3: Energy graph, running (x-axis represents 14 seconds, y-axis is energy)

When the device is placed in a backpack (as in figure 4), the force values are smaller than when the device is placed in a pocket, but the peaks still show the individual steps. The lower force values are most likely due to the damping effect from the backpack itself.

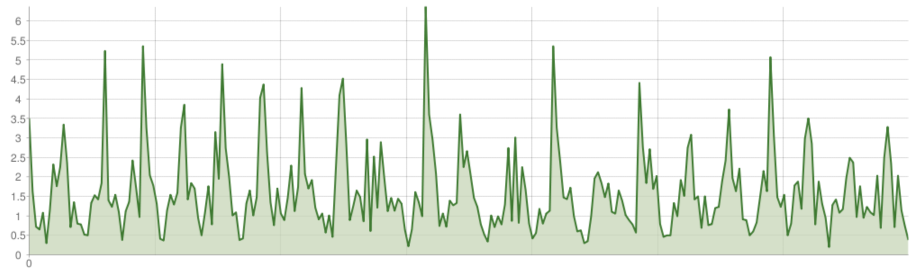

Figure 4: *Walking, device in backpack (x-axis represents 14 seconds, y-axis is energy)*

The data set in figure 5 is recorded by a device in the pocket a running 7-year old girl. In this case, the device was in a loose side-pocket, which could cause extra force. The running styles of the individuals have not been analyzed, but these can also contribute to how much force is measured.

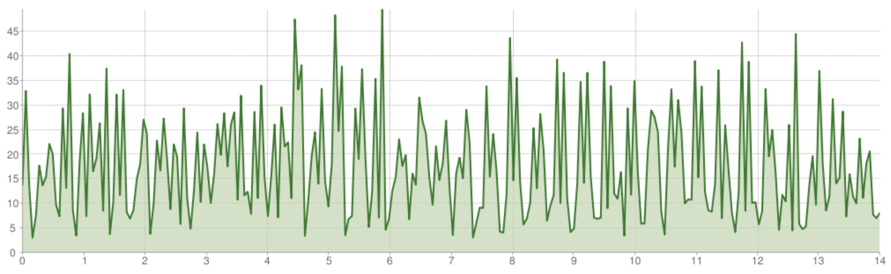

Figure 5: *Energy graph, child running (x-axis is seconds, y-axis is energy)*

Other situations In figure 6, the data for a device placed on the front seat of a driving car is shown. At the time of measurement, the car was driving with a speed of around 50km/h on an ordinary road. If data from this use case should be of any use, we should be able to either measure acceleration and deaccelleration, or to observe that the suspension system of a modern car would cause the data to appear distinctively different from a non-car use case. From observing data collected while driving, it is clear that it can not be used to determine that the user was actually driving. The spikes in the graphs are most likely caused by small bumps in the road or the phone sliding on the seat in a turn, but this can not be differentiated from data captured by device in the pocket of somebody making a cup of coffee.

Also, it is clear from inspecting data from an accelerating car, that evidence of acceleration and deceleration of the car will drown in the data caused by bumps and sliding.

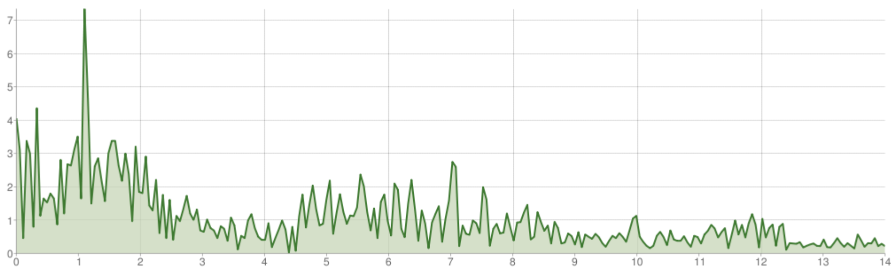

Figure 6: *Energy graph, driving car. Device is lying on passenger seat (x-axis is seconds, y-axis is energy)*

The same is true for riding a bicycle. In the data shown in figure 7, the device was placed in the backpack of a person riding an ordinary bicycle. Bumps in road contribute to a lot of noise, causing a clear pattern to disappear. It could be imagined that if the asphalt was completely flat, cyclic peaks would appears, since riding a bike involves cyclic motion of the body. If there was a requirement to strap the device onto the tight of a person, it could very well be feasible to detect biking, but such a requirement is specifically not in this project.

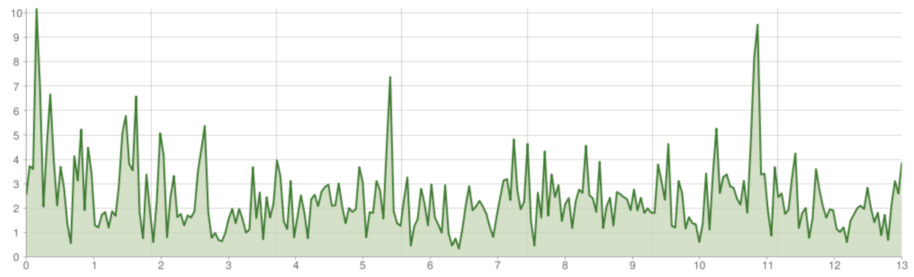

Figure 7: *Energy graph, biking. Device is in backpack (x-axis is seconds, y-axis is energy)*

For reference, figure 8 shows the data captured from a device lying idle on a table. This shows the noise caused by the inaccuracy of the hardware, which is expected in consumer-level phones. These values can be used for establishing when a device can be considered to be in an idle state.

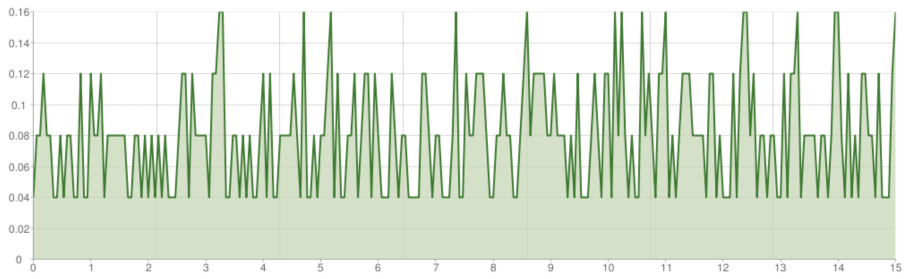

Figure 8: *Energy graph, idle. Device is placed on table (x-axis is seconds, y-axis is energy)*

2.1.2 Data quality

Generally, the data from the accelerometers is reliable

- The sample rate is uniform (to a certain degree)
  - Garbage collections or other CPU-intensive activity can have a small effect on sample rate, but not to a degree where false results are measured.
- Patterns can be reproduced
  - shaking or moving the device several times in the same manner results in the expected cyclic data output
- No unexpected spikes or dead periods have been observed.

2.2 Location

Location can - on the device chosen for this project - be determined in two ways

- GPS signal
- WIFI/Cell tower triangulation

Table 1 shows an overview of the relevant properties for the two methods.

	GPS	WIFI/CELL
Precision	high	low
Works without Internet connection	yes	no*
Battery usage	high	low
Works indoor	no	yes
Works in remote areas	yes	no**

** unless the device has a cache or preloaded list of WIFI and cell tower locations*
*** if an area is not within cell or WIFI coverage, location can not be terminated*

Table 1: *Location methods*

For this project, battery usage and the ability to work indoor are very important properties. As the application require the location frequently, battery usage becomes a concern. GPS receivers are known to use a lot of battery, in particular when they are started on a device placed indoor, as it will spend a lot of time on the radio listening for satellite signals that can not reach it. WIFI/Cell triangulation typically does not have this problem, as these signals are better received inside buildings. A good compromise is to use WIFI/Cell when it is available, and attempt to catch the GPS location if it fails. For this project, only WIFI/Cell is used for location determination.

When using WIFI/Cell location, the detected cell and WIFI points are compared against a list of known cells and WIFI points - for this device, the list is maintained by Google and queried on demand by the device. An import factor in this respect is the data quality and completeness of this list.

2.2.1 Data sampling

When the device uses cell and WIFI ids to determine location, it receives a series of location estimates. Such an estimate contains

- latitude and longitude
- timestamp
- accuracy

The accuracy the maximum range in meters from the lat-lon point that the device can be. When a WIFI id is detected, the accuracy is about 200m, though often even lower (around 80 m). When no WIFI is nearby, the device falls back to cell ids. As a cell can have a very long range, it will typically use several cells and triangulate an estimate. As official documentation is not available, the exact algorithm that the device uses in this case is not known, but the data collected in this project shows that no location is given unless it can be estimated with an accuracy on 5000m or lower. Roughly speaking, location based on WIFI will give you an estimate on what street or city block you are, while a cell id will tell you the city.

A typical scenario can be seen in figure 9, where the red line is the actual route traveled (drawn by hand). Along the route, blue circles represents the estimated locations along the route, where the radius of the circle is the accuracy. The number inside each circle is the minute since the trip began.

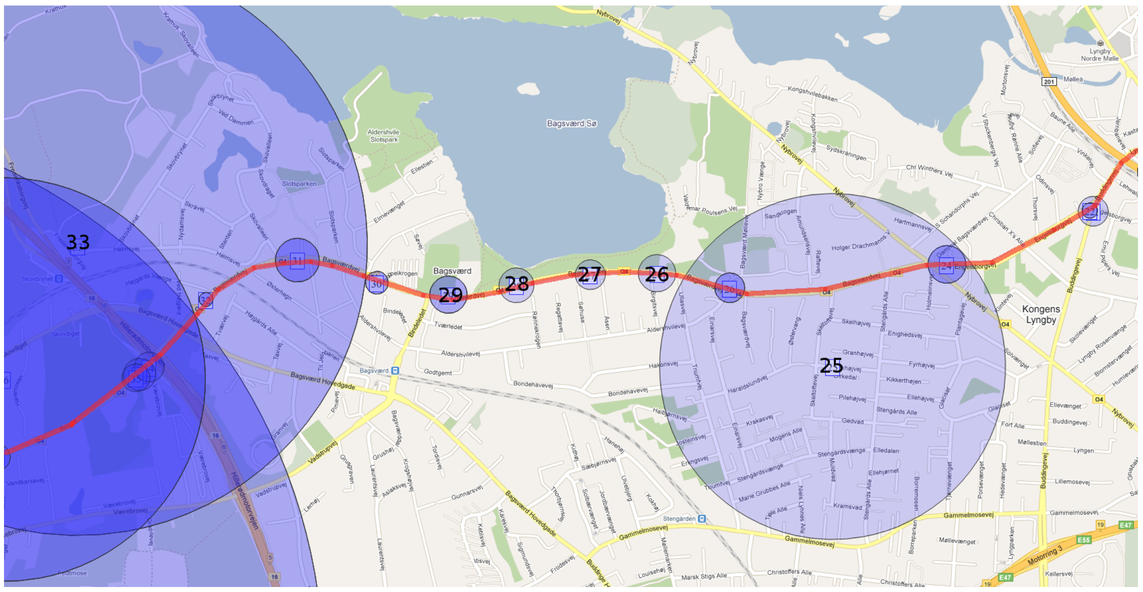

Figure 9: *Cell and WIFI points detected along a route. Numbers show the minute number that each location was determined. Blue circles represents accuracy.*

Several observations can be made from this figure. The small blue circles (minutes 26, 27, 28, 29 etc.) are most likely from WIFI signals, and the larger circles (minutes 25 and 33) are from cell ids. Note how the estimates from the WIFI signals are accurate - their accuracy values are in all cases too high, as the lat-lon points are typically on the route - while the detected cells often are centered on the route.

The area traveled in the route show in figure 9 is a densely populated rural area, with good cell and WIFI coverage. As expected, the location determination in areas like this is good, due to the amount of reference points available. In more sparsely populated areas, it is expected that the accuracy drops - there are less cell towers and very often no WIFI coverage at all.

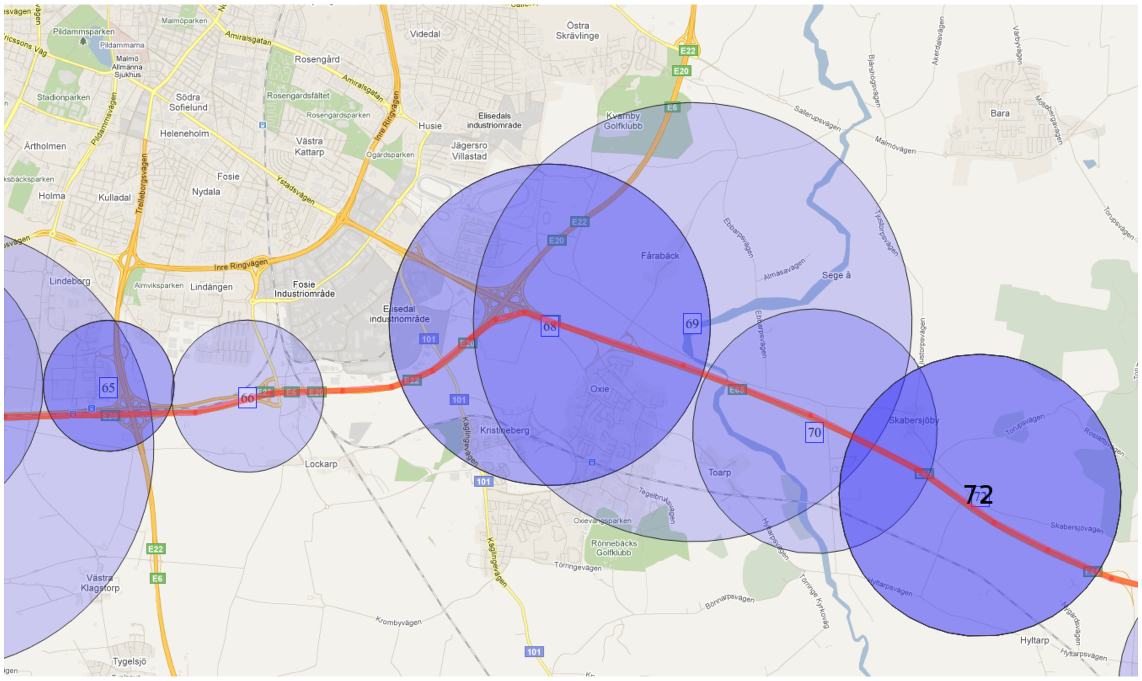

Figure 10: *Route in rural Sweden. The accuracy for point 72 is 1810 meters*

It has, however, also been observed that the location service can return invalid results. Figure11 shows several different errors. One issue can be observed with the points at minute 90 and minute 91A. For the most part, the accuracy circle from point 91A covers the route, but this is not the case for the point at minute 90. More seriously, a second point has been registered in the 91st minute. This point - which is marked 91B - is obviously wrong, and such a point is likely to cause problems for an algorithm that is not capable of filtering it out. In this case, the point 91A is blatantly wrong and filtering it should be possible, both other more subtle cases (such as point 90) is less obviously wrong.

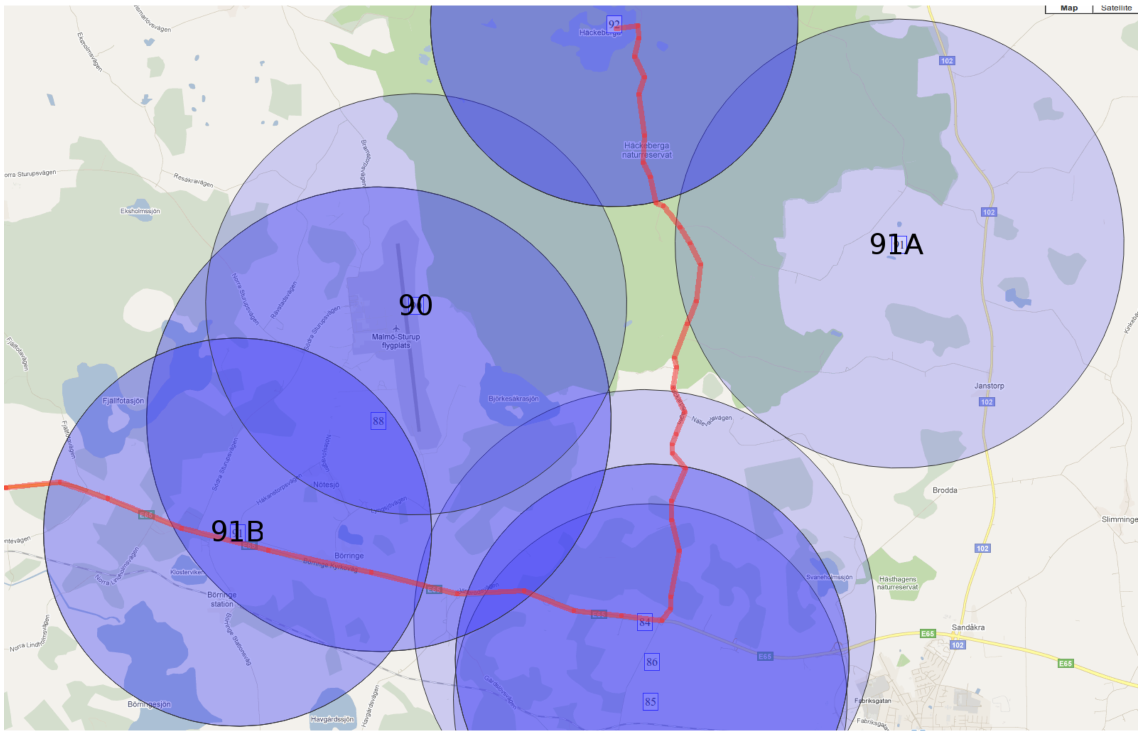

Figure 11: *Route in Swedish forest. The accuracy value for point 91 is 3415 meters. The red route is plotted manually.*

The cases shown in figure 11 shows issues with data sampled from cell ids. The same type of errors happen for samples that include WIFI points, but these seem less frequent. Also, as cells have much higher coverage radius, the false locations that can come from a bad reading based on cell ids will typically impact a calculation much more than a bad WIFI reading, since the error distance is greater.

2.2.2 Data quality

The data from the location service can not be said to be reliable. From the sampling done in this project, the conclusion is that densely populated areas has relatively good and accurate location data available, while less accuracy is available in rural areas. Both these observations are as expected. However, the false readings illustrated in figure 11 shows that data can not be trusted and must be filtered before being used.

Also, location data is not available at all if there is no cell or WIFI coverage. Although it has not been a problem in this project, it does restrict the area in which an application that make use of the location service can operate. In this case, falling back on using the GPS data is an option to consider.

3 Algorithms

3.1 Gait classification

From the data collected from the accelerometers, an algorithm to detect the current gait is needed. As seen in figure 2 and figure 3, a person running or walking produce a graph with distinct peaks. There are several documented approached this how to solve this problem, and for this project, Fourier transformation has been chosen.

3.1.1 Fourier Transform

A Fourier transformation moves a set of data from time-domain to frequency-domain. For the purpose of gait detection, this is relevant, as gaits typically contains cyclic behavior that will show up as distinct frequencies. Some gaits from quadrupeds can have advanced frequency properties, while human gaits typically have simpler frequency properties.

An example of a Fourier transform graph can be seen in figure 12.[1]

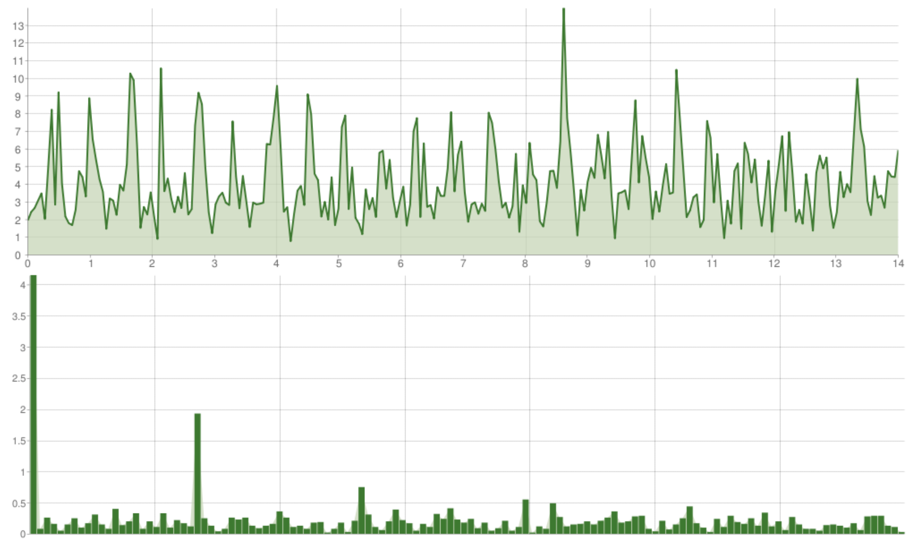

Figure 12: *The energy graph for a walking person and (below) the corresponding Fourier transform. The spike is at x=0,19*

In figure 12, which shows the accelerometer data and the corresponding frequency values from the Fourier transform, the relevant frequency is easy to single out in the frequency-domain graph, as the peak at x=0,19 is distinctly higher than the rest of the values. The same can be observed in figure 13, where the distinct peak frequency is at x=0,33. These two examples have data of good quality, and the corresponding Fourier transforms have distinct peaks. As shown in figure 14 - which is a more typical set of data - the accelerometer values shows a clear cyclic pattern that changes about halfway through. This can happen for a number of reasons - the surface type could change while walking, or

[1]For all Fourier transform graphs shown, the x-axis has a range from 0 to 1. For implementation reasons, the true frequency values are not used, but an abstract [0..1] range is used. The high value at x=0 indicates the average value in the time-domain.

the device moved from a hand into a pocket. Hence, the corresponding Fourier transform is less clear, but still shows a distinct peak.

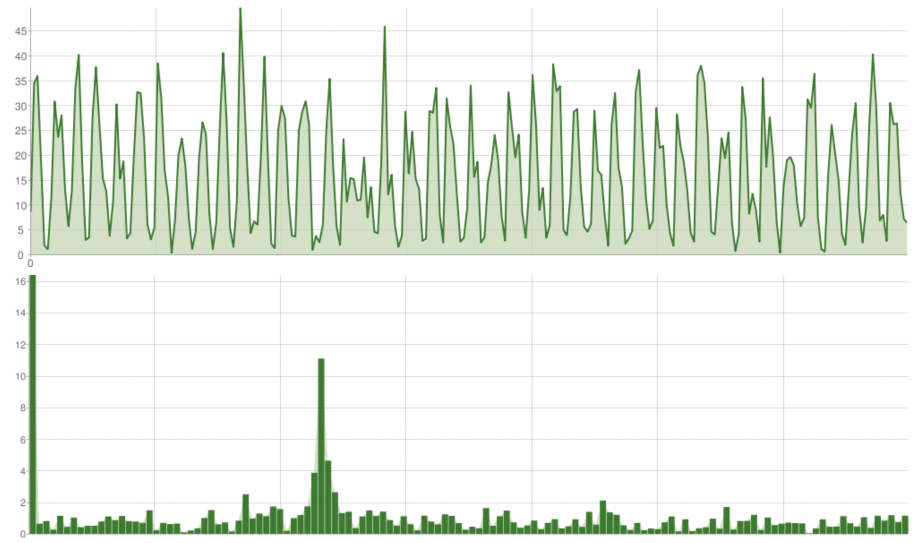

Figure 13: *The energy graph for a running person and (below) the corresponding Fourier transform. The peak is at x=0,33*

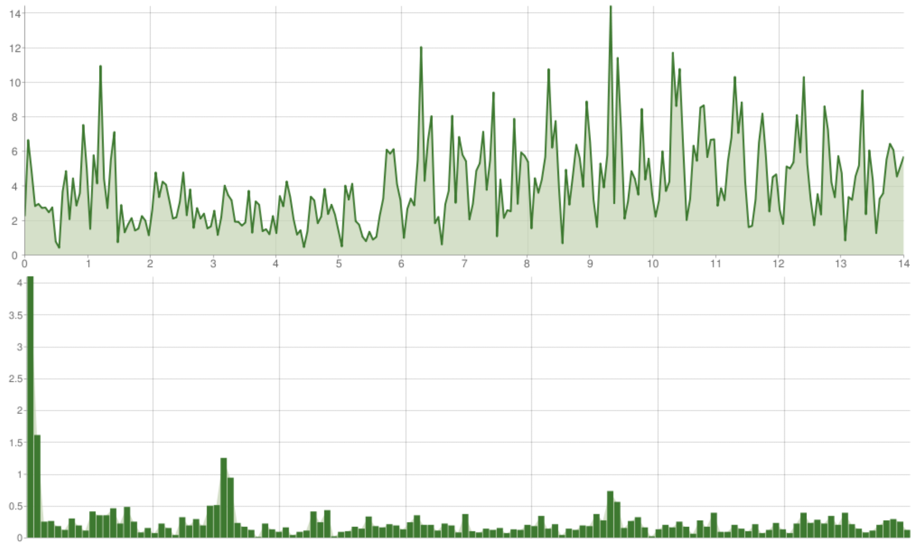

Figure 14: *The energy graph for a walking person and (below) the corresponding Fourier transform. The peak is at x=0,23*

The Fourier transform values in figure 12, figure 13 and figure 14 shows that it is possible to distinguish between walking and running gaits. To determine a gait, the following is required

- a **distinct peak** in the Fourier transform
- a **minimum** average value in the accelerometer data

A frequency x is a peak if

- $f(x)$ is higher than f(0)/7,5 AND
- $f(x)$ is 3 times higher than the average value of all frequencies

Frequencies are grouped by these rules if they are adjacent, forming a single peak with the highest value as the peak value. As it can be observed in figure 14, frequencies often rise around a peak. Grouping is nescessary to avoid adjactent frequencies to be considered as individual peaks.

A peak is a **distinct peak** if

- $f(x)$ is at least 15% higher than any other peaks

The minimum average value is chosen to be 1,5. If the average value of all accelerometer values for a 14 second period is below this value, gait analysis will not be performed. The above definition values have been chosen empirically, after examining a large set of collected data.

If a distinct peak x is established (which in many cases is not possible due to too much noise), it is straightforward to determine if the person using the device is running or walking

- $f(x) > 0,1$ and $f(x) < 0,25 \Rightarrow$walking
- $f(x) >= 0,25$ and $f(x) < 0,6 \Rightarrow$running

3.2 Speed estimation

To determine if the user of the device is in a moving vehicle, an estimate of the current speed is needed. If such an estimate can be established, any speed above running speed (which is around 10km/h) indicates that the user is using some kind of vehicle - either a bicycle, car, train or similar. For this, the location data described in section 2.2 is used[2].

To be able to measure speed at a certain time, a set of recently passed locations will be used. An important fact is that some points have better accuracy than others, so filtering the inacurate points out will like improve the speed estimate. Figure 15 shows an example of measured locations a, b, c, d, e, f and g along a route (green line). In this example, the points b and e have been discarded, as their accuracy is not as good as nearby points.

The length L of the estimated route is then

$$L = |a - c| + |c - d| + |d - f| + |f - g|$$

and the estimated speed S (where x_t is the time where location x was measured)

$$S = \frac{L}{g_t - a_t}$$

[2]A location is defined as latitude/longitude, accuracy and timestamp

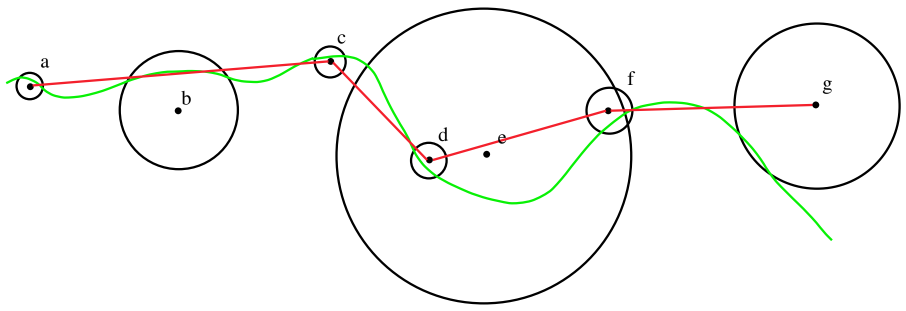

Figure 15: *Route through a set of locations. The green line is the actual route travelled, and the red line is the estimated route. The circles around each location represents the accuracy.*

Several things are worth noticing

- the estimated distance L is smaller than the actual distance travelled, as the recorded locations does not capture the curves and detours of the actual route
- not all locations are close to the route actually followed, which could increase inacuracy
- the point g is not very accurate - and it does make a difference if the point is used with our without its inacuracy value.
- if no locations with good accuracy is available, the calulated distance could be much higher than the actual distance

The example in figure 15 is not a complex case, and a speed estimate would in this case be fairly accurate. In reality, data is often not as unambiguous as this. In particular, the less accuracy the locations have, the more problematic this approach becomes.

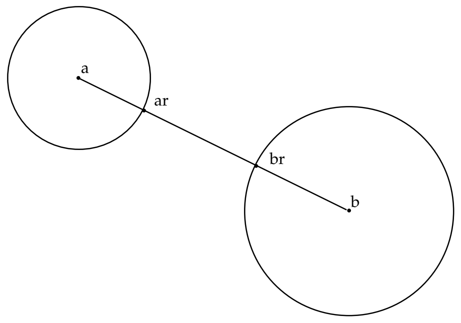

Figure 16: *Measuring the distance between two inacurate locations*

Figure 16 illustrates how speed between two inaccurate locations can be estimated. The locations a and b each has an accuracy value defined, shown as circles. Each circle defines the area in which the location could be in. The distance between the points ar and br then defines the minimum distance between a and b. Had the circles overlapped, no minimum distance could be calculated, as the device that provided the location potentially had not moved at all. By using the minimim distance value, the minimum speed S_{min} can be calculated

$$S_{min} = \frac{|ar - br|}{b_t - a_t}$$

This value will always be lower than the actual average speed, but never higher[3].

To calculate a good guess of the average speed during the last 4 minutes, two locations from the list of recent locations are found :

pa= location with best accuracy, age less than 1 minute

pb= location with best accuracy, age between 3 and 4 minutes

Using the method described in figure 16, the minimum average speed can be found

$$S_{min} = \frac{|pa - pb| - (pa_{accuracy} + pb_{accuracy})}{pa_t - pb_t}$$

This method has some shortcomings :

- if moving in an area where location data is not accurate,
  - it is possible that the accuracy circles of pa and pb overlap, which causes the estimated speed to be 0
  - it is possible that the accuracy circles of pa and pb are very close, causing a very low speed estimate
- if moving in a circle, the pa and pb could by coincidence be close, causing a low or no speed estimate

3.3 Resolving context

As shown in the previous sections, data from the accelerometer can be used to detect if a user is idle, walking or running, and data from the location service can be used to detect if the user travels with a speed high enough to indicate she is on board a vehicle.

These two data sources must be combined, so that one common result is returned - idle, walking, running or vehicle. Also, it should be ensured that the result - the current context - does not flip-flop between two states when the immediate data is not conclusive. Tests shows that while walking or running, some of the 14-second accelerometer datasets does not translate into the expected “walk” or “run” result, but when looking at the average picture of the last few minutes, it is much clearer what the context is.

A simple state tracker will be used to track the probablitites of the possible states at a given moment. The state tracker will keep three values

- walk probability
- run probability

[3]This assumes that the locations and their accuracy values are always correct. In reality, this is not always the case

- vehicle probability

Each of these values are in the range 0 to 1. The current context is the found by selecting the highest of the three values. If all values are below 0.1, the current context is considered to be idle.

The state changes are described in table 2 and table 3. The top row of these tables shows the possible results from accelerometer data - the **"lo"**, **"me"** and **"hi"** accelerometer results are used if neither walking or running can be detected, but the average accelerometer force measured is above idle. **"lo"** is used if the when little force is measured, **"me"** if medium force is measured and **"hi"** if high force is measured. These extra accelerometer values are used to indicate to the state tracker what might be going on in the absence of real results. The numeric values in the tables are applied to each state probability, depending on which accelerometer result is applicable. Table 2 is used if the measured speed is below 10 km/h, and table 3 is used if the measured speed is higher than 10 km/h. The speed is measured as described in section 3.2, meaning that is the minimum average speed. If the speed is greater than 10 km/h, it is possible that a vehicle is in use.

	idle	lo	me	hi	walk	run
walk probability	-0.5	-0.15	0.0	0.0	+0.25	-0.05
run probability	-0.5	-0.15	0.0	0.0	-0.05	+0.25
vehicle probability	-0.05	0.0	0.0	0.0	-0.15	-0.15

Table 2: *State change table for speed < 10 km/h*

As it can be seen in table 3, it is possible to get contradicting data. It should not be possible to walk and move with more than 10km/h at the same time. As a "walk" result from the accelerometer is considered to more reliable than a speed estimate, the walk probability is then incremented while the vehicle probability is decremented, even though the speed indicates otherwise.

	idle	lo	me	hi	walk	run
walk probability	-0.5	-0.15	**-0.25**	**-0.25**	+0.25	-0.05
run probability	-0.5	-0.15	**-0.25**	**-0.25**	-0.05	+0.25
vehicle probability	-0.05	**+0.05**	**+0.25**	**+0.25**	-0.15	-0.15

Table 3: *State change table for speed >= 10km/h*

An example of state tracking applied to data from a walking person can be seen in table 4. This examples shows that device starts idle, and is then picked up. On two occasions, the "walk" result is returned from the accelerometer, which increment the walk probability. When "me" is returned (indicating that the device is in motion), the walk state is not changed. When "lo" states appear, the walk probability decrease, as "lo" is too little force to be considered walking. As the device is idle, the walk probability has decreased back to 0.

One major issue with the state tracker is that the estimated speed uses data that is up to 4 minutes old, while the accelerometer data is current. A concrete problem this gives can be illustrated by walking up to a car and start driving. The walking is easily detected by the accelerometer, and the state tracker will soon pick up a high "walk" probability. After entering and starting the car, the accelerometer will start to return "hi", "me" and "lo" values, as the device is now subject to the movements of the car. As the speed change has not yet been picked up, the "walk" probability will very likely still remain high.

accelerometer	time	speed	walk	run	vehicle	result
idle	14	<10	0.0	0.0	0.0	idle
lo	28	<10	0.0	0.0	0.0	idle
walk	42	<10	0.25	0.0	0.0	walk
me	56	<10	0.25	0.0	0.0	walk
walk	70	<10	0.50	0.0	0.0	walk
me	84	<10	0.50	0.0	0.0	walk
hi	98	<10	0.50	0.0	0.0	walk
lo	112	<10	0.35	0.0	0.0	walk
lo	126	<10	0.20	0.0	0.0	walk
idle	140	<10	0.0	0.0	0.0	idle

Table 4: *Example of state tracking of walking person*

Not until a few minutes have passed AND sufficient cells or WIFI coverage areas have been passed will the increase in speed be detected and the "vehicle" state will start to increase.

4 Conclusion and future work

During the development of the mobile application associated with this project, a fair amount of data has been collected and analyzed[4]. Generally - and this is also evident from the sections above - the detection of walking and running is good, and the detection of being in a moving vehicle is less good, depending on the cell and WIFI coverage in the area. Personally, the experience has been good, with most of my travels going on bike between Ballerup and DTU in Lyngby. The time spent in a vehicle was almost always correct (26 minutes on estimated on average, actual average 29 minutes), and walking performed before and after each bike trip was accurately measured. The results, however, can not always be trusted, as spurious location errors can occur.

4.1 Future work

Location filtering to remove incorrect locations. As mentioned in section 2.2.2, incorrect locations is a problem. Some locations are obviously wrong (and place you in a different city, 30 kilometers away), and can easily be discarded, while others are exactly wrong enough to cause the results to be wrong, but not wrong enough to easily detect. One filtering method could be to use historical location data to estimate the probability of a current location to be correct. This could be verified by occasionally turning on the GPS receiver to validate the location.

This is a problem that might be smaller as time passes, as the location service feeds on data collected by (in this case) Google. If they improve their data, the issue about incorrect locations become smaller. However, if they do not improve their data, quality will deteriorate, as cell masts and WIFI routers are being replaced or relocated. An interesting experiment would be to use a different location service provider and compare their location data with the data obtained from Google. One such provider is Skyhook (www.skyhook.com).

Bike detection was hoped to be possible in this project, but no suitable indicator was found in the sensor data. One idea would be to filter out the noise of the accelerometer data (the road bumps) in the hope that cyclic movement could be detected. If this is possible or not has not been analyzed further, and no similar projects with this goal has been found.

Battery usage has not been a formal target of investigation for this project, but it was taken into consideration during the actual implementation. The basic strategy is to check the accelerometers every 30 seconds (or more - this is configurable) for movement. If the device is idle, it is put back to sleep for another 30 seconds. If movement is detected, recording of accelerometer data will be started and if movement continues for 30 seconds, the location service is started. The current battery usage has not been measured accurately with and without this application running, but it is evident that such a service does use some battery. Further improvements in this area could be to more intelligently put the device to sleep for longer periods, ie. during the night or other periods where it historically is known to be idle.

Applications that use the context data. The mobile application will continually publish the current context (idle, walking, running) from the last 14 seconds of accelerometer data, as well as the ongoing context (idle, walking, running vehicle) described in section 3.3. A wide range of application can be imagined - from apps that tell you if you need to run or walk more, to apps that measure time spent in your car (for accounting purposes).

[4]This data is available. See appendix A

5 References

[1] Toshiki Iso and Kenichi Yamazaki - Gait Analyzer based on a Cell Phone with a Single Three-axis Accelerometer. NTT DoCoMo Network Laboratories, 2006.

[2] Ričardas Braž inskas - Towards Context Awareness Using Mobile Sensors. Master thesis, DTU Informatics, 2008

[3] Sean Hollister - article on Engagdet. http://www.engadget.com/2010/11/27/nokia-toys-with-context-aware-smartphone-settings-switch-jigsaw/

[4] Sean Hollister - article on Engadget. http://www.engadget.com/2010/09/15/intel-building-a-context-aware-api-for-smartphones-and-tablet-pc/

[5] Davrondzhon Gafurov, Kirsi Helkala, and Torkjel Søndrol - Biometric Gait Authentication Using Accelerometer Sensor. Department of Computer Science and Media Technology Gjøvik University College, 2008

[6] Alexis Malozemoff, "Accelerometer-based gait recognition", McGill 2009

[7] Wikipedia article on location based services, http://en.wikipedia.org/wiki/Location-based_service

A Source code and collected data

Source code and collected data is available at
https://github.com/hrstrand/ContextResolver

- ActivityGuesser (source files)
  - Android service and app
  - Main application logic is implemented here
  - Data analysis tools
- TestGuessListener
  - Example Android application that listens for context intents published by ActivityGuesser
- Data
  - raw and processed data from fall 2010, used for data analysis in this project