




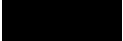
Department of Electrical Engineering and Information Technology
Distributed Multimodal Information Processing Group
Prof. Dr. Matthias Kranz

Map-Matching Algorithms for Android Applications

Map-Matching Algorithmen für Android Anwendungen

Ermin Sakic

Bachelor Thesis

Author:	Ermin Sakic
Address:	
Matriculation Number:	
Professor:	Prof. Dr. Matthias Kranz
Advisor:	Dipl.-Ing. Stefan Diewald
Begin:	01.07.2012
End:	23.10.2012



Department of Electrical Engineering and Information Technology
Distributed Multimodal Information Processing Group
Prof. Dr. Matthias Kranz

Declaration

I declare under penalty of perjury that I wrote this Bachelor Thesis entitled

Map-Matching Algorithms for Android Applications

Map-Matching Algorithmen für Android Anwendungen

by myself and that I used no other than the specified sources and tools.

Munich, October 19, 2012

Ermin Sakic

Ermin Sakic



Kurzfassung

Global Positioning System (GPS), Wi-Fi Positioning System (WPS) und das Cell ID System zur Positionsbestimmung repräsentieren die wichtigsten Lösungen für standortbezogene Dienste (engl. Location-based services (LBS)). Die Positionierungssysteme werden zu militärischen und kommerziellen Zwecken genutzt, zum Beispiel um die aktuelle Position des Benutzers zu ermitteln, eine Route zu navigieren oder interessante Orte (sogenannte Points of Interest) zu bestimmen. Der Anwendungsbereich ist breit und der Marktwert riesig. Es wird geschätzt, dass dieser bis 2015 eine Höhe von US \$29,8 Milliarden erreicht¹.

Map-matching erlaubt den Abgleich einer empfangenen Position (eines Punktes) oder einer Punktsequenz mit den vorhandenen Ortsinformationen einer digitalen Karte, um das richtige ("wahre") Straßensegment (auf welchem sich das Objekt bewegt) und die genaue Objektsposition auf dem Segment zu bestimmen. Die Ungenauigkeiten entstehen infolge der Abweichungen in der Synchronisation von Satelliten- und Geräteuhren, Mehrwegeausbreitung, ionosphärischen und troposphärischen Laufzeitverzögerungen, Ungenauigkeiten von Ortsinformationen der digitalen Karten oder Fehler in Berechnungen [1]. Sie können zu fehlerhaften und unbrauchbaren Ergebnissen bei den Abbildungen von GPS Aufzeichnungen auf die Straßen führen. GPS, im Vergleich zu WPS und Cell ID, bietet das beste Verhältnis von Genauigkeitsgrad und Systemverfügbarkeit. Zusätzlich wird GPS von den meisten modernen Mobilgeräten unterstützt. Aus diesen Gründen wurde beschlossen, GPS (und A-GPS) als die wichtigste Quelle für die Eingabeinformationen in unseren Algorithmen zu berücksichtigen.

Um die GPS Aufzeichnungen akkurat mit der unterliegenden Karte abgleichen zu können, wurden verschiedene Map-matching Algorithmen entwickelt. Diese sind in rein geometrische, topologische, probabilistische und fortgeschrittene Algorithmen eingeteilt [2] und werden in dieser Arbeit besprochen. Der generelle Ablauf, die Leistungsbewertungen, Vor- und Nachteile dieser Algorithmen sind Bestandteil der Arbeit. Abhängig vom Anwendungsbereich existieren offline und online (Echtzeitalgorithmen) Map-matching Algorithmen. Für unsere Zwecke werden nur generische Algorithmen, die in Echtzeit abgleichen und sich nicht auf zukünftige Informationen verlassen, betrachtet. Diese Algorithmen sind schnell und für die vorhandene Hardware in heutigen Mobilgeräten implementierbar. Ein kurzer Überblick von anderen, komplexeren Algorithmen, die auf Fuzzy-Logik und Partikel-Filter basieren, wird ebenfalls gegeben. Im zweiten, praktischen Teil der Arbeit wird eine optimierte Implementierung von einem fortgeschrittenen Map-matching Algorithmus (ST-Matching, erstmals eingeführt von Yin Lou et al. in 2009 [3]) präsentiert, ausgewertet, erzielte Ergebnisse diskutiert und Verbesserungsansätze vorgeschlagen. Der Algorithmus wurde in Java für das zur Zeit meistverbreitetste Mobiltelefon-Betriebssystem Android² implementiert.

¹http://www.prweb.com/releases/global_positioning_system/GPS/prweb4670914.htm

²http://www.comscore.com/Press_Events/Press_Releases/2012/9/comScore_Reports_July_2012_U.S._Mobile_Subscriber_Market_Share

Abstract

Global positioning system (GPS), Wi-Fi positioning system (WPS) as well as the cell tower positioning (Cell ID) are the most prominent solutions for location-based services (LBS). The positional systems are used for both military and commercial purposes, for example, for determining the user's approximate location, navigating a route, or looking for points of interest in the vicinity. The usage scenario of these services is broad and the value of the market immense, projected to reach US \$28.9 Billion by 2015³.

GPS, compared to WPS and Cell ID, offers the best balance between accuracy and system availability. Additionally, it is available on a broad range of today's mobile devices. For this reason, it has been decided to consider the GPS (and A-GPS) as main source of the input information processed in our algorithms. The process of map-matching allows for the GPS receiver to match the received positional observations (a point), or a sequence of positions (points), with the actual ("true") road segment the object is moving on and to determine the object's location on that segment. Inaccuracies resulting from the noisy input data, including but not limited to the differences in the clock synchronization between the GPS satellites and the receiver-device, multipathing, ionospheric and tropospheric propagation delays, inaccurate digital maps or hardware computation numerical errors may lead to erroneous and unusable results when projecting (matching) the GPS fix on the road [1].

To correctly reconcile the GPS observations with the underlying map, different map-matching algorithms, grouped into geometric, topological and probabilistic algorithms [2], have been developed and are discussed in this thesis. The general workflow as well as the advantages, disadvantages and a performance comparison of these is included. Depending on the usage scenario, post-processing (offline) and real-time (online) map-matching algorithms exist. For our purposes, only generic algorithms, which do not rely on future information and match in real-time will be considered. These algorithms are all performance-wise quick and possible to implement on the hardware used in modern cell phones. A short overview of other, more complex algorithms based on fuzzy logic and particle filtering is given. In the second part, that is, the practical part of the thesis, an optimized implementation of an advanced map-matching algorithm called ST-Matching, first introduced by Yin Lou et al. in 2009 [3], is presented, evaluated, achieved results after the matching process discussed and further optimizations proposed. The algorithm was implemented in Java for Android, which is currently the most popular mobile phone operating system⁴.

³http://www.prweb.com/releases/global_positioning_system/GPS/prweb4670914.htm

⁴http://www.comscore.com/Press_Events/Press_Releases/2012/9/comScore_Reports_July_2012_U.S._Mobile_Subscriber_Market_Share

Contents

Contents	v
1. Introduction	1
1.1. Global Positioning System	1
1.1.1. Global Positioning System - Error causes	3
1.1.2. Assisted-GPS	3
1.2. Wi-Fi Positioning System	4
1.3. Cellular Positioning	5
1.4. Comparison of the geolocation technologies and the reasons behind map-matching	6
2. Overview of map-matching algorithms	8
2.1. History	8
2.2. Geometric matching	9
2.2.1. The problem	9
2.2.2. Point-to-point matching	9
2.2.3. Point-to-curve matching	10
2.2.4. Curve-to-curve matching	11
2.3. A weighted topological map-matching algorithm for generic applications	12
2.3.1. Topological algorithms	12
2.3.2. The basic idea	13
2.3.3. Determining the correct link	14
2.3.4. Adjusting the weighting parameters	17
2.3.5. Determining the physical location of the vehicle on the link	17
2.3.6. Evaluation and further optimizations	19
2.4. Probabilistic matching	19
2.5. Advanced map-matching algorithms	21
3. Implementation of the ST-Matching algorithm on Android	22
3.1. The ST-Matching algorithm	22
3.1.1. Candidate detection	25
3.1.2. Spatial analysis	25
3.1.3. Temporal analysis	27

3.1.4. Final result matching and localization	28
3.2. Implementation	29
3.2.1. Google Android and its Network Location Provider	29
3.2.2. The application workflow and introduced optimizations	30
4. Evaluation	33
4.1. Test results	33
4.2. Possible improvements	35
5. Conclusion	38
Bibliography	39

Chapter 1.

Introduction

1.1. Global Positioning System

Developed and actively maintained by the United States government since 1978, the NAVSTAR-GPS, or simply "GPS", is a Global Navigation Satellite System (GNSS) used for position determination and navigational purposes. In addition to the GPS, other GNSS exist, with Russian GLONASS (GLObal NAvigation Satellite System) being the only other, fully globally functional GNSS. European Union's Galileo, China's Compass and Indian Regional Navigational Satellite System (IRNSS) are under construction as well. It should be noted that all algorithms described in this thesis are fully compatible with every one of these systems.

At first, the GPS was used as an advantage for the needs of the U.S. Forces and although becoming fully operational in 1994, the technology acclaimed commercial success only after in May 2000, the U.S. Government removed the previously enforced limitations on the technology. This removal allowed for non-encrypted GPS signals to be transmitted by the satellites and received by the GPS-enabled devices [1]. Since then, the technology has seen a huge market push and many different kinds of applications, using the location-based services, started emerging. For us, particularly interesting applications are those involving mobile end devices, letting the end-user enjoy all the benefits of positional and navigational services this system offers.

The system allows for the location-services to work properly whenever there is a clear sight of four or more GPS satellites and an unobstructed signal downstream from these. In total, there are 24 GPS satellites orbiting the earth in 12-hour orbits, transmitting the GPS signal. Based on the travelling time of the signal from satellite to a device, the distance between the two can be computed, and the actual position of the device determined. Since no return channel is necessary, an unlimited number of devices can receive these signals at once. At least four satellites are needed to compute the user's position in space. Three of these are needed to compute the latitude and longitude, the fourth one for altitude (3-dimensional positioning).

To lower the redundancy and strengthen the resistance to jamming, two different signals are

broadcast, L1 at frequency of 1575.42 MHz (wavelength of 19.05 cm) and the second, L2 at 1227.60 MHz (wavelength 24.45 cm). These signals are broadcast at low power, and for most receivers, the signal with strength of at least -130 dBm (typically somewhat higher, at -127.5 dBm) is required [4]. L1 is the free signal, carrying the navigation data and the standard positioning code (SPS) whereas the L2 only carries the precise (P-) code, and is normally used only in military applications, which require special receivers (designed for this purpose). The two carrier signals are modulated by three binary sequences. The Coarse Acquisition (C/A) code is a repeating 1023bit long pseudo-random code sequence, which modulates the L1 signal. The C/A sequence, broadcast periodically every 1 ms, is unique for every satellite transmitting the data, and allows for reduction of the interference liability and signal spreading of multiple satellites at the same frequency. The second code sequence, precise (P-) code, modulates both L1 and L2 signals, is encrypted and only accessible by the U.S. military. Finally, the navigation message modulates the L1 - C/A signal and carries the information related to the satellite clock, health and the precise orbit of the satellite (empheris). The last part of the navigation message, the so-called "almanach" code, contains information related to the positions of the other satellites in constellation, is needed for the initialization of the receiver and is typically updated once every 24h.

Let s_i denote the distance between the device j and satellite i , t_i the time at which the signal was broadcast by the satellite, t_j the time at which the signal was received (as seen on the device) and c the average speed of light in the air. Then the following holds true:

$$s_i = c \cdot (t_j - t_i) \quad (1.1)$$

However, this equation applies only when the satellite and receiver clocks are perfectly synchronized. Since the device's clocks are not as accurate as the atomic clocks of the satellites, these slight timing differences can lead to rather big errors in the distance measurements. For that reason, we refer to this propagation time as pseudorange ρ_j [5]. For the purpose of computing the device's clock offset Δt , the equation above is further expanded to:

$$\rho_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + c \cdot \Delta t \quad (1.2)$$

where (x_j, y_j, z_j) are the satellite positions (received in the empheris message) and (x_u, y_u, z_u) the user's coordinates. The user's position as well as Δt are at first unknown, but possible to compute after the GPS signals from at least four different satellites are received and a linear system consisting of the four equations solved.

1.1.1. Global Positioning System - Error causes

In addition to the clock discrepancies the following error sources also affect the GPS performance and lead to errors in computations:

- **Signal multipathing:** Multipathing can happen as a result of the transmitted signal being reflected by a tall reflecting surface (buildings, ground, rocks etc.). This leads to propagation delays before the signal reaches the device.
- **Troposphere and ionosphere side effects:** The atmosphere slows down the signals passing through it. This is the main reason why the satellites emit the signals on at least two frequencies. Almost all of the ionospheric effect can be removed by making measurements on two widely spaced frequencies and combining them [6]. The additional errors can be partially corrected by the built-in GPS error correction mechanisms and corresponding ionospheric-model parameters. However, these unpredictable errors remain and account for most errors in the calculations.
- **Obstructed sky:** Especially in dense urban areas, the line of sight between the GPS receivers and the satellites is limited. To be able to determine its own location, the receiver needs to lock to at least four satellites transmitting the GPS signal. This leads to unavailability of the locating services in most urban areas where tall buildings dominate, and to complete darkness in indoor and underground spots. However, these places are usually rich in Wi-Fi Access Points which can partially help in positioning where GPS signals do not get through.
- **Other errors include ephemeris (orbital) errors,** where satellites report false readings regarding their current position, and **satellite shading** where, as a result of tight grouping of satellites, interference of signals can happen.

All these lead to errors in the computations taking place locally on the GPS receiver. The process of map-matching, especially when backed up by high-quality digital roadmaps can make up for most of them and match the majority of the erroneous readings onto their respective, true positions on the road the vehicle is moving on.

1.1.2. Assisted-GPS

Time-to-first-fix, abbreviated as TTFF, is the time it takes the GPS-enabled receiver to fetch the information regarding the visible satellites relevant to its positioning after a cold start was initiated (or after the device reboot took place). During this time, the almanach data, a 15,000 bit long code sequence, containing the basic information about the current positions of other satellites in constellation, relevant for the receiver, from any visible satellite transmitting the GPS signals, is downloaded, and the device's position is computed. This code sequence contains general satellite

data, valid up to 6 days. The device usually takes up to 12.5 minutes to fully receive and process the data. As long as the data is valid, the receiver continues receiving the updated empheris (orbital) data, which is unique for every satellite the device gets the information from, so that faster location updates (every 20 seconds) are possible. Empheris data is much more detailed and valid only up to 4 hours. During the time the device is processing only the empheris and clock data, the device is considered "warm".

To shorten the 12.5 minutes it takes the device to determine the first fix, the A-GPS system was developed, which in addition to the built-in device GPS receiver also involves a mobile station for the locating purposes. The device relies on this mobile server to have the updated almanach data on its side most of the time, and to be able to transmit them to the receiver whenever there is a need for the data. This can shorten the TTFF down to approximately 15 seconds but would carry the burden of carrier billing with it in situations where the data providers count this transmission as a data access, which usually costs money. In case no data plan is available, the device would rely on the fallback mode and use the standalone GPS receiver without the advantages the A-GPS technology offers. However, many mobile phone manufacturers only offer partial GPS receivers in their cellphone offerings, which are A-GPS enabled but do not feature the full GPS functionality. A-GPS also helps in cases where a signal reflection (multipathing) might take place, for instance, in cities, and eventually, indoors, where visibility is limited and the time it takes for the device to seek the relevant satellites is prolonged. Originally, most of the final location computations took place on the remote servers as well. In this A-GPS mode, the device transmits a snapshot of the satellite data received to an assistance server, where the position fix is computed. This greatly reduces the CPU workload on the device. After the related standards were issued in 2001 [5], the receiver manufacturers were allowed for computing the final location locally, on the device, as well.

1.2. Wi-Fi Positioning System

Especially in dense urban areas where the functionality of the GPS-enabled receivers is limited, and a high number of Wi-Fi access points (AP) is deployed, the advantages of using the Wi-Fi Positioning System (WPS) techniques becomes obvious. These APs emit their signals in the wide area around them, and since they are stationary and rarely moved, their location is known and can be later used as a reference for the devices picking up the signal in question. Since only MAC address of the device and the corresponding signal strength matter for the localization, the Wi-Fi algorithms relying on the Wi-Fi positioning can use both encrypted and unencrypted signals without even connecting to the broadcasting device, and work well with both weak and strong signals.

Various algorithms focusing on Wi-Fi positioning have been developed, with the prominent being

those specializing on indoor spots while putting focus on the time-of-arrival using sensor measurements. These are the signal strength, time difference of arrival, time of arrival and angle of arrival measurements where the 2-way communication might take place and channel utilization has to be considered [7]. Additionally, both indoor and outdoor (generic) algorithms exist, consisting of two different phases, the so called training (offline) and online phase, in literature better known as radio mapping (fingerprinting, database correlation or pattern recognition) algorithms [8]. The training phase involves the periodical collection of "fingerprints", that is, the signal strengths and the unique MAC addresses from the emitting Wi-Fi APs at a known location (fix coordinates) and storing these in a large database. This process is also known as "wardriving" since it usually involves a vehicle and a laptop, with a GPS sensor mounted on, picking up the Wi-Fi signals, and storing those for the later use [8]. In the "online" phase, the device moving in an unknown area records its observed Wi-Fi signals and compares them with the fingerprints in the database, finding the closest match and determining its approximate location. This localization method can lead to very good positioning results in environments where a tight grouping of the Wi-Fi APs exists, resulting in accuracy measurements from 1 to 5 m [8–10].

However, all of these algorithms require a high number of APs set-up in the area, and are sensitive to location changes of the APs but serve as a great companion to the traditional GPS receivers, and are especially helpful in areas where the GPS signals do not come through. The results published by Cheng et al. [11] regarding the pioneer project in the field of the WPS technologies, Place Lab by Intel, reported average errors of 13 to 40 m following the positioning process. Nowadays, Skyhook Wireless is the most important provider of the fingerprinting algorithm, claiming sub-second TTFF, 10 to 20 meter accuracy and close to 100% availability where Wi-Fi APs exist¹. Skyhook Wireless was the only provider of Wi-Fi positioning services for Apple's iPhone from 2008 to April 2010 when Apple started utilizing its own WPS solutions². Today, the company offers its SDK for use in location-based applications for other operating systems (including Windows, Linux, Mac OS X and Android variants) as well³. Google manages its own database for the WPS available on Android, which contains the MAC and positional data of these. It is further populated by receiving new and updated information from the devices running the Android operating system, which act as collectors of the data.

1.3. Cellular Positioning

It is not rare for the GPS signals to be unable to reach the receiver, especially if no data connection is available and the device lacks the full GPS functionality (A-GPS). In remote areas, on the land and rural spots, no Wi-Fi access points might be available or no fingerprint associated with them

¹<http://www.skyhookwireless.com/location-technology/performance.php>

²<http://techcrunch.com/2010/07/29/apple-location/>

³<http://www.skyhookwireless.com/location-technology/index.php>

could exist in the database. However, the last resort, the possibility of locating using the base (mobile) stations might still exist. Although not as accurate as any of the above methods of determining the user's location, the cell networks are by far the most spread out and supported systems of positioning.

The structure itself consists of different cells (areas) which are accompanied by at least one cellular tower, which can communicate both ways with the device the connection was established with. Most popular method of locating using the cellular towers is "Cell ID", which, if device starts communicating with the station in question, assumes its location somewhere in the radius of the respective tower's signal broadcast service. Another way of locating, similar to the method mentioned under the Wi-Fi Positioning System sub-section, is using the so-called "Time of Arrival" algorithm, where multiple base stations send and receive the data to and from the device, suspected distance measurements are computed and the device's position in space approximated. This process is also known as trilateration since it requires at least three different base stations with known location which the device relates to.

Since the accuracy of the measurements depends on the number of cell towers in vicinity and the bandwidth of the cellular signal, the Global System for Mobile Communications (abbreviated as GSM) results with more reliable time of arrival measurements when compared to the Code Division Multiple Access (CDMA) [8]. Fingerprinting, similar to the methods used with Wi-Fi positioning, is possible, but not as popular since the efforts in the training phase are assumed to be too high and not necessary [8].

1.4. Comparison of the geolocation technologies and the reasons behind map-matching

Not many comparisons of the GPS, Wi-Fi and cellular positioning systems, benchmarked on mobile devices of newer generations, exist. However, the in-depth review of the iPhone 3G location services, published by Paul A Zandbergen in 2009, offers a detailed insight into the quality of the existing locating solutions [8]. The results are dissatisfying when compared to fully fledged standalone GPS receivers and require further refinements when used for navigational purposes where a need for accurate positioning is present.

Compared to the average median error of 1.4 m for multiple field tests using a Garmin GPSMAP 60Cx test unit the iPhone locating systems performed somewhat poorly:

- A-GPS positioning: Maximum error of 18.5 m and a root-mean-square-error (RMSE) of 8.3 m, still sufficient for most of the basic location-based services in iOS applications.
- Skyhook Wireless Wi-Fi positioning system: Median error of 74 m for 58 observations.

The system failed to meet the specifications and performance review published by Skyhook Wireless in 2008 [8].

- Cellular network positioning: Median error of 600 m for 64 observations. Seemingly, the technology used during the field test, at least in the iPhone 3G in 2009, was the basic Cell ID location lookup service.

Obviously, the raw results, presented in this form are rather unusable for any applications relying on accurate location measurements. Extreme care must be taken when using similar mobile devices for navigational and locating purposes in real-time, where the current position of the driver becomes of the highest priority and relies on the receiver to guide the way. Since the GPS receivers used in today's mobile phones are mostly of low-end quality, the need for the intelligent error correction functionality which helps minimizing the erroneous readings and offers reliable and accurate positioning appears. For the purpose of having the usefulness and reliability of the highest level in these intelligent transportation systems, during the last 15 years many different map-matching algorithms have been developed, ranging from purely geometric to probabilistic and advanced algorithms [12], specialized for different usage scenarios and various location-based services.

Chapter 2.

Overview of map-matching algorithms

2.1. History

Three generations of personal-navigation assistants (PNA) are in use today [13]. The first one, offering the advantages of a digital map, features the possibilities of searching for points of interest and other basic map functionality like scrolling, panning and zooming. The second generation devices enabled the positioning of the user and reconciling his/her position with his/her true location on the digital map. The last group of devices provides users with advanced navigational services. The main difference between the second and third generation of these PNAs is that in the second generation, the true location of the user could be anywhere on the map, whereas, when a need for navigating service appears, the user is naturally expected to be moving on a road in a vehicle. For the purpose of matching the raw GPS observations with the corresponding "true" road segments and determining the absolute location of the user on that segment, map-matching algorithms have been developed.

The simplest geometric map-matching algorithms started emerging already in 90s. Many different modifications of these have been developed, but all of them based on the primal principle of matching the location obtained from the GPS receiver to a closest node, road segment or an arc, sometimes considering the dead reckoning technique in their workflow as well [14]. Dead reckoning takes into account the previously determined direction of the vehicle (bearing, easting and northing, using gyroscope and compass), its speed (for instance by measuring the number of wheel rotations or using built-in accelerometer) and compares it with the newer data to approximate the new position of the vehicle. Since these algorithms had other usage scenarios as well, where the receivers were not always vehicle-based and input data for the dead-reckoning techniques unavailable, most of the work was done on the basic geometric techniques [13, 15]. The matching process showed up to be a rather complex task, and it was concluded by Bernstein and Kornhauser that, in order to get the feasible results, some kind of curve-to-curve matching must be incorporated and a considerable amount of attention should be given to topological information [15]. Since the results of these algorithms were based on trivial mathematical operations, especially at junctions

(crossroads) and U-turns, the geometric algorithms showed up to be extremely inefficient. The need for a new kind of algorithms which take the relation to previously matched points (historical data), timestamps of the GPS observations and relations based on digital map data (node to node relation, topological information) appeared. These advanced map-matching algorithms are called topological algorithms. They incorporate some kind of weighting score techniques when choosing between the candidate segments taken into observation and consider way information to avoid sudden switching between unconnected road lines [16, 17]. Additionally, probabilistic map-matching algorithms have been developed, considering probability theory when identifying the candidate segments and measuring the weighting scores and mutual influences between candidate points and segments [3, 18]. Later on, more complex algorithms emerged, relying on advanced techniques such as the extended Kalman filter [19] or fuzzy logic [20] but which require much more input data than the purely geometric and topological algorithms and usually result in slower matching times and higher hardware workload. All of these topological, probabilistic and advanced algorithms, in some way, incorporate the basic map-matching geometric techniques, usually, the point-to-line matching.

A general overview of the above described map-matching techniques will be given in this chapter.

2.2. Geometric matching

2.2.1. The problem

To formulate the problem of location matching, let the point p_0 (known latitude and longitude) represent the first GPS observation after the cold boot, as returned by the GPS sensor and let N denote the road network consisting of at least two different waylines, trajectory $A = (a_0, a_1, a_2 \dots a_n)$ and trajectory $B = (b_0, b_1, b_2 \dots b_n)$, where a_0 and b_0 represent the start nodes of the two trajectories, $(a_1, a_2 \dots a_{n-1})$ and $(b_1, b_2 \dots b_{n-1})$ the corresponding shape points. To simplify the problem, we assume that the trajectories consist of an infinite number of these shape points. Assuming that the GPS observation p_0 is not located on any of the two trajectories but somewhere between the two, the trajectory which corresponds to the true location of the user has to be determined, and the GPS observation reconciled with a shape point belonging to the trajectory in question, representing the true location of the user.

2.2.2. Point-to-point matching

The most obvious and simple way of determining the true location would be to find the shape point closest to the raw GPS observation, belonging to any of the trajectories in the close area. For this purpose, a "range query" with a reasonable radius would need to be conducted and the

shape point with the smallest distance to the GPS observation identified. This is the simplest form of geometric matching, also called "point-to-point" matching. This method is very fast and easy to implement. However, it is very sensitive to the number of shape points building the candidate trajectories. It might be expected for the results to be much more accurate when using digital maps consisting of waylines with a dense distribution of shape points, which also may be true (although it considerably enlarges the spatial map data set), but where the algorithm fails to match correctly when, for instance, challenged with the scenario shown in Figure 2.1. Assume that the vehicle is moving on trajectory A, and p_0 is the raw GPS observation. Since the shape point b_1 is closer to the p_0 than any of the shape points belonging to trajectory A, the point-to-point algorithm would falsely match the GPS observation to b_1 .

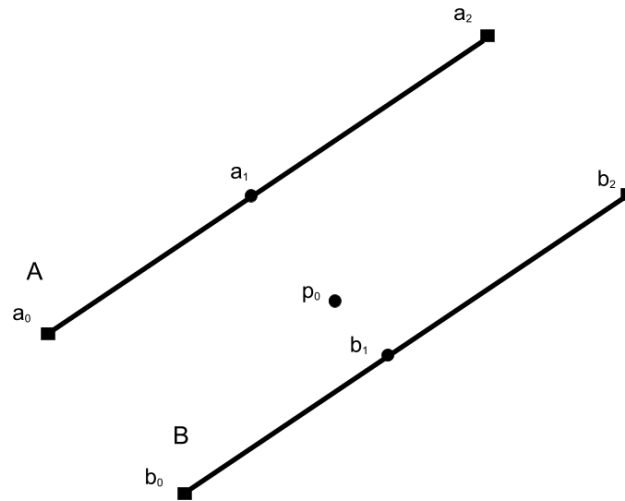


Figure 2.1.: Point-to-point matching

2.2.3. Point-to-curve matching

Different algorithms, based on the point-to-curve matching have also been developed, with the calculations of distance between the GPS observation and the linear segments building a curve (lines) being slightly more complicated than when measuring the point-to-point distances. Similar to point-to-point matching, a few close nodes (start/end nodes or shape points) in the area around the GPS point need to be identified. The line segments corresponding to these represent the candidate segments. As proposed by White et al. in 2000 [13], using the minimum norm projection, the distance to these segments can be computed and the closest one chosen as the true segment. Both, the approach and disadvantage of this method can be comprehended by observing the Figure 2.2, where p_0 , p_1 and p_2 would match correctly, on the road segment A, but in case of p_3 where the estimated distance to both arcs is shown to be equal, the algorithm could eventually match the GPS observation onto the wrong line segment B. Although it might

be obvious that the vehicle is still moving on the same trajectory, no historical or any other kind of relational data (consideration of the previously matched points) is included in the workflow of these algorithms.

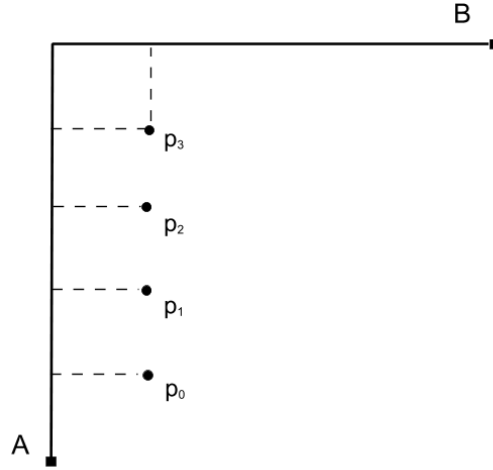


Figure 2.2.: Point-to-curve matching (figure adapted from White et al., 2000 [13])

2.2.4. Curve-to-curve matching

Curve-to-curve matching is based on distances between curve segments. Similar to the above methods, the relevant nodes in the area surrounding the GPS fix and the set of arcs incident to these nodes are identified. Piece-wise linear segments are built from the GPS observations ($p_0 \dots p_n$), requiring at least two consecutive GPS readings, where any previous fix acts as the start node with its follower representing the end node of the newly created line segment. The distance from this curve-segment to all the candidate segments in the area is measured, and the closest one taken as the correct one, projecting the point on it. The approach, when measuring the distance between the curve constructed by connecting the two GPS observations and the candidate road segment, as proposed in [13], is calculating the "subcurves" of equal length. As illustrated in Figure 2.3, the distances between the GPS observations (actual lengths of the newly created arcs) were measured (30 m and 10 m here, respectively), and the points m_1 , 30 m along the road segment A, and m_2 10 m further from the point m_1 on the same segment, used for distance measuring purposes. The distance d between the two curves can then be computed as follows:

$$d = \|p_0 - m_1\|_2 + \|p_1 - m_2\|_2 \quad (2.1)$$

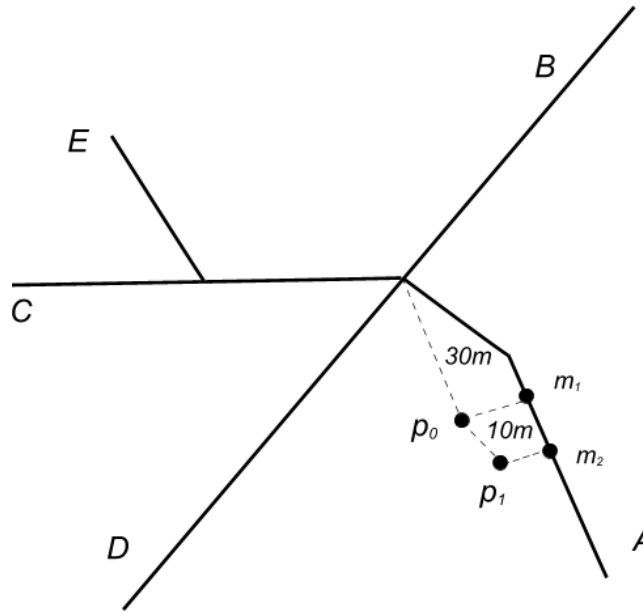


Figure 2.3.: Curve-to-curve matching (figure adapted from White et al., 2000 [13])

In case this distance is the shortest one - compared to the distances to other candidate segments, the points m_1 and m_2 are identified as the correctly matched points for this set of GPS observations.

2.3. A weighted topological map-matching algorithm for generic applications

2.3.1. Topological algorithms

The authors of the most important map-matching algorithms all agree that, in order to have an algorithm resulting with the majority of correct reconciliations, initial identification of the correct link between all the candidate segments represents the most important step [12, 16, 21]. If executed poorly, a wrong initial match can result with incorrect matchings for the succeeding observations. Also, close attention should be paid to introducing as many topological aspects in the algorithms as possible, which are especially important when faced with intersections, parallel road arcs and urban areas where complicated network road maps dominate [2, 15].

The first such properties were taken into account in an algorithm published by White et al. in 2000 [13]. In addition to the geometric measurements, the heading information of the GPS observation were compared with the arc headings. In case of big differences in the heading, the compared candidate arcs would be discarded. This concept was further developed in the algorithm published

by Greenfeld in 2002 [16], which compares the orientation of a GPS line created from two GPS observations with the orientation of the candidate road links. However, the vehicle heading was not taken into account, and for that reason, the algorithm performed poorly at crossroads and U-turns, where different road segments with the same orientation appeared. A solution to this problem was given in the paper published by M. A. Quddus et al. in 2003 [17], who recognized the value of combining the GPS and dead-reckoning headings to identify such examples and correctly match the GPS observations on the map. Also, in situations where a vehicle is considered stationary - mostly at intersections, it has been empirically proved for the GPS receiver to report false readings more often than when moving with higher speeds. In the same paper, this limitation was looked at and the solution of considering a single position fix in such situations proposed. Since this algorithm was developed and expanded further, becoming the most-cited topological algorithm¹, a detailed review and the optimizations that followed in later publications will be given in this section.

2.3.2. The basic idea

The algorithm outlined here, heavily based on the solutions published by Greenfeld in 2002 [16], and Quddus et al. in 2003 [17], assumes that both the positional information from the GPS observations, and the topological information (including speed and heading data) are available as input data for the algorithm. The weighting factors of the algorithm are determined empirically and, depending on the importance given to the relevant sensor readings and computations (detailed below), the total weighting score for a candidate segment is computed and the segment providing the highest score identified as the correct one.

The matching process consists of a search for the candidate segments in the area by identifying the relevant nodes and road lines connected to it. After the correct candidate segment has been identified and physical location of the vehicle on the link determined, verification whether the next GPS observation can still be matched on the same link is done. If that is the case, the physical location on that link is computed (using the speed and bearing readings to measure the distance the vehicle moved from the last matched location), or else the observation is taken as the initial point from which the matching process should begin again. This means that the most important task in establishing the guarantee of high accuracy matchings is identifying the correct first segment from the possible candidates. Greenfeld proposed considering the orientation (the similarity in orientation of a line created by two consecutive GPS readings and the candidate street arc) and proximity (distance measured between the GPS observation and the candidate arc) of the GPS fix compared to the candidate road segments. By changing the weighting parameters, the effect of the two parameters on the final weighting score for a candidate arc is adjustable.

¹http://scholar.google.de/scholar?hl=en&as_sdt=0,5&q=topological+map-matching+algorithm

The total weighting score S_t for a line segment is computed as follows:

$$S_t = S_{ho} + S_p + S_i + S_l \quad (2.2)$$

S_{ho} represents the weighting score considering the vehicle heading and orientation of the link. S_p considers the distance (proximity) of the GPS observation to the candidate line. S_i is the weighting score considering a possible intersection between the line built from two GPS observations and the candidate road segment, and S_l the weighting score depending on the position of GPS observation in regards to the link.

2.3.3. Determining the correct link

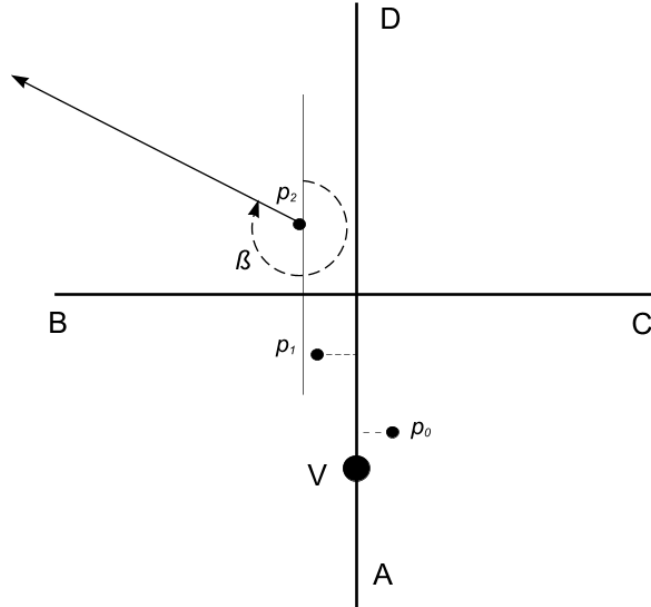


Figure 2.4.: Vehicle heading and orientation of the link (adapted from Quddus et al., 2003 [17])

Assuming the vehicle V , represented as a dot in the Figure 2.4, is moving on the trajectory A. A few moments later, the vehicle comes to an intersection and turns left (continues its movement on the way segment B). The GPS observations p_0 and p_1 are matched correctly, to the road segment A, but to be able to match the p_2 , one has to choose between the line segments B, C and D. For that purpose, the GPS/DR heading, the vehicle has at the time the p_2 is returned, is utilized. The angle β represents this heading, measured relative to the northerly direction. The angles β'_i with $i = (A, B, C, D)$ represent the same relation, but for the candidate links in question (so that $\beta'_B = 270^\circ$, $\beta'_C = 90^\circ$ and $\beta'_D = 0^\circ$). With that in mind, the weighting score S_{ho} for vehicle heading and orientation of a candidate link i can be computed as follows:

$$S_{ho} = A_h \cdot \cos(\Delta\beta'_i) \quad (2.3)$$

where $\Delta\beta'_i = \beta - \beta'_i$, and where A_h represents an adjustable weighting parameter.

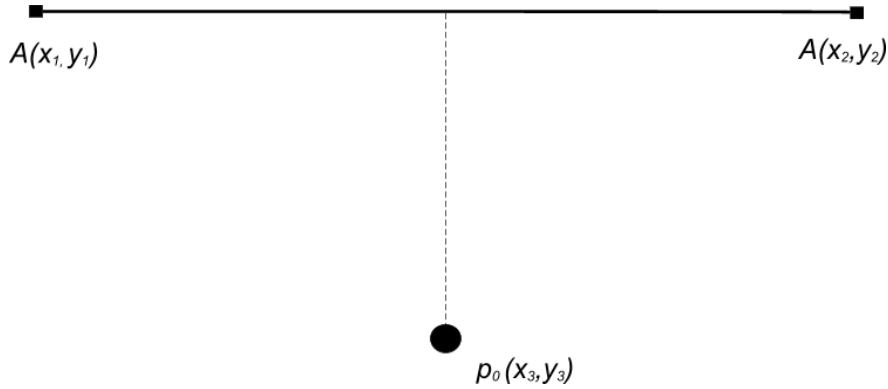


Figure 2.5.: Distance of the GPS observation to the candidate link (adapted from Quddus et al., 2003 [17])

For proximity measurement score, the GPS observation is projected onto the line of interest and the perpendicular distance to the line segment is determined. Assuming the scenario presented in Figure 2.5, the distance d from the GPS fix p_0 to the candidate line segment is:

$$d = \frac{x_3 \cdot (y_1 - y_2) - y_3 \cdot (x_1 - x_2) + (x_1 \cdot y_2 - x_2 \cdot y_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \quad (2.4)$$

It is understandable that the distance between the link and the GPS observation should be inversely proportional to the proximity score declared to this candidate line, since the larger distances also mean a lower probability that the line segment in question should be considered the correct match for the GPS observation. With that in mind, one can define the weighting score for the distance (proximity) of the GPS observation to the candidate line as follows:

$$S_p = \frac{A_p}{d} \quad (2.5)$$

where A_p represents an adjustable weighting parameter for the proximity score. The further away from the candidate line the point is, the lower its proximity and overall score will be.

Also, an occurrence described by Quddus as an almost certain indication of a link being the correct match, is when the line built from two successive GPS observations (with the observations being the start and end nodes of the line) intersects with a candidate link. An example is presented in Figure 2.6, where the line built using the two GPS observations p_1 and p_2 intersects the road

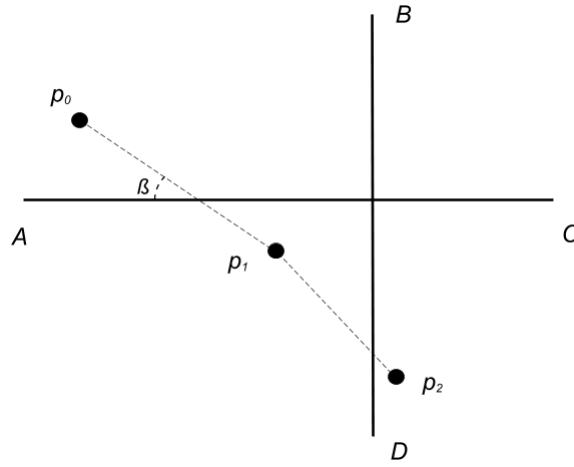


Figure 2.6.: Intersection of the two lines

segment A, building the angle β . In such case, the additional score should be added for that candidate link, the weighting score for an intersection S_i :

$$S_i = A_p \cdot \cos(\Phi) \quad (2.6)$$

where A_p represents the weighting parameter used before in the distance score measurement.

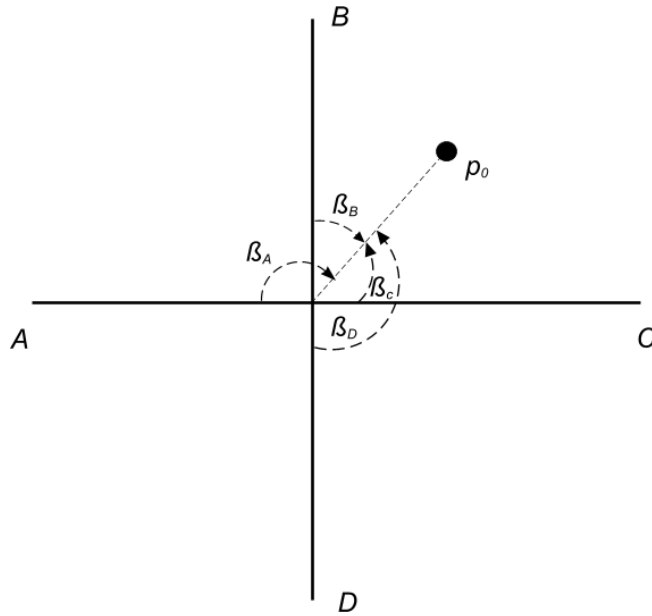


Figure 2.7.: Angle of GPS observation in regards to the link (adapted from Quddus et al., 2003 [17])

Finally, the last value affecting the overall score for a candidate link is its relative position on the plane when compared to the GPS observation. In Figure 2.7, let the angles β_i represent the positional relation between the candidate segments A, B, C, D and the GPS observation p_0 . The larger the angle β_i is, the lower the relationship score should be for the corresponding link. The closer (smaller angles) ones should lead to a higher overall score. For that reason, a cosine relationship is used in the equation, and the weighting score S_l , depending on the position of GPS observation in regards to the link, can be obtained as follows:

$$S_l = A_k \cdot \cos(\beta_i) \quad (2.7)$$

where A_k represents an adjustable weighting parameter for the fix - line segment relationship score. The sum S_t of the four scores (Equation 2.2) represents the overall score for a candidate line segment. The segment for which the highest score was computed, is taken as the most probable match, and the physical position of the vehicle on that segment determined.

2.3.4. Adjusting the weighting parameters

It has been proven empirically that a higher importance should be given to the weighting score for heading (S_{ho}) than to the score for the relative position (S_l) [17]. Also, the weighting score for the relative position (S_l) should affect the end-score stronger than the sum of the two proximity scores ($S_p + S_i$). This leads us to the recommended weighting parameters when using this algorithm for a real-world matching problem. The relationship between these (A_h , A_p and A_k) was proposed by Quddus et al. as follows [17]:

$$\begin{aligned} A_h &= a \cdot A_p \\ A_k &= b \cdot A_p \end{aligned} \quad (2.8)$$

with a and b representing the new weighting factors, determined individually for different sensors used for the measurements, and dependent on their statistical accuracy.

2.3.5. Determining the physical location of the vehicle on the link

To determine the vehicle's physical position on the link, either a perpendicular projection on the link may follow or the GPS/DR data including vehicle speed and bearing should be used. Assuming that the position of the vehicle at the point $p_1(E_1, N_1)$ is known (as depicted in Figure 2.8), as well as its average speed v on the segment, the increments in the easting and northing can be computed as follows:

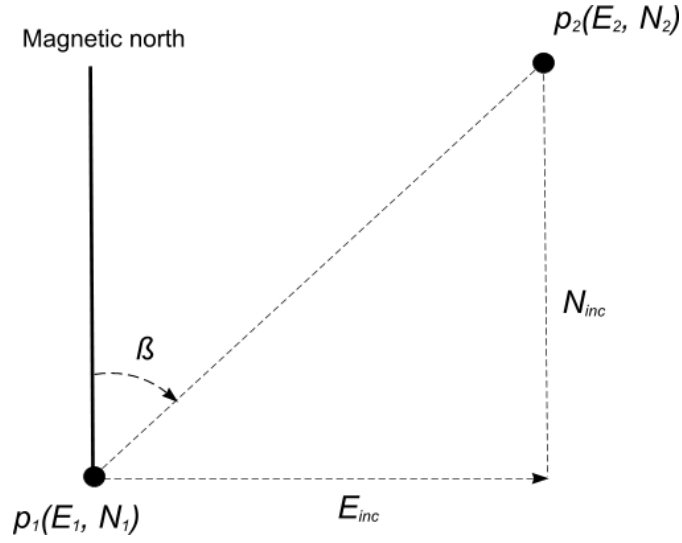


Figure 2.8.: Determining the location on the link (adapted from Quddus et al., 2003 [17])

$$\begin{aligned} E_{inc} &= v \cdot \sin(\beta) \\ N_{inc} &= v \cdot \cos(\beta) \end{aligned} \quad (2.9)$$

where β represents the bearing between the two location, and the new location $p_2(E_2, N_2)$ determined by adding the incremental easting and northing values to the known location coordinates:

$$\begin{aligned} E_2 &= E_1 + E_{inc} \\ N_2 &= N_1 + N_{inc} \end{aligned} \quad (2.10)$$

Assuming that an initial matching process was concluded successfully, a link match has been found and the physical location on the link has been determined, any succeeding GPS observation will be examined for whether the vehicle is still moving on the same road segment, or has switched to another link (turns, different streets etc.). In case it is still moving on the same road, only the incremental easting and northing values will be computed (with regard to the previously determined position), and if not, a new initial matching process will take place and the candidate segments in the vicinity evaluated. Three optimal methods were proposed in a later publication [21] by the same author:

- Detection of an increasing trend in heading for about 2 to 5 seconds
- The absolute difference in headings of two consecutive GPS observations is bigger than 30°

- The absolute difference in headings of current and the second last GPS observation bigger than 35°

If any of these assumptions are identified as true, the vehicle is assumed to be taking, or to have taken a turn, and the map-matching process is executed anew.

2.3.6. Evaluation and further optimizations

Results from tests executed on a complex road network in London are presented in the paper published in 2007 by Quddus et al. [2]. They show that the algorithm correctly identifies 88.6% road segments and has an average horizontal accuracy of 18.1 m. The tests were performed with various scenarios, different driving maneuvers including U-turns, waiting for traffic lights and traveling through tunnels. Improvements to the basic weighting algorithm presented in another paper by Ochieng et al. [21] included the optimized process of outlier identification (which will be discussed in the Chapter 3), more frequent use of historical data, additional methods for detection of turns and intersections and the candidate segment locating using error ellipse (discussed in the next section). However, the idea of using the weighting algorithm with different scores for orientation, proximity and link relation remained the same, and was not a subject to change in any updates to the algorithm.

2.4. Probabilistic matching

With probabilistic map-matching algorithms, combinations of techniques and methods based on topological information and probability theory are meant. As published by Zhao in 1997 [22], the variance and covariance error values, as received by the GPS receiver, can be used for computing an error region in which the correct road segment is assumed to be found. In the usual cases where multiple candidate segments exist, the algorithm would rely on proximity and orientation measurements to identify the correct one. For this step, combination of different techniques dependent on topological aspects of the situation (including the one presented in the previous section) could be used.

This concept was first used in a real-world implementation by Ochieng et al. [21], who divided the matching process in two different parts, the Initial Matching Process (IMP) and the Subsequent Matching Process (SMP). Since the IMP is used for the initial match, which succeeding matchings are based on, big importance was given to the identification of the candidate segments, and the correct link. In the IMP, this concept was used for determining an error ellipse around the GPS observation and the initial match, using the variance-covariance information as returned by the sensor, whereas, the second process helped in matching the consecutive GPS fixes onto the same

segment. As proposed by Zhao in 1997, the error ellipse parameters a and b (Figure 2.9) can be computed as follows:

$$\begin{aligned} a &= \sigma_0 \cdot \sqrt{1/2 \cdot (\sigma_x^2 + \sigma_y^2 + \sqrt{(\sigma_x^2 - \sigma_y^2)^2 + 4 \cdot \sigma_{xy}^2})} \\ b &= \sigma_0 \cdot \sqrt{1/2 \cdot (\sigma_x^2 + \sigma_y^2 - \sqrt{(\sigma_x^2 - \sigma_y^2)^2 + 4 \cdot \sigma_{xy}^2})} \\ \Phi &= \pi/2 - 1/2 \cdot \arctan\left(\frac{2 \cdot \sigma_{xy}}{\sigma_x^2 - \sigma_y^2}\right) \end{aligned} \quad (2.11)$$

with σ_x^2 and σ_y^2 being the positional error variances as returned by the GPS sensor, σ_{xy} the corresponding covariance, Φ the orientation of the ellipse relative to the North and σ_0 the expansion factor (compensation for the GPS errors discussed in Chapter 1, particularly the orbital errors, ionospheric and tropospheric delays and signal multipathing). For optimal results, Ochieng et al. recommended an expansion factor value of 3.03 (empirically concluded) [21].

In the same algorithm, the computation of the error region is only used for initial matching and at intersections, mostly for performance reasons, but also to eliminate the possibility of finding a false match when multiple candidate links have been identified around the road link the vehicle is moving on. Also, when determining the physical location on the link, a novel solution was presented in this paper, considering different error sources relevant to the sensors and spatial map data set in question.

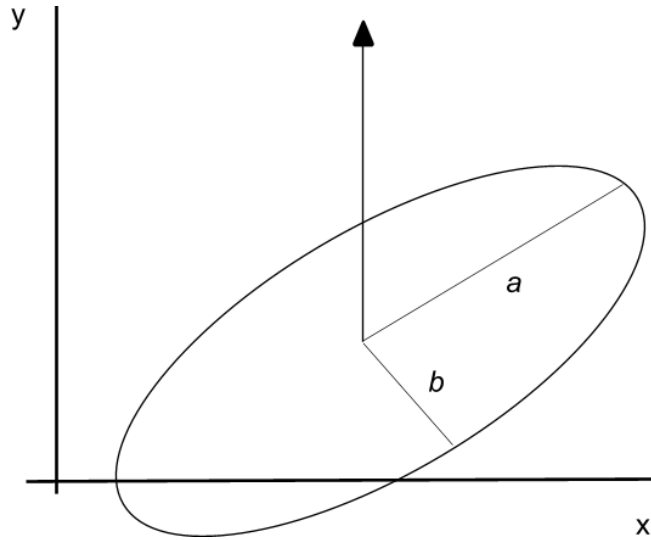


Figure 2.9.: Elliptical error region (adapted from Ochieng et al., 2003 [21])

2.5. Advanced map-matching algorithms

The algorithms based on the more complex concepts such as particle filtering, fuzzy logic or extended Kalman filter do usually result in better matches when processing whole trajectories (built from GPS observations) and identifying the corresponding true path. However, these are mostly used in situations where offline processing is an option and the processing backed up by, performance-wise, better hardware than what is available in the conventional mobile devices.

First introduced by Fu et al. in 2004 [23], the use of fuzzy logic model for determining the correct line segment for a GPS observation showed up to be a rather unreliable solution when confronted with the vehicle moving at low speeds. The lower speeds negatively affect the accuracies of headings and positional information of the readings. As input for the algorithm, the difference in the headings of the vehicle and the candidate segment, as well as the proximity (distance) between the GPS observation and the candidate line segment have been used. Since no historical data was considered either, Quddus et al., in 2006 [20], proposed their solution to the matching problem, relying on the link connections and speed information as well. Three set of fuzzy rules have been defined, one for each of the three phases of the algorithm - initial matching, subsequent matching and matching at junctions.

Another advanced algorithm, relying on particle filtering (Bayes filter), and dead reckoning techniques, was published in 2002 by Gustafsson et al. [24]. Except the digital map information, which is needed to reduce the possible paths of movement, the algorithm required the knowledge of the dead reckoning (wheel rotation speed) information for its computations. The tests while using this matching algorithm have shown a bad performance in the initial matching, but getting almost sub-meter accurate results by using the recursive Bayesian filter operations for a longer time period. The initial position is determined using a traditional built-in GPS receiver, or any other method of positioning, to put the constraints on the area space and limit the number of the particles. This makes the algorithm a reasonable companion to such a positioning system (after the initial match, no further need for any other positioning systems exists). The algorithm has performed almost equal in rural, and better than traditional GPS receivers in dense urban areas (cities).

Chapter 3.

Implementation of the ST-Matching algorithm on Android

3.1. The ST-Matching algorithm

In this section, a detailed review of a modern algorithm will be presented, originally published as a low-sampling-rate map-matching solution for generic matching problems [3]. However, for our purpose, it can be optimized for fast, high-sampling-rate observations, as they are possible today. The ST-Matching (S for Spatial, T for temporal analysis), in contrast to the topological and probabilistic matching algorithms reviewed above, not only takes the topological (positional) aspects of the area as input data, but also employs temporal analysis in its workflow. This makes it very flexible not only when used in scenarios where a high number of GPS observations in a short term interval is available, but also for trajectories where the GPS "ticks" appear with a lower rate. Most of the other algorithms are incremental algorithms, which consider up to two previous GPS points when matching a new tick to its candidate road position. They all rely on using the local topological data for the matching purposes, making the algorithms almost obsolete when confronted with problem scenarios where the GPS observations appear only in longer intervals. The ST-Matching, however, can be used both as a global algorithm (matching an entire trajectory with the road network), or as shown in this thesis, as a modified local (incremental) algorithm, with a minimum fixed window size of two GPS ticks (although, using only spatial/proximity measurements, the first fix can be matched right away, as well).

To better explain the advantage of the temporal analysis for the matching process, consider Figure 3.1. It shows a service road segment *A* (max. speed 30 km/h) next to the highway segment *B* (max speed 120 km/h). A vehicle with an on-board GPS receiver is moving on the segment *B*. Although the two GPS ticks are found to be closer to the service road *A*, the ST-Matching would map the two points correctly, on the highway segment *B*, since it involves computing the average speed between the two point as the base for the temporal analysis and comparing the average speed with the spatial map data, where available (at most locations, using the Open-

StreetMaps data sets). Obviously, the speed information can be very useful in matching processes.

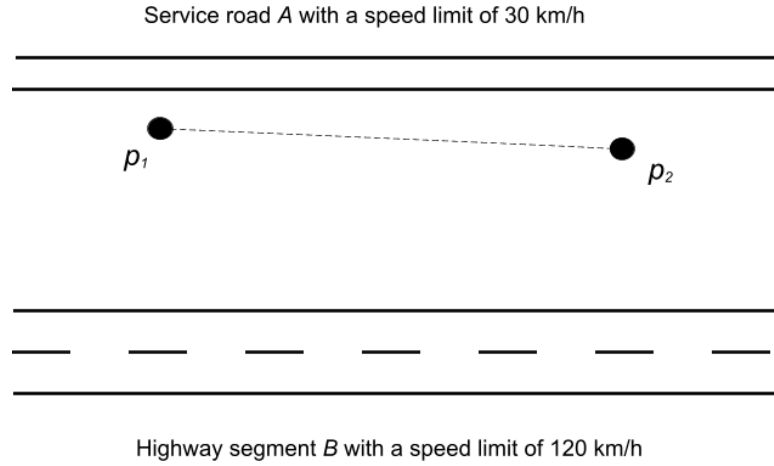


Figure 3.1.: Matching onto the correct road segment (adapted from Lou et al., 2009 [3])

Since the algorithm does not rely on dead reckoning or accelerometer readings, but only requires GPS (A-GPS) observations with basic positional information (latitude, longitude, heading, timestamp) and accurate spatial map data, all kinds of mobile devices and positional observations can be used in the matching process, independent of the built-in GPS receiver. Actually, not only A-GPS observations can be used, but Wi-Fi and cellular positioning data as well, although usually negatively affecting the accuracy, as explained in Chapter 1. The application developed for this thesis uses the speed information, which can either be obtained directly from the GPS receiver or computed locally, on the device, using the distance and temporal information. The heading information is used when available (for A-GPS observations), but in situations where only raw Wi-Fi or cellular positions (which do not contain any speed or heading information) are received, the bearing between at least two such fixes is computed locally on the device. Bearing information is used for the spatial analysis and in another modification originally included in this application, the so-called "outlier identifier" process, which identifies the points which may appear randomly as a result of the A-GPS inaccuracies. We refer to such illogical GPS observations as "spikes". The details around this correction process will be given later. An additional reason for choosing the ST-Matching over other popular map-matching algorithms is its fast running time and low hardware requirements, making it an ideal matching solution for implementation on the mobile devices of today.

The algorithms itself, is composed of four steps, those are:

- Candidate detection: For matching-process relevant nodes and the set of connected links in a reasonable radius around the GPS observation are identified and the corresponding data is filtered and prepared for the processing. Then, the process of interpolation of the road

segments (by using a PostGIS query) to increase the density of the shape points, which build the road segment in question, takes place. After that, the determination of physical location on the link takes place.

- **Spatial analysis:** Concerns the proximity measurements, including computing the likelihood that a road node represents the correct match based on the distance measurements and topological aspects such as determining the probability of two road nodes building the segment the vehicle moved on - probability that the two of them build the "true" path.
- **Temporal analysis:** Determines the average moving speed between two neighboring points, takes the speed constraints as included in the spatial map data into consideration, and helps when matching a GPS trajectory (the line built from two succeeding GPS observations) with a road segment by comparing the available speed information of the two segments.
- **Final result matching:** After the spatial and temporal results are available, the correct segment the vehicle was, or is still, moving on, and the physical location on it can be determined. This is done by building a graph (Figure 3.2) after at least two GPS observations p_i (here, p_1 and p_2) are available which shows the relations of all the candidate positions identified after both GPS readings were returned. The candidate points c_i^j represent the nodes of the graph and the edges of the graph $c_{i-1}^t \rightarrow c_i^s$ - the shortest paths between the neighboring candidate points. The points are assigned weight values based on the results of spatial and temporal analysis and the most probable link (route, with the highest score) is determined.

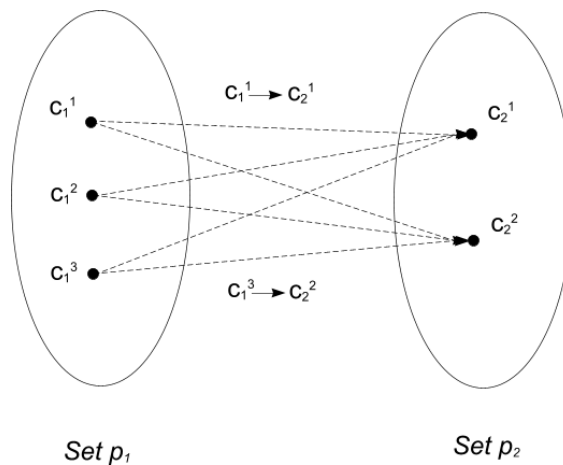


Figure 3.2.: Building a graph

3.1.1. Candidate detection

To find the candidate points the scoring process should be executed for, we first identify the road segments in a defined radius around the GPS observation, in which the true location is expected to be found. The radius can be defined either by using a reasonable constant, in our case, a radius of 120 m for A-GPS observations, or by using the value returned from the Android's `Location.getAccuracy()` method if the value is available. The accuracy returned in meters represents the Circular Error Probable Google is using for the approximation of the error, and represents a radius of a circle in which the matching result is expected to be found. This value, however, does not always seem to be correct. For that reason, it is multiplied with 1.5, to further increase the circle boundaries. The relevant candidate segments are identified by using PostgreSQL + PostGIS database queries, and furthermore, interpolated, resulting in a set of points which correspond to all the nodes building the candidate segments. Those nodes which are geographically positioned inside the circle defined earlier, are returned and represent the candidate points used in the spatial and temporal scoring processes.

3.1.2. Spatial analysis

For the spatial analysis of the candidate points, we define two different probability values which determine the final score, the *observation probability*, considering the candidate point's proximity (distance) to the GPS point, and the *transmission probability*, responsible for the topological aspects of the locating process.

The *observation probability* represents the likelihood of a candidate node being a correct match for the GPS observation, based solely on the distance between the two points, not considering the neighboring points - the candidate points corresponding to the previous GPS sample. This likelihood $N(c_i^j)$ can be described as a normal (Gaussian) distribution of the distance between the two points, as follows:

$$N(c_i^j) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x_i^j - \mu)^2}{2\sigma^2}} \quad (3.1)$$

where:

- c_i^j represents the candidate point j for the GPS observation p_i ,
- x_i^j the shortest (air-path) distance between the candidate point c_i^j and the GPS point p_i ,
- σ the standard deviation,
- σ^2 the corresponding variance and

- μ the mean (expectation).

As in the originating paper, a zero-mean normal distribution with a standard deviation of 20 meters (concluded empirically) is used. As mentioned, the observation probability does not consider the historical data (position of previously matched points) but only the distance between the current candidate and the GPS point, which sometimes can result in erroneous matching, as can be observed in Figure 3.3. Assuming the vehicle is moving on the road A, comes to the intersection, but does not turn and just proceeds moving on the same street, the p_1 and p_3 would match correctly. However, p_2 would not, since the observation probability does not account for the sudden switch of the candidate segments corresponding to the points p_1 and p_2 and only looks for the closest road node to project on, in this case, the c_2^1 .

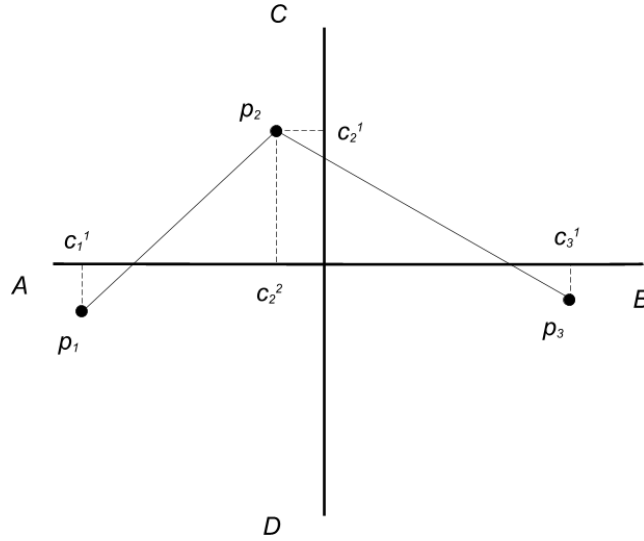


Figure 3.3.: GPS raw positions and the candidate nodes

To solve this problem, the *transmission probability* is introduced, which represents the relation between the distance measured between two successive GPS observations and the distance measured between the two corresponding candidate points. The transmission probability of two points a and b , basically represents the likelihood that the line segment which has a and b as the start and end nodes, equals the true path the vehicle accomplished. The distance of the two GPS samples $d_{p_1 \rightarrow p_2}$ would need to be compared with the distances between the respective candidate nodes $d_{c_1^1 \rightarrow c_2^1}$ and $d_{c_1^1 \rightarrow c_2^2}$. The second distance, concerning the correct node point c_2^2 , would naturally result with a better score. The *transmission probability* value $V(c_{i-1}^t \rightarrow c_i^s)$ can be computed as follows:

$$V(c_{i-1}^t \rightarrow c_i^s) = \frac{d_{p_{i-1} \rightarrow p_i}}{d_{c_{i-1}^t \rightarrow c_i^s}} \quad (3.2)$$

After the computation of the both probability values, a multiplication of the two would equal the final *spatial analysis function score* $F_s(c_{i-1}^t \rightarrow c_i^s)$, defined as follows:

$$F_s(c_{i-1}^t \rightarrow c_i^s) = N(c_i^s) \cdot V(c_{i-1}^t \rightarrow c_i^s) \quad (3.3)$$

This score basically represents the likelihood of the object moving from the position c_{i-1}^t to the position c_i^s and is computed for any pair of two neighboring candidate points (as can be seen in Figure 3.2). Since this distance represents the shortest (air) distance between the two candidate points and is unlikely to be the only distance the vehicle passed (after turns for instance), the observation probability cannot be ignored. Both values are needed.

3.1.3. Temporal analysis

A situation where temporal analysis would help in determining the correct link has already been showcased in the introduction of the algorithm (Figure 3.1). In such a case, relying exclusively on spatial analysis and not considering the speed limit constraints, would result in a wrong match.

To compare the speed information between the two paths, the true average speed of the vehicle has to be determined. Assuming that, when moving from the candidate point c_{i-1}^t to the position of the candidate point c_i^s , the road segments $(r_1, r_2 \dots r_n)$ in between have been passed and their corresponding lengths $(l_1, l_2 \dots l_n)$ are known (using the digital spatial map information), the average speed of the vehicle on the path $c_{i-1}^t \rightarrow c_i^s$ can be determined as follows:

$$v_{c_{i-1}^t \rightarrow c_i^s} = \frac{\sum_{i=1}^n l_i}{\Delta t_{c_{i-1}^t \rightarrow c_i^s}} \quad (3.4)$$

where $\Delta t_{c_{i-1}^t \rightarrow c_i^s}$ represents the time passed in between the two GPS observations p_i and p_{i+1} . Since every road segment $(r_1, r_2 \dots r_n)$ has also a speed limit value $(v_1, v_2 \dots v_n)$ associated to it (speed information extracted from digital maps), the *temporal analysis function* can be defined as follows:

$$F_t(c_{i-1}^t \rightarrow c_i^s) = \frac{\sum_{i=1}^n v_i \cdot v_{c_{i-1}^t \rightarrow c_i^s}}{\sqrt{\sum_{i=1}^n v_i^2} \cdot \sqrt{\sum_{i=1}^n v_{c_{i-1}^t \rightarrow c_i^s}^2}} \quad (3.5)$$

As in the spatial analysis, the *temporal analysis function* is computed for any pair of two neighboring candidate points.

3.1.4. Final result matching and localization

After both spatial and temporal analysis have been concluded, a candidate graph, consisting of sets of candidate points related to the GPS observations and the edges of the graph, connecting any two neighboring candidate points, can be constructed (Figure 3.4). All candidate points have an observation and every edge a transmission probability value associated with them. Additionally, for the edges connecting two candidate points a temporal analysis score value is established as well. The ST function of an edge $c_{i-1}^t \rightarrow c_i^s$ is defined as:

$$F(c_{i-1}^t \rightarrow c_i^s) = F_s(c_{i-1}^t \rightarrow c_i^s) \cdot F_t(c_{i-1}^t \rightarrow c_i^s) \quad (3.6)$$

with $2 \leq i \leq n$, since at least two GPS observations, i.e. the corresponding candidate points, are needed to build a graph edge.

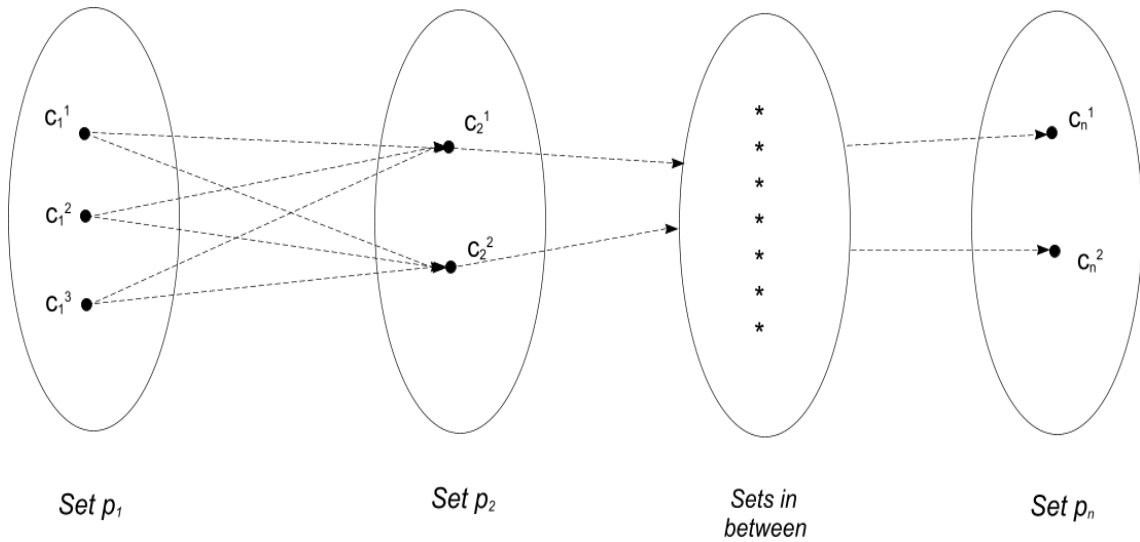


Figure 3.4.: A graph consisting of candidate points and the edges connecting the candidate points (adapted from Lou et al., 2009 [3])

The general ST-Matching was developed to be used for offline trajectory matching. First, a set of the GPS observations and their candidate points is identified. Then, the scoring process takes place and a candidate sequence path $P_c : c_1^{t_1} \rightarrow c_2^{t_2} \rightarrow \dots \rightarrow c_n^{t_n}$ with the highest sum $F(P_c) = \sum_{i=2}^n F(c_{i-1}^{t_{i-1}} \rightarrow c_i^{t_i})$ of individual ST function scores of the trajectories in question, is taken as the most probable road path for the trajectory built of GPS observations. However, for our purpose, where online processing in real-time is desired, the algorithm had to be modified to work as an incremental, localized, algorithm. This can be achieved by building "partial graphs", consisting of at least two sets (after the first two observations), and similar to above, determining

the best ST function score for the edges in that graph. When using the fixed window size of two sets, this localized ST-Matching algorithm would identify the most probable path the device moved on, during the time passed since the last GPS observation, and match the current GPS observation onto the most probable candidate position the vehicle moved to (highest overall score). To reduce the computational complexity of the algorithm when determining a match for current GPS observation p_i , the ST function score can be computed only for the edges coming from the candidate point identified as the correct match for the previous GPS observation p_{i-1} . With the fixed graph size of two sets, the algorithm can be used for incremental, online matching.

3.2. Implementation

3.2.1. Google Android and its Network Location Provider

Since the Android platform provides robust means of estimating the user's current location, using either the readings received by the GPS/A-GPS sensor (which most Android devices have built-in) or using Google's Network Location Provider which can return the most probable position relying solely on Wi-Fi and cell tower positioning techniques, this platform has been chosen for the implementation of the discussed ST-Matching algorithm. However, during our research, it has been decided to focus on the GPS (A-GPS) readings. Not only because of the better accuracy the GPS offers, but also because of the availability (available in virtually all open environments). The drawbacks of the GPS solution is its performance in tunnels and the shorter battery life span than when using the other methods of locating. However, these disadvantages have not been proven critical for our research. Since the ST-Matching can match using the Wi-Fi and Cell ID systems as well, with particularly Wi-Fi access points being very dense in the urban areas like cities, this particular algorithm has shown to be the most doable and efficient one for all purposes.

Although any of the three location providers can be used for location updates, in our application, for accuracy reasons, only GPS and Wi-Fi providers are listened for updates, with a minimum time interval in which the position observations may arrive, defined. After the application is provided a *LocationManager*, whose instance is requested by calling *getSystemService(Context.LOCATION_SERVICE)* and which can register for periodic location updates using the defined location provider(s), the objects of the class *android.location.Location* are returned. They provide the application with the current positional observation and the corresponding values including coordinates (longitude and latitude) and, when available, the speed, altitude, bearing and accuracy information. The accuracy parameter is especially important for the range query since it defines the radius around the GPS observation in which the candidate line points should be identified. When requesting location updates, additionally, *android.location.Criteria* parameters

can be taken as an argument, which would automatically determine the best location provider (choosing between the GPS, WPS and Cell ID readings) based on the parameters defined (cost/no cost, finer accuracy requirements, the needed location properties etc.).

3.2.2. The application workflow and introduced optimizations

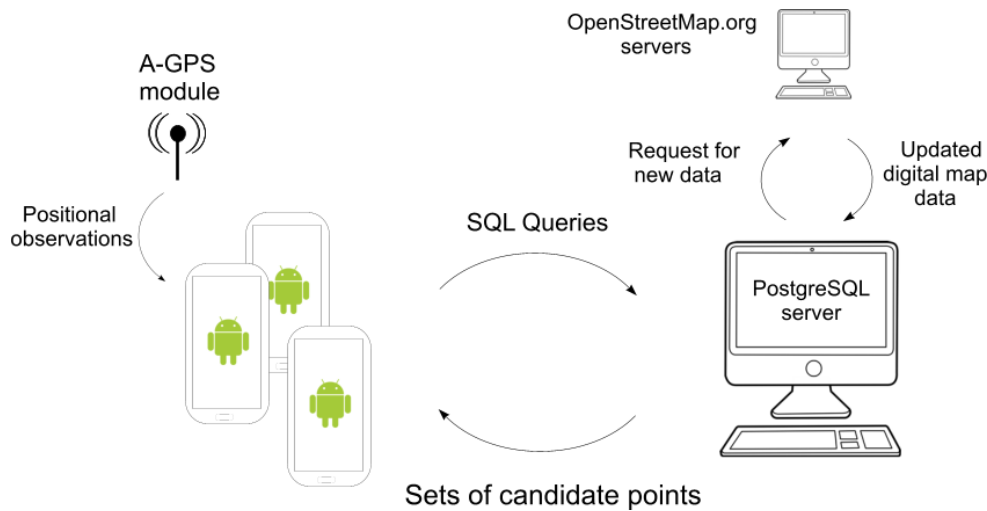


Figure 3.5.: An overview of the system setup

In addition to the device, a server running a relational database containing the digital map information is needed. An overview of the system is depicted in Figure 3.5. The spatial map data (using the OpenStreetMap data sets), consisting of nodes, waylines and the corresponding properties (positional, speed data, type of the streets and additional useful information), have been imported in a PostgreSQL database deployed on a remote Linux server using the *osm2psql* utility which allows for importing of the road data (provided as XML files on openstreetmap.org). This data is remotely fetched using the SQL query functionality, but also extended to the PostGIS commands¹ which further support the relations between the geographical objects in the database. For the purpose of communication with the server, the Java Database Connectivity (JDBC)² plugin for relational databases is employed in the Android application.

After the arrival of GPS/WPS/Cell ID observations, the communication with a remote server the spatial data is deployed on, occurs, and the candidate line segments information, together with their shape points (street points), are returned. The interpolation, where using the start and end nodes of the partial line segments to construct additional street points, takes place on the remote server as well. The points found in the determined error radius around the raw observation are then evaluated locally, on the device. These candidate points are saved as the objects of class

¹<http://postgis.org/documentation/>

²<http://docs.oracle.com/javase/6/docs/technotes/guides/jdbc/>

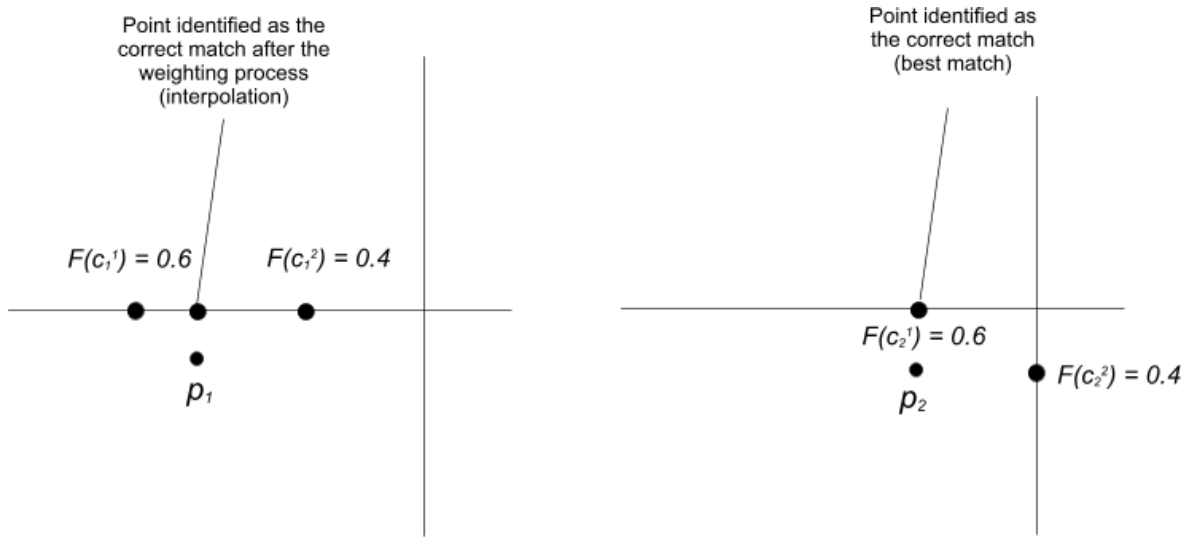
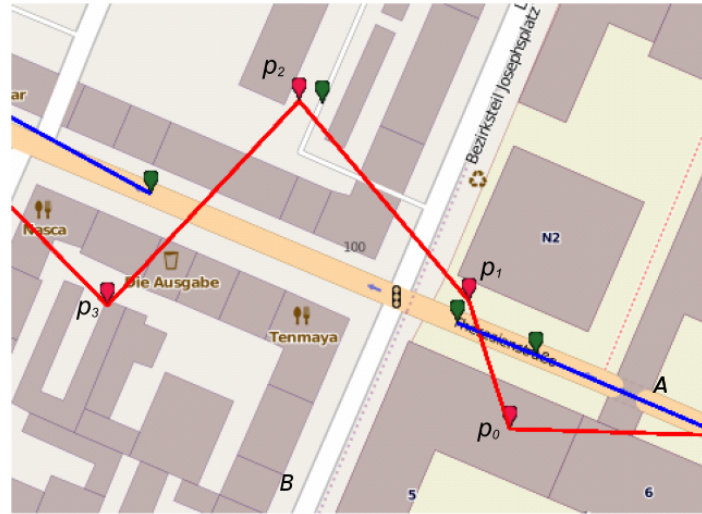


Figure 3.6.: Matching on the road segment or candidate node position

CandidateNode and assigned the observation probability. After the second GPS observation, the transmission probability for the new candidates, in regards to the previous points are computed and the spatial and temporal analysis function values for the paths determined. Before identifying the candidate point with the highest ST function score as the correct match for the current observation, the weighting on the road line connecting the two points with the highest scores, if on the same street segment, results with a third interpolated point. This point is then taken as the final position the observation should be matched on. In a situation where the two points with highest score are positioned on different segments, the final match for the current GPS observation is the highest scoring candidate point (both scenarios are depicted in Figure 3.6). In order to reduce the computing time for the succeeding candidate points, the connectivity to the last correct match only, is considered.

Important modifications to the general ST-Matching algorithm, introduced in this application, are the already discussed weighting on the road segments, *outlier identification* process and *merging at intersections*. With the "outliers", the spikes in the raw observations are meant, resulting in readings which completely differ from the previous heading (changes of $\Delta\beta = 45^\circ$ between the consecutive headings) and cannot be logically incorporated into the true path of the vehicle. To identify such points, sudden changes in the headings of succeeding samples are evaluated. As an example, a vehicle is moving on a road *A* as depicted in Figure 3.7. The red points represent the GPS observations, and green points the corresponding matched points on road segments. The red lines represent the path constructed from the GPS readings, whereas the blue ones build the "true" path (a matched sequence). After correctly matching the positional observations p_0 and p_1 , a sudden spike p_2 is returned. This outlier can be identified only after another, third observation took place. Sometimes such sudden changes in headings might actually occur (at U-turns for

Figure 3.7.: Outlier p_2 identified

instance) and before simply excluding the matched point from the "true" path, an observation of the next few points to check for possible connection is proposed.

It has been concluded by multiple sources [13, 17] that in cases where the speed of the vehicle gets very low (stops at intersections, traffic jams), the readings by the GPS sensors may result in very inaccurate positions and headings perceived. The solution proposed and incorporated in this application is a single matching (matching only one from many GPS observations), whenever the vehicle's speed gets close to zero (distance between two raw observations very small), which is very often the case at intersections.

Chapter 4.

Evaluation

To test the implementation for correctness, two different routes were taken into account. The routes were created using the GPSies.com track creator utility¹ which allows for manual input of GPS observations and the average speed the object is moving with. They resemble the real-world paths a vehicle would move on and consider various error sources in the positioning of the GPS ticks. These, typical for dense urban areas, further result in multiple incorrect GPS ticks at intersections, "spikes" in consecutive readings, and, for instance, some extreme cases with more than 150 m of error distance between the expected "true" point and the corresponding GPS tick.

4.1. Test results

Most of these erroneous readings, as can be seen in Figure 4.1, can be corrected using the modified ST-Matching implementation. The *MapView* shown in the main window is rendered using the open-source MapsForge API² and the Mapnik toolkit³. The open-source Mapnik toolkit allows for rendering the map tiles (size of 256x256 pixels) downloaded from the tile server (tile.openstreetmap.org). The red points correspond to the GPS observations and the green ones to the matched positions. Two trajectories, using these points as shape points, can be built. The red trajectory (starting at point *A*, ending at point *B*) takes the GPS ticks in account, whereas the blue one, created using the matched locations as shape points, represents the "true" path the vehicle is assumed to have passed. The difference in correctness between the two paths is emphasized in Figure 4.2, where the advantages of using the map-matching become obvious. To construct the path, 50 consecutive GPS ticks have been matched on the road, incrementally in real-time. Although the algorithm matched more than 95% of the observations correctly, one obvious flaw can be observed at point *E* in Figure 4.1 where an incorrect matching at an intersection had occurred. Since no changes in bearing between the previous and the matched point

¹<http://www.gpsies.com/createTrack.do>

²<http://code.google.com/p/mapsforge/wiki/OverlayAPI>

³<http://wiki.openstreetmap.org/wiki/Mapnik>

were observed and no heading data (if and how vehicle turned at the intersection) was used as the input, the spatial analysis returned the highest spatial score for the incorrectly matched point (closest to the GPS tick) failing to identify the erroneous match during the matching process. To detect and avoid such mistakes, special importance should be given to matching at intersections. The intersections should be detected (taking the available speed measurements into account), and by using dead reckoning techniques (e.g. by utilizing the built-in compass, where available) the changes in heading identified. Merging at intersections is utilized in our route (for unifying multiple "close" GPS observations into a single point), resulting in a lower number of matched locations seen in the image.

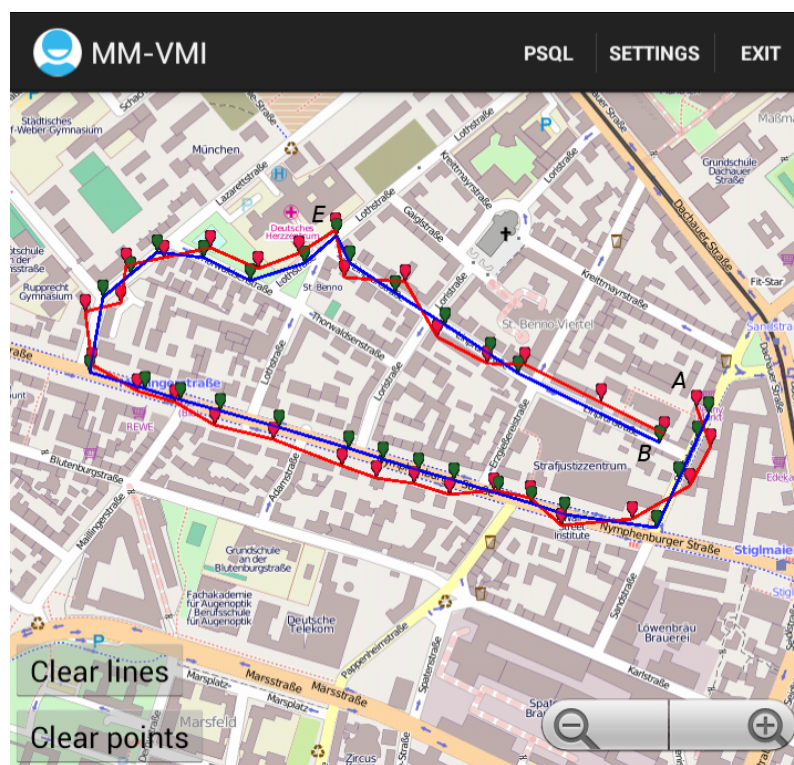


Figure 4.1.: The first test route where *A* and *B* represent the start and end nodes respectively and point *E* a false match after passing the nearby intersection

A second test route, built from 58 consecutive positional observations (red dots) and the corresponding matches (green dots) can be observed in Figure 4.3., where, as with the previous sample route, erroneous readings do happen at intersections (*E1* and *E2*). There, the spatial matching function favors the closest matches, failing to identify the correct roads the vehicle continued moving on after passing the intersections. As proposed, the built-in compass could be used to determine the headings in such scenarios. Still, the algorithm performs very well in most cases. It is able to identify the possible spikes (*E3*) using the outlier identifier method and discard such a match before including it into the "true" path of the vehicle.

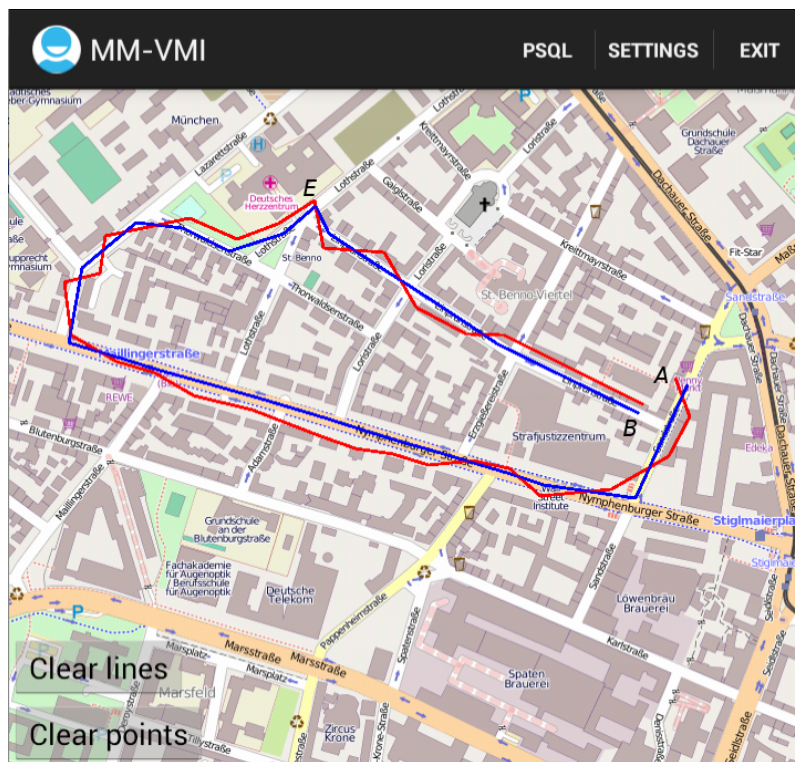


Figure 4.2.: The red path is constructed out of GPS observations. The blue path represents the "true" path constructed out of matched points.

4.2. Possible improvements

In addition to the proposed implementation of dead reckoning methods into the existing algorithm, some additional tests and improvements to the outlier identifier function are needed. Specifically, the possibility of multiple consecutive outliers ("spikes") happening, against which the algorithm was not tested, should be considered. Furthermore, currently, the device communicates with a server on which a PostgreSQL relational database is installed (and which contains all the road data - updated from the OpenStreetMaps XML data sets). Although the PostgreSQL, by itself, has geometry types, additional functions to process the spatial data are needed. This database's support for geographical objects is extended by using the open-source PostGIS functions. Although no considerable delays in sending and receiving the responses to the SQL queries were noticed, for the app to function properly, a constant Internet connection is needed. It has been discussed that a possible local implementation of the database information should be considered. This would ask for importing the spatial data into a local SQLite relational database (on the device). Since the SQLite⁴ is the only relational database the Android OS provides, rewriting the JDBC and PostGIS parts of the code to use the Spatialite libraries and, possibly, porting some of the functions might

⁴<http://www.sqlite.org/>

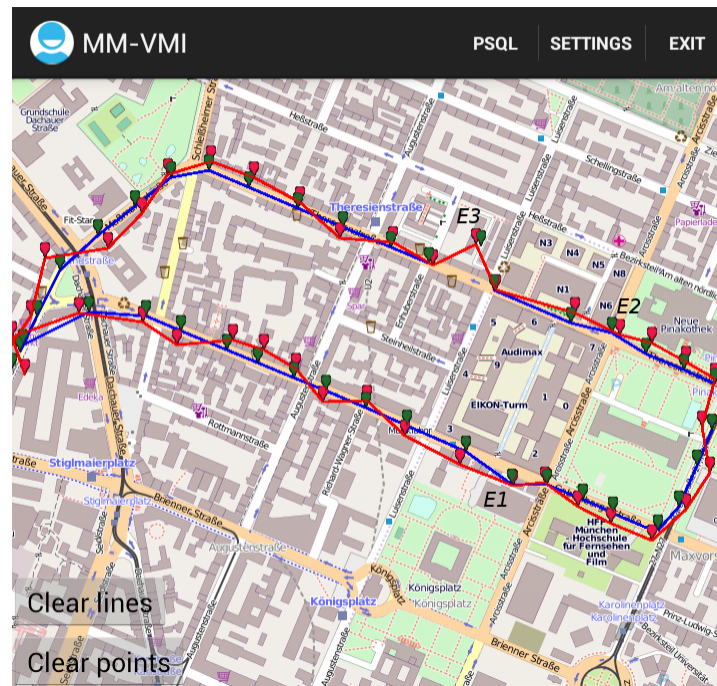


Figure 4.3.: The second test route where $E1$ and $E2$ represent false matches at intersections

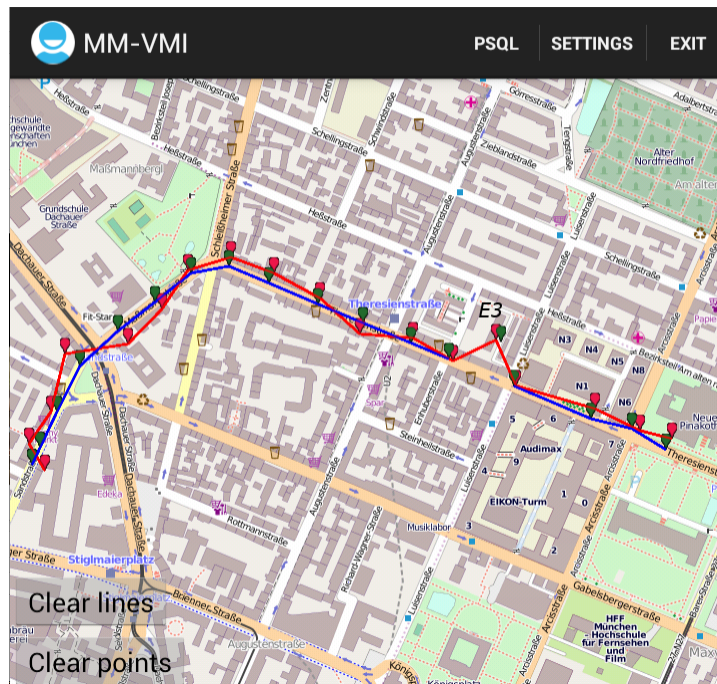


Figure 4.4.: Successful identification of an outlier at point $E3$

be needed. SpatialLite⁵ is an open-source library which extends the SQLite database by providing

⁵<http://code.google.com/p/spatialite-android/>

the vector geodatabase functionality. Since the map data is fairly big in size (402 MB compressed file, for the Bavaria region), only parts of the complete country database sets would be downloaded and imported into the database, depending on the user's current location (or choice of the data set).

Chapter 5.

Conclusion

In this thesis, an overview of modern map-matching algorithms, used to reconcile a positional observation onto the road using geometric, topological, statistical and advanced techniques in real-time, has been given. The usual methods of positioning, as well as the sources of errors associated to these methods have been discussed and the motivation for the process of matching presented. Furthermore, a scoring-based map-matching algorithm, called ST-Matching, which takes into account spatial, topological and temporal information of the setting, has been implemented for the mobile platform Android. The original workflow of the algorithm was additionally optimized and offers a robust detection of input "spikes" and erroneous observations when confronted with lower moving speed of the vehicle. For the use case on mobile phones, in addition to the GPS, it has been extended to also use Wi-Fi and Cell ID signals for positioning purposes. The implementation of this incremental algorithm has been tested and evaluated against multiple road tracks, in a dense urban area, and has resulted in a high percentage of correct matches. Although the algorithm matched the majority of observations correctly, in order to estimate the probability of correct matches, a performance comparison against other up-to-date map-matching algorithms is proposed. Furthermore, additional attention should be given to matching at intersections and an execution of multiple real-world tests considered in order to make the algorithm fully usable in practice.

Bibliography

- [1] R. W. Sturdevant, *Societal Impact of Spaceflight*. National Aeronautics and Space Administration, 2009.
- [2] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, p. 312–328, 2007.
- [3] Y. Lou, C. Zhang, X. Xie, Y. Zheng, W. Wang, and Y. Huang, "Map-Matching for Low-Sampling-Rate GPS Trajectories," *GIS '09 Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 352–361, 2009.
- [4] C. Fernandez-Prades, L. Presti, and E. Falletti, "Satellite Radiolocalization From GPS to GNSS and Beyond: Novel Technologies and Applications for Civil Mass Market," *Proceedings of the IEEE*, vol. 99 (11), pp. 1882–1904, 2011.
- [5] G. M. Djuknic and R. E. Richton, "Geolocation and Assisted-GPS," *Computer*, vol. 34, pp. 123–125, 2001.
- [6] J. A. Klobuchar, "Ionospheric effects on GPS," *Global Positioning System: Theory and applications. Vol. 1 (A96-20837 04-17)*, pp. 485–515, 1996.
- [7] S. A. Golden and S. S. Bateman, "Sensor Measurements for Wi-Fi Location with Emphasis on Time-of-Arrival Ranging," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 1185–1198, 2007.
- [8] P. A. Zandbergen, "Accuracy of iPhone Locations: A Comparision of Assisted GPS, WiFi and Cellular Positioning," *Transactions in GIS*, vol. 13, pp. 5–25, 2009.
- [9] E. Mok and G. Retscher, "Location Determination Using WiFi - Fingerprinting Versus WiFi - Trilateration," *Journal of Location Based Services*, vol. 1, pp. 145–159, 2007.
- [10] M. Wallbaum, "A priori error estimates for wireless local area network positioning systems," *Pervasive and Mobile Computing*, vol. 3, pp. 560–580, 2007.
- [11] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, "Accuracy Characterization for Metropolitan-scale Wi-Fi Localization," *MobiSys '05 - Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pp. 233–245, 2005.

- [12] N. R. Velaga, M. A. Quddus, and A. L. Bristow, "Developing an Enhanced Weight-Based Topological Map-Matching Algorithm for Intelligent Transport Systems," *Transportation Research Part C: Emerging Technologies*, vol. 17 (6), pp. 672–683, 2009.
- [13] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, pp. 91–108, 2000.
- [14] J. Kim, "Node Based Map-Matching Algorithm for car navigation system," *Proceedings of the International Symposium on Automotive Technology & Automation*, pp. 121–126, 1996.
- [15] D. Bernstein and A. Kornhauser, "An Introduction to Map Matching for Personal Navigation Assistants," *Technical report, New Jersey TIDE Center*, 1996.
- [16] J. S. Greenfeld, "Matching GPS Observations to Locations on a Digital Map," *Proceedings of the 81st Annual Meeting of the Transportation Research Board*, 2002.
- [17] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland, "A general map matching algorithm for transport telematics applications," *GPS Solutions*, vol. 7, pp. 157–167, 2003.
- [18] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An Interactive-Voting Based Map Matching Algorithm," *MDM '10 - Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, pp. 43–52, 2010.
- [19] L. Zhao, W. Y. Ochieng, M. A. Quddus, and R. B. Noland, "An Extended Kalman Filter Algorithm for Integrating GPS and Low Cost Dead Reckoning System Data for Vehicle Performance and Emissions Monitoring," *The Journal of Navigation*, vol. 56, pp. 257–275, 2003.
- [20] M. A. Quddus, R. B. Noland, and W. Y. Ochieng, "A high accuracy fuzzy logic based map matching algorithm for road transport," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, pp. 103–115, 2006.
- [21] W. Ochieng, M. Quddus, and R. Noland, "Map-Matching in complex urban road networks," *Brazilian Journal of Cartography (Revista Brasileira de Cartografia)*, vol. 55, pp. 1–18, 2003.
- [22] Y. Zhao, *Vehicle Location and Navigation Systems*. Artech House Inc, 1997.
- [23] M. Fu, J. Li, and M. Wang, "A hybrid map matching algorithm based on fuzzy comprehensive Judgment," *Proceedings of the 7th IEEE Conference on Intelligent Transportation Systems*, pp. 613–617, 2004.
- [24] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, and R. Karlsson, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 425–437, 2002.