Path tracing via comparison of received signal strength within WiFi landscapes

by Alvaro Gauterin

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computational Visualistics

at the

University of Koblenz

Faculty of Computer Science

and the

Fraunhofer Institute for Integrated Circuits

Department of Communication Networks in Nuremberg

First reviewer	Prof. Dr. J. Felix Hampe
Second reviewer	Dr. Stefan Stein
Supervisor	Dipl.-Inf. Steffen Meyer

March 2011

Acknowledgements

This work is one part of a triple bachelor theses project. When my fellow students, Christian Stehr and Philip Boedts, and me were looking for a professor willing to supervise our triple theses project we were on the point of believing that we won't find anyone. We knew that the triple theses project was extraordinary and entailed a significant amount of work for the supervisors. Especially as we intended to write our theses externally in a research facility 350 kilometers away from our home university a lot of professors were scared off since it was not obvious to them how their institute will benefit from this project. In the end Prof. Dr. J. Felix Hampe and his research assistant Dr. Stefan Stein from the department of commercial information technology were kind enough to take on the burden of supervising our triple theses project. I would like to thank them for this decision.

Up next I would like to thank Steffen Meyer who was our supervisor at the *Fraunhofer IIS*. He helped us to define the subject area of each thesis so that we could work on them independently from each other. Writing one's thesis in a company offers the great advantage of always having professionals from the subject area (of WiFi localization in our case) around you. They can explain you things you are having trouble with in a much more intuitive way than a book can do that you are reading at home alone. I would like to mention Thorsten Vaupel, a very helpful employee who always pointed us in the right direction when we got stuck in a tricky issue. I really do recommend to students to write their theses in external companies or research facilities due to the valuable experiences one gains while working there.

Finally I would like to thank Christian and Philip for the great teamwork on project *Schnitzel*. I will never forget how much fun it is to record WiFi measurement data when you are doing it out of a 26-year-old classic convertible.

Abstract

This bachelor thesis deals with the comparison related to the similarity of recorded WiFi patterns during the tracing of a path through the streets of a large city. Both MAC address only comparison has been investigated as well as the incorporation of RSSI values, whereby the localization accuracy has been evaluated. Methods for the detection of different types and combinations of loops in the path are demonstrated likewise the attempt to estimate the degree of urban development in the environment of the user by assessing the received signal strength and signal-to-noise ratio of GPS satellites and GSM cell towers. In order to observe a user's proximity to a certain spot on a large public square the absorption of WiFi signals by the human body has been taken into account. Finally, the results of a comparison of the computing performance of a modern smartphone versus the alternative of remote calculation on a server including data transmission via cellular data network are presented.

Contents

1 Introduction

Mobile phones have revolutionized the way people communicate with each other. Before the era of cellphones you were calling a landline which meant you were calling a specific place. With the invention of the mobile phone the aspect of locality became irrelevant. A person with a cellphone is virtually reachable at all times no matter where he is. The mobile phone became enormously popular and today in the 21st century it is basically self-evident for a citizen of the western civilization to own a cellphone.

Since their widespread introduction in the 1980s cellphones have developed dramatically. Originally intended simply for placing phone calls today's mobile phones have evolved into high-performance hand-held computers for the pocket. The first devices offering relatively high computing power, called PDA, were introduced in the 1990s. These *personal digital assistants* served businessmen as a mobile office. For private consumers those products became interesting ever since the device generation that was able to playback rich multimedia content. The disadvantage of the PDA was that people had two devices to take with them, the PDA and a separate cellphone. It was only a question of time until these two device classes merged into one versatile all-in-one gadget around the turn of the century. The resultant device class is called *smartphone*, as these gadgets are *smart* enough to combine the functionality of a full-featured cellphone and the office and multimedia capabilities of a PDA. Today's smartphones offer virtually the same range of functions as full-sized desktop computers, just with the limitation of a relatively small display. Modern smartphones feature processor clock speeds not available on desktop computers just ten years ago, feature dedicated graphic chips that are capable of processing millions of triangles per second and offer a wide range of connectivity options for wireless data transmission, such as WiFi, Bluetooth and UMTS/HSPA. Additionally, today's smartphones feature built-in sensors, such as GPS, compass, accelerometer, gyroscope and ambient light sensor, enabling them to react to their surrounding environment.

This context-awareness allows for interesting novel applications like real-life games, in which the player assumes the role of an agent who chases other agents through a big city supported by the useful information that his smartphone calculates. The investigation of the technical fundamentals of such a game is the major subject of this bachelor thesis. In fact, this bachelor thesis emerges in conjunction with two other bachelor theses authored by Christian Stehr and Philip Boedts also from the University of Koblenz. The reading of these two other bachelor theses is highly recommended in order to understand the full extent of this collaborative project.

2 Location determination on modern smartphones

Modern Smartphones determine their current geographic position not only by utilizing signals from GPS satellites but also by analyzing the pattern of WiFi access points and GSM cell towers present in their current environment [Laitinen01]. The crucial disadvantage of GPS localization is its insufficient precision in urban places with a high density of development. Inside buildings or in the middle of a narrow alleyway the GPS technology can only deliver position information with an accuracy of several 100 meters or is not able to gain a satellite fix at all.

This is where wireless standards, such as WiFi and GSM excel, since they do not require unobstructed view at the sky [BillCapKofahlMundt04]. In a real world situation WiFi and GSM signals can travel through walls while GPS signals normally cannot. The reason for this phenomenon is the much greater signal strength of WiFi and GSM radiation compared to GPS signals. GPS satellites orbit the earth at an altitude of over 20,000 kilometers. This is the distance the GPS signal has to travel before reaching the user's smartphone. The typically received signal strength of GPS is around -130 dBm, whereas WiFi and GSM transmitters are positioned at ground level (compared to outer space), resulting in a small fraction of the transmission distance of GPS signals. The typically received signal strength of WiFi in an outdoor environment ranges from -70 to -100 dBm, while the official *Android* API indicates GSM signal strength in the range from -51 to -113 dBm. If we want to compare signal strength values it is recommended to convert the values from the logarithmic scale of decibel to the linear scale of watt. In order to do this we can use the following formula where P is the power in watt and x is the power ratio in dBm:

$$P = 10^{\frac{x-30}{10}} \tag{1}$$

If we compare the typical received signal strength of WiFi at -80 dBm with the one of GPS at -130 dBm, we compare $10^{-11}W$ to $10^{-16}W$, which means the receiving power of WiFi is 100,000 times greater than the receiving power of GPS. With this comparison in mind it is easier to understand why GPS signals are blocked out by the stone walls of buildings. On the other hand WiFi and GSM signals are strong enough to travel through urban development to a limited extent. This is why we can place mobile phone calls inside large buildings and connect to the Internet via a WiFi access point located a few rooms further away. In conclusion, one can say that GPS delivers indeed high precision localization in the countryside or suburban areas, however within dense urban areas only wireless technologies like WiFi and GSM remain usable for position fixing.

In addition to that, it should be mentioned that position fixing via WiFi is much more precise than via GSM, due to the much greater cell radius of a GSM tower compared to the range of a WiFi access point. GSM cell towers have a coverage of several kilometers while WiFi access points have a reach of around 100 meters. Furthermore, the amount of WiFi access points is much higher than the amount of GSM cell towers heard simultaneously, in a typical urban environment you hear 30 access points versus 7 cell towers. In general, the following rule of thumb holds true for localization accuracy: Better many distinguishable transmitting stations each with a short range (WiFi) than few distinguishable transmitting stations each with a long range (GSM). In order to deliver the best possible location determination modern smartphones combine the measurement results of both GSM and WiFi, as well as GPS. Especially this makes a step by step refinement of the positioning accuracy possible. The result from GSM localization is the first to be available, since a mobile phone is informed of the present GSM cells at all times. Next the result from WiFi localization becomes available, since it takes a few seconds to scan for present access points. Finally, the result from GPS localization becomes available, since it takes up to several minutes for the GPS chip to get a fix on at least three satellites. However, on a smartphone the satellite fix is accomplished much faster than on an ordinary GPS navigation device. Since the approximate location is already known, as it is retrieved by GSM and WiFi positioning, the evaluation of the GPS signals can proceed at a much higher speed. This enhancement to the conventional proceeding of GPS positioning is called *Assisted GPS* (A-GPS). It allows for a much shorter *Time To First Fix* (TTFF) [WeynSchrooyen08].

3 Functional principle of WiFi fingerprinting

Before WiFi positioning can be used on mobile devices, preparatory work has to be done. During the so-called training phase, a reference carpet of WiFi patterns linked to their corresponding geographic coordinates is being created [Xiang04]. In the area in which WiFi positioning is intended to be available every street needs to be traversed systematically. During this phase the MAC address and received signal strength of all present WiFi access points is being recorded continuously as well as the current GPS position consisting of latitude and longitude. Notabene, the data being collected does not contain any privacy-related information.[1] The execution of the training phase is done by commercial companies like *Skyhook Wireless* and *Google Inc.*, which have deployed drivers to survey virtually every single street worldwide.

Finally, the gathered data is processed and entered into an online database with an interface for smartphones to use as a free service. If a smartphone wants to perform a localization via WiFi it scans for present access point in its environment and uploads this WiFi pattern consisting of MAC addresses and RSSI values to the web service of a company offering WPS. The web service compares the user's WiFi pattern to the database to find the best matching WiFi pattern recorded during the training phase, whose GPS coordinates are finally returned to the user's smartphone. Probably the final coordinates are being interpolated if there are multiple WiFi patterns in the database that have an equally high matching rate to the user's WiFi pattern. The accuracy of WPS within dense urban areas is about 20 to 30 meters [QuaderLiPengDempster07].

[1] In May 2010 *Google Inc.* got into criticism of the media when the company admitted to have accidentally recorded private data from WiFi transmissions while generating a reference carpet for WiFi positioning with one of their *Street View* cars.

3.1 WiFi fingerprinting through crowdsourcing

Apple Inc. has originally used the *Skyhook* database for WiFi positioning on its first generation *iPhone* and *iPod touch* since these devices did not feature a built-in GPS chip. With the launch of the *iPhone 3G* incorporating a GPS receiver *Apple* started to build its own WiFi fingerprinting database by using its sold *iPhone 3G* devices in the hands of the customers as data collection facilities. Each *iPhone* constantly collects WiFi pattern/GPS coordinates pairs and uploads these fingerprinting packages to the *Apple* server every 12 hours on the availability of a WiFi Internet connection[2]. Taking into account the tremendous amount of *iPhone* users spread all over the globe, *Apple* was able to build up their own world-spanning reference carpet within a relative short amount of time on a nearly zero cost basis, since this method does not require the deployment of drivers. Additionally, the fingerprinting database is being updated on a permanent basis from virtually all places of the world simultaneously. This approach is called *crowdsourcing*. *Skyhook*, in addition to their employed drivers, also uses crowdsourcing to refresh their fingerprinting database - they call this technique *Automated Self-Healing Network*.

4 Theory of radio signal propagation

When a signal is sent over a physical medium, like copper or fibre-optic cable, it travels along a predefined path. As opposed to this, air provides no fixed path for signals to follow, signals travel without guidance. Ideally, a wireless signal would travel directly in a straight line from its transmitter to its intended receiver. This type of propagation, known as LOS (line-of-sight), uses the least amount of energy and results in the reception of the clearest possible signal. However, because the atmosphere is an unguided medium and the path between a transmitter and a receiver is not always clear, wireless signals do not usually follow a straight line. When an obstacle stands in a signal's way, the signal may pass though the object or be absorbed by the object or it may be subject to any of the following phenomena: Reflection, diffraction or scattering. The object's geometry (shape and size) governs which of these three phenomena occurs [Dean09, p. 367]. So as to classify the size of WiFi radiation we must calculate the typical wavelength λ of WiFi signals with the following formula:

$$\lambda = \frac{c}{f} \tag{2}$$

where c is the velocity and f is the frequency of the wave. Since WiFi normally travels through air at the speed of light with about $3 * 10^8 \frac{m}{s}$ and typically operates at a frequency of 2.4 GHz, WiFi signals have a wavelength of roughly 12.5 centimeters.

[2]In July 2010 this fact has been made public via *Apple*'s response to a letter by two U.S. Representatives requesting for information regarding *Apple*'s privacy policy and location-based services. The fact that *Apple* makes use of such methods has been unknown up to that point in time.

- In the case of a **reflection** the wave bounces off a surface according to the rule *angle of incidence equals angle of reflection.* Reflection appears if the surface has a much larger dimension than the WiFi signal's wavelength. This applies to large table tops, walls and ceilings.
- **Diffraction** describes the phenomenon where waves can bend around corners. Typically this happens at sharp edges like the edge of a table or the corner of a wall.
- **Scattering** means that the wave is reflected in multiple different directions resulting in a diffusion of the signal. It typically happens if the obstacle encountered by the wave has a much smaller dimension than the signal's wavelength. In outdoor environments scattering appears at the presence of raindrops, snowflakes, mist droplets and hailstones.

These three phenomena have one great advantage and one great disadvantage. On the one hand, thanks to these phenomena direct unobstructed line-of-sight between the sender and the receiver in a WiFi network is not necessary, since the signal can bounce off walls in order to reach areas behind a large obstacle. On the other hand, these phenomena lead to the fact that one and the same signal reaches its destination multiple times over different paths through the air. All these instances of the same signal reach the intended receiver with a slight time shift, causing interference that can be either constructive or destructive. If two waves featuring identical frequency and amplitude interfere phase-shifted by half the wavelength the resulting signal cancels out. In this case finally no signal at all arrives at the receiver. The phenomenon of a signal reaching a destination over several ways is called *multipath propagation.*

With the previously explained behavior of radio signal propagation in mind we can take a look at another phenomenon, called *slow and fast fading*, that helps understand the behavior of WiFi signal strength in a real-world environment. The measurement data of WiFi received signal strength is normally bound to permanent noise. Sudden boosts and drops in received signal strength are caused by interference (see *multipath propagation*) dependent on only small changes in the relative distance between the sender and the receiver. A movement of the receiving device in the scale of a few centimeters, due to $\frac{\lambda}{2} \approx 6\ cm$, decides on constructive or destructive interference. This phenomenon is called *fast fading.* If the receiving device is moved around a corner behind a wall further away from the transmitting station, the averaged received signal strength gradually decreases due to the shadowing effect of the wall. This phenomenon is called *slow fading* and is evoked by a relative movement of several meters. Typically, slow fading is superimposed by fast fading [ChandraLide06, p. 205].

5 Components of a WiFi pattern

The device used for recording WiFi data throughout the work on this bachelor thesis was a smartphone called *Hero*, manufactured by *HTC Corporation* and released to the European market in July 2009. The *Hero* runs the *Android* operating system version 2.1 developed by *Google Inc.*. At the *Fraunhofer IIS* an application called *BoxRecorder* is commonly used to measure WiFi patterns on *Android* devices and to save them to a text file on the device's flash memory for later processing. *BoxRecorder* records all hearable WiFi access points at a fixed interval of one second. The data collected from one access point includes its MAC address and RSSI value.

MAC stands for *Media Access Control* and is a unique identifier assigned to network interfaces like WiFi access points. Therefore every access point has its own MAC address and thus can be clearly distinguished from other access points. A typical MAC address consists of six groups of two hexadecimal digits.

RSSI stands for *Received Signal Strength Indication* and describes the power level of the WiFi signal reaching the recording device. Signal strength is specified in decibels, which is a logarithmic unit that indicates the ratio of a power value relative to a reference level. In the case of WiFi measurements the signal strength is indicated in decibel milliwatt (dBm), which describes the measured power referenced to one milliwatt (mW). The following formula shows the definition of power level L_P, which describes the power P in proportion to the reference power of one milliwatt [Papamanthou08]:

$$L_P(dBm) = 10 \log_{10}\left(\frac{P}{1\ mW}\right) \tag{3}$$

The following table gives a short overview of transmission and reception power levels typical in the domain of WiFi signals:

13 to 20 dBm	20 to 100 mW	typical transmission power WiFi router
-10 to -30 dBm	100 to 1 μW	max reception power WiFi
-60 to -80 dBm	1,000 to 10 pW	typical reception power WiFi indoor
-80 to -100 dBm	10 to 0.1 pW	typical reception power WiFi outdoor

Regarding the typical reception power of WiFi in outdoor environments it should be mentioned that *Android*-powered devices do not record access points with received signal strengths below -100 dBm. At least the official *Android* SDK (software development kit) does not pass MAC addresses below this level through its API (application programming interface). On a low level in the system there clearly takes place a distinct amount of preprocessing to the WiFi data before it is passed on to the third-party software developer for use in applications like the *BoxRecorder*. Probably, it would be possible to gain access to less processed and more original measured data by investigating the less popular *Android* NDK (native development kit). Written in C/C++ it allows for deeper insight into the system on a low-level basis.

The following listing shows an extract from a text file created by *BoxRecorder* showing MAC addresses and RSSI values from 26 different access points detected within one second. This is an example of a typical WiFi pattern:

```
00.1C.28.2B.F3.57,-78.0          00.26.4D.93.FB.E8,-80.0
```

```
00.04.0E.D8.35.CF,-81.0          00.1C.4A.45.C7.08,-81.0
00.26.4D.66.C3.12,-81.0          00.1C.28.AA.4E.3F,-83.0
00.1C.4A.06.C6.AD,-85.0          1C.AF.F7.83.06.B6,-86.0
00.1F.3F.D7.BD.09,-86.0          00.26.4D.25.6D.85,-88.0
00.1C.4A.48.01.9E,-88.0          00.1B.2F.A6.9D.70,-90.0
00.04.0E.D1.EA.09,-90.0          00.1A.2A.C2.11.C9,-91.0
00.1D.19.06.04.34,-91.0          00.13.49.B0.F5.EB,-92.0
00.1F.3F.0F.40.DB,-93.0          00.04.0E.9C.48.F0,-94.0
00.24.FE.FE.74.01,-95.0          00.1F.3F.45.7F.59,-95.0
00.18.F3.85.69.04,-96.0          00.26.4D.09.73.72,-96.0
00.1C.4A.01.FA.FB,-96.0          00.1A.4F.9B.14.26,-97.0
00.1C.4A.07.52.4A,-97.0          00.16.38.B4.1C.91,-98.0
```

5.1 Functional principle of WiFi pattern recording

A distinction is made between two different types of WiFi scanning modes: *active scanning* and *passive scanning*, where the former is the more commonly used scanning method. When scanning passively the recording device does not transmit any WiFi signals, it only receives them. In this mode the recording device is a quiet listener to the WiFi signals, which are already present in the air due to the communication between other WiFi-enabled devices. This procedure is colloquially called *sniffing*, since the recording device stays invisible to other gadgets. A data packet sent over WiFi does always contain the MAC address of the sender and normally the MAC address of the recipient. These data packets are then intercepted by the recording device, although it is not the intended receiver, the sending MAC address is read from the packet header and the received signal strength is measured.

During active scanning the recording device becomes visible or rather hearable to the WiFi-enabled devices in its environment. The recording device broadcasts a so-called *probe request*, which contains its return address but no address of an intended receiver, since the recording device still does not know about its surrounding gadgets and their MAC addresses. Next all the devices that have received this probe request will reply to it by sending back a so-called *probe response* containing their MAC address, network name (SSID), encryption type used (i.e. WEP, WPA, WPA2) and other information. While the recording device receives all those probe responses it can measure each signal strength and has consequently learned about the WiFi pattern at its current location.

6 Filters for preprocessing

Since the *BoxRecorder* is a digital product, it produces time-discrete measurement data, i.e. WiFi patterns are being recorded once per second. If you plot the amount of different MAC addresses heard in one second or the sum of received signal strengths from a real-life measurement, i.e. you draw a line through all the measuring points, the resulting curve shape will have a very noisy appearance. WiFi patterns are subject to strong fluctuations, due to environmental influences manipulating signal dispersion, resulting in random signal noise. From one second to another, MAC addresses appear and disappear on the list of heard access points and the received signal strength of one access point can vary significantly, even if you are standing still at a fixed position without any movement. In order to make the plotted measurement data more readable and to make it easier to visually detect major trends in the signal curve, it is essential to apply certain preprocessing steps to the original measurement data.

6.1 Exponential smoothing

A common method to smooth out a noisy signal curve is to apply a *low-pass filter* on it. A low-pass filter blocks out high frequencies, that means sudden changes, and only allows low frequencies, that means gradual but constant changes, to pass through. A simple low-pass filter for discrete-time application is the *exponentially-weighted moving average*, originating in resistor/capacitor circuits from the field of electrical engineering. Its formula [Hyndman08, p. 13, (2.2)] looks like this:

$$y_i = \alpha x_i + (1-\alpha)\, y_{i-1} \qquad \textit{where} \quad 0 < \alpha < 1. \tag{4}$$

α represents the smoothing factor whereas a low value towards zero means a greater smoothing impact. In order to understand this, the operating principle of the filter has to be comprehended. The filter weights the current value x_i with α and the predecessor result from the second before y_{i-1} with 1-α. A value of 0.1 for α has delivered satisfying results throughout different applications during work on this bachelor thesis. In this case the new value only has a 10% impact on the result while the *history value*, compound of past iteration steps, has 90% weight. The compound nature of the *history value* explains the exponential feature of this filter:

$$\begin{aligned}
y_i &= \alpha x_i + (1-\alpha)\, y_{i-1} \\
&= \alpha x_i + (1-\alpha)(\alpha x_{i-1} + (1-\alpha) y_{i-2}) \\
&= \alpha[x_i + (1-\alpha)x_{i-1}] + (1-\alpha)^2 y_{i-2} \\
&= \alpha[x_i + (1-\alpha)x_{i-1} + (1-\alpha)^2 x_{i-2}] + (1-\alpha)^3 y_{i-3} \\
&\ldots
\end{aligned}$$

Here the equation has been recursively inserted into itself with adjusted indices to represent the values of one more step back in the past. As you continue the substitution, you see that the result value of this formula y_i consists of the weighted average of all the past observation values x_i, x_{i-1}, x_{i-2}, ... The weight assigned to these past observation values 1, $1-\alpha$, $(1-\alpha)^2$, ... is getting exponentially smaller the more those observation values lie back in the past.

Unfortunately, this filter causes a time delay as it reacts late to significant changes in the signal curve. A solution to overcome this issue is to apply the filter twice: forwards in the first step and backwards in the second step. Consequently, the filtered curve shape is neatly aligned to the course of the original curve.

6.2 Access point history filter

There is a constant coming and going of MAC addresses in the list of heard access points. Frequently, a MAC address drops out of the list and emerges shortly afterwards, i.e. a few seconds later. Normally, access points with low received signal strength are more prone to this behavior than access points with a strong reception. However, this phenomenon does also happen to the last-named access points evoked by signal cancellation caused by interference during wireless transmission. To overcome this issue the *access point history filter* fills these temporal gaps of short-term absent access points. The filter keeps MAC addresses in memory, i.e. adds them to subsequent WiFi patterns, even though originally they have not been recorded any more. The timeframe of how long MAC addresses are still considered in subsequent calculations can be adjusted. Practical experiences have shown that a duration of five seconds delivers good results, as it meets the requirements of real world conditions.

7 Principle of the gameplay

In the course of this collaborative project consisting of three bachelor theses, many algorithms have been developed that operate on the output data from sensors measuring acceleration, magnetic field, WiFi, GPS and GSM. The development of these algorithms took place with the idea of a real-life game in mind. The principle of the gameplay is inspired by a mixture of a paperchase, geocaching and the board game *Scotland Yard: The Hunt for Mister X*, by the German game company *Ravensburger Spieleverlag GmbH*. One player represents *Mister X* and several other players assume the role of agents who spy on *Mister X* and try to catch him as they trace his escape route.

Our team has developed two different approaches to the gameplay: one simplified and one advanced. The simplified approach is called the *static gameplay* where the agents trace a path, which *Mister X* has walked a long time ago, i.e. between path creation and path tracing lies an extended period of time. As the creation of *Mister X*'s path is finalized before the agents begin to trace it, the entire path is predetermined and completely known to the system. The situation is different with the advanced approach, which is called the *dynamic gameplay*. Here *Mister X* is chased in real-time, which means that the path to be traced is constantly evolving over the course of the game. *Mister X* and the agents start at the same location, whereas *Mister X* is granted a short temporary head start. The game is considered finished if *Mister X* and at least one agent are located at the actual same position, or if *Mister X* succeeds in staying uncaught for a certain period of time. In the static gameplay there is no time pressure and the agents can take as long as they need to reach the end of the path.

7.1 Technical principle of path tracing

The assessment if an agent is located on *Mister X*'s path, called *reference path* from now on, and the assessment of where an agent is located on the reference path is done by comparing WiFi patterns. Nevertheless, there is one fundamental principle that is absolutely essential for the understanding of this approach to path tracing: We do not know anything about geographical coherences. To understand this, we need to take a look once more at the way modern smartphones do WiFi positioning:

Location $\Rightarrow$ typical measurement data $\Rightarrow$ coordinates $\Rightarrow$ semantic meaning of the location

At a certain location the smartphone records WiFi measurement data that is typical for this specific location. This WiFi pattern corresponds to specific geographical coordinates (latitude, longitude) looked up from a database (see Section 3). These geographical coordinates can have different semantic meanings depending on the context, e.g. at the coordinates 49.45, 11.083333 is situated the city center of Nuremberg. As opposed to this, the way WiFi positioning works within the framework of our game looks like this:

Location $\Rightarrow$ typical measurement data $\Rightarrow$ semantic meaning of the location

The coordinates as geographical reference have dropped out. Here the recorded WiFi patterns are directly correlated to the semantic meaning of the location, e.g. at the location where this specific WiFi data has been measured *Mister X* has been there at second 500 out of his 2000-second-long trail. Notice that the semantic meaning has a temporal reference not a geographical one. The system only knows which WiFi pattern has been heard in which second and finally has to assess the path tracing process based on this information alone. This is what makes our game concept such a novel approach compared to conventional GPS-based real-life games[3]. Please make sure you have understood the last paragraph thoroughly before you continue reading.

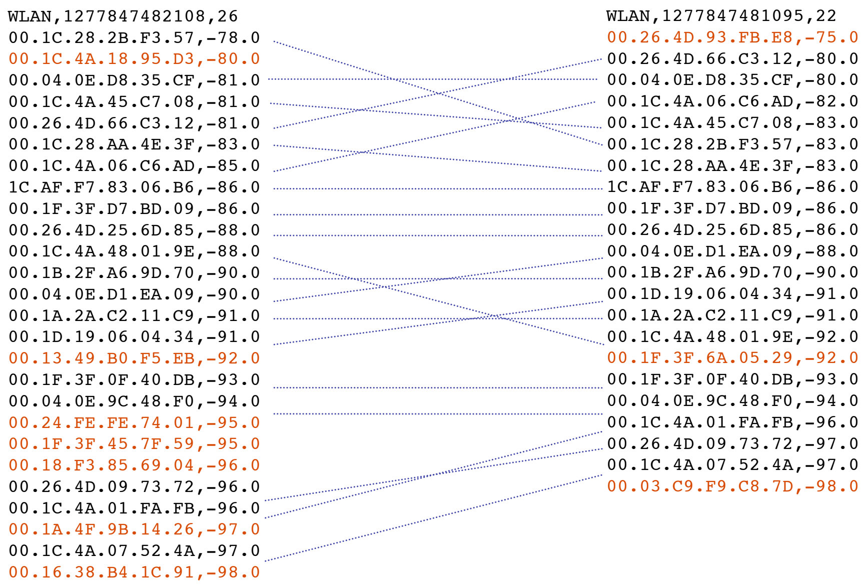
The application scenario from a technical point of view looks like this: While *Mister X* flees through the streets the current WiFi pattern is recorded every second. The agents' devices are supplied with this reference path of WiFi patterns, either as a whole in the static gameplay or continuously in the dynamic gameplay. Each of the agents' devices is also recording own WiFi patterns by itself whereby it gathers information about its current whereabouts. Eventually, the most recent WiFi pattern recorded by an agent is compared to all the WiFi patterns recorded by *Mister X* available up to that point once a second. The second of the best matching WiFi pattern, which is the pattern with the

[3]In 2009 students at the University of Bonn released the game *Mister X Mobile* for *Android* and *iPhone*. However, the technology used here for location determination is GPS delivering absolute coordinates presentable on a geographical map.

highest similarity, represents the current position of the agent on the reference path. The degree of resemblance of this best matching pattern is a measure of the probability that the agent is actually situated on the reference path.

8 Estimating the probability of the agent to be situated on the reference path

The procedure of WiFi pattern comparison should be explained with an example from real-life shown in Figure 1.



```
WLAN,1277847482108,26
00.1C.28.2B.F3.57,-78.0
00.1C.4A.18.95.D3,-80.0
00.04.0E.D8.35.CF,-81.0
00.1C.4A.45.C7.08,-81.0
00.26.4D.66.C3.12,-81.0
00.1C.28.AA.4E.3F,-83.0
00.1C.4A.06.C6.AD,-85.0
1C.AF.F7.83.06.B6,-86.0
00.1F.3F.D7.BD.09,-86.0
00.26.4D.25.6D.85,-88.0
00.1C.4A.48.01.9E,-88.0
00.1B.2F.A6.9D.70,-90.0
00.04.0E.D1.EA.09,-90.0
00.1A.2A.C2.11.C9,-91.0
00.1D.19.06.04.34,-91.0
00.13.49.B0.F5.EB,-92.0
00.1F.3F.0F.40.DB,-93.0
00.04.0E.9C.48.F0,-94.0
00.24.FE.FE.74.01,-95.0
00.1F.3F.45.7F.59,-95.0
00.18.F3.85.69.04,-96.0
00.26.4D.09.73.72,-96.0
00.1C.4A.01.FA.FB,-96.0
00.1A.4F.9B.14.26,-97.0
00.1C.4A.07.52.4A,-97.0
00.16.38.B4.1C.91,-98.0
```

```
WLAN,1277847481095,22
00.26.4D.93.FB.E8,-75.0
00.26.4D.66.C3.12,-80.0
00.04.0E.D8.35.CF,-80.0
00.1C.4A.06.C6.AD,-82.0
00.1C.4A.45.C7.08,-83.0
00.1C.28.2B.F3.57,-83.0
00.1C.28.AA.4E.3F,-83.0
1C.AF.F7.83.06.B6,-86.0
00.1F.3F.D7.BD.09,-86.0
00.26.4D.25.6D.85,-86.0
00.04.0E.D1.EA.09,-88.0
00.1B.2F.A6.9D.70,-90.0
00.1D.19.06.04.34,-91.0
00.1A.2A.C2.11.C9,-91.0
00.1C.4A.48.01.9E,-92.0
00.1F.3F.6A.05.29,-92.0
00.1F.3F.0F.40.DB,-93.0
00.04.0E.9C.48.F0,-94.0
00.1C.4A.01.FA.FB,-96.0
00.26.4D.09.73.72,-97.0
00.1C.4A.07.52.4A,-97.0
00.03.C9.F9.C8.7D,-98.0
```

Figure 1: Comparison of two different WiFi patterns. Lines indicate corresponding MAC addresses. MAC addresses in red color do not have a matching partner.

The example presents a reference pattern with 26 heard access points and an agent's pattern with 22 heard access points. Altogether there are 19 corresponding MAC addresses in both WiFi patterns. However, on one hand the reference pattern features 7 access points that have not been heard by the agent's device and on the other hand there are 3 MAC addresses in the agent's pattern that do not appear in the reference list. Now the question is how well do these two WiFi patterns coincide? The first attempt to find a measure for the conformity is to simply ignore signal strength values and to base the comparison on the amount of corresponding MAC addresses only:

$$P = \frac{19}{26} \approx 0.73 \Rightarrow 73\% \text{ conformity} \tag{5}$$

This trivial attempt has two disadvantages: First, the amount of access points recorded by the agent's device that do not match to the reference pattern is not considered at all. Whether the agent's WiFi pattern contains 3 or 10 non-matching MAC addresses is not reflected in the final conformity value. However, a pattern with only 3 non-matching MAC addresses should be rated as more compliant than a pattern with 10 non-matching MAC addresses, regardless of the amount of corresponding access points. Second, the reception quality of heard access points remains disregarded. Nevertheless, the received signal strength is an important criterion as the following comparison shows:

```
00.26.4D.93.FB.E8,-75.0             00.03.C9.F9.C8.7D,-98.0
```

If you take a look at these two access point data, you will notice that the former access point delivers a much higher signal strength than the latter. This means the former access point has a great dominance in the WiFi landscape, whereas the latter access point is listed next to not heard at all (remember that the *Android* API only lists access points above the -100 dBm limit). It is likely that one second later in the next recorded WiFi pattern the latter access point will not be listed anymore while the probability is high for the former access point to still make an appearance in the next pattern. In fact, there is a lot of fluctuation going on towards the lower end of the listed WiFi pattern, assuming it is sorted by received signal strength in descending order while the top of the list stays rather stable. Eventually this means that it is much more severe for the assessment of conformity if a WiFi pattern does not contain the former MAC address than it is if it would not have heard the latter access point. Therefore, we need a solution to take into account the received signal strength of each access point.

The approach discussed next can be summarized by the term *summed up differences in signal strength levels.* The signal strength values of compatible MAC addresses from the two WiFi patterns are subtracted from each other and the absolute values of all these differences are added up. If one pattern features a MAC address that the other pattern does not, this access point is assumed to be present at a signal strength of -100 dBm in the pattern where it was originally not listed. The smaller the summed up differences the higher the accordance between the two patterns. A sum of zero would imply 100% conformity. In the worst case scenario not one MAC address matches at all, which results in a sum of 284 in the previously mentioned example. The sum of 284 is calculated by subtracting all signal strength values of the example reference pattern from -100 dBm and adding up the differences. Actually, this sum will even be greater the more non-matching MAC addresses the agent's pattern features. Nevertheless, a sum of 284 or higher leads to a 0% conformity. To come back to the example, the degree of conformity calculated with the new approach results in this:

$$P = \frac{103}{284} \approx 0.36 \text{ non-conformity} \Rightarrow 64\% \text{ conformity} \tag{6}$$

This result is slightly different from the initial calculation and now it is not certain to tell which result is the more accurate one. However, practical experience has shown that the advanced method of calculation copes reasonably with real-life WiFi signal behavior and delivers believable valuations that seem to reflect real-life incidents quite well. This means the conformity value drops the moment the agent departs from the reference path and rises again when the agent returns to the reference path. The usability of this algorithm has been evaluated by a test run in the historical downtown of Nuremberg. In the course of the path tracing the agent has deliberately deviated from the reference path every now and then and returned back to the track shortly afterwards. In the aerial view of Figure 2 you see indicated two distinct path progressions: The trail in blue color represents the reference path while the red track signifies the agent's trail.

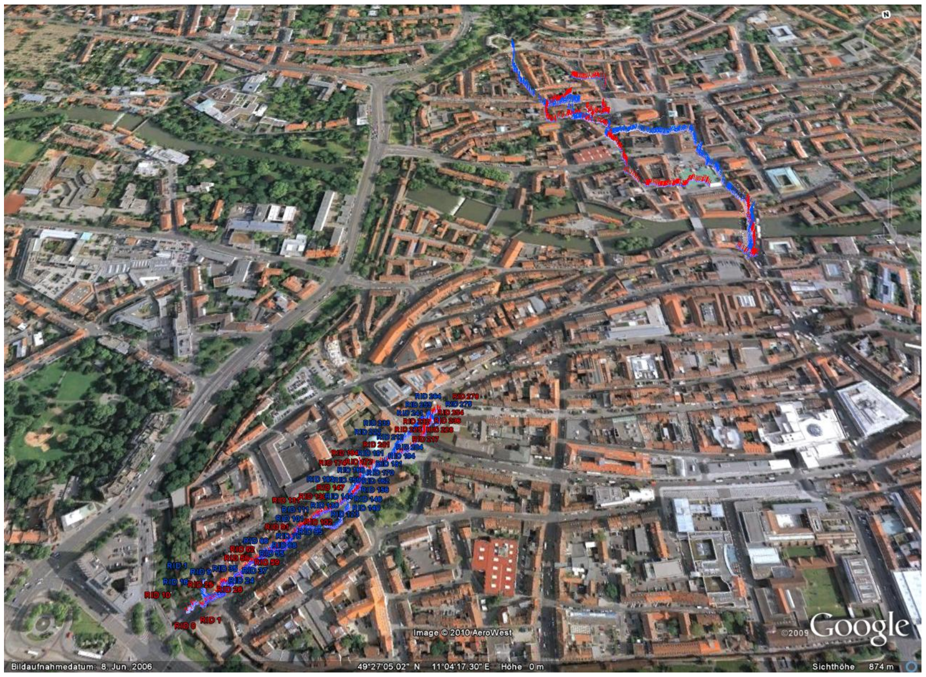


Image © 2010 AeroWest
©2009 Google
Bildaufnahmedatum: 8. Jun. 2006
49°27'05.02" N 11°04'17.30" E Höhe 0 m
Sichthöhe 874 m

Figure 2: Example trail in the historical downtown of Nuremberg walked twice partially with deviations

As you can see, the first part of the reference path, originating in the lower left corner, has been followed accurately. Hereafter a subway ride ensued, which is why no GPS data could be recorded. Subsequently, the agent followed the reference path for another short segment but finally deviates significantly: The agent passed the central marketplace from below, while *Mister X* passed it from above. Thereupon the two paths reunite and so forth. The plot in Figure 3 shows how the algorithm evaluated the process of this exemplary path tracing:

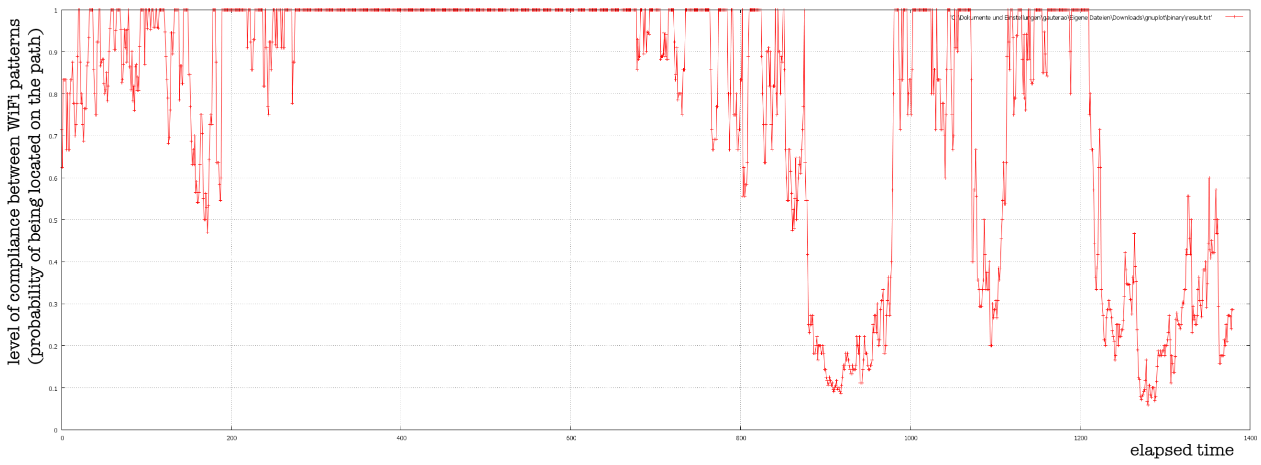


Figure 3: Probability of the agent to be situated on the reference path shown in Figure 2

The abscissa represents the chronological process of the path tracing in seconds (which is identical to the index of the measured WiFi pattern), while the ordinate indicates the probability of the agent to be situated on the reference path which means technically the degree of compliance of the agent's current WiFi pattern with the best matching WiFi pattern in the reference path. During the subway ride, which took place between second 250 and 650, no valid WiFi pattern could be obtained due to the absence of WiFi signals on the subway track[4]. While the degree of conformity generally stayed above 50% in the first part of the path tracing process, it suddenly drops to 10% compliance around second 900, which reflects the agent's deviation from the reference path around the central marketplace. Soon after, the compliance meter surges back to 70 to 100% as the agent heads back to the reference path. The question, starting from which compliance level the system should assume that the agent has actually left the reference path, is not easily answered. As practical experience has shown, even if the agent is situated directly on the reference path the degree of conformity varies between 50 and 100% due to interference phenomena between

[4]In 2008 the Nuremberg Transport Corporation (VAG) introduced autonomically operating subway trains outside the downtown area. These coaches stream live CCTV footage via WiFi to the supervisory center for security purposes. Maybe WiFi positioning would be possible on these subway lines. See [Kawaguchi09] for a test run of WiFi positioning in the subway.

WiFi signals (see Section 4). If the determination of a threshold is necessary for the gameplay a value of 30% should be reasonable. If the compliance level stays below this threshold for longer than a few seconds the probability is high that the agent has indeed left the reference path.

9 Estimating the position of the agent on the reference path

An agent who is chasing after *Mister X* does not only want to know if he is still walking along his escape route but would also like to have a measure of his progress at his disposal. The agent would like to know how close he is on *Mister X*'s heels (in the dynamic gameplay) or how far away the end of the reference path still is (in the static gameplay). Possibly the agent walks indeed along the reference path, however in the wrong direction. What the agent needs are means to orientate himself. However, as we have discussed in Section 7.1, the agent cannot be provided with a geographical orientation. The agent's current position on the reference path can only be stated in a temporal manner like, *You are standing at second 500 of the 2000-second-long reference trail, which equals* 25% *progress.* This means the agent is standing at the geographical position that *Mister X* has reached after 500 seconds of walking since he started recording his escape route. Be aware that the question of *where* on the reference path is misleading and is used here with a different meaning.

Actually, the position information has incidentally already been calculated in the previous Section 8. The algorithm introduced there picks out the WiFi pattern from the reference path, which matches best with the agent's latest recorded WiFi pattern. The information how well this WiFi pattern matches indicates the probability of the agent to be situated on the reference path whereas the index of this WiFi pattern (the second when it has been recorded by *Mister X*) indicates the agent's position on the reference path. As described in the previous section, the algorithm can operate with two different approaches in order to find the best matching WiFi pattern: Either the amount the matching MAC addresses has to be maximal, or the summed up differences of signal strength values have to be minimal. To better understand this selection process of the best matching WiFi pattern, it should be visualized with the aid of real-life measurement data. Imagine the agent has accurately followed the reference path with a length of 360 seconds for 185 seconds. The plot in Figure 4 visualizes the result of the algorithm operating with the simple method of MAC address comparison at second 185:

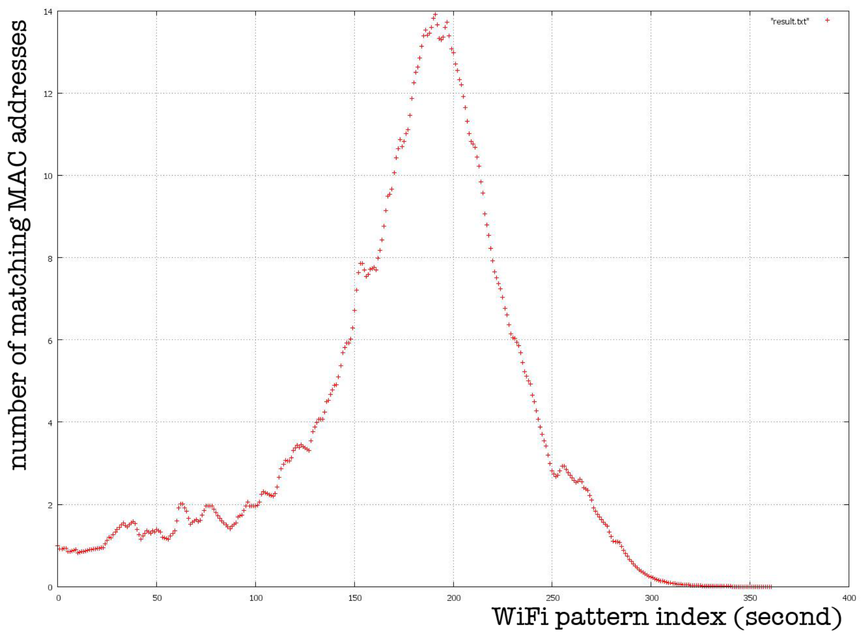


Figure 4: WiFi pattern recorded by agent at second 185 compared to every WiFi pattern from the reference trail with simple MAC-address-only approach (data has been low-pass filtered)

Here the WiFi pattern recorded by the agent at second 185 is compared against each WiFi pattern of the reference path. Assuming the agent's walking speed has been constant over time, the index of the WiFi pattern with the highest amount of matching MAC addresses is approximately 190, which is fairly close to the input index of 185. However, it has to be taken into account that under certain circumstances this result can be a fallacy. If for example, the agent had to wait at a red traffic light from second 70 to second 90, which did not happen when *Mister X* walked this path, the agent will from second 90 on arrive 20 seconds later at the geographical positions where *Mister X* used to be (assuming constant walking speed). In this case after 185 seconds of walking, the system would tell the agent that his position on the reference path was second 165. This example illustrates why the result of this position determination cannot be used as a reference for localization accuracy which is discussed in the subsequent Section 9.1. The plot in Figure 5 visualizes the result of the algorithm operating with the advanced comparison method incorporating signal strength values at second 222:

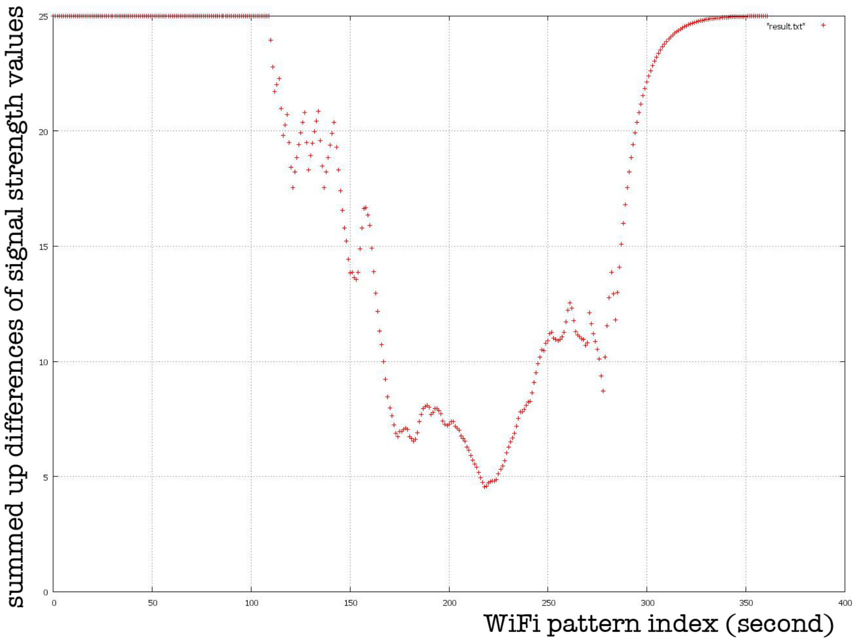
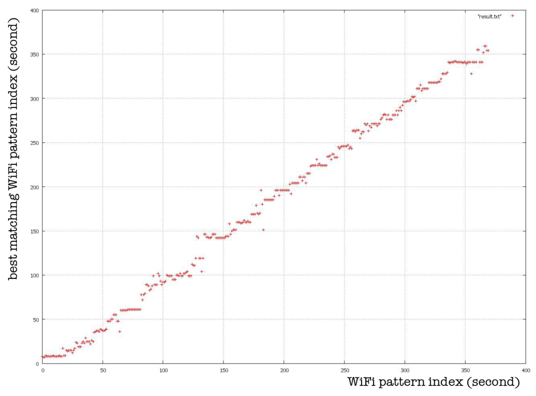
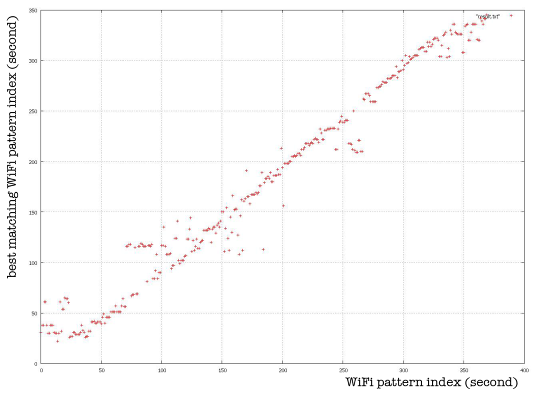


Figure 5: WiFi pattern recorded by agent at second 222 compared to every WiFi pattern from the reference trail with advanced approach incorporating RSSI values (data has been low-pass filtered)

Here the smallest sum of added up differences of signal strength values has the WiFi pattern with the index 220, which again is fairly close to the input index of 222. Assuming the agent has followed the reference path precisely until the end, which took him roughly the same amount of time as *Mister X*, and we plot the positioning results of every second from the agent's path tracing, we will get a diagonal line like in Figure 6, which represents calculation results from both algorithmic operation modes respectively. As you can see, the diagonal line from the advanced operation mode features more unwanted scattering than the diagonal line from the simple operation mode. The incorporation of signal strength values into the calculation process may decrease the quality of the result. This is because signal strength values are unpredictable, due to interference between WiFi signals (see Section 4). The criterion of heard MAC addresses offers much more stable behavior and features an unbeatable performance to implementation effort ratio.



(a) MAC addresses only



(b) including RSSI values

Figure 6: Positioning results of each second from the agent's path tracing. For an enlargement see Figure 20 in the appendix.

9.1 Localization accuracy

Although it has been mentioned several times that we do not know anything about geographical coherences (see Section 7.1) nevertheless it would be interesting to evaluate the localization accuracy of the algorithm, which has been applied in the previous Section 9. For this purpose we need a method to convert the temporal position information measured in seconds to geographical metrics in the unit meter. A practical approach was the placement of checkpoints along a predetermined path about every 100 meters at distinctive positions like crossroads. The exact distances between successive checkpoints has been measured by the help of *Google Earth*. After this preparation phase the predetermined path has been followed twice. On reaching each checkpoint a marker has been set in the *BoxRecorder* software, which saves the time information in seconds. As we know the metric distance between checkpoints and the time it took to get from one checkpoint to the next we have established a connection between position information in seconds and in meters. Positions between two checkpoints are linearly interpolated. Now we can calculate the distance in meters between a position information given in seconds from the first time walking the path and a position information given in seconds from the second time walking the path.

For example, imagine the algorithm from Section 9 has determined that the WiFi pattern recorded at second 185 during the second time walking matches best with the WiFi pattern recorded at second 178 during the first time walking. Assuming second 185 of the second time walking corresponds to meter 278 on the predetermined path and second 178 of the first time walking corresponds to meter 267, the position estimation of the algorithm yields an error of 11 meters in this specific example. If you calculate the positioning errors for the whole path you get the total mean error, which is a measure for the localization accuracy of this algorithm. Two test runs in different types of environment have been conducted: The first test run took place in the residential area of *Ziegelstein*, which is a district of Nuremberg where the environment is characterized by single family detached houses. Here the algorithm delivered a mean localization accuracy of about 10 meters, the 90% quantile was 21 meters while the 95% quantile was 28 meters [5].

The second test run took place in the historic city center of Nuremberg around the central marketplace. Here the algorithm delivered a mean localization accuracy of about 8 meters, the 90% quantile was 19 meters while the 95% quantile was 24 meters. The reason why the algorithm delivered an even higher localization accuracy in the second test run is the amount of available access points, which was about 30 on the average in the downtown area while it was only 10 in Ziegelstein. The more different MAC addresses present in a WiFi landscape the better the possibilities for high localization accuracy. Please keep in mind that these localization accuracy values are the result of one-dimensional determinations of the position along the reference path, this is not a two-dimensional location determination on a map. This is why the mean localization error seems to be so remarkably low here when compared to conventional WiFi positioning systems (WPS) [Cheng05] like the one developed by *Fraunhofer IIS*, which features an accuracy of 20 to 30 meters according to their own statements[6].

9.2 Aging of WiFi patterns

Normally in the static gameplay the agents will start tracing the reference path shortly after it has been created by *Mister X*. An interesting question is, how well will the path tracing work if there lies a long lapse of time, maybe a year, between the creation and the tracing of the reference path? How reliable will be the information which the algorithm calculates to support the agents during the path tracing process (if on path and where on path)? In the course of time WiFi landscapes change: Old MAC addresses disappear and new ones emerge as people replace their obsolete access points with more modern models. This entails disadvantages for the quality of WiFi positioning since newly recorded WiFi patterns cannot reach the same high level of compliance as it used to be possible at the time of creation of the reference path. To investigate this effect more precisely it has been attempted to simulate the aging process of WiFi patterns. This has been achieved by replacing MAC addresses from the original WiFi pattern with fictional MAC addresses, which should be an artificial modification close enough to real-life proceedings. The amount of altered MAC

[5] If the 95% quantile is 28 meters this means that 95% of the positioning results feature an accuracy of 28 meters or better. The 95% quantile is determined by sorting the positioning results by accuracy in descending order and cutting off the lower 5% of the list. The now last value of the list is the 95% quantile.

[6] According to a paper titled *Structure of the Fraunhofer IIS WLAN algorithm* which is for internal use only and not publicly accessible.

addresses in one WiFi pattern defines the degree of aging. The following table shows the results of an aging experiment relating to the localization accuracy using the data recorded during the second test run mentioned in the previous Section 9.1:

Degree of aging	Mean accuracy	90% quantile	95% quantile
0%	8 meters	19 meters	24 meters
25%	9 meters	20 meters	25 meters
50%	10 meters	22 meters	28 meters
75%	23 meters	47 meters	99 meters
95%	82 meters	189 meters	225 meters

An aging degree of 95% means that only one access point per WiFi pattern is left featuring its original MAC address. If you take a look at the table you will notice a surprising fact: The localization accuracy does not get any worse than 82 meters in the average. The moment that each and every MAC address would be replaced by a fictional one a location determination would of course not be possible at all. However, as long as at least one original MAC address per WiFi pattern remains a reasonable localization result is still available. This phenomenon will be explained with the aid of the diagram in Figure 7 illustrating a simplified real-life scenario:

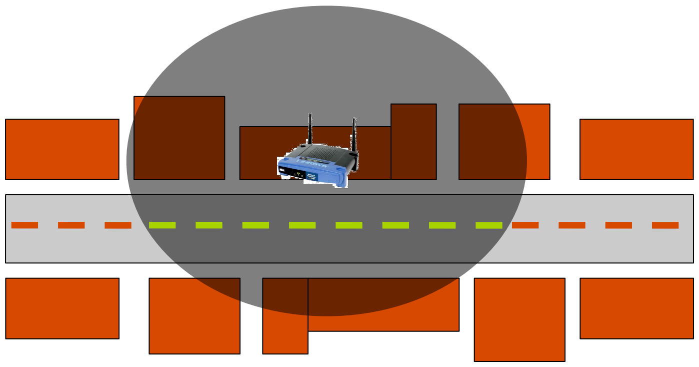


Figure 7: Simplified real-life scenario for an explanation on WiFi access point range

Here we see a street with buildings on each side. The reference path represented by the dashed line runs through this street. In one of the buildings is situated an access point whose WiFi coverage is indicated by the gray translucent ellipse. The part of the reference path where the dashed line is red lies outside the coverage area of the access point while in the part of the reference path where the dashed line is green the access point can be heard. If this access point's MAC address appears in the latest recorded WiFi pattern and it is therefore known that this access point could be heard then we know that the agent must be located somewhere on the path segment indicated by the dashed line in green color. If we assume that the access point radiates WiFi signals 40 meters up the street and 40 meters down the street then the segment of the reference path where this access point could be heard has a length of 80 meters. Consequently, the one-dimensional localization accuracy cannot become worse than 80 meters. This worst case scenario would occur if the agent stood in fact at the right end of the green dashed line; however, the positioning algorithm assumed he would be standing at the left end of the green dashed line. In order to improve the precision of localization in this case the incorporation of signal strength values would be helpful.

The following thought experiment illustrates the relationship between the range of access points and the accuracy of WiFi positioning: If the access point in the preceding example would have featured a smaller coverage area the green dashed line would be shorter and therefore the localization accuracy would be higher since the largest possible error would be smaller. This leads to the rule of thumb already mentioned in Section 2: Better many distinguishable access points each with a short range than few distinguishable access points each with a long range.

9.3 Device calibration

As mentioned in Section 5 the device used for all measuring sessions involved during this bachelor thesis was the *HTC Hero*. However, what happens to the quality of the results delivered by the algorithm (if on the path and where on the path) if the agents use a different device model than the one used by *Mister X*? Different types of devices have different characteristics of receiving WiFi signals since the built-in WiFi chips feature different sensitivities. For example, a recording device with an external antenna attached will deliver overall higher received signal strength values. To what extent do more (or less) sensitive WiFi chips influence the algorithmic results?

For this thought experiment we assume the device of an agent to record WiFi patterns with 10 dBm higher signal strength values. The assessment of where on the reference path the agent is located remains unaffected since the best matching WiFi pattern is chosen based on a relative comparison against all other WiFi patterns. If the sum (of added up differences of signal strength values) of each WiFi pattern is increased by the same amount the proportion between the sums stay identical. Therefore, even after a great increase in signal strength values the selection of the best matching WiFi pattern will be the same. However, what will change is the probability of the agent to be situated on the reference path since the degree of compliance (calculated using absolute values) will decline as the sum of added up differences of signal strength values rises. Due to this discrepancy some kind of device calibration seems to be appropriate.

10 Urban environment assumptions

This section introduces attempts to extract additional information about the environment *Mister X* wanders through. The initially discussed algorithm tries to detect crossroads along the reference path by analyzing successive WiFi patterns while the second presented algorithm estimates the degree of urban development in proximity to *Mister X*. The latter investigation is based on signal strength values from GPS satellites and GSM cell towers. Both algorithms generate helpful hints, which support the agents even further during path tracing.

10.1 Crossroad detection

The theoretical fundamental idea of crossroad detection is simple: Imagine you are walking along a straight street flanked by high buildings without any gaps between them and your device receives on average 20 different access points. After a while you reach a crossroad and while you traverse it straight ahead your device suddenly hears 30 different access points. After you have left the intersecting street behind you the amount of access points drops back down to 20. Due to the high gapless buildings on each side of the street this type of development evokes the formation of a tunneling effect, which means that WiFi signals propagate alongside the street canyon. If you are standing in the middle of a crossroad, you are situated on the intersection of two street canyons, each one of them featuring their own access points. This way the recording device logically captures a higher than average amount of access points the moment you step onto a crossroad. This rise and fall of heard access points is exploited by the crossroad detection algorithm.

Now, the challenge is to distinguish real crossroads from seeming crossroads, since not every growth in the amount of heard access point must be an indication of a crossroad. The occurrence of these so-called *false-positives* (the algorithm assumes the existence of a crossroad although no crossroad is present in real-life) must be prevented as far as possible. Three different approaches have been developed to detect significant increases and decreases in the signal curve of either the amount of heard MAC addresses or the sum of received signal strength values. Each approach makes use of one of the following mathematical tools:

1. Dynamic mean as low-pass filter
2. Sum of mean and variance within a sliding window
3. High and low points of the signal curve

Strictly speaking, the totaling of received signal strength values does not make sense from the viewpoint of physics since the question *What meaning does the sum of signal strength values from individual access points have?* remains unresolved. However, since this technique has proved advantageous and delivers reasonable results it is an accepted tool for WiFi signal processing.

The **first approach** uses the dynamic mean as a low-pass filter which is in fact the *exponentially-weighted moving average filter* mentioned in Section 6.1. The algorithm is looking for large coherent segments of the measurement curve that are situated above the mean curve. The area between the measurement curve and the mean curve must be greater than a specific threshold to be concidered a crossroad. The plot in Figure 8 shows measurement data from a single crossroad traversal.

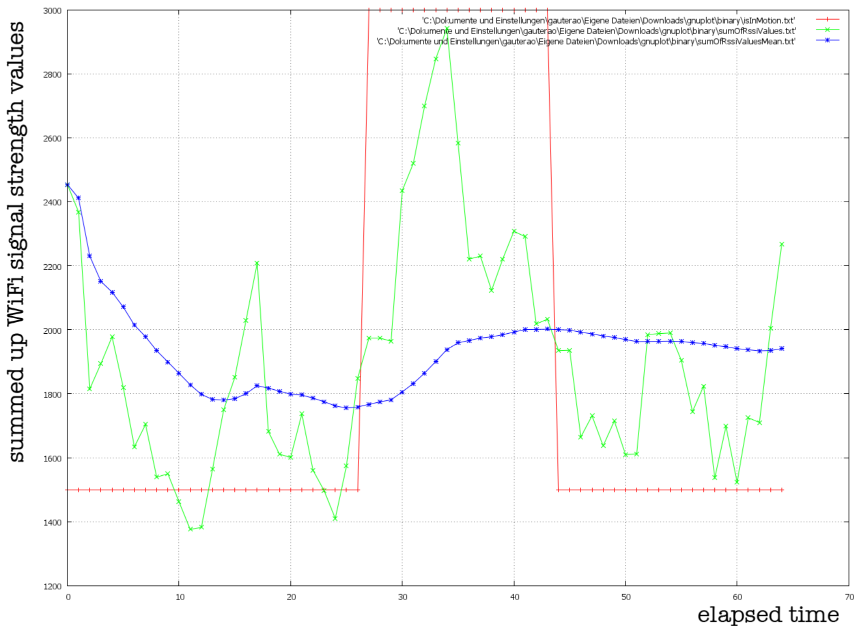


Figure 8: WiFi measurement data from a single crossroad traversal

The green line represents the measurement data as summed up signal strength values from all heard access points. The blue line indicates the dynamic mean of the summed up signal strength values while the red line simply marks the beginning and the end of the crossroad as seen during recording. As you can see there are a few small areas above the mean curve, which should be ignored by the algorithm. However, the real crossroad flanked by the two vertical red lines makes the measurement curve dominate the mean curve for about 20 seconds. The resulting area is large enough to be assessed by the algorithm as a potential crossroad. With the previously shown plot being an isolated example of only one crossroad Figure 9 presents a test run along several blocks of houses in the southern part of Nuremberg.

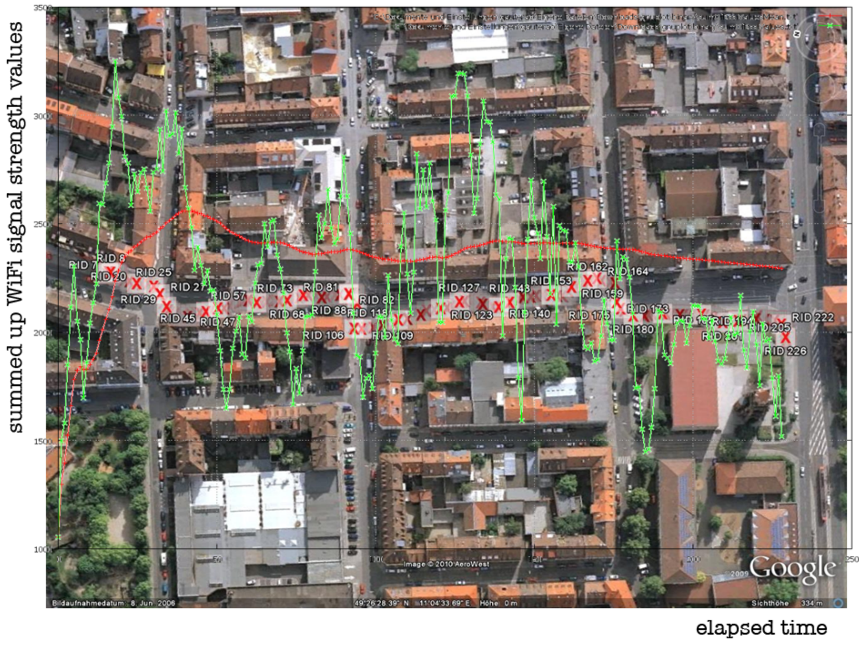


Figure 9: Overlaid WiFi measurement data from a test run along several blocks of houses

The aerial view is overlaid with the precisely fitting curve of summed up signal strength values. The red crosses indicate the course of the path. As you can see, the places where crossroads are present feature large portions of the measurement curve above the mean curve. However, difficulties are caused by the following situations:

- Entries to backyards are like one-sided crossroads. The accumulated WiFi signals from all the apartments in the backyard emerge from the gateway and deceive the algorithm (refer to the beginning of the example measurement curve).
- Low rooftops between high rooftops evoke the same effect as entries to backyards. Accumulated WiFi signals from the backyard radiate over the lower rooftop causing an increase in summed up signal strength values and the amount of heard access points respectively (take a look at the middle of the example measurement curve).
- Large public squares and parks hamper the detection of crossroads. If a crossroad lies directly at the side of a square or park, the block of houses at the fourth corner is missing resulting in an overall lower amount of

WiFi signals (see the end of the example measurement curve). There are indeed access points emitting WiFi signals over the square or park from the other side, however their received signal strength is relatively low due to the distance.

- Passing a student hostel almost certainly fools the algorithm in assuming the existence of a crossroad. Nowhere else is the density of access points per square meter that high, since nearly every student appartment features its own access point.

The **second approach** to detect crossroads uses the technique of a sliding window. This window has a specific width and traverses from left to right over the measurement curve. Within this window, the sum of mean and variance is calculated and used as a threshold value above which measurement data has to lie to be considered part of a potential crossroad. The added up combination of mean and variance delivers a threshold value, which cuts off common noise reasonably. Measurement data that is even higher than this threshold value normally belongs indeed to an existing crossroad. The plot in Figure 10 represents the measurement data recorded during a 35 minute walk (2107 seconds as seen on the abscissa) through the southern neighborhood of Nuremberg. The ordinate indicates the quantity of heard access points.

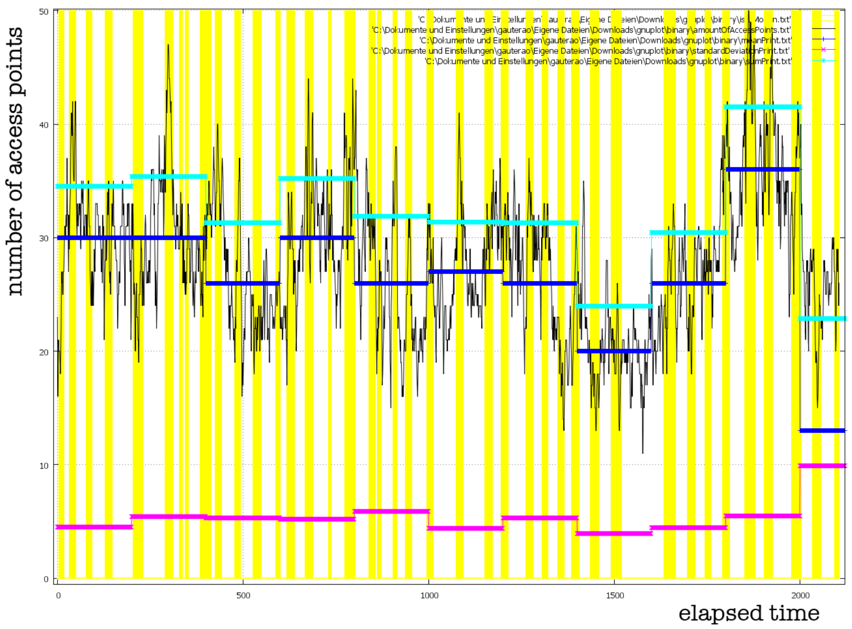


Figure 10: Crossroad detection on a 35-minute-long trail via the sum of mean and variance

Dark blue lines indicate the mean, magenta lines show the variance while cyan lines stand for the sum of mean and variance. The yellow vertical bars represent the nearly 40 crossroads encountered during path recording. The window size used here is a width of 200 seconds. Actually, it is recommended to traverse the measurement curve multiple times applying different window sizes. This way the algorithm is able to capture both small and large crossroads. A small crossroad is characterized by a short traversal duration and a not so high dominance of WiFi signals while a large crossroad is characterized respectively. If you use a large window size the algorithm will miss small between large crossroads as the calculated threshold value, appropriate for the detection of larger crossroads, may be too high for smaller crossroads and thus they will be overlooked. On the other hand, if you choose too small a window size the algorithm will have difficulties to detect large crossroads properly, as they are too big to fit into the narrow window size and thus will be cut off on one or both sides.

The **third and last approach** to detect crossroads is oriented towards the high and low points of the measurement curve. The idea behind this technique is that steep surges and drops respectively between a high and a low point are indicative of a crossroad. The moment you step onto a crossroad, the amount of heard access points increases drastically all of a sudden in the course of a few successive WiFi patterns. The same holds true for the moment you leave the crossroad, however with the measurement data developing in the opposite direction. Both the increase and the decrease happen in a fairly short span of time, which is characteristic for a typical crossroad. On the other hand, if you experience a slow increase or decrease you are likely just moving on to an area featuring a generally dense and light WiFi landscape respectively. The plot in Figure 11 presents a low-pass filtered measurement curve of the above-mentioned 35-minute-long trail, including detected high (small blue marks) and low points (small magenta marks). This time the ordinate indicates summed up signal strength values.

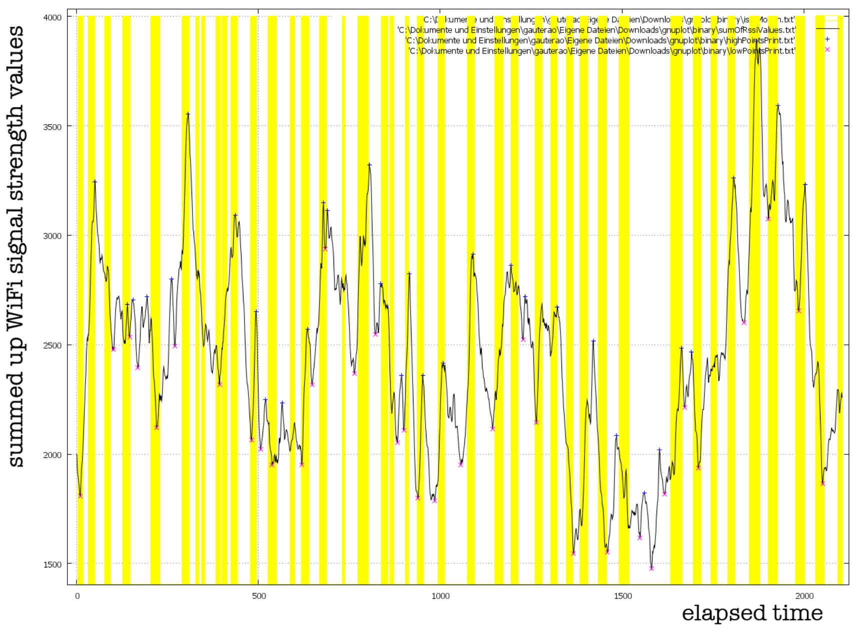


Figure 11: Crossroad detection on a 35-minute-long trail via high and low points

Finally, the aerial views in Figure 21 (refer to the appendix) give an overview of the pathway of the 35-minute-long trail, featuring markings for the results of the crossroad detection algorithm working in the operation mode described by the first approach. In the green segments of the path the algorithm could not find any crossroads. Red marks indicate that the algorithm is quite sure about the existence of a crossroad at this place while yellow spots mark positions where the algorithm is undecided and cannot deliver a definite statement. Usually this *maybe*-marking appears around small crossroads that do not feature pronounced WiFi patterns.

In fact, the results presented in this figure are commonly non-reproducible. The second time walking the exact same path about 24 hours later partially produced considerably different assumptions about the existence of crossroads (for a comparison see Figure 21 in the appendix). Places where the algorithm did not realize any crossroads on the first run, although existent in real-life, have been clearly detected on the second attempt. Unfortunately, the reverse case holds true as well, as previously detected crossroads were not recognized anymore. On average the recognition rate lies around 70%, while the proportion of false-positives is roughly 30%.

An interesting phenomenon, which has been observed during the recording of measurement data intended for investigations described later in Section 10.2, was the fact that the signal-to-noise ratio of available GPS satellites increased to a higher-than-average level for the duration of stay on a crossroad. This can be explained by the improved reception quality of GPS satellites due to a more open urban development situation on the middle of a crossroad. In this way the WiFi-only crossroad detection can be assisted by signal strength values retrieved from GPS satellites, exposing student hostels to not being a crossroad, since the reception quality of GPS satellites will normally not improve when walking alongside large buildings. Another application example of sensor fusion will be discussed in Section 10.2 where GPS is supported by GSM to improve the estimation of site density.

10.1.1 Distinguishing between traversing and turning off on a crossroad

Next up is the presentation of a small experiment undertaken out of curiosity, which has unfortunately never been finished for lack of time. However, as the underlying idea sounds promising in theory, the concept will be introduced here. The vision was to develop an algorithm that is able to differentiate if the agent or *Mister X* has walked over a crossroad straightforward or if he has turned off on this crossroad. The challenge was not to use the compass nowadays built into every smartphone but to base this distinction solely on WiFi signals. This might be useful within indoor environments, since the proper functioning of electric compasses is severely hampered inside buildings as thick concrete walls attenuate and large metal parts modify the terrestrial magnetic field, causing the compass to produce unreliable results.

The functional principle of the algorithm is described as follows: The collection of access points hearable ahead of the crossroad is compared to the collection of access points hearable behind the crossroad. If the agent had walked straight over the crossroad, the signal strength of beforehand heard access points received in retrospect should be higher than if he had turned off on the crossroad. This assumption is based on the tunneling effect of street canyons described in Section 10.1. If WiFi signals simply have to go straight ahead over the crossroad they experience only the attenuation of air. However, if the WiFi signals have to go around the corner, they will lose a great portion of their energy due to absorption the moment they are reflected by the exterior wall of a building (for an in-depth discussion on radio signal propagation see Section 4). Initial practical experiences with a prototype algorithm have resulted in an unsatisfying hit ratio of 50%. Presumably the hit ratio can be improved by further investigations incorporating more advanced analysis techniques.

10.2 Estimating the degree of development

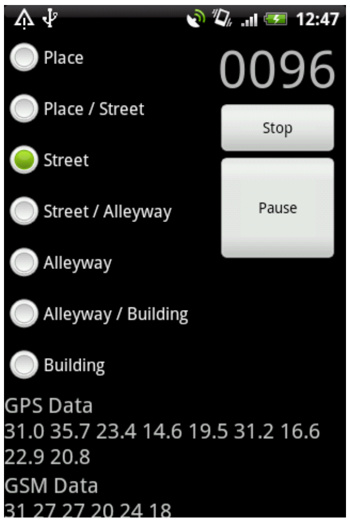
Usually pursuits pass through various cityscapes like spacious squares, broad streets, narrow alleyways and public buildings like shopping centers or the central station. For the agents it would be a tremendous assistance if they could keep track of the different cityscapes *Mister X* has walked through on his escape route. For example, if *Mister X* were walking along a wide shopping promenade and suddenly turns off into a small alleyway a hint like *At this point Mister X has entered an alleyway* would certainly be advantageous for the agents, as otherwise they would presumably overlook the alleyway and let it pass by. The obvious difference between a broad street and a narrow alleyway is the site density, as walls of facing houses are closer together.

This observation has led to the following intuitive assumption: The denser the urban development the lower the received signal strength of GPS satellites and GSM cell towers. The deeper the street canyon the more energy-reducing reflections of signal waves are involved before they reach the recording device, whereas on a large public square the view of the sky is unobstructed and wide open enabling strong signal reception of GPS satellites and GSM transmitting stations. The evaluation of WiFi signals does not seem to be appropriate in this application, as WiFi landscapes are not directly related to the density of development. An alleyway can feature higher-than-average quantities of access points if apartment windows face the alleyway or can feature no access points at all if it is uninhabited. GSM and especially GPS signals originate from a remote location, which makes them dependent on the density of development, which defines how well they reach receiving devices, whereas WiFi is a more local radio technology operating in closer proximity to the receiving device whereby it is not that much influenced by site density.

Consequently, the algorithm bases its assumptions on the degree of development on the signal-to-noise ratio of available GPS satellites in the first place and consults GSM signal strength in inconclusive situations as described later on. The categories used by the algorithm for classification of urban environment types are as follows sorted by descending expectable received signal strength:

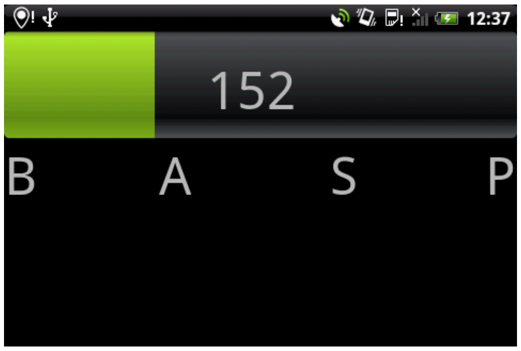
$$\text{Place} \Rightarrow \text{Street} \Rightarrow \text{Alleyway} \Rightarrow \text{Building (indoor)}$$

In fact, the algorithm works with additional inter-categories between these four main categories as an unambiguous classification is not always possible. For the purpose of this experiment a handy *Android* application has been developed as a convenient recording facility. This *Android* application features two distinct modes, one for recording measurement data and one to see the algorithm in action as illustrated by the screenshots in Figure 12.



GPS Data
31.0 35.7 23.4 14.6 19.5 31.2 16.6
22.9 20.8
GSM Data
31 27 27 20 24 18

(a) Recording mode



(b) Try-out mode

Figure 12: Screenshots of *Android* recording application for estimating the degree of development

The recording mode offers a radio button for each category mentioned previously. While walking through the various cityscapes, the user subjectively assesses the degree of development and selects the appropriate category in the list of radio buttons. In the meantime, the application continuously (see the seconds counter on the upper right) recordes GPS and GSM data displayed in the lower portion of the screenshot on the left. These signal-to-noise ratio values are added up separately and the two sums are finally saved in the category that was selected in the list of radio buttons at that moment. When the recording is finished, the value range of the sums collected within each category is determined. Eventually, these value ranges are characteristical for their corresponding category of site density.

The recording of GSM data has a mannerism that needs special treatment in order to retrieve workable measurement data. Firstly, the received signal strength of the serving cell, which is the cell the mobile phone is currently connected to, delivers unusable results, since this value hardly changes over time, at least within the scope of the official *Android* SDK. More promising results are produced by the neighboring cells which are cells that are indeed available, however the mobile phone is not connected to them, since they do not offer the same high reception quality as the serving cell. Fortunately, the neighboring cells deliver finely differentiated signal strength values. Furthermore, there is a problem with serving cell switches: The moment the mobile phone decides to change its serving cell in favor of another cell offering higher reception quality, all neighboring cells seem to be gone for a differing period of time. When this happens, which is about every few 100 meters outdoors and nearly every 50 meters inside buildings within city centers, the list of neighboring cells, provided by the *Android* SDK, is empty. The driver of the GSM chip discards everything it knows about the GSM landscape and begins to retrieve this information from the ground up every time the serving cell is switched. Step-by-step the list of neighboring cells fills up with cells that initially have the invalid signal strength value *99* before the real value is determined. Until the whole list is re-established normally 5 to 20 seconds pass by.

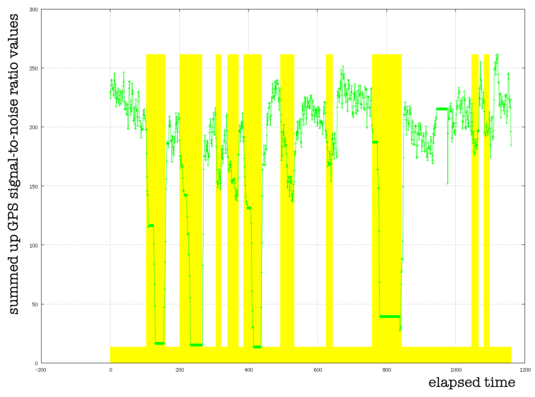
However, a lot can happen within this timeframe, as *Mister X* may have left a broad street, walked through a narrow alleyway and stepped on a spaciously square. During this timeframe an algorithm based exclusively on the evaluation of GSM data is rendered incapacitated, which calls the usefulness of GSM technology for the estimation of site density into question. Admittedly, this issue can be fixed superficially in post-production by assuming the list of neighboring cells from before the serving cell switch to still be valid as long as it takes to build up the new list from scratch. However, the problem persists that no statements about changes in the degree of development can be made. Due to this unreliability of GSM technology on the *Android* platform, it has been decided to make GPS the technology on which the algorithm first bases its assumptions, as GPS convinces with permanent availability. Nevertheless, this does not mean that GSM is completely useless for assessing the urban environment, as GSM technology has proven advantageous in inconclusive situations described later on. It should be pointed out once again that we indeed record GPS data, however do not extract geographical coordinates but only use the signal-to-noise ratio of available satellites.

The screenshot on the right side of Figure 12 shows the try-out mode in action, which in fact works opposite to the recording mode. Here all typical value ranges for the different categories of site density are already predefined and the current sum of added up signal-to-noise ratio values from GPS satellites (see the number in the middle of the bar) is displayed on a *B A S P* scale (Building Alleyway Street Place). Imagine the green bar filling up to the right as reception quality increases. This prototype try-out mode works on GPS data only and does not incorporate GSM data.

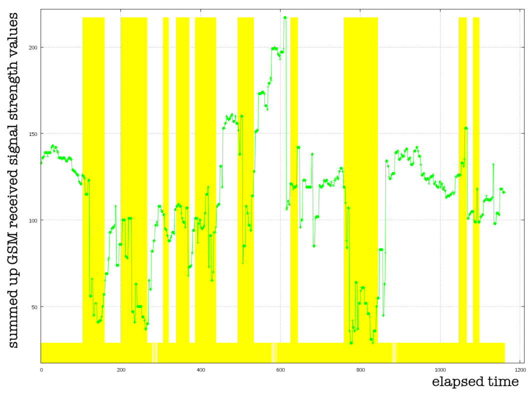
Typical value ranges for the different categories of site density using both GPS and GSM have been determined in major cities like Hamburg, Munich and Nuremberg. Surprisingly the value ranges from the three cities have been fairly consistent among themselves which is good news since it would be impractical if you had to determine value ranges for every city you would like to assess site density in. Obviously, the cityscape of these three major cities behaves equally in relation to received signal strength values of GPS and GSM. The following table reflects the measurement results of the three test runs:

Category	GPS	GSM
Place	> 210	> 130
Street	170 - 210	100 - 130
Alleyway	50 - 170	60 - 100
Building	< 50	< 60

In fact, the unit of these values is decibel milliwatt (dBm), however, as they are linearly summed up logarithmic values, it is safer to consider these values dimensionless. An interesting observation, which has been made in the historic downtown of Nuremberg at 1 a.m. in the night was the fact that the GPS values throughout the categories experienced a boost of about 20 dBm compared to previous recording sessions. Responsible for this phenomenon were the atmospheric conditions, as the sky was cloudless and starry that night, resulting in improved reception quality of GPS signals, whereas during the previous recording sessions the sky looked overcast and hazy. Possibly some kind of device calibration should be contemplated to counteract this issue. To get a feel for GPS and GSM signal strength values in various types of development, take a look at Figure 13, which illustrates the same 20-minute-long trail through the pedestrian areas of Nuremberg, once based on GPS and once based on GSM.



(a) GPS



(b) GSM

Figure 13: Signal strength values in various types of development. For an enlargement see Figure 22 in the appendix.

The yellow bars represent special occurrences of development types, as pointed out by the following list in chronological order: shopping center, shopping center, alleyway, alleyway, passage, alleyway, alleyway, shopping center, colonnade, colonnade. White areas signify normal shopping streets. As you can see, the GPS signal strength completely breaks down inside shopping centers as expected, whereas GSM signal strength indeed decreases significantly, however does not lean towards zero. Additionally, it can be observed how alleyways evoke reductions in received signal strength for both GPS and GSM, setting them clearly apart from ordinary streets. Colonnades are open on one side but roofed over making them behave like alleyways on average. An interesting observation can be made at passages that are transitions through buildings open on both ends. From the viewpoint of GPS these passages seem like buildings, however from the perspective of GSM they behave more like alleyways. Based on GPS data alone the algorithm would assume the whereabouts to be inside a building, which would not describe the situation adequately. Here the algorithm definitely benefits from the combination of measurement data provided by different sensors, called *sensor fusion*, enabling the algorithm to provide more sophisticated results. Nevertheless, the algorithm may still be tricked by several different constellations of development. For example, standing underneath a glass dome on the highest floor of a shopping center makes the algorithm think of an alleyway rather than a building, because glass lets much of the signal waves originating from GPS satellites and GSM cell towers pass through it, or at least it does not attenuate the signal energy as much as concrete walls do.

In fact, there is an approach to differentiate between indoor and outdoor environments based exclusively on WiFi signals. This approach compares the variance of received signal strength values recorded indoors on the one part and outdoors on the other part. Indoor environments typically feature many walls, corners, probably pillars, furnishings and other items made of different materials in a relatively small amount of space all causing the effects of signal propagation discussed in Section 4, like reflection, diffraction and scattering. All these phenomena contribute to the occurrence of interference, resulting in noise identifiable by the variance of received signal strength. In contrast to the indoor situation, the environment outdoors is spacious with much open space and objects set wide apart from each other. Here radio signal waves can propagate moderately without too much interference going on. Consequently, the variance of received signal strength from WiFi access points is considerably lower outdoors than it is within indoor environments.

11 Loop detection

When *Mister X* is on the run from the agents he will try to get away as far as he can. However, in the heat of the moment, he may sometimes accidentally return to a spot where he had already been before. In this case *Mister X*'s escape route contains a loop. If an agent is still situated ahead of this loop, he can save time and take a shortcut by heading for where the reference path has been recorded more recently. In this section an algorithm will be introduced that is able to detect different types of loops and is capable of handling complex combinations of loops. Additionally, a routing algorithm will be presented that can guide the agent out of a confusing network of multiple encapsulated loops. The operating principle of this two-step algorithm will be described with the aid of an example trail from real-life shown in Figure 14.

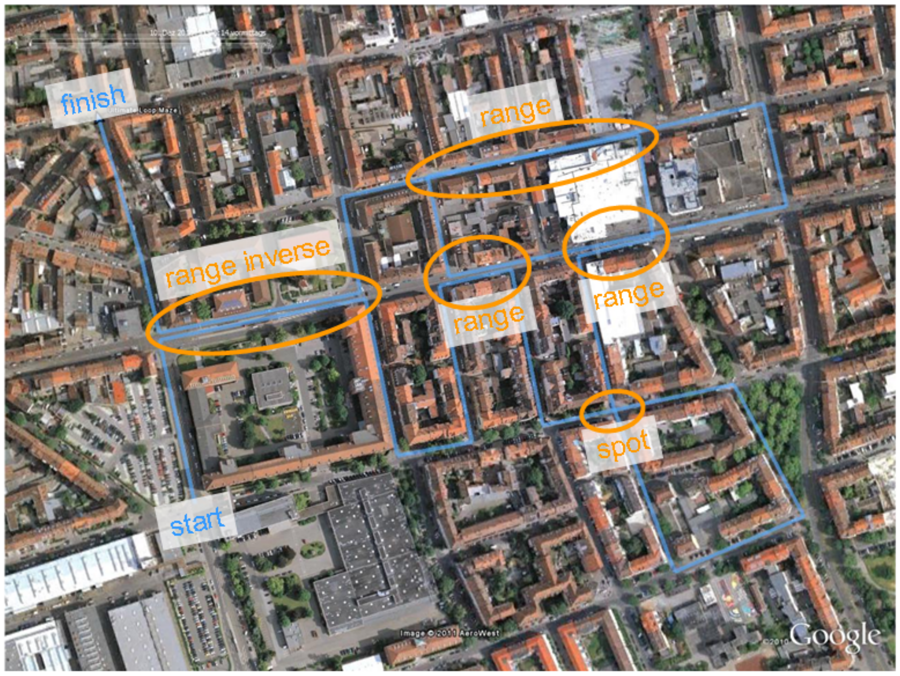


Figure 14: Example path in the southern part of Nuremberg containing different types of loops

11.1 Distinction between *spot loops* and *range loops*

As you can see in Figure 14 the algorithm needs to differentiate between *spot loops* and loops over a longer range. In the case of a *spot loop*, *Mister X* has crossed his own escape route at a single position forming the shape of a crossroad. By contrast, the *range loop* features a longer segment of the path that has been walked along twice. Now, how does the algorithm detect loops in general?

One after another, each WiFi pattern of the reference path is compared to all other WiFi patterns of the reference path. If later on in the reference path there is another sequence of WiFi patterns that is highly similar to the currently regarded WiFi pattern, this indicates a loop. So the phenomenon that the algorithm is looking for has the appearance illustrated in Figure 15. Here we can see the percentage of accordance (indicated on the ordinate) of all WiFi patterns of the reference path to the WiFi pattern recorded at second 120. As expected, the WiFi patterns in immediate proximity to the WiFi pattern observed at second 120 feature high conformity, since they have been recorded at the relatively same geographical position. As *Mister X* walked on, the distance to the position at second 120 became greater and therefore the degree of compliance gradually decreases towards zero percent as seen in Figure 15. However, 200 seconds later the percentage of accordance starts rising again as *Mister X* approaches the geographical location that he has been to at second 120. At second 360 *Mister X* visited the exact same position a second time as the degree of compliance reaches another maximum. As you can see, this high point does not reach the level of second 120 as a full 100% match is next to impossible in a real-life application due to permanent variance in the WiFi landscape. Finally, *Mister X* moves on and the accordance percentage starts falling again.

To sum up, the algorithm is looking for the existence of a second *hill* in the compliance curve of the WiFi pattern. Since the algorithm is doing this check on each and every WiFi pattern of the reference path, coherences between WiFi patterns become visible step-by-step. For example, it may become obvious that the WiFi patterns recorded from second 100 to 200 correspond to the WiFi patterns recorded from second 700 to 800. In this case we would have a *range loop*, since there are apparently multiple successive WiFi patterns featuring a high degree of compliance to another group of successive WiFi patterns. If for example, we would find that seconds 100 to 105 were in accord with seconds 700 to 805, we would have a *spot loop*, as the number of coherent WiFi patterns is relatively small. Admittedly, it is not always unambiguous to tell if a loop is a *spot loop* or a *range loop*, as you have to determine a threshold value for the number of coherent WiFi patterns.

Figure 14 features four *range loops* and one *spot loop*, whereby one of the *range loops* is an *inverse range loop*. This means that the beginning of the loop from the first visit (second 100) corresponds to the end of the loop from the second visit (second 800) and the end of the loop from the first visit (second 200) corresponds to the beginning of the loop from the second visit (second 700)[7]. The seconds in-between match accordingly inverse as well. If you plot the compliance results of all WiFi patterns into one diagram you get something like Figure 16. The abscissa indicates the index (second) of the regarded WiFi pattern with the index of the best matched WiFi pattern from the second *hill*, if existent, marked on the ordinate. Please notice that only WiFi patterns with a compliance level of over 70% to another WiFi pattern of the second *hill* make an appearance in this diagram.

[7]The words *first visit* and *second visit* may be swapped to illustrate the other variant of an *inverse range loop*.

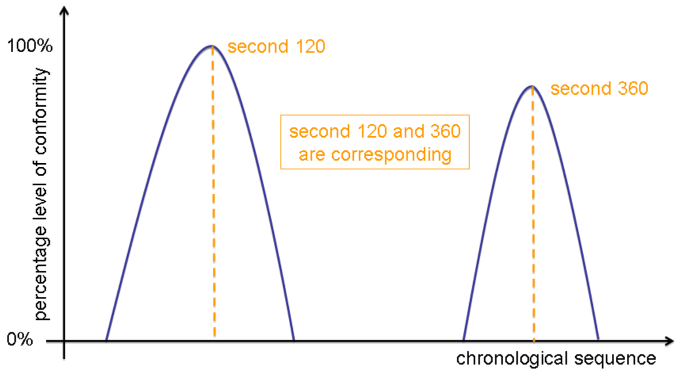


Figure 15: The WiFi pattern recorded at second 120 compared to every WiFi pattern in the trail. The *hill* on the right is the phenomenon the algorithm is looking for during loop detection.

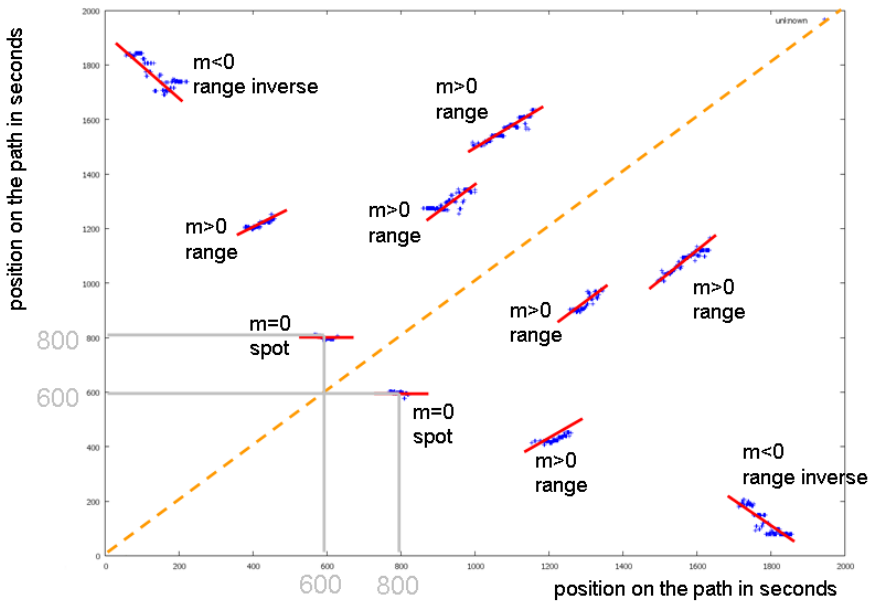


Figure 16: Compliance results of all WiFi patterns in the trail

The most conspicuous feature is that this diagram is axisymmetrical, indicated by the orange dashed diagonal line, because if second 600 corresponds to second 800, then as well will second 800 correspond to second 600. This example has been marked on the plot. As you can see, the point clouds are easily identifiable, as they are set widely apart from each other both vertically and horizontally. This correlates to the course of the path, how distant corresponding loops are in a temporal sense, and to the selected threshold value for the minimum compliance level (here set to 70%). A lower minimum compliance level will result in extended point clouds, since WiFi patterns featuring a lesser degree of accordance will likewise make an appearance in the diagram. If you choose a threshold value too low, the point clouds will start to overlap, if you choose it too high, less pronounced loops will begin to disappear from the plot.

Since Figure 16 is a representation of what the algorithm works with in a situation depicted in Figure 14, the same five loops show up again in the plot. Here you can see why the distinction between *spot loops* and *range loops* is not a simple decision since both loop types are represented by point clouds to a certain extent. Of course, *range loops* generally feature more widespread point clouds, however there is another criterion that facilitates the decision-making process, and that is the slope of the line of best fit through a point cloud (see the short red lines). If the slope m is positive the point cloud represents a normal *range loop* and if the slope is negative the point cloud represents an *inverse range loop*, while a slope of zero is an indication of a *spot loop*. The reason for this is that a *spot loop* has the shape of a crossroad. There is this one specific position (second) in the middle of the crossing from the first visit to this location that matches best with all the positions (seconds) shortly before and after the crossing from the second visit. The same holds true vice versa when you swap the words *first visit* and *second visit*. Time moves on on the abscissa of Figure 16, as *Mister X* crosses his own path, however as opposed to *range loops*, time on the ordinate stands still around this one best-matching index (second).

In the case of a *range loop* we can now read out the beginning and the end of this loop (point cloud) and its corresponding loop. As experience has shown, the length in seconds of two related *range loops* typically does not coincide perfectly. Normally, a *range loop* is either longer or shorter by several seconds than its matching partner. This happens when *Mister X* has taken a different amount of time to walk along the same path segment for the second time, as he has traveled at a presumably higher or lower speed.

11.2 Routing algorithm for complex loop constructions

At this stage the algorithm has overviews of which successive seconds correlate to the same geographical locations of other successive seconds. Based on this knowledge the algorithm has the ability to inform the agents if they are approaching a loop and if they have proceeded into the right direction, provided that the agents have chosen to take the shortcut. Make no mistake, the algorithm cannot tell the agents if they have to turn left or right or if they have to walk straight ahead when approaching a loop and they want to save some time. Again, the algorithm does not have any knowledge about geographical positions of loops, it only knows temporal coherences. Therefore the algorithm cannot give directions in a geographical sense of where to walk in order to reach a path segment that is situated in closer proximity to the finish of the path, which means it features higher index numbers (seconds), than the path segment the agents are currently situated on. In a real-life situation the procedure of a successful loop avoidance could look like this: An agent approaches a *spot loop* and is informed about this incident. The agent decides to turn right and is notified by the algorithm that he is about to walk the loop in reverse direction, which would mean a waste of time. Then the agent decides to walk straight ahead and is notified that this time he is about to walk the loop in forward direction, which again would be a waste of time. As only one option remains, the agent finally turns left while the algorithm confirms that he has just omitted x seconds of *Mister X*'s escape route. Before the algorithm has the ability to comment on the meaningfulness of a direction decision, the agent has to walk several steps into this direction, so that the algorithm can compare the freshly recorded WiFi patterns against its knowledge base about the coherencies of loops.

Let us suppose an agent is situated in the middle of a complex *maze* consisting of multiple encapsulated loops as already known from Figure 14. In this moment the gameplay grants the agent a one-time guidance out of this *loop maze* to the finish of the reference path. For this task to be accomplished the algorithm needs to be extended by some kind of routing facility. Figure 17 is a graphical representation of the knowledge base about the temporal coherencies of loops built from the conditions depicted in Figure 16. In fact, this graphical representation is an undirected weighted graph. All numbers in the boxes depict either a point of time in seconds or a period of time in seconds. The green box at the top showing second zero represents the beginning of *Mister X*'s escape route while the green box at the bottom represents the finish after 1963 seconds of walking. Boxes in turquoise color stand for loops; they are the nodes of the graph. *Range loops* have three numbers in their boxes, the beginning time, end time and duration, while *spot loops* have only one number in their boxes, which stands for the point in time of their occurrence. White boxes are the weights of the graph, representing the costs to move between the two adjacent nodes, meaning how long it took *Mister X* to walk from one loop to the next. Bold lines depict the course of the path while thin lines connect corresponding loop nodes, optionally with an *inverse* label where applicable.

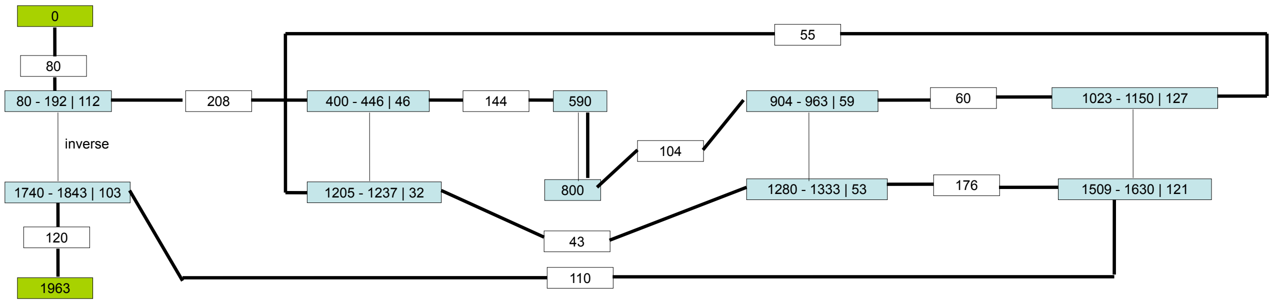


Figure 17: Graph structure used for routing purposes within the *loop maze* illustrated in Figure 14

11.2.1 Comparison to *Dijkstra*'s algorithm

Dijkstra's algorithm is the de-facto standard for determining shortest paths within graphs. Routing algorithms used in car navigation systems typically are advancements of *Dijkstra*'s algorithm. This algorithm determines in an iterative process for every node the fastest way to reach from this node all the other nodes of the graph. This way, car navigation systems know the shortest path to get from node *Nuremberg* to node *Koblenz*. In the example of the complex *loop maze* the algorithm has to find out the shortest path from the agent's current position, which in this case becomes a temporary node itself, to the green-boxed finish node. The most substantial difference between car navigation systems and our routing algorithm for *loop mazes* is that the former system bases its calculations on geographical distances while the latter bases on temporal distances.

Additionally, there is one feature that makes the development of a routing algorithm for *loop mazes* so special and that is not possible in car navigation systems: *teleportation*. As one would expect, this *teleportation* has its meaning in a temporal sense, not a geographical one. Take a look at Figure 17: For example, if an agent is situated at second 400 he is located at second 1205 at the same time since second 400 has been recorded at the same geographical location as second 1205. If an agent is situated at second 192 he is located at second 1740 as well due to the inverse relationship between these two *range loops*. Consequently, temporal transitions are possible between corresponding nodes, indicated by the thin lines, which of course do not carry white boxes since there are no costs for the algorithm to transition into a corresponding node.

Next up is an introduction to the operating principle of the routing algorithm inspired by *Dijkstra*. The procedure begins at the *loop node* that contains the second the agent is currently standing at or, if there is no loop node at this second, an additional node is inserted into the graph at the chronologically appropriate position. In the former case, if the agent is standing at a *spot loop*, or in the latter case, this node is assigned the distance *zero*. In the former case, if the agent is standing on a *range loop*, the distance to both the beginning and the end of this loop is calculated and saved into that node. In general, nodes representing *spot loops* only save one distance value while nodes standing for *range loops* need to store two distance values, one for the beginning and one for the end, as the beginning and the end of a *range loop* normally have distinct distances[8]. Now the distances to both the preceding and the subsequent nodes are calculated and saved into them as well as a reference to the node of origin. In addition to each distance value the node stores namely the list of nodes that consecutively lead to this node from the node the agent is currently standing at. If a corresponding loop node is existent its distance and reachability information is updated accordingly.

Following this, each node that has been processed in the previous step is recursively checking its two neighboring nodes, which are the corresponding node and the loop node that is not the node of origin. On each check the just calculated distance is compared to the distance value already stored in this node and replaces it if the new distance is smaller. In the case of a replacement, the path of nodes leading to this node is updated as well. If the newly calculated distance is not smaller than the current one this branch of recursion ends at this point, which ensures the existence of a termination condition. This way the level of recursion does not get infinitely deep. When the algorithm has finished every node has been visited and contains the information of the shortest distance and path of reachability from the node the agent is currently situated at. Consequently, the quickest way out of the *loop maze* to the green finish node from Figure 17 is finally known, which is what we were interested in in the first place.

[8]The following example clearly shows the importance to dissociate the beginning from the end of a *range loop* node in relation to the distance value and the reachability path. Let us assume that the agent is located at second 900. From there he can reach the beginning of *range loop* 1023-1150 within 123 seconds by moving on in chronological order. To reach the end of this *range loop* he could simply walk to its beginning and on until the end which would take him 250 seconds (123 seconds to the beginning plus the 127-second-length of this *range loop*). However, a much faster way would be to *teleport* from second 904 to 1280, move on to *range loop* 1205-1237 and finally reach the end of *range loop* 1023-1150 within 134 instead of 250 seconds.

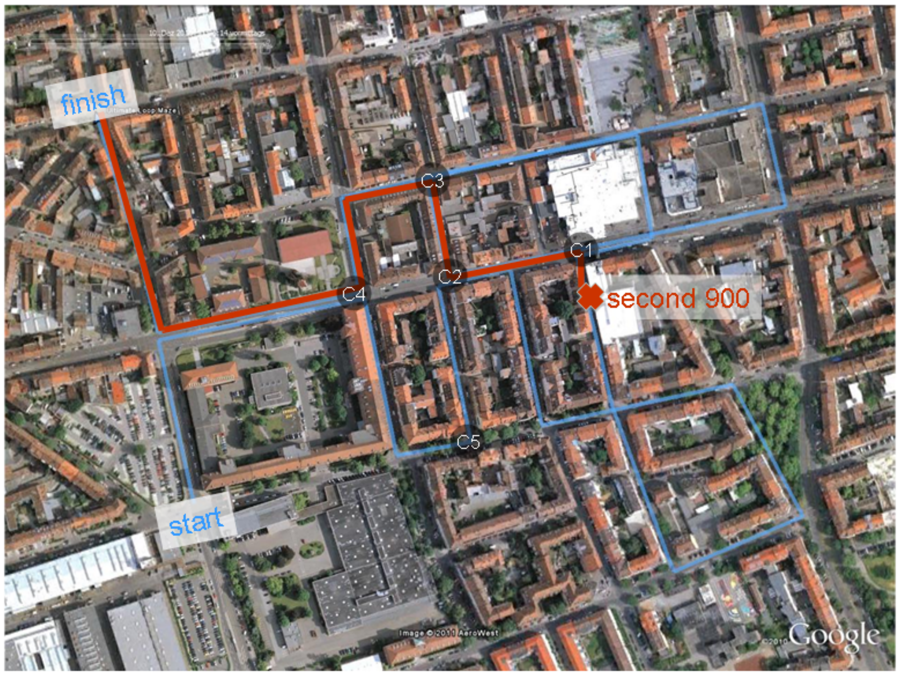


Figure 18: Escape route out of the *loop maze* proposed by the routing algorithm

Figure 18 illustrates the resulting escape route out of the *loop maze* (indicated by the red line) as proposed by the routing algorithm for the case that the agent is currently located at second 900. Please notice the proposed escape route from checkpoint C1 to C2 and from C2 to C3 running in the reverse direction to the original movement direction of *Mister X*. In order to reach checkpoint C4 from C2 the escape route makes a detour along C3 instead of proceeding straight ahead on a direct line from C2 to C4. The reason for this is that the algorithm is not aware of the existence of this connecting road, since *Mister X* did not walk along this street and therefore no WiFi patterns have been recorded there. So the algorithm has decided to let the escape route run over C3. Another possibility would have been to proceed over C5, but as you can see, the detour over C5 is longer in a metrical sense than the detour over C3. However, make no mistake, this is only the indirect reason why the algorithm has chosen to go for checkpoint C3. The direct reason is that it took *Mister X* less time to walk the path along C3 [9] than it took him to walk the path along C5. If for some reason it would have taken *Mister X* less time to move along

[9]In fact, *Mister X* had never walked the path from C2 via C3 to C4 as a whole. This part of the escape route consists of the part from C2 to C3, which *Mister X* had actually walked in the opposite direction and the part from C3 to C4, which *Mister X* had originally walked himself.

C5, despite the metrically longer path, as he may have been running instead of just walking, than it took him to move along C3, then the algorithm would assume the path via C5 to be shorter and therefore would have proposed an escape route running via C5 instead of C3.

Of course the *loop maze* illustrated in Figure 14 is not at the maximum in terms of complexity; you can make these *loop mazes* arbitrarily complicated as you choose. For example, the algorithm could be extended so that it can handle a *spot loop* running through a *range loop*. However, at this stage the abilities of the algorithm cover the most common cases of loop constellations.

12 Observing the agent's proximity to a certain spot on a large public square

To make the gameplay in the beginning more interesting, the exact starting point (or *spawn point* as it is called in computer games) of *Mister X* is obscure to the agents. The only information they have is that *Mister X*'s spawn point is somewhere on this specific public square that is known to the agents. Hence, a tool had to be developed that supports the agents in finding *Mister X*'s spawn point. Once again, the approach of choice for this purpose was WiFi pattern comparison. However, a simple MAC-address-only comparing like the one described in Section 8 will not help us in this situation, since basically every access point can be heard at every spot on a large public square. Here the signal waves of each access point around the square can propagate in an unimpeded way and reach every corner as long as the transmit power is high enough to travel these straight distances. However, the value that will change while wandering around the square is the received signal strength of each heard access point, since signal strength is directly correlated to distance. Therefore, the advanced technique, descriped in Section 8, incorporating signal strength values came into use here.

What the tool does in the end is observe the agent's proximity to this spot on the square where *Mister X* began recording his escape route. This proximity is determined by the evaluation of the compliance level between the agent's current WiFi pattern and *Mister X*'s first WiFi pattern. The compliance level should rise as the agent approaches the starting point and should fall again as his movement direction diverges from the starting point's location. This procedure reminds of the German childrens' game *Topfschlagen*, which literally translates to *beat the pot*[10]. The compliance level indicates how *hot* or *cold* the agent's proximity to *Mister X*'s spawn point is. Initial function tests of this tool on the central marketplace of Nuremberg (which has a size of about 90 times 90 meters) were disillusioning as the test subject stood directly on the starting point while the compliance level was still fairly low. As it turned out later, one important

[10]A blindfolded child taps on the ground with a wooden cooking spoon on the search for a metal cooking pot under which a bounty is typically hidden. As the blindfolded child is wandering about on the ground the other children provide guidance by shouting either the word *hot* or *cold* depending on the current proximity to the metal cooking pot.

phenomenon in radio signal propagation had not been taken into account and that was the absorption of WiFi signals by the human body.

12.1 Absorption of WiFi signals by the human body

Two-thirds of the human body is water. Since water absorbs the energy of radio signal waves the received signal strength of a WiFi access point is significantly higher in front of a human body than it is behind in the shadow of the body. Therefore, it makes a difference for the measurement of the received signal strength if the carrier of the recording device is facing or if he is averted from the access point [KaemarungsiKrishnamurthy04]. To substantiate this allegation, Figure 19 presents two diagrams depicting the received signal strength of two distinct access points measured from every angular direction. The test subject turned around his own axis slowly (one full rotation took about two to three minutes to complete) two times while holding the recording device (*HTC Hero* smartphone) tight in front of his body.

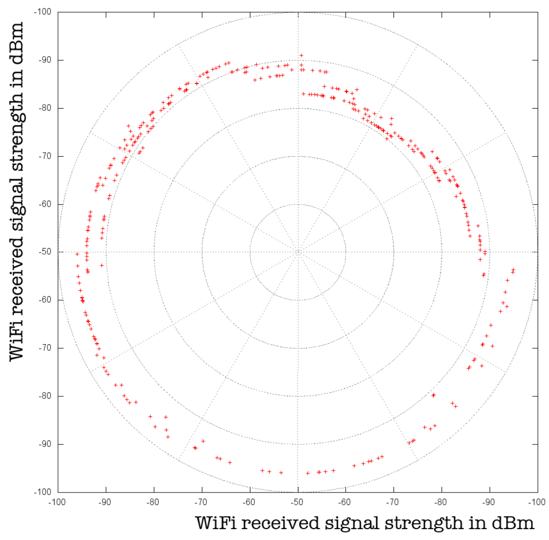
For presentation purposes the polar coordinate system has been chosen since it illustrates the angular property of this measurement appropriately. Since the polar coordinate system is oriented counterclockwise and begins on the right side of the x-axis north is on the right, east is at the top, south is on the left and west is at the bottom of this diagram. In order to match the measurements to the cardinal directions, the built-in compass of the recording device has been used. The center of the polar coordinate system in this example marks a received signal strength of -50 dBm while the outer edge marks -100 dBm. As a consequence, the more a measurement reading (indicated by the small red dots) is located towards the center the higher the received signal strength it represents. The major difference between the two diagrams in Figure 19 is that the second one presents the measurement data of an access point that has delivered overall higher signal strength values than the first access point did. If you take a closer look at the distribution of the measurement readings you will see that the received signal strength is indeed dependent on the orientation of the test subject, as the rings of measured data in both diagrams clearly show a pronounced indentation towards the center in one specific range of directions. The moment the test subject faced toward this range of directions, this specific access point could be heard louder than when facing toward any other direction.

By the way, from this observation we can estimate the position of this access point as we now know the direction from which the WiFi signals originate. In the case of the first diagram of Figure 19, this access point is located presumably north-east while the access point delivering the higher signal strength is probably situated south-west of the test subject's position. Whereas the location determination of access points was not the intended object of study[11] this experiment has made obvious the need to consider the absorbing effect of the human body on WiFi signals[12]. As a consequence, the direction from which

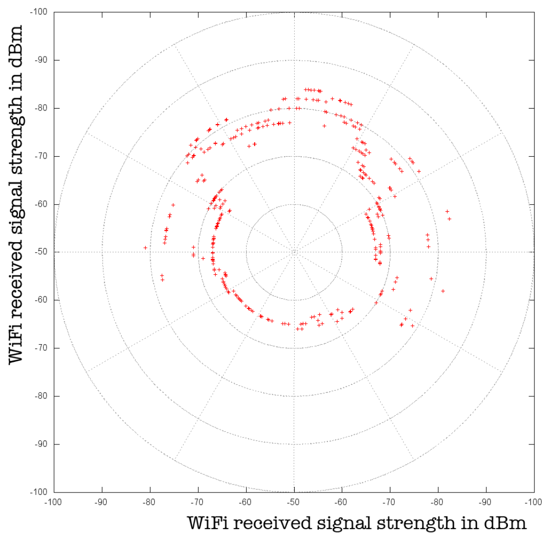
[11]At about the same time of evolution of this bachelor thesis another student at the *Fraunhofer IIS* has indeed investigated this research theme.

[12]The *Fraunhofer IIS* has investigated possibilities to take advantage of the absorbing effect

the agent approaches *Mister X*'s spawn point has to be taken into account (see [LiQuaderDempster08] for an inclusion of directional information into WiFi positioning).



(a) first observed access point



(b) second observed access point

Figure 19: Received signal strength values from two distinct access points recorded during two full rotations around the test subject's axis. For an enlargement see Figure 23 in the appendix.

12.2 Enhancing proximity sensing by incorporating compass heading

The most obvious sensor to use for direction determination is the compass, which is nowadays built into most modern smartphones. However, not only during locating *Mister X*'s spawn point can the compass be used, but also when *Mister X* is about to start recording his escape route he has to run through a compass-involving process necessary to gather information about this position. When *Mister X* has chosen his preferred spawn point on a large public square he is asked to slowly rotate around his own axis two times. For the duration of this process WiFi patterns including assigned direction information (degree values) are stored in a reference database for later access. When an agent tries to locate *Mister X*'s spawn point, the direction information of his latest

of the human body on WiFi signals. The idea was to estimate on a rough scale (in steps of 100) the quantity of people standing within a large crowd by analyzing signal strength values as more human bodies absorb more WiFi signals resulting in lower signal strength values. However, this experiment named *Body Mass Detection* ended up being less promising.

WiFi pattern is extracted and looked up in *Mister X*'s reference database. The WiFi pattern recorded by *Mister X* in the same direction as the agent's current viewing direction is finally compared to the agent's WiFi pattern. This approach enabled a much more workable tool for the locating of certain spots on a large public square. For an example where digital compasses enhance a WiFi-based indoor positioning system see [King06].

However, there was yet another problem with the compliance level when going across the reference point. The WiFi data provided by the *Android* SDK has a time lag of a few seconds. This fact exacerbates the assessment of the agent's proximity to the reference point in the case of fast movement. The following scenario illustrates this problem: An agent approaches the reference point while the conformity level gradually rises. When the agent hovers directly over the reference point the latest WiFi pattern, which is available at that moment, represents in fact the WiFi landscape of a few seconds before.

However, it is not only the time lag that causes difficulties, but also the movement of the agent plays a part in contributing to the problems. What we do here is compare WiFi patterns recorded during movement, meaning continuous changes in position, (in the case of the agent) to WiFi patterns which have been recorded at the exact same position (in the case of *Mister X*). We try to capture the WiFi landscape at a specific position when passing by it, with the quality requirement of a stationary recorded WiFi pattern. To get a solid comprehension of the full extent of the WiFi landscape at a particular location the recording device needs to be held in a steady position and orientation for at least several seconds. Subsequently, the measurement data should be averaged to filter out the noise originating from interference (see Section 4). Actually, this indeed happens in the prototype application during the recording of *Mister X*'s spawn point. Therefore, you cannot expect the proximity sensing to work very well when just passing by. However, neither can you expect from the agent to stand still for a short period of time every few meters to check for proximity, which would considerably constrain the fun in playing the game. Hence, the use of thresholds for the compliance level has been implemented.

Real-life experiments have shown that, when hovering over the reference point in constant motion, the compliance level typically rises to over 60% but stays beneath 70%, which makes 60% compliance a suitable threshold value for a position to potentially be *Mister X*'s spawn point. This means, if the agent's latest recorded WiFi pattern has a compliance level of over 60%, the agent is requested to stop walking. If now the compliance level increases further to over 80%, as now the recording device has the time to capture every detail of the WiFi landscape, this position indeed seems to be the searched-for starting point. If the compliance level does not rise significantly within a few seconds, the initial assumption turns out to be a false alarm and the agent needs to look further around.

13 Performance of the mobile platform vs. calculation on a remote server

Nowadays smartphones already offer an impressive level of computing power despite their small form factor. However, when it comes to heavy calculation tasks these handheld devices are reaching their limits rather quickly. Annotating algorithms like crossroad and loop detection are executed on a powerful desktop computer or server anyway. Nevertheless, there is one ongoing calculation taking place on each agent's device every second and that is the comparison of WiFi patterns for the determination of the own position (see Section 9) and for the assessment if one is still situated on the reference path (see Section 8). The complexity of one iteration of the comparison process depends on the length of the reference path on the one hand and on the amount of access points heard by the agent's device in this second on the other hand. Here we assume that the algorithm compares the agent's current WiFi pattern to each WiFi pattern of the whole reference path from start to finish every second. As a consequence, the longer the reference path the higher the amount of WiFi patterns to compare to. Assuming the dynamic gameplay (see Section 7), the complexity of the comparison process increases as the game continues. Furthermore, the more access points the agent's current WiFi pattern contains the longer the comparison with each WiFi pattern of the reference path takes to complete. Therefore the comparison takes longer if the agent is situated in the downtown area featuring 40 access points than it would take if he was standing on the outskirts featuring 10 access points.

The desired refresh rate for the position and probability information is once per second. However, if the comparison process takes longer than one second the agent cannot be provided with an update every second. In this case the local computing power of the smartphone is insufficient. Either we accept the slow refresh rate which affects the gaming pleasure negatively or we outsource the expensive comparison process *into the cloud* (see *cloud computing* in Section 14.1). However, the delegation of tasks *into the cloud* is not always profitable since the data transmission to and from the remote server takes a certain amount of time. If the computation expenditure is low a local calculation will be faster than a remote one. From which point on the utilization of the *cloud* is useful depends on the performance of the smartphone. A more modern device featuring a faster processor will reach this point at a later date. In order to experimentally determine this point a benchmark test on two different smartphones has been conducted. The first device was the *HTC Hero* featuring a 528 MHz processor while the second device was the *Google Nexus One* offering 1 GHz of computational power. For the remote calculation part of the test a standard desktop computer with an *Intel Core 2 Duo* processor running *Windows XP* has been set up as a server. The smartphones make requests to the server via a web service called *Restlet* which is a *RESTful* web framework for *Java*. For this purpose both mobile devices are capable of Internet connections via the HSPA standard. See the following table for results of the benchmark:

testing conditions path length / pattern size	HTC Hero	Google Nexus One
100 sec / 5 APs	47 / 2,870	7 / 887
100 sec / 10 APs	124 / 2,577	13 / 516
100 sec / 20 APs	170 / 2,978	23 / 510
100 sec / 40 APs	361 / 3,421	51 / 685
500 sec / 5 APs	272 / 2,474	40 / 517
500 sec / 10 APs	462 / 3,475	83 / 488
500 sec / 20 APs	928 / 3,102	121 / 661
500 sec / 40 APs	1,759 / 2,710	233 / 561
1,000 sec / 5 APs	492 / 3,536	79 / 499
1,000 sec / 10 APs	856 / 2,688	136 / 482
1,000 sec / 20 APs	1,637 / 2,576	234 / 662
1,000 sec / 40 APs	3,134 / 2,482	448 / 512
2,000 sec / 5 APs	961 / 2,640	151 / 584
2,000 sec / 10 APs	1,704 / 3,283	251 / 667
2,000 sec / 20 APs	3,228 / 2,550	473 / 477
2,000 sec / 40 APs	6,240 / 2,434	881 / 526
5,000 sec / 5 APs	2,975 / 2,618	365 / 804
5,000 sec / 10 APs	4,473 / 2,668	637 / 475
5,000 sec / 20 APs	8,521 / 2,775	1,242 / 570
5,000 sec / 40 APs	1,8763 / 2,463	2,225 / 561
10,000 sec / 5 APs	5,543 / 3,074	760 / 463
10,000 sec / 10 APs	9,145 / 2,548	1,282 / 484
10,000 sec / 20 APs	26,193 / 2,842	2,339 / 497
10,000 sec / 40 APs	34,866 / 2,665	4,550 / 634

If you take a look at the column named *testing conditions* you will see that the benchmark runs through different combinations of reference path length and WiFi pattern size which is equal to the number of access points (APs). In the course of one iteration both position and probability are calculated with the aid of the simple MAC-address-only algorithm introduced in Section 8. For remote calculation the smartphone uploads its current WiFi pattern to the server whose response consists of two integers, the calculated position and probability. The numbers in the two device columns indicate the duration in milliseconds it took until the result was available. The first number before the slash specifies local calculation while the second number after the slash stands for remote calculation.

As you can see the duration for all remote calculations remains constant on average for each device independent of the testing condition. The reason for this is that even the last testing condition in the table makes for a ridiculously low computation expenditure from the point of view of a modern desktop computer. Eventually, it makes no difference to a server whether it has to compare a WiFi pattern consisting of 5 MAC addresses with 100 other WiFi patterns or whether it has to compare a WiFi pattern consisting of 40 MAC addresses with 10,000 other WiFi patterns. In both cases the calculation result is available in under 100 milliseconds. What makes up the greatest share of total duration during remote calculation is the data transmission via cellular network. For some reason remote calculation on the *Nexus One* is consistently much faster than it is on the *Hero*. The *Nexus One* seems to keep the connection to the server alive between requests while the *Hero* appears to re-establish the connection on each request. The more intelligent connection management on the *Nexus One* may be related to the newer version of the *Android* operating system, 2.2 vs. 2.1 in this case.

The situation is different with local calculation where the testing condition has a great impact on calculation time which varies between 50 milliseconds and 35 seconds on the *Hero* in the course of the benchmark. With the aid of these measurement results we can estimate the point from which on remote calculation *in the cloud* makes more sense than local calculation. If you take a look at the table and do a little linear interpolation - which is valid since the complexity scales in a linear way - you will come to the conclusion that the searched-for point is reached after roughly 25 minutes of playing time assuming 30 access points on average which is characteristic of the downtown area where the pursuit will mainly take place. In the case of the *Nexus One* this point is reached after about 35 minutes. This value seems to be quite low which is down to the fact that remote calculation on the *Nexus One* takes considerably less time than it does on the *Hero*. If we would assume a duration of 2.5 seconds for a remote calculation on the *Nexus One* like it is the case on the *Hero* the searched-for point will be reached as late as after two hours of playing time. This result illustrates the difference in computational power between these two devices. You will also notice this when you compare the duration of local calculation with the testing condition depicted in the last iteration of the benchmark. This calculation lasted 35 seconds on the *Hero* while less than 5 seconds on the *Nexus One* which results in a factor of seven relating to performance.

Admittedly, the operating principle of the algorithm to determine position and probability is fairly inefficient. Of course, it is not necessary to compare the agent's current WiFi pattern to each and every WiFi pattern of the whole reference path from start to finish every second. Once the position of the agent on the reference path is known we also know his approximate position in the following second since he cannot go far within one second. Therefore, the utilization of a *window* makes sense which is positioned centrally in relation to the most recently determined position. A window size of 100 seconds should be sufficient even for fast moving agents. This way the algorithm analyzes 50 seconds of WiFi patterns in both directions which takes into consideration that the agent might turn around and walk in the opposite direction. If you take a look at the benchmark results once again you will see that even the *HTC Hero* is capable of finishing the comparison to 100 WiFi patterns within a third of a second, even with 40 access points in the agent's WiFi pattern. As a consequence, the computation expenditure remains independent of the length of the reference path.

In the event of the agent reaching a loop (see Section 11) the algorithm needs to apply two distinct windows since the agent has two possibilities to continue, either he follows along the original progression of the reference path like *Mister X* created it or he takes the leap in time and continues the path tracing at an earlier or later point in the reference path respectively. After the agent has walked away from the junction the algorithm knows upon which continuation of the path the agent has decided and can stop the tracking of the irrelevant window. Fortunately, the *Hero*'s computing power offers enough reserves to allow for the sequential processing of two windows resulting in a temporary total of 200 comparisons of WiFi patterns per second.

However, there is a situation which breaks the applicability of the window technique and this occurs the moment the agent leaves the reference path completely. If the algorithm is not able to determine the agent's position since none of his current WiFi pattern's MAC addresses appear in any of the reference path's WiFi patterns then the algorithm has no clue of the agent's whereabouts. Due to this fact, the agent could make his reappearance anywhere on the reference path which is why the algorithm is required to take the whole reference path from start to finish into consideration. The moment the agent steps somewhere on the reference path his location can be determined again and the window technique can finally be reapplied. From this point on the smartphone can switch back to local calculation. However, at times of uncertainty about the agent's whereabouts the *cloud* has to take over the computation task. Consequently, a dynamic alternating operation between local and remote calculation is imaginable as the circumstances require it.

Finally, one last remark on remote calculation should be mentioned. Although response from the server is typically available after about three seconds on the *HTC Hero* this does not mean that a refresh on position and probability information occurs only every three seconds. This information is still updated every second, however it describes the situation from three seconds ago. Therefore, the refresh rate remains constantly high as desired but the content presented lags behind by a short amount of time. This is achieved via asynchronous requests to the server. In the beginning the smartphone sends its first three requests without waiting for any reply from the server until it receives the response to the first request in the fourth second. In the fifth second the smartphone sends its fifth request while receiving the response to its second request and so on. As a matter of fact, a delay of several seconds for the information refresh hampers the rapid course of the game significantly which is why utilization of the window technique makes sense in oder to avoid remote calculation whenever possible. Nevertheless, as we have seen in the benchmark the *Nexus One* has demonstrated that remote calculation can indeed be real-time capable.

14 Review and outlook

Just five years ago, the realization of this game idea would not have been possible with that ease, as we experienced it during our work on this collaborative project. Relating to hardware, today's smartphones offer a wide array of sensors, each one of them allowing for the acquisition of the user's behavior or surroundings. On the software side the development of mobile applications has been greatly simplified with the introduction of the *Android* and *iPhone* platform around 2007. Getting started with mobile software development has never been this easy thanks to free of charge and well-documented SDKs. Countless Internet forums offering assistance for developers by developers have formed large communities around these popular platforms. Distribution facilities like the *Android Marketplace* and the *Apple App Store* have enormously simplified the global distribution of self-developed applications. Now even hobby programers can develop so-called *apps* in their spare time and sell them on a worldwide market without leaving their home. This has led to a flood of low-quality apps, which made it harder for established software companies to stand out from the mass to promote their professionally developed products.

The situation is different on the *Sony PlayStation 3* platform, which is much more challenging to program for. According to an interview given by Kazuo Hirai, the CEO of *Sony Computer Entertainment, Inc.*, to the *Official PlayStation Magazine* in February 2009, the *PlayStation 3* platform has been designed to be intentionally demanding for developers, so that game studios cannot exhaust the full power of the hardware all at once but rather need several years in order to achieve that. This way *Sony* wants to extend the life cycle of one *PlayStation* device generation. What is more, *PlayStation* is normally not available to private persons as a development platform, since you need an expensive developer license from *Sony* to write your own games for this platform.

The high popularity of smartphones has not only enabled a pleasant development process for us, but also allows for a widespread deployment of our game among potential players. According to a report by *Nielsen Company*[13] in March 2010, smartphones will overtake feature phones[14] in terms of market share in the U.S. around the end of 2011. At the end of 2009 feature phones still had a share of 83% in the U.S., according to American Internet marketing research company *comScore*. The triumph of the smartphone proceeds in favor of the *Android* platform, which has taken the lead in smartphone market share by the 4th quarter of 2010, according to analytics firm *Canalys*. This fact accommodates the marketing opportunities of our game, since the hardware facilities are already available to many potential players. Interested parties just need to install the application via the *Android Marketplace*, which is an uncomplicated process, and can start playing right away in whatever city they live. This instant-game-start feature is made possible by the fact that our game is not de-

[13]The *Nielsen Company* is a global marketing and advertising research company headquartered in New York.

[14]*Feature phone* is a term used to describe a low-end mobile phone that offers less computing power and features than a smartphone.

pendent on any preparatory operations like the fingerprinting process necessary for the functionality of WiFi positioning systems (WPS). The only requirement is the presence of a potent WiFi landscape, which is why our game has to be played downtown.

Actually, GPS and WiFi positioning behave opposite to each other in terms of accuracy. While GPS delivers its highest precision in the countryside with low density of development, WiFi positioning would presumably fail completely, due to the absence of any access points. However, the deeper you move into the city, which accommodates an enormous amount of access points, the higher the accuracy of WiFi positioning, while GPS falls far behind relating to precision as site density increases. Therefore, WiFi positioning demonstrates its strengths primarily in downtown and indoor locations, which made it the first choice as the positioning technology of our game.

The excitement of playing our game lies in the impreciseness of the hints the algorithms calculate and pass through to the agents. The players need to be aware that the provided information is only a computer-produced hint, which might be highly inaccurate and should not be relied on at all times. Agents should always call the generated hints into question and make their own logical reflections about how *Mister X* might have behaved in a certain location. In the end it is quite astonishing how much you can get out of WiFi technology, although its intended purpose is actually just wireless data transmission. As it turns out, projects that try to use common technology for unconventional purposes end up being the most interesting ones.

Several sub-projects, which emerged from work on this bachelor thesis have the potential to be deployed in other products. For example, the algorithm capable of detecting crossroads and the one assessing the density of development around the user could be interesting for autonomous robots doing simultaneous localization and mapping (SLAM). By the incorporation of these algorithms, the environment exploration, heretofore based on image processing and laser scanning, could be further improved to capture advanced details about the vehicle's surroundings.

14.1 The future of smartphones

Over the course of the next ten years smartphones are expected to gain so much computing power that they will reach the performance level of notebook computers. They will probably even become a reasonable replacement for desktop computers. Imagine you have a docking station connected to a mouse, a full-size keyboard and a large monitor on your desk, both at home and in the office. The actual computer is the smartphone you carry around in your pocket and which you plug into the docking station if you need a full PC-like experience. Since a smartphone will still have to meet the requirements concerning portability its dimensions will stay the same. Therefore, the built-in display cannot exceed a diagonal screen size of around five inches, which makes the use of an external monitor necessary for serious work. Admittedly, smartphones will probably never come into question as a desktop replacement for power users who depend on high-end workstation-class performance. However, the majority of people using a computer in everyday life is doing computationally less demanding tasks, such as surfing the web, writing emails, browsing photo collections and streaming online video content. In the end, most people do not need that much computing power anyway.

So long as smartphones are still in the process of evolution, aspiring to reach desktop computing performance, the IT industry needs to rely on another trend emerging these days. *Cloud computing* means to smartphones that they can offload computationally intense tasks onto the *cloud*, which is basically a cluster of powerful servers on the Internet. The challenge is for the smartphone to decide when a remote calculation will be profitable which means faster availability of the result on the one hand and less energy consumption on the other hand. Of course, the actual calculation on a remote server takes a negligible amount of time however, the transfer of request data to the *cloud* and the transmission of result data back to the smartphone are typically responsible for the largest portion in total duration. Therefore the smartphone needs to forecast the amount of time the data transmission to and from the remote location will take based on the current quality of its mobile Internet connection. If the connection quality is fairly low at the moment a local calculation will presumably be completed in less time making it the more sensible choice. Likewise conceivable would be to conduct both a local and a remote calculation and take whichever result is available earlier. However, we also need to take care of energy consumption since users appreciate smartphones featuring long-lasting battery life. On the one hand local calculation implies heavy load on the CPU which means high energy consumption while remote calculation on the other hand entails wireless data transmission which means high energy consumption as well. Doing both calculation methods would be an inefficient and irresponsible handling of system resources. Therefore the efficiency of *cloud computing* lies in the intelligent decision whether to use it or not.

Another idea of *cloud computing* is that all your multimedia data is not stored locally on the smartphone but remote *in the cloud*. Your music library is streamed wirelessly to your mobile device as well as your family photos and home movies whenever you like to enjoy them. However, for this to work reasonably a broadband Internet connection is required which is not always that highly available as it needs to be in the mobile environment. Additionally, it is not only your multimedia which is expected to move into the *cloud* but also your personal data like your whereabouts, the places you are visiting in everyday life. This enables for example advertising companies to deliver location-aware offerings to you and allows your friends and family to know where you are which could be relevant to social networking communities. The fact that smartphones know a lot about their users' behavior is appropriately pointed out by John Brandon, a contributor to the high-tech lifestyle website *Digital Trends*: *The phone - and the cloud-based server side intelligence behind it - will know you, your location, your social networks, and your preferences in food, media, and communication. It will predict your next moves.* The intention behind this artificial intelligence is to require less interaction by the user and to let the smartphone evaluate the user's behavior on its own. For example the event that you have entered a certain coffee shop to stay there for a while can be detected by looking up heard MAC addresses in an database annotated with public places. If the access point which is part of the caf's furnishings can be clearly heard the user seems to be situated within this caf which can then be published automatically on social networking sites such as *Facebook Places*. As a matter of fact this kind of artificial intelligence would be technically not difficult to realize however, companies offering location-based services are still struggling with privacy to keep private data secure (see the chapter on privacy protection in [Kuepper05]). In addition to that, the question is if people will want this permanent surveillance or if they are intimidated at the thought of an impersonal commercial company keeping track of every step they take making them to become transparent men. See [RekimotoMiyakiIshizawa07] for thoughts on continuous location logging of a whole human lifetime.

14.2 The future of WiFi

According to Sanjit Biswas, CEO and Co-Founder of American wireless networking company *Meraki, Inc.*, in general everyday data traffic on WiFi networks has doubled from 2010 to 2011 and is expected to continue doubling each year. On the one hand this is caused by an increasing amount of WiFi-enabled devices that we use in everyday life. Initially only notebooks were using WiFi connections to transfer data, nowadays it is also smartphones, tablet PCs, gaming consoles, digital cameras and others that communicate via WiFi networks. In the future, it wouldnt be inconceivable to imagine electrical household appliances like microwaves communing with a building automation system over a wireless connection. On the other hand, also responsible for the rise in wireless data traffic is the ever-growing size of typical multimedia items especially video streaming in high-definition resolution. Since people do not want to relinquish watching video streams while travelling mobile service providers are expanding their network of public WiFi access points in order to take the load off their cellular network. So as to sustain a high level of operability while the penetration of wireless networks increases the perfomance of WiFi connections needs to be adapted in order to be able to keep up with tomorrow's requirements. New WiFi standards need to improve upon range, robustness and throughput. Eventually, WiFi will reach the level of robustness and speed needed to make it a reasonable replacement for wired Ethernet connections.

A new field of application in which WiFi technology is about to emerge is the automotive sector. Vehicles are equipped with WiFi in order to exchange information between each other which is referred to as *Car2Car communication*. This way oncoming vehicles can warn the own car about for example forthcoming traffic congestions and construction sites to improve safety in road transport. *Car2Car communication* is intented to support multi-hop data transmission so that information can be forwarded from one car to the next achieving greater range. Here the challenge is to ensure a fast connection establishment between two cars moving in opposite directions. As these two cars may be fast moving on a highway there is not much time for the information exchange before the distance becomes too large.

Another interesting new extension to the WiFi standard is called *Wi-Fi Direct* announced by the *Wi-Fi Alliance* in October 2010 and promises easier manageable direct ad-hoc connections between WiFi-enabled devices. This feature should be useful for our game to detect the event when one of the agents successfully catches *Mister X*. Up until now we did not have a workable solution for the detection of this event. If all agents and *Mister X* would be constantly uploading their geographical coordinates which would then be compared server-sided relating to distances this approach would be too slow in terms of response time - especially as the pursuit becomes hectic near the end of the game - as well as too imprecise since the accuracy of GPS is not sufficient within dense urban environments.

Another possibility would be to detect the proximity to a certain device by catching the probe requests the other device broadcasts while scanning for WiFi networks in order to analyze the received signal strength of these probe requests from which conclusions can be drawn relating to the distance between the sender and the receiver[15]. This technique is known as *sniffing* and is rather a *hack* than a serious solution. However, *Wi-Fi Direct* seems promising to facilitate the detection of an encounter between an agent and *Mister X* since this extension enables direct device-to-device links without the detour via an access point. For one thing the fact that an ad-hoc connection can be established between the two devices means that *Mister X* cannot be far away anymore, he must be within sight distance or one block of houses away at the most, and for another thing the agent's device can still analyze the received signal strength while communicating with *Mister X*'s smartphone. Unfortunately, the official *Android* SDK does not provide access to ad-hoc links or *Wi-Fi Direct* via one of its APIs at the time of writing although *Android* smartphones are supporting this feature on the hardware side (firmware) since its announcement in 2010.

14.3 Updating the WiFi measurements of the reference path by subsequent followers

In Section 3.1 a technique called *Automated Self-Healing Network* has been mentioned which the company *Skyhook* uses to keep the database on which its WiFi positioning system (WPS) is based up to date via user contribution. This concept might be applicable to the path tracing process of our game as well. As it has been demonstrated in Section 9.2 we have a decline in localization accuracy the farther *Mister X*'s recording of his escape route lies back in time. The idea is to counteract this problem by updating the WiFi measurements of the reference path by subsequent followers.

Imagine a street which is part of *Mister X*'s escape route. When *Mister X* was here his recording device has captured an access point with a very high sig-

[15]In order to determine the identity of the sender the MAC address contained in the probe requests needs to be extracted. Therefore *Mister X*'s MAC address must be known a priori to all the agents.

nal strength. A few weeks later a group of agents walks along the same street[16]. Although their recorded WiFi patterns feature a high level of compliance with those recorded by *Mister X* none of the agents' patterns contains the MAC address of the access point which *Mister X*'s device was able to recognize so distinctly. If the devices of several groups of agents are experiencing this same phenomenon the system can assume that this access point is not existent any more. This way the subsequent followers' devices give feedback about the current condition of *Mister X*'s escape route relating to the WiFi landscape. The challenge for the system is now to intelligently evaluate the feedback in order to make reasonable updates to the WiFi patterns constituting *Mister X*'s reference path.

However, this technique is dangerous since its operating principle relies on the assumption that subsequent followers walk the reference path along the exact same geographical line as *Mister X* did which is typically not the case. Walking along the same street but on the other side already produces considerably different WiFi patterns. What we try to do here is to improve the accuracy of a localization system which contains out-dated data after having determined our position with the aid of this already imprecise system. Consequently, our suggestion for improvement of WiFi patterns is based on an already inaccurate location determination. What we need is an independent positioning system whereby we could tune the WiFi positioning system. If we had a system that would give us absolute coordinates we could tell if the agent making a suggestion for improvement of WiFi patterns was indeed standing at the same geographical position as *Mister X* used to. Unfortunately, the accuracy of GPS is doubtful within dense urban areas. The fact that the contribution back to the system by one group of agents is based on the system already updated by the feedback of the previous group of agents means that erroneous updates are adding up from game to game. The moment a group of agents contributes an update which is actually a great improvement for the worse the following update will build upon this faulty update and chances are that it will make the operating precision of the system even worse. The system will converge to a very low level of performance finally rendering the game unplayable.

[16]We are assuming the static gameplay here, see Section 7.

Glossary

A-GPS Assisted GPS. Technique to support GPS via other technologies, such as WiFi and GSM.

AP (Wireless) Access Point. Device that allows wireless devices to connect to a wired network.

API Application Programming Interface. Collection of commands for use in self-written software via which functions of the underlying system can be called.

CCTV Closed-circuit television. The use of surveillance cameras.

CEO Chief Executive Officer. The head of a company.

CPU Central Processing Unit. The portion of a computer system that executes the instructions of a computer program.

GPS Global Positioning System. Satellite-based system for worldwide location determination.

GSM Global System for Mobile Communications. Second generation of the global standard in digital mobile communications.

HSPA High Speed Packet Access. Enhancement for UMTS allowing for faster data transmission.

IIS (Fraunhofer) Institute for Integrated Circuits. One of 60 institutes of the German research organization Fraunhofer Society.

IT Information Technology. Generic term for electronic information and data processing.

LED Light-Emitting Diode. Modern light source featuring low energy consumption, long lifetime, high robustness, small size, fast switching, and great durability and reliability.

LOS Line Of Sight. Unobstructed direct connection between two points.

MAC Media Access Control address. Unique identification number for network devices.

NDK Native Development Kit. More advanced development tools enabling access to low-level system functions.

PC Personal Computer. General-purpose computer for use by private individuals.

PDA Personal Digital Assistant. Mobile handset device offering advanced office and multimedia functionalities.

RSSI Received Signal Strength Indication. Power of a radio signal reaching a receiving device.

SDK Software Development Kit. Collection of development tools for the creation of software programs.

SLAM Simultaneous Localization And Mapping. Technique used by robots and autonomous vehicles to build up a map within an unknown environment while at the same time keeping track of their current location.

SSID Service Set Identifier. Name of a wireless network.

TTFF Time To First Fix. Required time to determine the current location via GPS.

UMTS Universal Mobile Telecommunications System. Third generation of the global standard in digital mobile communications.

WEP Wired Equivalent Privacy. Ciphering method for wireless networks, insecure from a present-day perspective.

WiFi Wireless Fidelity or Wireless Local Area Network (WLAN). Technology for wireless data transmission between computers in the broader sense.

WLAN Wireless Local Area Network. See WiFi.

WPA Wi-Fi Protected Access. Ciphering method for wireless networks, more secure than WEP.

WPA2 Wi-Fi Protected Access II. Ciphering method for wireless networks, even more secure than WPA.

WPS Wi-Fi Positioning System. System enabling location determination based on WiFi signal patterns.

References

[Hyndman08] Rob J. Hyndman, *Forecasting with exponential smoothing: the state space approach.* Springer Berlin Heidelberg, 2008.

[Dean09] Tamara Dean, *Network+ Guide to Networks.* Cengage Learning, 2009.

[ChandraLide06] Praphul Chandra and David R. Lide, *Wi-Fi telephony: challenges and solutions for voice over WLANs.* Newnes, 2006.

[WeynSchrooyen08] Maarten Weyn and Frederik Schrooyen, *A WiFi Assisted GPS Positioning Concept.* University College of Antwerp, Department of Applied Engineering, 2008.

[Cheng05] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca and John Krumm, *Accuracy Characterization for Metropolitan-scale Wi-Fi Localization.* Intel Research Seattle, 2005.

[LiQuaderDempster08] Binghao Li, Ishrat J. Quader and Andrew G. Dempster, *On outdoor positioning with Wi-Fi.* Journal of Global Positioning Systems (2008), Vol. 7, No. 1 : 18-26.

[QuaderLiPengDempster07] Ishrat J. Quader, Binghao Li, Wendi (Patrick) Peng and Andrew G. Dempster, *Use of Fingerprinting in Wi-Fi Based Outdoor Positioning.* International Global Navigation Satellite Systems Society, 2007.

[Papamanthou08] Charalampos Papamanthou, Franco P. Preparata and Roberto Tamassia, *Algorithms for Location Estimation Based on RSSI Sampling.* Brown University, Department of Computer Science and Center for Geometric Computing, 2008.

[Laitinen01] Heikki Laitinen et al., *Cellular Location Technology.* CELLO Consortium, 2001.

[BillCapKofahlMundt04] Bill, Cap, Kofahl and Mundt, *Indoor and Outdoor Positioning in Mobile Environments - A Review and some Investigations on WLAN-Positioning.* Geographic Information Sciences, Vol. 10, No. 2, December 2004.

[Kawaguchi09] Nobuo Kawaguchi, *WiFi Location Information System for Both Indoors and Outdoors.* Nagoya University, Japan, 2009.

[RekimotoMiyakiIshizawa07] Jun Rekimoto, Takashi Miyaki and Takaaki Ishizawa, *LifeTag: WiFi-Based Continuous Location Logging for Life Pattern Analysis.* University of Tokyo and Sony Computer Science Laboratories, 2007.

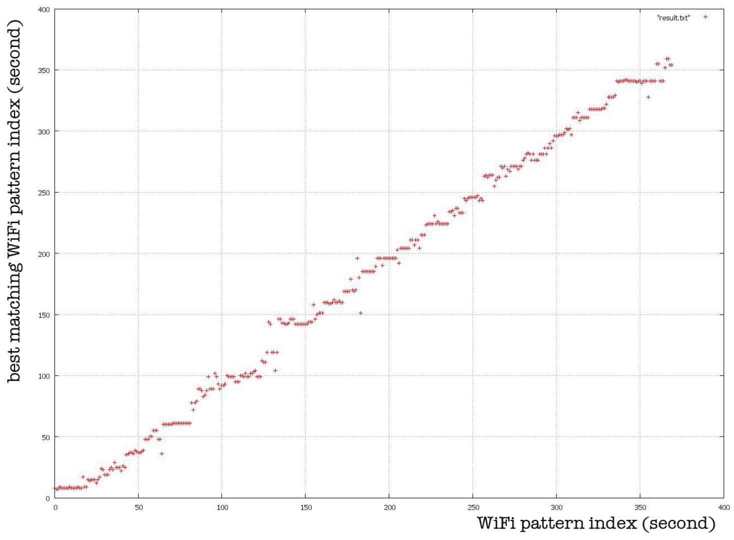
[King06] Thomas King et al., *COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses.* Department of Computer Science, University of Mannheim, 2006.

[Xiang04] Xiang, Song, Chen, Wang, Huang and Gao, *A wireless LAN-based indoor positioning technology.* IBM China Research Laboratory, Vol. 48, No. 5/6, September/November 2004.

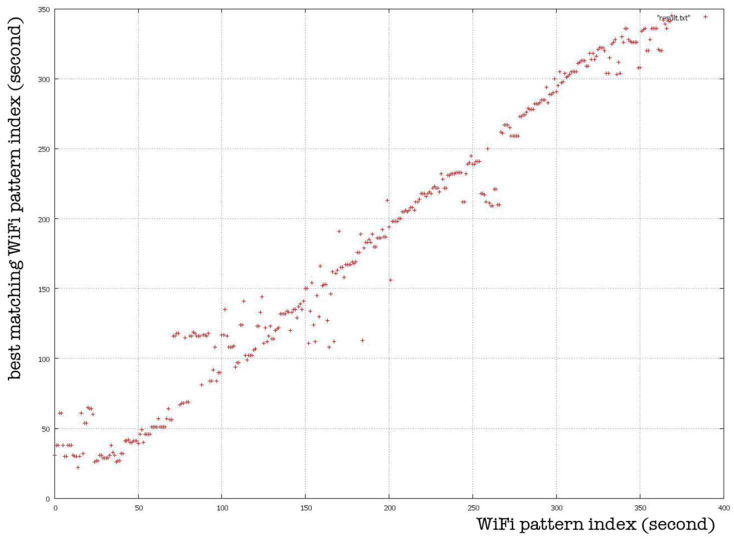
[KaemarungsiKrishnamurthy04] Kamol Kaemarungsi and Prashant Krishnamurthy, *Properties of Indoor Received Signal Strength for WLAN Location Fingerprinting.* School of Information Sciences, University of Pittsburgh, 2004.

[Kuepper05] Axel Kuepper, *Location-based Services, Fundamentals and Operation.* Wiley, 2005.

A Additional figures



(a) MAC addresses only



(b) including RSSI values

Figure 20: Positioning results of each second from the agent's path tracing

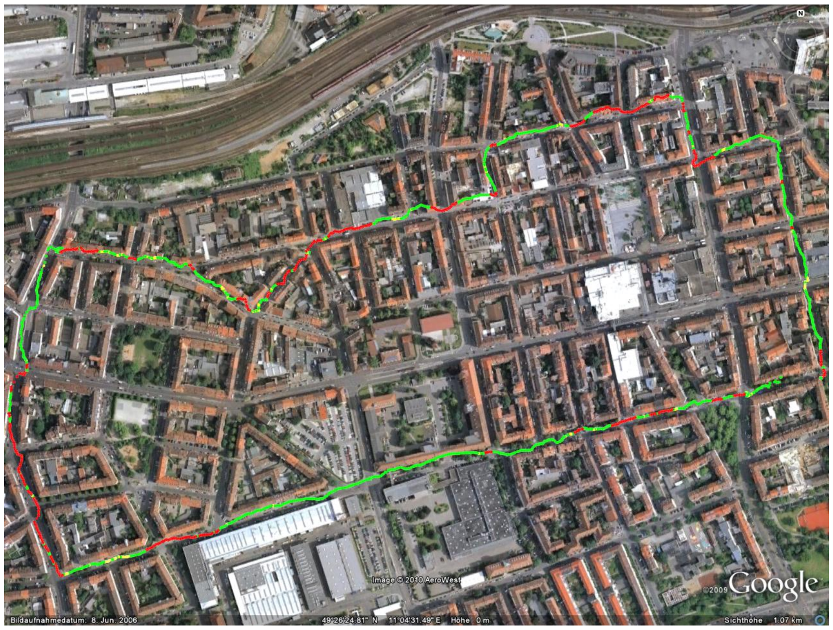


Image © 2010 AeroWest
©2009 Google
Bildaufnahmedatum: 8. Jun 2006
49°26'24.61" N 11°04'31.49" E Höhe 0 m
Sichthöhe 1.07 km

(a) First test run

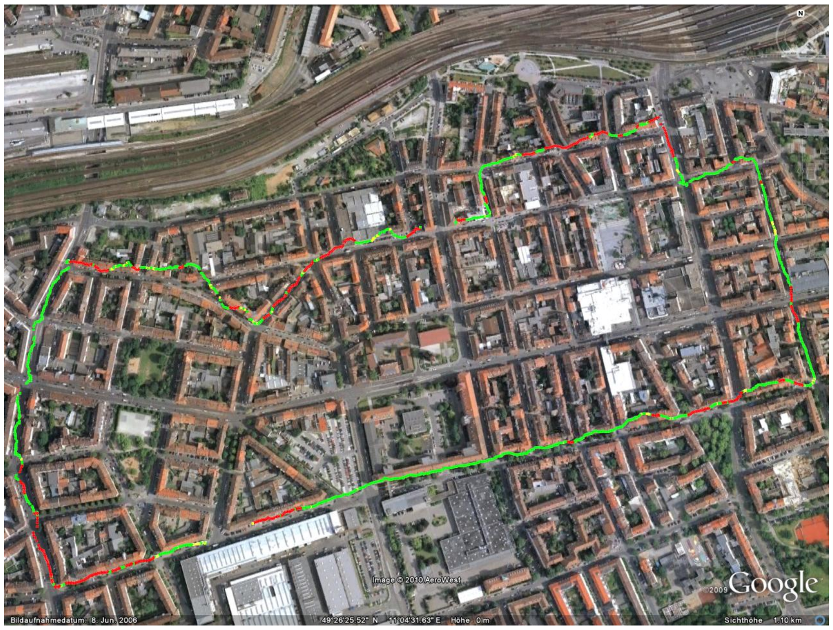
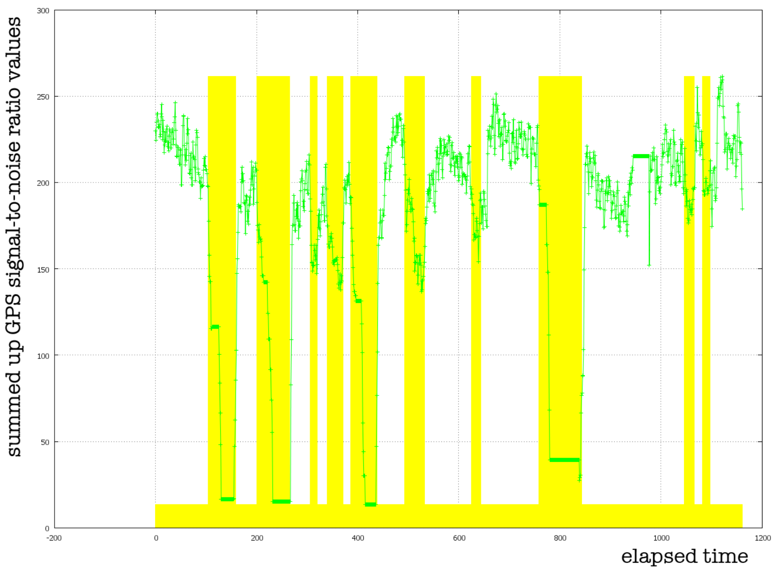


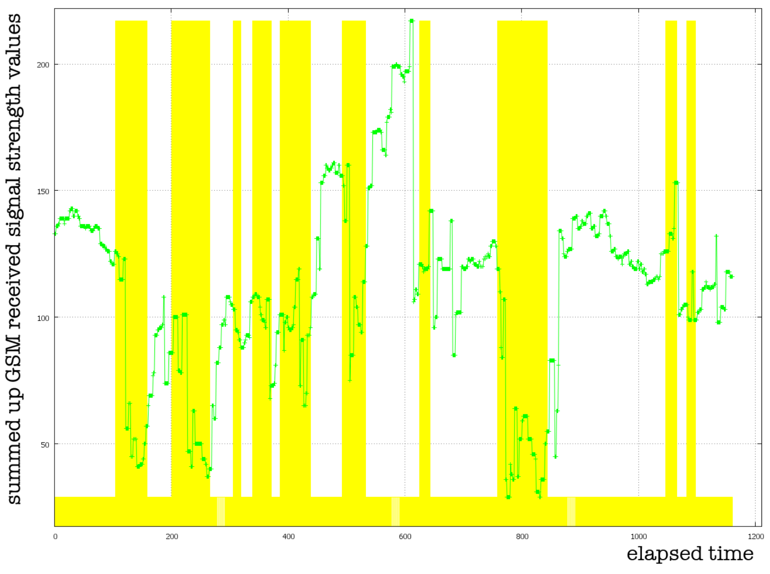
Image © 2010 AeroWest
©2009 Google
Bildaufnahmedatum: 8. Jun 2006
49°26'25.52" N 11°04'31.63" E Höhe 0 m
Sichthöhe 1.10 km

(b) Second test run 24 hours later

Figure 21: Results of the crossroad detection in the southern part of Nuremberg

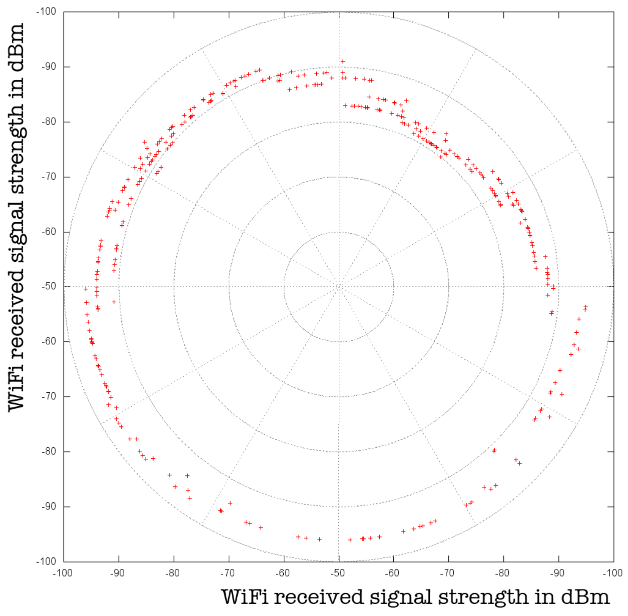


(a) GPS

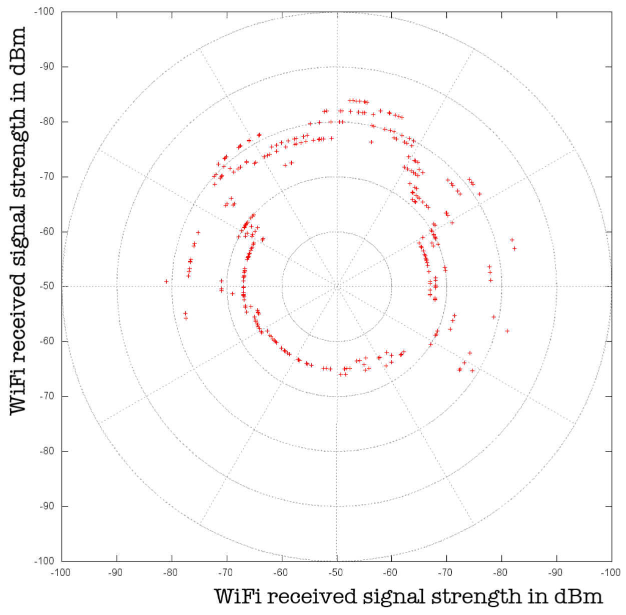


(b) GSM

Figure 22: Signal strength values in various types of development



(a) GPS

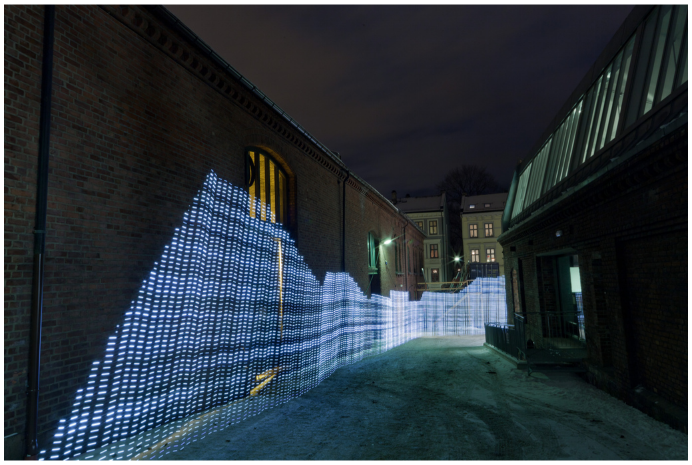


(b) GSM

Figure 23: Received signal strength values from two distinct access points recorded during two full rotations around the test subject's axis

B Immaterials: Light painting WiFi

In February 2011 three students from the School of Architecture and Design (AHO) in Oslo, Norway have published their work on an art project called *Immaterials: Light painting WiFi*. The idea was to make WiFi landscapes visible as they run through the streets of a large city. This was achieved by walking through the cityscape with a four-meter-long rod featuring 80 LEDs attached to it while taking a photograph of the scene with a long exposure time. The greater the WiFi penetration in a certain spot the higher the bar of lit LEDs. Since all the changes in WiFi signal strength are recorded within one exposure the resulting image is a signal curve drawn directly into the cityscape. Therefore this illustration indicates how WiFi penetration is directly correlated to its environment. This project connects the assessment of WiFi coverage from a technical point of view with an artistically appealing presentation and fits perfectly to what has been investigated in the context of this bachelor thesis. The images below are provided by courtesy of the three artists. For more information on this project please refer to their blogs[17].



[17]`yourban.no/2011/02/22/immaterials-light-painting-WiFi` or `nearfield.org/2011/02/WiFi-light-painting`