Advanced Map Matching Technologies and Techniques for Pedestrian/Wheelchair Navigation

by

Ming Ren

B.E., Shenyang University of Technology, 1993

M.E., Chinese Institute of Academy, 1998

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2012

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Ming Ren

It was defended on

March. 16, 2012

and approved by

Dr. Daqing He, Associate Professor, School of Information Sciences

Dr. Paul Munro, Associate Professor, School of Information Sciences

Dr. Prashant Krishnamurthy, Associate Professor, School of Information Sciences

Dr. Linda Van Roosmalen, Adjunct Professor, School of Health and Rehabilitation Sciences

Dissertation Advisor: Dr. Hassan Karimi, Associate Professor, School of Information Sciences

Advanced Map Matching Technologies and Techniques for Pedestrian/Wheelchair Navigation

Ming Ren, PhD

University of Pittsburgh, 2012

Due to the constantly increasing technical advantages of mobile devices (such as smartphones), pedestrian/wheelchair navigation recently has achieved a high level of interest as one of smartphones' potential mobile applications. While vehicle navigation systems have already reached a certain level of maturity, pedestrian/wheelchair navigation services are still in their infancy. By comparing vehicle navigation systems, a set of map matching requirements and challenges unique in pedestrian/wheelchair navigation is identified. To provide navigation assistance to pedestrians and wheelchair users, there is a need for the design and development of new map matching techniques.

The main goal of this research is to investigate and develop advanced map matching technologies and techniques particular for pedestrian/wheelchair navigation services. As the first step in map matching, an adaptive candidate segment selection algorithm is developed to efficiently find candidate segments. Furthermore, to narrow down the search for the correct segment, advanced mathematical models are applied. GPS-based chain-code map matching, Hidden Markov Model (HMM) map matching, and fuzzy-logic map matching algorithms are developed to estimate real-time location of users in pedestrian/wheelchair navigation systems/services. Nevertheless, GPS signal is not always available in areas with high-rise

buildings and even when there is a signal, the accuracy may not be high enough for localization of pedestrians and wheelchair users on sidewalks. To overcome these shortcomings of GPS, multi-sensor integrated map matching algorithms are investigated and developed in this research. These algorithms include a movement pattern recognition algorithm, using accelerometer and compass data, and a vision-based positioning algorithm to fill in signal gaps in GPS positioning.

Experiments are conducted to evaluate the developed algorithms using real field test data (GPS coordinates and other sensors data). The experimental results show that the developed algorithms and the integrated sensors, i.e., a monocular visual odometry, a GPS, an accelerometer, and a compass, can provide high-quality and uninterrupted localization services in pedestrian/wheelchair navigation systems/services. The map matching techniques developed in this work can be applied to various pedestrian/wheelchair navigation applications, such as tracking senior citizens and children, or tourist service systems, and can be further utilized in building walking robots and automatic wheelchair navigation systems.

TABLE OF CONTENTS

TABLE OF CONTENTS VI

LIST OF TABLES X

LIST OF FIGURES XII

PREFACE XVII

1.0 INTRODUCTION 18

1.1 PROBLEM STATEMENT 20

1.2 OVERVIEW OF MAP MATCHING IN PEDESTRIAN/WHEELCHAIR NAVIGATION SYSTEMS/SERVICES 23

1.3 MAP MATCHING CHALLENGES FOR PEDESTRIAN/WHEELCHAIR NAVIGATION 25

1.4 GOAL AND OBJECTIVES 29

1.5 CONTRIBUTIONS 30

1.6 ORGANIZATION 31

2.0 BACKGROUND AND RELATED WORK 32

2.1 SIDEWALK NETWORKS 33

2.2 GEO-POSITIONING TECHNOLOGIES AND TECHNIQUES 34

2.3 MAP MATCHING TECHNIQUES IN NAVIGATION SYSTEMS 39

2.4 COMPUTER VISION IN NAVIGATION SYSTEMS 43

2.5 MOBILE TECHNOLOGY IN NAVIGATION APPLICATIONS 48
3.0 ADAPTIVE CANDIDATE SEGMENTS SELECTION ALGORITHM 49
3.1 RELATED WORKS 50
3.2 SPATIAL NETWORK DATA REPRESENTATION 52
3.2.1 Hierarchical Clustering Tree 52
3.2.2 Clustering Road Segments 53
3.3 ADAPTIVE SEARCHING ALGORITHM 56
3.3.1 A Binary Tree Structure from the Clustering Tree 56
3.3.2 Searching Algorithm 58
3.3.3 Adaptive Search Window Set 59
3.3.4 Adaptive Search Window Update 60
3.4 PERFORMANCE ANALYSIS 61
3.4.1 Datasets 61
3.4.2 Construction Cost 61
3.4.3 Searching Cost 64
3.5 SUMMARY 67
4.0 GPS-BASED MAP MATCHING TECHNIQUES 68
4.1 CHAIN-CODE-BASED MAP MATCHING 69
4.1.1 Eight-Direction Chain Code 70
4.1.2 Chain-Code-based Map Matching Technique 73
4.1.3 Map Matching Process 78
4.1.4 Validation 81
4.2 HMM-BASED MAP MATCHING ALGORITHM 89

4.2.1 Hidden Markov Model 90

4.2.2 A Hidden Markov Model for Map Matching 92

4.2.3 HMM-based Map Matching Process.....98

4.2.4 VALIDATION 100

4.3 FUZZY-LOGIC-BASED MAP MATCHING ALGORITHM.....105

4.3.1 Fuzzy Logic Map Matching.....105

4.3.2 Fuzzy Logic Map Matching Process 111

4.3.3 Validation.....115

4.4 COMPARISON.....119

5.0 MULTI-SENSOR INTEGRATED MAP MATCHING ALGORITHMS 124

5.1 CLIENT/SERVER ARCHITECTURES FOR MAP MATCHING.....125

5.1.1 Lightweight Client/Heavyweight Server Architecture 125

5.1.2 Heavyweight Client/Lightweight Server Architecture 126

5.2 MOVEMENT PATTERN RECOGNITION ASSISTED MAP MATCHING FOR PEDESTRIAN/WHEELCHAIR NAVIGATION.....128

5.2.1 Movement Pattern Recognition.....131

5.2.2 Movement Pattern Recognition Assisted Map Matching.....136

5.2.3 Experiments 142

5.3 MULTI-SENSOR MAP MATCHING USING MONOCULAR VISUAL ODOMETRY TECHNIQUE FOR PEDESTRIAN/WHHELCHAIR NAVIGATION ……152

5.3.1 Multi-Sensor Map Matching Algorithm Using Monocular Visual Odometry 154

5.3.2 Accelerometer-Assisted Monocular Visual Odometry for Motion Estimation 156

5.3.3 Integrated Map Matching 170

5.3.4 Experiments and Analysis 172

6.0 SUMMARY, CONCLUSIONS, CONTRIBUTIONS AND FUTURE RESEARCH 186

6.1 SUMMARY 186

6.2 CONCLUSIONS 187

6.3 CONTRIBUTION 189

6.4 FUTURE RESEARCH 191

REFERENCES 193

LIST OF TABLES

Table 2.1 Outdoor geo-positioning technologies potential for integration with GPS 35

Table 2.2. Comparison among performances of various map matching algorithms for vehicle navigation (after table in Quddus, 2007).....42

Table 3.1. Tree features of three road networks 63

Table 3.2. Statistics of the searching cost 66

Table 4.1. RBF neural network structure 84

Table 4.2. Map matching evaluation using RBF neural network 84

Table 4.3. Linear-model map matching results 85

Table 4.4. Comparing the linear model and the non-linear model 89

Table 4.5. Performance results 104

Table 4.6. Parameters of the fuzzy logic map matching 107

Table 4.7. Rules of the fuzzy logic map matching 108

Table 4.8. Performances of Experiments 119

Table 4.9. Accuracy of GPS-based map matching algorithms for pedestrian/wheelchair navigation 122

Table 4.10. Time performance of GPS-based map matching algorithms for pedestrian/wheelchair navigation 122

Table 4.11. Overall comparison of three GPS-based map matching algorithms.....122

Table 5.1. Selected features.....135

Table 5.2. Classifier accuracy in identifying four different movement behaviors 145

Table 5.3. Confusion matrix of cross-validation on feature classification of movement behavior 146

Table 5.4. Map matching performance (efficiency and accuracy) 150

Table 5.5. Key points and matched points 176

LIST OF FIGURES

Figure 1.1. Components of a navigation system.....19

Figure 1.2. Map matching process in pedestrian/wheelchair navigation systems.....24

Figure 1.3. Sidewalk network (a) versus road network (b) in the same area.....26

Figure 1.4. No path in map database.....27

Figure 1.5. Poor GPS signals (PDOP between 2.5 and 11.4).....28

Figure 1.6. Comparison of GPS data from a professional GPS receiver (green) and a smartphone (red).....29

Figure 2.1. Example image with landmarks (Steinhoff et.al, 2007).....44

Figure 3.1. An example of road network.....54

Figure 3.2. Corresponding matrix (20-by-20).....55

Figure 3.3. Corresponding clustering tree.....55

Figure 3.4. Data structure of a binary tree for segment clustering.....57

Figure 3.5. A GPS point is located within the range of a Bounding Box.....57

Figure 3.6. The clustering tree of Pittsburgh campus.....63

Figure 3.7. Query results changing with scenarios of moving object's positions.....65

Figure 3.8. A scenario on a large-scale map.....66

Figure 4.1. Perpendicular distance.....68

Figure 4.2. 8-Direction chain code.....70

Figure 4.3. Digital map with GPS data 71
Figure 4.4. Example of chain-code-based map matching 72
Figure 4.5. Linear model 74
Figure 4.6. Non-linear model 75
Figure 4.7. RBF neural network for map matching evaluation 77
Figure 4.8. Flowchart of chain-code-based map-matching algorithm 80
Figure 4.9. University of Pittsburgh's campus 82
Figure 4.10. Training with RBF neural network 83
Figure 4.11. Route 1 comparing map-matching result with GPS raw data on campus sidewalk map 86
Figure 4.12. Route 2 comparing map-matching result with GPS raw data on campus sidewalk map 87
Figure 4.13. Route 3 comparing map-matching result with GPS raw data on campus sidewalk map 88
Figure 4.14. Architecture of a HMM 91
Figure 4.15. The hidden Markov model for map matching 93
Figure 4.16. An example of GPS points overlaid on sidewalks on campus 97
Figure 4.17. An abstracted sidewalk network model 97
Figure 4.18. State transition matrix 97
Figure 4.19. Map matching locations versus GPS positions 98
Figure 4.20. Flowchart of HMM-based map matching process 99
Figure 4.21. Route 1 comparing map-matching result with GPS raw data on campus sidewalk map 101

Figure 4.22. Route 2 comparing map-matching result with GPS raw data on campus sidewalk map 102

Figure 4.23. Route 3 comparing map-matching result with GPS raw data on campus sidewalk map 103

Figure 4.24. Two inputs and the output in the fuzzy logic map matching 107

Figure 4.25. Examples of entering mode 113

Figure 4.26. Flowchart of the fuzzy logic map matching process 115

Figure 4.27. Route 1 comparing map-matching result with GPS raw data on campus sidewalk map 116

Figure 4.28. Route 2 comparing map-matching result with GPS raw data on campus sidewalk map 117

Figure 4.29. Route 3 comparing map-matching result with GPS raw data on campus sidewalk map 118

Figure 4.30. Comparison among the three GPS-based map matching algorithms on one route. 121

Figure 5.1. Lightweight client/ heavyweight server architecture for map matching 126

Figure 5.2. Heavyweight client/ lightweight server framework for map matching 127

Figure 5.3. An example of GPS error in the scenario in which a user is stopped on a sidewalk. 129

Figure 5.4. Overview of movement pattern recognition 132

Figure 5.5. 3D accelerometer 133

Figure 5.6. Movement recognition decision tree 136

Figure 5.7. Multi-sensor data integrated map matching 137

Figure 5.8. Accelerometer Data (acceleration in m^2/s) Figure 5.9. Orientation Data (angle in degree) 138
Figure 5.10. Timing diagram for synchronization 139
Figure 5.11. Flowchart of the movement pattern-recognition-assisted map matching algorithm 141
Figure 5.12. Motorola Backflip smartphone and the direction of its 3D accelerometer 143
Figure 5.13. A sample of a log file recording GPS, accelerometer, and orientation data 143
Figure 5.14. Route 1 comparing map matching result with GPS raw data 148
Figure 5.15. Route 2 comparing map matching result with GPS raw data 148
Figure 5.16. Route 3 comparing map matching result with GPS raw data 149
Figure 5.17. Flowchart of multi-sensor map matching algorithm using monocular visual odometry 155
Figure 5.18. Overview of visual odometry process 156
Figure 5.19. Estimation of rotation matrix $R_{i-1,i}$ and translational vector $T_{i-1,i}$ in the motion between video frame Fr_{i-1} and Fr_i 159
Figure 5.20. Frame-to-frame motion estimation 166
Figure 5.21. Flowchart of vision-based positioning algorithm 168
Figure 5.22. Scale adjustment 169
Figure 5.23. Flowchart of map matching approach 171
Figure 5.24. A checkerboard to calibrate camera 173
Figure 5.25. A sequence of images extracted from a video 174
Figure 5.26. SIFT features of a street view image 175
Figure 5.27. Matched SIFT feature points 176

Figure 5.28. Feature points in image 1 vs. Epipolar lines in image 2 177
Figure 5.29. Geo-positioning results by using visual odometry, top view (left) and street view (right) 178
Figure 5.30. Map matching results overlaid on Google Maps 180
Figure 5.31. A sample log file to compare accuracy of GPS positions 181
Figure 5.32. Comparison of GPS-based map matching results with multi-sensor map matching results 184

PREFACE

I would like to express my deep gratitude to my advisor, Dr. Hassan Karimi, for all his support during my study to make my research meaningful. I greatly appreciate all the advice and suggestions from my dissertation committee members. I also would like to thank the faculty in the School of Information Sciences for their teaching and support.

I am happy to make friends during my study and thank my former and current colleagues at the Geoinformatics Laboratory for their help in my research. I specially thank Jon Walker for his help in my study.

I dedicate this dissertation to my parents to make them proud.

1.0 INTRODUCTION

Over the past two decades, due to the improved accuracy and increased affordability of the Global Positioning System (GPS), vehicle navigation systems have experienced a tremendous increase in demand. Vehicle navigation systems assist in keeping track of vehicles and finding desired paths to destinations. Motivated by the success of vehicle navigation systems and the ubiquity of GPS, building navigation systems and services for pedestrians and wheelchair users is the focus of this dissertation.

A navigation system consists of a map database, a user interface and several navigation functions including geo-positioning, map matching, routing and guidance. These components are briefly described below.

- Map Database. A map database contains the geometry, topology, and attributes of a map network (e.g., a road network or a sidewalk network).
- Geo-positioning. Geo-positioning is the process of measuring location updates of an object in real time. The measurement of positions can be through GPS or other sensors, such as Dead Reckoning (DR), which are subject to noise, interference from the environment, and other of errors.
- Map Matching. Map matching is the process of determining the current vehicle's location on the road segment, using the geographic coordinates obtained by the geo-positioning component and the map database component.

- Routing and guidance. Routing computes user-preferred routes and guidance gives step-by-step instructions on how to travel on routes.
- User interface. A user interface accepts the user's requests for directions. It responds with map-matched positions on the map and audible or displayed directions at the right time.

Figure 1.1 shows the interrelations among the components of a navigation system. Real-time position data are obtained through geo-positioning sensors, such as a GPS receiver. To accurately locate a vehicle, map matching determines the road segment on which the vehicle is and estimates a position on the segment. The user interface shows matched positions in real time. The routing component provides users with their desired routes to their destinations, taking user preferences into account.

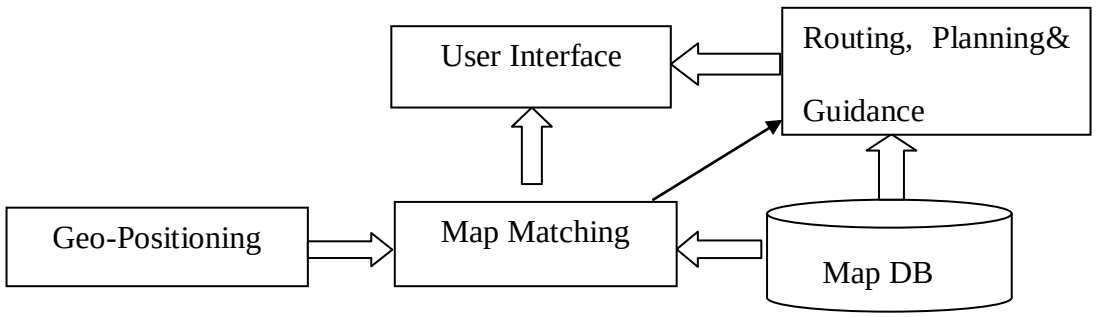


Figure 1.1. Components of a navigation system

Although vehicle navigation systems are currently the most popular, pedestrian/wheelchair navigation systems/services are gradually growing in prominence. With the progress of mobile technology, indoor/outdoor mobile systems (e.g., Nokia) are

being developed to provide location-based services, such as tour guidance for pedestrians (Fritz et al., 2006; Steinhoff et al., 2007). For wheelchair users, research has focused on developing "smart" wheelchairs that can provide navigation assistance mainly in indoor environments (Levine et al. 1999; Simpson et al. 2004). Outdoor wheelchair navigation systems/services are emerging as applications in their own right and are more challenging to develop than vehicle navigation systems are (Ding et. al. 2007). Inspired by the increasing demand for pedestrian/wheelchair navigation systems/services, this dissertation focuses on investigating and developing advanced map matching technologies and techniques to assist pedestrians and wheelchair users with mobility while outdoors.

1.1 PROBLEM STATEMENT

Vehicle navigation systems have been widely researched and developed over the past decades. However, with the rapid growth of the mobile device market, there is an increasing demand for pedestrian navigation services as mobile applications on mobile devices. As the populations of both disabled and senior citizens increase, there is a corresponding need to meet the travel demands of these groups so that they can maintain their quality of life. As a result, improvements in wheelchair navigation are required in order to provide wheelchair users with appropriate travel routes that can safely, accurately, and efficiently guide them to their destination. The backbone of these types of pedestrian/wheelchair services is a spatial database that represents the underlying

sidewalk network, which is used for tracking, routing, and guiding users to their destination.

Most of the commercially available pedestrian/wheelchair navigation systems/services were originally designed as vehicle navigation systems, and are now sold as with only minor modifications, such as the "Walking Navigation" feature on Google Maps for Mobile on Android, or extended pedestrian navigation functionality on some Nokia smartphones. These applications use road networks, like vehicle navigation systems, to provide navigation assistance to pedestrians and wheelchair users. However, road networks cannot provide pedestrians and wheelchair users with optimal services for several reasons. First, road networks typically include paths that are accessible to vehicles, but pedestrians and wheelchair users usually move or operate on sidewalks and do not travel on roads. Second, road networks do not contain information about footpaths, accessible areas for pedestrians, or connecting links to indoor environments. Third, map matching and routing on road networks do not provide the localization and routes that are of use for pedestrians and wheelchair users.

To provide appropriate navigation assistance to pedestrians/wheelchair users, pedestrian/wheelchair navigation services must support sidewalk networks (Kasemsuppakorn and Karimi, 2008). An analysis of the differences between pedestrian/wheelchair navigation and vehicle navigation reveals that map matching in pedestrian/wheelchair navigation services has unique requirements unfound in vehicle navigation systems.

First, while vehicle navigation systems track cars on roads, requiring road networks in the map database component, pedestrians/wheelchair navigation

systems/services track movement of users on sidewalks, requiring sidewalk networks. In general, sidewalk networks are much denser than road networks are in the same area, compounding the challenge of finding the correct sidewalk segment in the process of map matching.

Second, different from car driving, pedestrians walk or wheelchair users operate at lower speeds and their outdoor activities are usually closer to buildings. Under such circumstances, the Global Navigation Satellite System (GNSS), e.g., GPS, provides less accurate and reliable positioning information due to noise and multipath problem, compared to it received in vehicles. Further, in narrow streets (those with widths less than 10 m), GPS receivers (those available on mobile devices) have difficulty in determining the side of the street on which the user is travelling. Therefore, map matching has to deal with the positioning problem raised by GPS.

Third, pedestrians or wheelchairs are free to advance, stop and make turns at will on sidewalks. Since they have more flexibility than vehicles driven under traffic rules, people can always travel in both directions on sidewalks and in open areas, such as squares and parks. Rules for cars' travelling on road that have been utilized as constraints in map matching for vehicle navigation must be replaced with new rules in map matching for pedestrian/wheelchair navigation according to pedestrian or wheelchair users movement behaviors.

Existing map matching techniques based on road networks in vehicle navigation systems do not take these characteristics unique to pedestrian/wheelchair navigation into account, thus they are not suitable for pedestrian/wheelchair navigation. For this, to provide appropriate navigation assistance, special designs and further customization of

map matching are required to meet the specific needs of both pedestrians and wheelchair users.

1.2 OVERVIEW OF MAP MATCHING IN PEDESTRIAN/WHEELCHAIR NAVIGATION SYSTEMS/SERVICES

In general, map matching algorithms integrate estimated locations, from positioning sensors such as GPS and DR, with a road network map to identify the correct link on which a vehicle is traveling and to determine the location of a vehicle on that link (Karimi et al. 2006; Quddus 2006; Ochieng et al. 2004). Map matching plays a crucial role in a navigation system whose logic is heavily dependent upon the characteristics of the underlying positioning sensors. The success of a navigation application mainly depends on the suitability of its positioning sensors and the map matching algorithm.

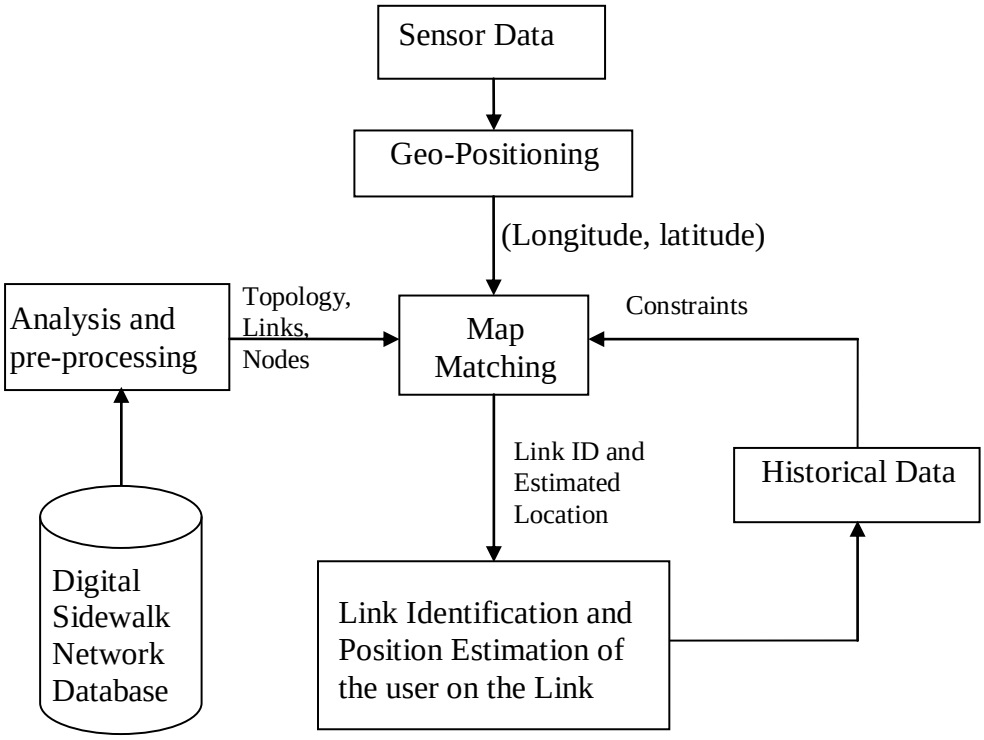


Figure 1.2. Map matching process in pedestrian/wheelchair navigation systems

Figure 1.2 highlights the map matching process in a pedestrian/wheelchair navigation system/service where a sidewalk map, instead of a road map, is used. The sidewalk network database includes sidewalk geometries, sidewalk topologies and relevant information for personal accessibility. Information such as sidewalk conditions (e.g., grade, steps, smoothness) and building properties (e.g., accessible entrance and elevators) are also needed. Geo-positioning data come from sensors, such as GPS, accelerometer, and compass sensors. Prior to map matching, a pre-processing task is performed on the sidewalk network database to prepare geometrical and topological information of sidewalk segments with appropriate attributes for map matching. The map

matching process incorporates geo-positioning data and pre-processed spatial data. Once a correct sidewalk segment is identified, each new positioning data is projected onto the segment to estimate a new position. Moreover, a user's historical trajectory is used as a means to speed up the map matching process.

1.3 MAP MATCHING CHALLENGES FOR PEDESTRIAN/WHEELCHAIR NAVIGATION

Existing map matching algorithms are designed for vehicle navigation systems. Utilizing road networks, instead of sidewalk networks, for map matching does not appropriately address the navigation needs of pedestrians/wheelchair users. As shown in Figure 1.2, in addition to map data, geo-positioning techniques also play a role in map matching. Some existing map matching algorithms for vehicle navigation use other sensors, in addition to GPS, such as DR and gyroscope. Such sensors are not usually available in pedestrian/wheelchair navigation services. As a result, map matching algorithms that are based on the sensors that are only available on vehicles are not directly applicable for pedestrian/wheelchair navigation. Furthermore, existing solely-GPS-based map matching algorithms may work for pedestrian/wheelchair navigation only after considering those specific requirements discussed in Section 1.1. Analyzing the characteristics of pedestrian/wheelchair navigation and comparing them with those of vehicle navigation reveal the challenges that must be addressed in developing map matching algorithms for pedestrian/wheelchair navigation.

First, in a pedestrian/wheelchair navigation system/service, a sidewalk network is required, which exacerbates the map matching process. Since most roads in an urban area have sidewalks on both sides, a sidewalk network is much denser than its corresponding road network is. Determining the side of the road on which the pedestrian/wheelchair is moving by pedestrian/wheelchair navigation systems/services is a challenging task. Figure 1.3 compares the density of a sidewalk network and that of its corresponding road network.

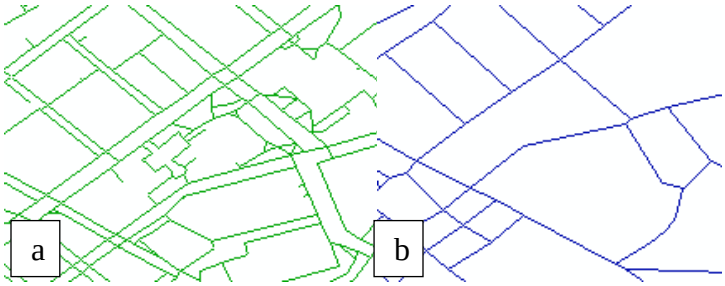


Figure 1.3. Sidewalk network (a) versus road network (b) in the same area

Additionally, pedestrians and wheelchair users sometimes move on a random path rather than follow the sidewalk. This further compounds the map matching process. Figure 1.4 shows the trajectory of a user along a route with no corresponding sidewalk in the area's map database.

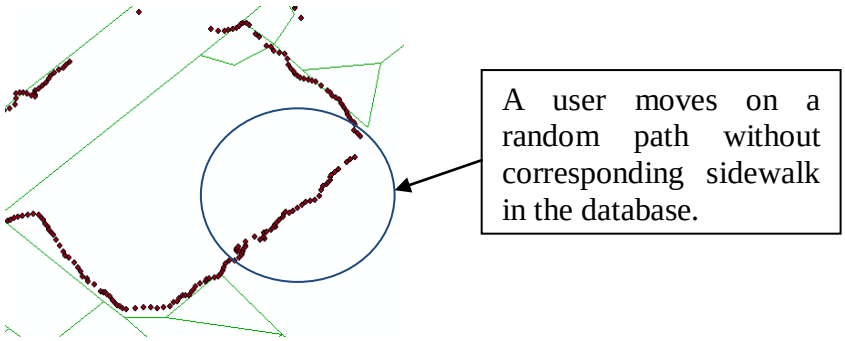


Figure 1.4. No path in map database

Second, GPS errors cause issues with change in location, time or weather, especially in dense urban areas, where high buildings, among other obstacles, block satellite signals. Since pedestrians and wheelchair users are on sidewalks close to buildings, the navigation system/service is more susceptible to GPS signal loss or signal degradation than vehicle navigation systems are. Figure 1.5 shows the problem caused by GPS errors in places with high-rise buildings. Position Dilution of Precision (PDOP), as a measure of overall uncertainty of a GPS position, represents quality of GPS signals. A PDOP value of 1 indicates a good satellite configuration and high-quality data; conversely, PDOP values above 8 are considered poor. The quality of the data decreases as the PDOP value increases. The PDOP of GPS positions, shown in Figure 1.5, ranges between 2.5 and 11.4 which are considered poor and very poor, respectively.

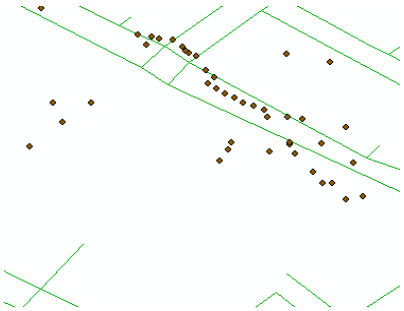


Figure 1.5. Poor GPS signals (PDOP between 2.5 and 11.4)

Moreover, GPS position fixes are less reliable at speeds of less than 3.0 m/s (Ochieng et al. 2004; Taylor et al. 2001), which are often the case with pedestrians and wheelchair users. At such a low speed, the uncertainty in positions could impede the derivation of heading based on displacement (Taylor et al. 2006).

Obviously, since geo-positioning only by GPS is insufficient to support navigation and tracking, there is a need for advanced map matching techniques and/or additional data from other sources. An alternative to geo-positioning by GPS is a multi-sensor map matching approach. Due to recent advances in computing and mobile device technologies, smartphones, like iPhones and Android phones, are growing in popularity. Sensors such as cameras, accelerometers, compasses, and gyroscopes in mobile phones can be employed by navigation systems. With these sensors, a multi-sensor map matching approach, using a smartphone as the platform, can provide a solution to seamless tracking in pedestrian/wheelchair navigation systems/services. However, consumer-grade GPS receivers built on smartphones cannot provide positioning data as good as professional-grade GPS receivers. Figure 1.6 shows a route collected both

through a professional-grade GPS receiver and through a consumer-grade GPS receiver on a smartphone. The green points represent data obtained through a professional-grade GPS receiver, and the red points are GPS data obtained through a smartphone.

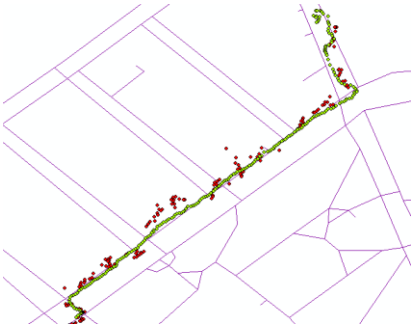


Figure 1.6. Comparison of GPS data from a professional GPS receiver (green) and a smartphone (red)

In summary, map matching in pedestrian/wheelchair navigation systems/services is more complex and challenging than map matching in vehicle navigation systems is, primarily due to high density of sidewalk networks and poor GPS signals.

1.4 GOAL AND OBJECTIVES

The overall goal of this research is to contribute to the realization of a fully functional pedestrian/wheelchair navigation system/service, by developing advanced map matching techniques and testing them on real sidewalk networks to validate their accuracy and performance. Such a navigation system/service will find applications in many areas including route guidance, tour guides, pedestrians/wheelchair user tracking and

monitoring, accident and emergency responses, and many more.

To achieve this goal, the following objectives are set forth in this dissertation:

- To develop an efficient candidate segments selection algorithm.
- To develop a number of map matching algorithms based on different techniques such as fuzzy logic, chain code, temporal probabilistic reasoning, computer vision, and the integration of motion sensors with vision and GPS.
- To validate the developed map matching algorithms using real-world field data.

1.5 CONTRIBUTIONS

This research contributes the following:

1. An adaptive candidate segments selection algorithm using a clustering technique;
2. A set of advanced GPS-based only map matching algorithms for pedestrian/wheelchair navigation systems/services;
3. Techniques based on accelerometer and compass data for recognizing a user's movement pattern to assist map matching.
4. Advanced map matching techniques based on computer vision;
5. A multi-sensor map matching approach that provides seamless map matching services in a pedestrian/wheelchair navigation system/service. This is accomplished by integrating motion sensors, such as an accelerometer and a compass, with vision and GPS data.

1.6 ORGANIZATION

This dissertation is organized into six chapters. Chapter 1 states the motivation of this research, gives an overview of the challenges in this study, and states its goal, objectives and contributions.

Chapter 2 provides a background on pedestrian/wheelchair navigation systems/services. It also reviews the related literature on existing map matching techniques, computer vision techniques, relative geo-positioning techniques and mobile technology for navigation applications.

Chapter 3 presents an adaptive candidate segments selection algorithm to efficiently find a set of candidate segments as the first step in map matching.

Chapter 4 discusses three advanced map matching algorithms for pedestrian/wheelchair navigation systems/services: chain-code-based map matching algorithm, Hidden Markov Model (HMM)-based map matching algorithm, and fuzzy logic map matching algorithm.

Chapter 5 presents two multi-sensor map matching algorithms based on a smartphone client/server platform. A movement pattern, recognition assisted, map matching algorithm for improving efficiency and accuracy of map matching is discussed. Multi-sensor integrated map matching, using a monocular visual odometry technique to provide users with seamless positioning services, is presented.

Chapter 6 summarizes the research and its contributions, discusses conclusions, and provides recommendations for future research.

2.0 BACKGROUND AND RELATED WORK

The first vehicle navigation and positioning system was built by the electronics industry in 1975 (Krakiwsky, 1993). In the thirty years since then, the number of such systems has grown rapidly. Today, most of the leading car manufacturers have developed Global Navigation Satellite Systems (GNSS) based in-car navigation systems.

In general, a vehicle navigation system is a satellite navigation system designed for use in automobiles. It typically uses a navigation device to acquire position data and to locate the user on a road based on an embedded map database. Moreover, the system can provide users directions to other locations along the road network.

Similar to vehicle navigation systems, pedestrian navigation systems aim to provide continuous positioning and tracking of a mobile user with a certain positional accuracy and reliability. Not present in vehicle navigation systems, a very challenging task for pedestrian navigation is to navigate in urban environments with a mixed indoor and outdoor environment as pedestrians travel in spaces where existing location methods cannot work continuously in stand-alone mode (Retscher et al., 2006).

Research has also focused on high maneuverability and navigational intelligence for driving a wheelchair in domestic environments (Pires, 1997; Simpson et al., 2004). These wheelchairs are usually equipped with sensors to detect obstacles or environmental

markers for localization and navigation. While some work on outdoor wheelchair navigation using GPS and other sensors has been reported (Imamura et al. 2004, Wu et al. 2005), their contributions are on obstacle detection and autonomous navigation.

2.1 SIDEWALK NETWORKS

Sidewalk networks are required to provide connections among commercial, institutional, municipal, educational and recreational facilities in any geographic area. The sidewalk map database consists of geometrical, topological, and attribute information (Kasemsuppakorn and Karimi, 2007). Geometrical information contains the geospatial coordinates of sidewalk segments. Topological information represents the connectivity of sidewalk segments. Finally, attribute information in a sidewalk map contains the characteristics of sidewalk segments.

Sidewalks along roads normally have two sides. Compared to a road network, the sidewalk network in the same area must be generated with more geometrical and topological data; therefore, sidewalk networks are much denser than road networks. Moreover, since sidewalks are connected to entrances of buildings that pedestrians and wheelchair users need to access, accurate map matching on the correct side of a road segment is one of the most important tasks for map matching in pedestrians/wheelchair navigation. Due to the density of sidewalk networks, this is more complex than map matching on roads.

In addition to sidewalk segments along roads, any path that can be used by

pedestrians and wheelchair users should be included in the sidewalk map database. With this increased number of segments, branches and paths in the sidewalk network database, map matching algorithms have more challenges to overcome.

2.2 GEO-POSITIONING TECHNOLOGIES AND TECHNIQUES

The geo-positioning technologies used in vehicle navigation systems have undergone a major evolution over the last few years. Several geo-positioning techniques are being used in vehicle navigation systems or under research worldwide. These include GNSS, WiFi Positioning System (WPS), Cellular Positioning System (CPS) and Dead Reckoning (DR) (Rizos et al., 2005, LaMarca et al., 2008). For pedestrian navigation, in addition to sensors used in vehicle navigation systems, sensors such as a low-cost attitude sensor (digital compass) providing the orientation and heading of the person being navigated and a digital step counter or accelerometers for travel distance measurements can be employed (Retscher et al. 2006). Similarly, wheelchair navigation systems also use both absolute positioning techniques, like GPS, and relative positioning techniques, like DR, odometry and Inertial Navigation System (INS) (Anousaki et al., 2007; Venkatraman et al., 2009). Additionally, other sensors may also be utilized in outdoor wheelchair navigation, like ultra-sonar and laser, which are mainly used for avoiding obstacles, not for positioning.

Among all these geo-positioning technologies, satellite-positioning technologies (i.e., GNSS) are widely employed for outdoor navigation. GPS is the most popular GNSS. The achievable positional accuracies of GPS-based navigation systems range from a few

meters to 10 m in stand-alone mode and sub-meter to a few meters in differential mode, (e.g., Differential GPS or DGPS). However, GPS often suffers from availability and accuracy issues. Due to obstructions, a sufficient number of satellites may not be available for a short period of time. In urban areas, especially downtown areas, GPS signals could be very weak compared with those in rural areas. Accuracy of GPS data also can be influenced by weather and it may also fluctuate in the same location with over time.

Therefore, a solution to this problem is the addition of new sensors (i.e., multi-sensors) to bridge the absence of satellite signals. These additional sensors are critical for pedestrian/wheelchair navigation to obtain the necessary data for localization.

Possible outdoor geo-positioning technologies (see Rizos et al., 2005; Retscher et al., 2006), both absolute and relative, include GPS, WPS, CPS, Vision-based Positioning System (VPS), DR, and INS. The outdoor geo-positioning technologies that can be integrated with GPS are listed in Table 2.1.

Table 2.1 Outdoor geo-positioning technologies potential for integration with GPS

Outdoor geo-positioning	**Type**	**Accuracy**	**Coverage**
WPS (WiFi)	Absolute	High ± 1 – 3 m	limited
CPS (Cellular)	Absolute	Low ± 50-100m	good
VPS (Vision-based)	Absolute/Relative	High	middle distance
DR (Odometry+INS)	Relative	Low ± 20 – 50 m per 1 km	–

WPS has grown rapidly in recent years and provides reasonable positional accuracies, but it suffers from limited signal coverage. WPS cannot locate targets when they are out of range of Wi-Fi signals. WPS accuracy also depends on Wi-Fi hotspot databases, which are built by fingerprinting wireless access points and must be constantly updated to keep up with Wi-Fi hotspot changes (LaMarca et al., 2008). Moreover, only a few commercial companies build databases of sufficient size to be used for Wi-Fi positioning. To build and maintain such a database requires considerable effort on the fingerprinting techniques which are not the focus of this dissertation; therefore, WPS is not chosen in this research.

In contrast to WPS, CPS has good signal coverage in urban areas, but it is less accurate. Kitching (2000) proposed integrating GPS and CPS at two levels: (a) at the measurement data level and (b) at the infrastructure level. Although the addition of cellular network Base Stations (BSs) can improve the horizontal accuracy of GPS positions, a number of infrastructure modifications are required to enable the cellular ranging measurements necessary in a positioning solution. Some commercial companies, like Qualcomm, are currently working on such integrations.

Integrating DR sensors, which are based on relative positioning techniques, is one alternative to overcome GPS errors in navigation systems/services. DR sensors, e.g., gyroscope and accelerometer, obtain the travelled distance from velocity and acceleration measurements and estimate direction of motion or heading and height difference.

A wheelchair's position can be estimated based on distances measured with odometer devices mounted on both wheels of the wheelchair. Accelerometers can provide relatively high position accuracy in a relatively short time. Due to its bias drift, the position error will grow over time. Another approach to measure a wheelchair’s movement is to use

wheel revolution counters (Lankton et al., 2005), which can determine the distance and time of travel by using seat occupancy sensor and tilt meter. A seat occupancy sensor reports when a user is seated in the wheelchair and the tilt meter measures the position of the wheelchair in angular degrees. However, in practice, problems such as wheel slippage and sidewalk surface condition contribute to poor accuracy. Also since localization of pedestrians or wheelchair users is based on horizontal distance, wheel revolution counters, which measure slope distance, are not considered in this dissertation.

For pedestrians, positioning data can come from accelerometer measurements based on an INS or from a step-counter and step-length estimator from a typical pedometer. Accelerometer measurements used for pedestrians have the same problem as in wheelchairs, i.e., positioning errors would be accumulated over time. Regarding pedestrian dead reckoning (PDR), the position accuracy in the pedometer/GPS integration relies mainly on estimations of the number of steps (counted by the accelerometer) and the length of the steps (calculated by the pedometer). For a pedometer to measure distance, the average step length of a user must be measured, which requires users to walk in a consistent pace. As PDR sensors are most effective when they are mounted on user's feet, they are not suitable for wheelchair users. Considering these above issues of DR/INS for pedestrians and wheelchair users, DR/INS techniques are not chosen for pedestrian/wheelchair navigation in this dissertation.

On the other hand, accelerometers have been used as motion detectors (DeVaul &Dunn 2001), as well as, for body-position and posture sensing (Foerster, Smeja, & Fahrenberg 1999). Acceleration measurements from accelerometers can be used for activity recognition which is usually formulated as a signal processing and classification

problem (Ravi et al., 2005). Researchers in this area mainly focus on the identification of physical activities, such as walking, jogging, resting, standing, climbing, running, etc. Modeled from other research studies using accelerometer, compass, barometer, gyroscope, or combinations of them to identify pedestrian's activities (i.e., walking, jogging and going upstairs or downstairs) (He et.al., 2009), this dissertation classifies user's movement as patterns including no movement, walking, running and turning in order to identify the user's movement behavior .

Computer vision is a growing research field (Koller et al., 1997; Chen and Shibasaki, 1999; Malis et al., 2002). Many "smart" wheelchairs use computer vision to help avoid obstacles and to explore indoor environments. Computer vision also can be used to compute distance and to estimate indoor and outdoor locations. Vision-based positioning (Henlich, 1997) is currently an active research topic (Chen and Shibasaki, 1999; Tardif et al., 2008). Feature extraction for positioning is not an easy task due to diversity of images and the wealth of information captured in images. Object recognition is one of the most difficult tasks in computer vision, which involves feature extraction, object clustering or classification and location/pose estimation, etc. However, compared to the aforementioned geo-positioning technologies, vision-based positioning technology has several advantages. First, vision-based positioning techniques are less influenced by the environment. This can help geo-positioning in areas without good GPS signals. Second, feature extraction using computer vision techniques in recent years has advanced and matured to guarantee high positional accuracy since feature extraction is critical in estimating motion in vision models. Finally, using a camera as a sensor for vision-based positioning is relatively inexpensive and practical. Realizing the advantages and

disadvantages of modern geo-positioning technologies and techniques, vision-based positioning is an attractive and emerging technology and thus, chosen for integration with GPS for map matching in this dissertation.

2.3 MAP MATCHING TECHNIQUES IN NAVIGATION SYSTEMS

Over the past two decades, many map matching algorithms have been proposed and evaluated in various scenarios. Most algorithms have been developed for vehicle navigation, a few for outdoor pedestrian navigation, but very few have focused on map matching for outdoor wheelchair navigation.

Current map matching algorithms can be divided into three main approaches, geometric map matching, topological map matching and advanced map matching (Quddus et al. 2007). Geometric map matching consists of point-to-point map matching, point-to-curve map matching and curve-to-curve map matching. Topological map matching utilizes both geometrical and topological data to make matching decisions. Advanced map matching applies models, such as probability theory, Kalman filter, and fuzzy logic, to either geometrical map matching or topological map matching, so the algorithm obtains better matching results with an increase in the complexity of the computations required.

In point-to-point map matching each newly obtained position is matched to the closest “node” or “shape point” of a road segment. While point-to-point map matching is both easy to implement and computationally fast, it is very sensitive to the geometry of

the road network. In point-to-curve map matching, each newly obtained position is matched to the closest "line segment" (curve) in the road network which is selected as the segment on which the vehicle is traveling. Although point-to-curve map matching can identify road segments more accurately than the point-to-point map matching, in dense networks, such as those for urban areas, it may not be able to produce good solutions. In curve-to-curve map matching, a vehicle's trajectory (current travelling curve) is matched to road segments (network curves). Curve-to-curve map matching finds a matched road segment in three steps. In the first step, it constructs piecewise linear curves using candidate nodes through point-to-point map matching. In the second step, it constructs piecewise linear curves using the vehicle's trajectory. In the third step, it calculates the distance between vehicle's trajectory (step 2) and the curves corresponding to road segments (step 1). The road segment closest to the vehicle's trajectory is selected as the solution. Curve-to-curve map matching may not always produce a good solution because of outlier sensitivity and its reliance on point-to-point map matching (Ochieng et al. 2004).

Topological map matching (Meng et al. 2006; Quddus et al. 2003) takes into account both geometrical and topological information of the road network, as well as, history of GPS data. In topological map matching, the vehicle's trajectory and topological features of the road (e.g., road turn, road curvature, and road connection) are matched. However, in some cases, topological map matching resorts to a post-processing mode to identify the correct road segment, and in other cases it can rely on a global matching strategy. Neither of these cases is suitable for real-time applications. Modifying the weighting scheme with additional criteria and parameters (i.e., vehicle's speed, position relative to candidate road

segments, heading information (directly from GPS data), or position data obtained by integrating GPS and DR), will improve the performance of topological map matching.

Both geometrical and topological map matching algorithms are used as the basis for developing other advanced map matching algorithms. These advanced algorithms employ additional techniques to improve performance, such as a Kalman Filter or an Extended Kalman Filter, a flexible state-space model and a particle filter, and a fuzzy logic model (Quddus et al. 2007; Jagadeesh et al. 2004).

Quddus (2006, 2007) reported a comparison of performances of some map matching algorithms. These algorithms, as well as, two algorithms by Wu et al. (2007) and Liu et al. (2008) are presented in Table 2.2.

Table 2.2. Comparison among performances of various map matching algorithms for vehicle navigation (after table in Quddus, 2007)

Authors and year of publication	Navigation sensors	Test Environments	Correct Link Identification (%)	Horizontal Accuracy (m)
Kim et al. (2000)	GPS	Suburban	–	10.6 (100%)
Kim and Kim (2001)	GPS/DR	Urban and suburban	–	15m (100%)
White et al. (2000)	GPS	Suburban	85.8	–
Pyo et al. (2001)	GPS/DR	Urban and suburban	88.8	–
Taylor et al. (2001)	GPS + Height	Suburban	–	11.6 (95%)
Bouju et al. (2002)	GPS	Suburban	91.7	–
Yang et al. (2003)	GPS	Suburban	96	–
Quddus et al. (2003)	GPS/DR	Urban and suburban	88.6	18.1 (95%)
Syed and Cannon (2004)	GPS/DR	Urban and suburban	92.8	–
Ochieng et al. (2004)	GPS/DR	Urban and suburban	98.1	9.1 (95%)
Quddus et al. (2006b)	GPS/DR	Urban and suburban	99.2	5.5 (95%)
Wu et al. (2007)	GPS	Urban	95.14	-
Liu et al. (2008)	GPS	Urban	99.4 in one case	-

The percentage of correct link identification ranges from 86% to 99%. The 2-D horizontal positional accuracy ranges from 18 m to 5.5 m (95%). It shows that many of these algorithms are based on utilizing multiple geo-positioning technologies in combination to obtain results with good accuracy. Quddus (2006) concluded that the fuzzy-logic map matching algorithm produces best results when GPS and DR are considered. It should be noted that since those map matching algorithms in Table 2 were tested on different areas with different road maps, the performance in accuracy is not

only a function of the geo-positioning technologies and the map matching algorithms they use, but also a function of the road networks used in the tests.

2.4 COMPUTER VISION IN NAVIGATION SYSTEMS

Previous work on vision-based navigation systems mainly consists of research on automatic driving systems (Castro et.al, 2001) and pedestrian navigation systems (Fritz et.al, 2006; Steinhoff et.al, 2007). Both systems use vision-based positioning techniques to provide either absolute or relative positioning. Absolute positioning computes the absolute position of an object by measuring distances from it to other known objects, like buildings, on the basis of recognizing known objects from images. Relative positioning calculates incremental positions by measuring movement or rotation/orientation step by step.

In pedestrian navigation systems, landmarks are used as references for absolute positioning. Given known locations of landmarks and estimated distances from users to those landmarks, pedestrian navigation systems can provide users with their absolute positions. For all known landmarks, locations of these landmarks are recorded in the database server in advance. An example scenario follows. A user on a tour captures pictures and sends them to the server. Once landmarks shown in the pictures are matched with the known landmarks in the server, the tour system can estimate the user's location by retrieving the known landmark's location in the database server and computing the distance from the user to the known landmarks. Figure 2.1 shows an example of user test case in a tour system from Steinhoff et.al (2007).

Figure 2.1. Example image with landmarks (Steinhoff et.al, 2007)

In this figure, the red points on the image are landmarks, which are used as reference points. When a user travels on a tour, his/her locations are computed and marked on the image, which are shown as query points. Steinhoff et.al (2007) concluded that the absolute positions obtained for the waypoints in the tour system were more accurate than GPS.

However, since mobile devices have limited computation capability and memory capacity, the tour system has to be built on a client/server network. In the client/server network, mobile devices communicate with the server using the wireless communication infrastructure. Using landmarks as distinctive features in the environment, images with landmarks are collected in advance and the extracted features, such as color or shape features, are stored in the server. On the client side, mobile users capture pictures and send them to the server. On the server side, the server responds to clients with their location by doing image retrieval. If a landmark in an image captured by a user is recognized, the server then can locate the user on the map near the known landmark's

location. Further, the tour system can prompt the user through a set of directions to provide guidance on his/her tour.

By contrast, automatic driving systems are based on more sophisticated computer vision techniques. Computer vision is not only for positioning but also for avoiding obstacles and driving without people in control. Automatic driving systems use a relative positioning approach in their vision-based positioning. By tracking visual features observed from a moving camera, relative positioning uses the transformation relationship between an image coordinate system and a world coordinate system to estimate a vehicle's location. To accomplish this, correspondence between image pairs or sequences must be obtained. Once an image set has been matched, bundle adjustment techniques can be used to compute the camera position, which is a surrogate for the vehicle's position.

With the location of a starting point, the automatic driving system can also mark a vehicle by its absolute positions on a map by adding relative movements, which is called visual odometry in robotics (Levin et al., 2004, Olson et al., 2001). Tardif et al. (2008) presented a system for motion estimation of a vehicle using an omnidirectional camera which successfully performed high precision camera trajectory estimation in urban scenes with a large amount of clutter. The visual odometry algorithm in their paper mainly includes feature extraction and tracking, motion estimation, and structure computation. Other systems prefer to use two cameras as a pair to improve accuracy.

In both automatic driving systems and pedestrian navigation systems, feature extraction and object recognition are the key techniques. Global features and local features are two types of image features widely used in object recognition algorithms

(MOBIS 2005). Global features describe an image as a whole. They have the ability to generalize an entire object with a single feature vector. Local features, on the other hand, focus on image patches. They are computed at multiple points in an image and are consequently more robust to problems of occlusion and clutter (MOBIS 2005). Global features mainly consist of color, texture and shape. Color features capture the chrominance information in the image. For example, sky, ground, and vegetation can be classified by colors. Texture and shape represent illuminance. Textures are also often used as local features. For instance, local region features can be described by their scale and texture, e.g., the Scale-Invariant Feature Transform (SIFT) approach (Lowe, 1999). Boundaries of segmented regions or a boundary of an entire object are often used as shape features. Although global features are widely used in various applications, they are not suitable for applications with large changes of background, viewpoints, occlusion, resolution, lightening and environment (MOBIS 2005).

Lowe (2001) defined local features as having intermediate complexity, which means that they are distinctive enough to determine likely matches in a large database of features and are sufficiently local to be insensitive to clutter and occlusion. In general, local features extraction includes four steps:

1. Detect a set of local features in an image
2. Compute a description for each local feature
3. Use descriptors to find similar local structures in images for matching or an image model for recognition
4. Verify object

Three basic types of local features each with several subtypes were identified in MOBIS (2005). These include interest points or “corners”, groups of line segments, and distinguished regions.

An “interest point” (Mikolajczyk et al., 2004) is a point in an image which has a well-defined position and can be robustly detected. An interest point can be a corner (Derpanis, 2004), an isolated point of local intensity maximum or minimum, line endings, or a point on a curve where the curvature is locally maximal. "Corner", "interest point" and “feature” are used somewhat interchangeably because corners are very stable features detectable even in the case of substantial viewpoint or photometric changes. They are also insensitive to clutter and occlusion. However, in order to gain distinctiveness of each individual corner feature, a small local neighborhood of the corner has to be considered (Mikolajczyk et al., 2004). Various computer vision algorithms have been proposed for feature extraction (Tuytelaars et al. 2008), such as Harris corner detector (Harris and Stephens, 1988), Harris-Affine (Mikolajczyk et.al., 2002, 2004) and Hessian-Affine detectors (Mikolajczyk et.al., 2002, 2004), SIFT(Lowe, 1999) and SURF(Bay, 2008).

Line segments are also often used as local image features. For example, Bay (2005) proposed line segments extracted by Canny edge detector. Shapiro (2002) presented a local feature called Consistent Line Cluster, defined as a collection of line segments grouped by colors, orientation and spatial features.

Distinguished regions are image elements (subsets of image pixels) that posses some distinguishing, stable property. Tuytelaars and Van Gool (2001) proposed Intensity Extrema-based Region Detector (IBR) that utilizes local extrema in the image intensity as anchor points. To improve the efficiency of this approach, Matas et al. (2002) presented

maximally stable extrema regions (MSER). Salient Regions and Scale-invariant Shape Features are other examples of the distinguished region approach. However, characteristics of objects in the images will ultimately determine what local features are chosen for matching or recognition.

Moreover, Deselaers et al. (2007) compared performances of a large variety of visual descriptors, both global and local features, which include Appearance-based Image Features, Color Histograms, Tamura Features, Global Texture Descriptor, Gabor Histogram, Gabor Vector, Invariant Feature Histograms, LF Patches, LF SIFT etc. One of their conclusions is that the global search of SIFT features extracted at Harris points performs best on the ZuBuD database (Shao 2003) which supports image-based building recognition.

2.5 MOBILE TECHNOLOGY IN NAVIGATION APPLICATIONS

Mobile phones have become a compelling platform for location-based services. Most current smart phones have a built-in global positioning system (GPS) receiver, cameras and low-cost MEMS motion sensors such as an accelerometer or a gyroscope, and/or a magnetometer. To achieve the goal that a pedestrian/wheelchair navigation system can continuously provide position and heading information on a digital map to guide users, the ubiquity, portability, good connectivity, the trend towards increased performance and inclusion of multi sensors of smart phones make them an ideal target device in this dissertation.

3.0 ADAPTIVE CANDIDATE SEGMENTS SELECTION ALGORITHM

Generally speaking, the first and most important step in map matching is to identify the correct segment on which an object is moving. Once the correct segment is determined, estimating the location of the object on that segment, though with a degree of uncertainty, is straightforward. Finding the correct segment requires that map matching algorithms identify a set of candidate segments based on a received GPS position and compare and analyze them to decide the most likely segment. Therefore, finding the set of candidate segments is imperative in map matching algorithms. This involves two types of situations. One is initializing a search range when the first position data is received and the other is updating the search window continuously for new positions. If the network is large, the search process is a time consuming operation. To identify the segment on which an object is moving not only demands correctness, but demands efficiency as well. Especially for a real-time navigation, the efficiency of finding the correct segment is even more important.

At this stage, our goal is to select a set of candidate segments close to the received GPS position. A widely used approach for determining the search window is to create a buffer centered at the GPS point and identify segments within this window as candidates. The search process is often based on an indexing technique which traverses a tree, for example an R- tree, down to its leaf nodes to find those segments within the coverage of

the search window. Since the topology of segments is very important to match GPS points, it is appropriate to consider those adjacent segments to GPS points as candidate segments. However, an indexing technique, like R- tree, does not consider topology of segments in constructing the tree. It is likely that adjacent segments are stored in different subtrees because the tree is constructed in a certain order (e.g., Hilbert order to build Hilbert R- tree). As a result, retrieval of candidate segments in a given network requires several passes through the R- tree. The searching complexity depends significantly on the distribution patterns of segments on the network. Furthermore, the same segment may be indexed more than once in the tree, requiring traversal of multiple paths. To avoid redundant searching in indexed spatial databases, in this chapter, we address the problem of efficiently finding candidate segments in spatial networks given GPS points of a moving object. The algorithm is a bottom-up approach that builds a segment-clustering tree to achieve both adaptability and efficiency in finding candidate segments.

3.1 RELATED WORKS

Current research on searching objects in spatial networks mainly employ indexing techniques with the goal of improving the efficiency of queries (Tele Atlas, 2008; Zhao et al, 2001). Spatial indexing for spatial and spatio-temporal queries has been an active research topic over the past decades. Since 1984 when R-tree by Guttman (1984) was presented, several indexing techniques have been developed for efficient spatial queries. Basic indexing techniques with variant structures such as R-tree, quadtree, B-tree, and

grid (Zhao, 1997) have been employed in different experiments focused on increased spatial query efficiency (Lin 2008; Chen et al. 2006; Kalashnikov et al. 2002).

R-tree was developed as an index structure for the efficient management of multi-dimensional and spatial data. Common operations performed on an R-tree include point location queries, range queries and nearest neighbor queries. In the map matching context, road or sidewalk segments can be represented as polygon objects and recorded on the leaf node level in an R-tree. Each leaf node holds two items for each data record. One is the bounding box of the object, and the other is information to place the object in the real world. Other nonleaf nodes of R-tree hold two items for each of its children: a bounding box of the child, and a pointer to the child.

A number of bulk loading approaches were developed to build R-trees (Bercken and Seeger, 2001). Top-down Greedy Split (TGS) and bottom-up approach are the two main methods (Alborzi. et al., 2007). A bottom-up approach builds an R-tree from leaf nodes level to upper level until it reaches the root node. In the lowest level, n data rectangles are sorted according to a predetermined sort order and m data rectangles are grouped in the upper level. The construction process is an iterative procedure from the bottom up to the root of R-tree. By contrast, a top-down approach first builds the higher levels of R-tree. The data rectangles are sorted according to a predetermined sort order and then split to build subtrees for the children recursively down to the final leaf nodes. Both approaches have to predetermine an order of objects stored in the tree. This causes a problem where the pre-ordered leaf nodes cannot represent adjacent objects in a spatial space appropriately. There is a lack of efficiency when spatially adjacent objects in R-tree are queried as a group, but were stored in multiple paths.

3.2 SPATIAL NETWORK DATA REPRESENTATION

3.2.1 Hierarchical Clustering Tree

Unlike indexing techniques, where objects are organized in a certain order, clustering techniques group objects by their characteristics. One important characteristic of spatial networks is the connectivity of segments that we use to cluster them.

In the family of clustering algorithms for different types of applications, the two most common branches are partitioning and hierarchical clustering algorithms (Huang, et al. 1998; Kotsiantis, 2004). Partitioning algorithms create a "flat" decomposition of a data set into a set of clusters. Examples of partitioning algorithms include k-means, k-medoids algorithms and density-based approaches. They generally need some input parameters that specify either the number of clusters that a user intends to find or a threshold for point density in clusters. However, it is difficult to determine what parameters are needed and what values they should have, as the parameters may not even exist.

In contrast, hierarchical clustering algorithms do not actually partition a data set into clusters but compute only a hierarchical representation of the data set reflecting its possible hierarchical clustering structure. Hierarchical clustering algorithms are more robust and less influenced by cluster shapes. Additionally, they are less sensitive to largely differing point densities of clusters, and they can represent nested clusters (Sander, Jörg et al. 2003). Since they partition without knowing the number of clusters (e.g., distribution of segments in a spatial network) ahead of time, hierarchical clustering

becomes an attractive alternative to spatial indexing techniques. The next section will describe clustering segments in a hierarchical clustering tree by using the average-linkage method.

3.2.2 Clustering Road Segments

In graph theory, a graph is formally represented by <V, E>, where V represents vertices in the graph and E represents edges in the graph. A general adjacency matrix in graph theory is $A=\{a_{ij}\}$, where a_{ij} represents the weight (e.g., distance) between i^{th} vertex and j^{th} vertex. Consisting of intersections and segments, a spatial network can be represented as a graph where intersections are the vertices and segments are the edges. To better perform map matching for navigation applications, segments are clustered instead of intersections, because intersections only represent geometric information, whereas segments represent both geometrical and topological information.

To build the hierarchical-clustering tree of a spatial network, a new matrix A' is introduced in this algorithm. Generalizing a spatial network as a non-directional graph clustered by distance, A' becomes a symmetric matrix. To define A', two matrices are first defined. One matrix represents the topology of a spatial network, denoted by $T=\{ t_{ij} \}$, where t_{ij} is 0 if the i^{th} and the j^{th} vertices are on the same segment (this occurs when i and j represent the same vertex or i and j are the two vertices of a segment) and t_{ij} is 1 when the i^{th} vertex is not directly connected to the j^{th} vertex. The other matrix is an adjacency matrix $A=\{a_{ij}\}$, which computes the Euclidian distance between any two vertices to represent the closeness of the intersections in geometric space. As a result, a new matrix $A'= A.*T = \{ a_{ij} * t_{ij} \}$ is defined to combine the two factors, geometrical

distance and topological relationship from matrices A and T. The weight of each element in the matrix A' not only considers the distance between the vertices (i.e, intersections), but also considers the topology of the spatial network. Based on matrix A', we build a hierarchical clustering tree from the bottom-up. Segments on the lowest level of this tree, as the smallest units in the structure, are grouped together based on the distance of each other and the clustering process is continued until the root of the tree is reached. For example, Figure 3.1 shows 10 segments, labeled by ID from 0 to 9. Every segment has two vertices, so 10 segments have twenty vertices, from 1 to 20. Matrix A' therefore becomes a 20-by-20 matrix. Since it is symmetric, only it's upper or lower triangluar (off diagonal) needs to be stored. The weights in the upper triangular and the lower triangular are all set as 0s shown in Figure 3.2.

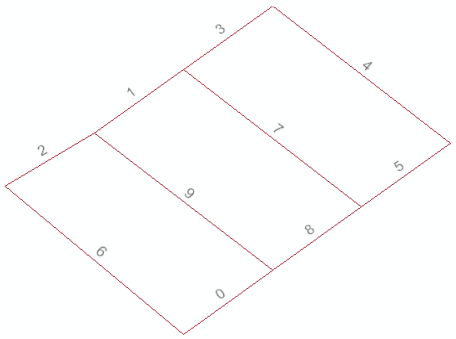


Figure 3.1. An example of road network

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0.2062	0.25	0.2126	0.2062	0.25	0.3202	0.3	0.3202	0.2	0.3	0	0.2126	0.2	0.25	0.1	0.2	0.1	0.2062
2	0	0	0.2062	0.2062	0.253	0.2062	0.2062	0.25	0.2	0.25	0.1	0.2	0.1	0.253	0.1	0.2062	0	0.1	0	0.2062
3	0	0	0	0	0.0943	0	0.1	0.2	0.3202	0.2	0.25	0.3202	0.2062	0.0943	0.25	0.1	0.2062	0.25	0.2062	0
4	0	0	0	0	0.1942	0.1	0	0.1	0.25	0.1	0.2062	0.25	0.25	0.1942	0.2062	0	0.2062	0.2062	0.2062	0.1
5	0	0	0	0	0	0	0.1942	0.2941	0.402	0.2941	0.3206	0.402	0.2126	0	0.3206	0.1942	0.253	0.3206	0.253	0.0943
6	0	0	0	0	0	0	0.1	0.2	0.3202	0.2	0.25	0.3202	0.2062	0.0943	0.25	0.1	0.2062	0.25	0.2062	0
7	0	0	0	0	0	0	0	0	0.25	0.1	0.2062	0.25	0.25	0.1942	0.2062	0	0.2062	0.2062	0.2062	0.1
8	0	0	0	0	0	0	0	0	0.2062	0	0.2062	0.2062	0.3202	0.2941	0.2062	0.1	0.25	0.2062	0.25	0.2
9	0	0	0	0	0	0	0	0	0	0	0.1	0	0.3	0.402	0.1	0.25	0.2	0.1	0.2	0.3202
10	0	0	0	0	0	0	0	0	0	0	0.2062	0.2062	0.3202	0.2941	0.2062	0.1	0.25	0.2062	0.25	0.2
11	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.3206	0	0.2062	0.1	0	0.1	0.25
12	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0.402	0.1	0.25	0.2	0.1	0.2	0.3202
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.25	0.1	0.2	0.1	0.2062
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3206	0.1942	0.253	0.3206	0.253	0.0943
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0.1	0.25
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2062	0.2062	0.2062	0.1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2062
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.25
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.2. Corresponding matrix (20-by-20)

Given the matrix A', a hierarchical clustering tree is built using the average-linkage clustering method. The average linkage clustering is based on measuring the proximity between two groups of objects. Here we use the average distance between all pairs of objects in cluster r and cluster s as the measurement. Its definition is as follows:

$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj}) \qquad (1)$$

Where n_r is the number of objects in cluster r and n_s is the number of objects in cluster s, and x_{ri} is the ith object in cluster r, and x_{sj} is the jth object in cluster s.

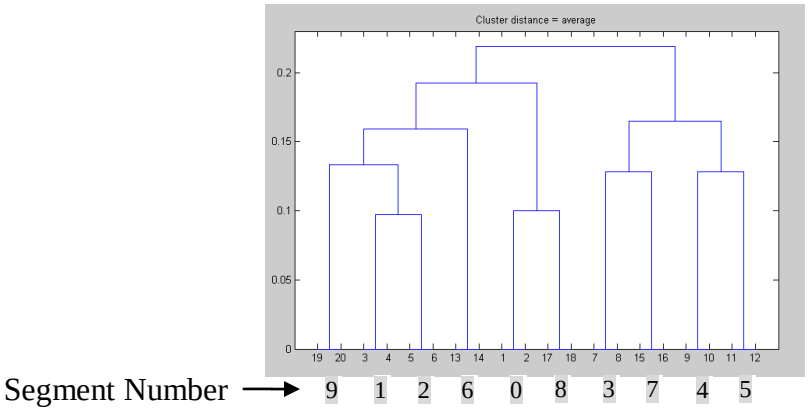


Figure 3.3. Corresponding clustering tree

After applying the average-linkage clustering method, Figure 3.3 shows how the hierarchical clustering tree is built from the matrix A' in Figure 3.2. The clustering process starts from the bottom where average distances of two vertices on the same segment as a cluster are set to zero in A'. Segments are grouped together in an order that clusters those with closer distances until all the clusters are grouped together and the root is reached. The root of the clustering tree represents the entire spatial network as one group.

3.3 ADAPTIVE SEARCHING ALGORITHM

Corresponding to the built clustering tree, a binary tree structure is generated and an adaptive searching algorithm is designed correspondingly. To support map matching, searching for segment candidates is conducted on the binary tree. The search starts from the entire spatial network and stops if a group of segments is found with the required criterion. Given the binary search tree and GPS points, instead of fixing a search window size for candidate segments, grouped segments are dynamically obtained with changes of the geometric and topological relationship between the GPS point and segments.

3.3.1 A Binary Tree Structure from the Clustering Tree

Since each group only joins one of the two members (one of which may be compound) in the clustering tree shown in Figure 3.3, the clustering tree is a binary tree. Therefore, the map matching process equates to finding the best-matched segment in all the candidate

segments indexed in the binary tree. Considering that segments have different positions, length and orientation, we use minimum bounding rectangles (MBR) (SAGAYARAJ et al., 2006) in the nodes of the tree. Similarly, each MBR in upper levels of the tree represents a group of segments. It is possible that sibling nodes have overlapping areas due to MBRs. The binary tree is constructed from bottom to top, so its root represents the bounding box of the entire spatial network. Figure 3.4 displays a small portion of the binary tree. Figure 3.5 indicates a scenario where a GPS point is located within the range of the MBR of a group of segments.

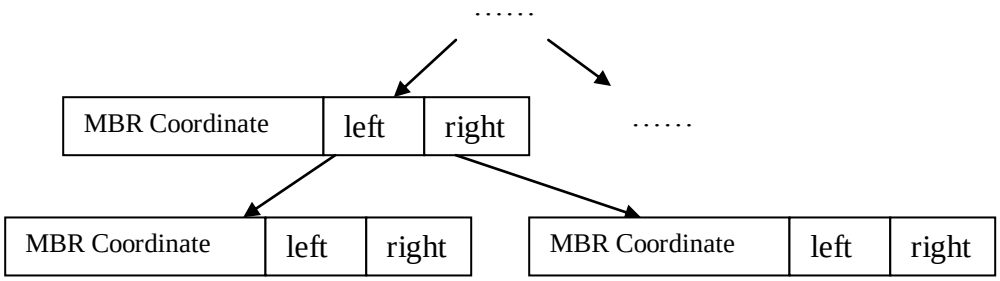


Figure 3.4. Data structure of a binary tree for segment clustering

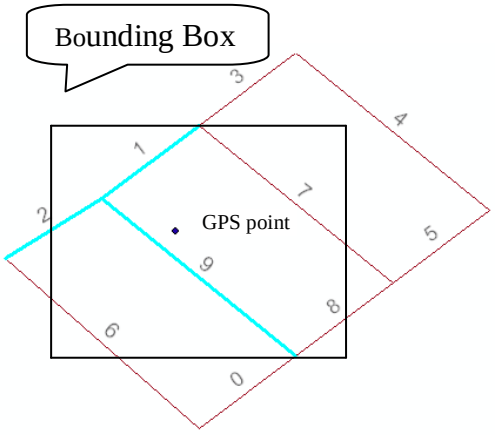


Figure 3.5. A GPS point is located within the range of a Bounding Box

3.3.2 Searching Algorithm

As shown in Figure 3.4, the range of MBR is narrowed down from top to bottom. Therefore, in order to determine the search window for map matching, the binary tree is traversed from top to bottom and will stop at a group node when a halt criterion is met. The halt criterion is very important; it determines the map matching window size. The searching algorithm is as follows:

1. Input the binary tree and a GPS point.
2. Set a criterion to halt searching.
3. If the GPS point is included in the boundary of the current group node and the halt criterion is not met, then search its sub-trees. Otherwise, search the sibling node.
4. If the halt criterion is met, then traverse all the leaf nodes of this sub-tree for the candidate road segments.

Since the candidate segments selection algorithm aims to find corresponding candidate segments given GPS positions, the halt criterion is determined by the relationship between the GPS point and the segment groups.

We note a GPS point as p(x, y), and a bounding box as B(minx, miny, maxx, maxy). Next, c refers to the center of the bounding box. Distance(p, c) calculates the distance between the GPS point (p) and the center point (c) and Within(p, B) evaluates whether p is within the bounding box or not. Based on the notation, the criterion is defined as:

IF Distance(p, c) < threshold and Within(p, B) THEN stop searching.

The threshold is related to the density of grouped road segments. To generalize the criterion, we set a threshold by testing on actual spatial networks. When the threshold is set relatively small, it would guarantee that the GPS point is close enough to the center of a selected group, in order to avoid misidentifying candidate segments by the GPS point located on the boundary of two clustered sub-groups.

3.3.3 Adaptive Search Window Set

In spatial indexing techniques, a search window for candidate segments is normally a fixed-size rectangle centered at a given GPS position. In order to retrieve those segments within the search window, searching algorithms need to go through the indexing tree several times. In contrast to such indexing techniques, a clustering tree provides an adaptive search window by clustering spatially closed segments into groups and only goes through the tree once. The output of the searching algorithm is a list of candidate segments.

The number of segments included in the list of candidate segments depends on three factors. One factor is the density of the spatial network. The second factor is the relevant dynamic position of GPS points in a clustered group. The third factor is the threshold in the halt criterion, which is set depending on GPS error range.

Similar to indexing techniques, nodes in the clustering tree have overlapping MBRs. It is possible that a GPS point is located within two overlapping MBRs. To justify which cluster should be selected based on its MBR, we compare distances between the GPS point and the center of each MBR. The one with a closer distance has a

higher possibility to be chosen as the selected cluster if the distance is less than the threshold. If a GPS point is located on the boundary of the two groups, then the search window will be adaptively adjusted to their parents, which is the higher-level clustered group. For instance, in Figure 3.5, if a GPS point is located on the corner of segment 2 and segment 6, the search window should cover the higher group that includes segments 9, 1, 2 and 6. In Figure 3.3, this group includes the group corresponding to the MBRs shown in Figure 3.5 and another group that contains only segment 6. This searching procedure starts from the root of the tree, as shown in Figure 3.3 and once the halt criterion is met, it will stop on the level which includes segments 9, 1, 2 and 6, rather than searching the lower level.

3.3.4 Adaptive Search Window Update

With an object moving, a map matching search window needs to follow its movement. The decision to update a search window or keep searching in the previous window depends on both the direction and speed of the object's movement. With the previous candidate segments in memory, and by knowing the successive points to estimate the object's movement, searching in the same segment group continues. However, with the object moving out of a known segment group, a new sub-tree has to be initiated. Under this circumstance, updating the search window just repeats the initial searching procedure.

3.4 PERFORMANCE ANALYSIS

To evaluate the candidate segments selection algorithm, we used two datasets, the University of Pittsburgh campus road map and the Allegheny County road map and built two clustering trees based on the maps. By simulating the movement of an object in different scenarios, the performance of the algorithm is analyzed in this section.

3.4.1 Datasets

We employed road maps in the Pittsburgh area from the US TIGER data files and collected some GPS positions on the campus of the University of Pittsburgh and tested our algorithm on different GPS locations, in order to evaluate it in various scenarios when an object was moving. Our algorithm was implemented using Matlab and tested on a PC machine with "Intel Core 2 2.13G HZ" CPU and 2GB memory.

3.4.2 Construction Cost

Memory usage and time complexity of constructing a clustering tree depend on the scale of a digital map. In order to save both space and time, our strategy was to split a large-scale digital map into a set of sub-maps and then build its corresponding clustering tree for each sub-map, so the original map corresponds to a forest structure. Indexing each tree in the forest is straightforward. As shown in Figure 3.4, we need intermediate memory to build matrices to calculate average distance of clusters, and the matrix before

compression is n by n, so the memory usage is proportional to n^2, where n is the number of intersections.

In this algorithm, average-link clustering merges the pair of clusters with the highest cohesion in each iteration. Based on this recursive computation of cohesion, the time complexity of average-link clustering is $O(n^2 \log n)$. However, Murtagh (1992) compared various hierarchical clustering approaches in computational time complexity and concluded that $O(n^2)$ time implementations exist for most of the widely known hierarchical clustering methods, and some methods can perform close to O(n) expected time for hierarchical clustering. Therefore, the construction cost of our hierarchical clustering tree can be further reduced by more sophisticated techniques. In the current implementation, a recursive approach is used to construct the hierarchical clustering tree. Table 1 provides features of the constructed clustering tree corresponding to two different map scales. Constrained by memory limitation in Matab, experiments are limited by the size of spatial networks. In spite of this, our algorithm can be expanded to any large scale map by splitting it into sub-maps and organizing it as a forest structure rather than a large tree structure.

A balanced binary tree has depth, $\log_2 n$, but hierarchical clustering trees that are built from spatial networks are not guarantee to be balanced. Therefore, maximum and minimum depths are recorded in analysis. Figure 3.6 shows the result of the tree construction by using the campus of the University of Pittsburgh as an example, which has the maximum depth of 14 and minimum depth of 10 as shown in the first row in Table 3.1.

Table 3.1. Tree features of three road networks

Road network	Road segments (leaf nodes)	Maximum depth of hierarchical clustering tree	Minimum depth of hierarchical clustering tree
Pittsburgh campus	171	14	10
Oakland area	1643	20	10

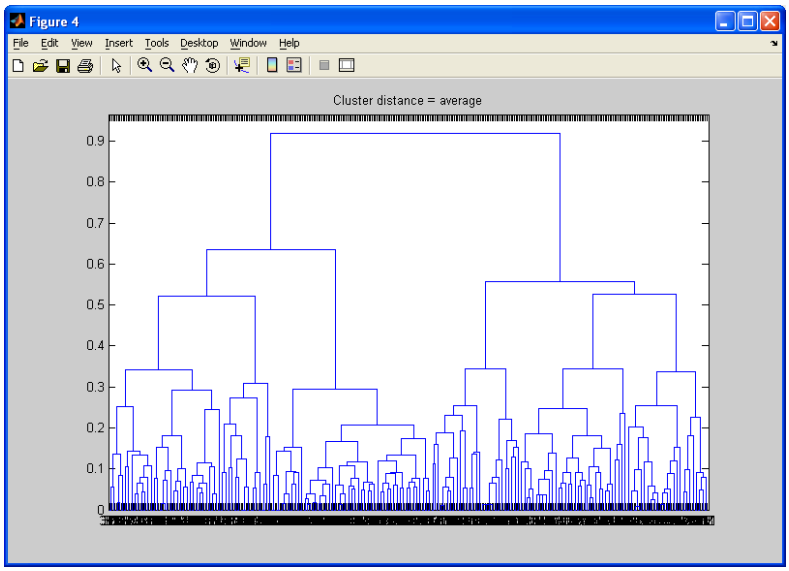


Figure 3.6. The clustering tree of Pittsburgh campus

3.4.3 Searching Cost

In theory, if n is the number of segments, then the time complexity of searching on binary trees is $O(\log_2 n)$. However, two factors influence query efficiency. One is the density of a spatial network and the other is changes to a moving object from an intersection to its linked segments. Because geometry and topology of spatial networks decide the clustering tree, once a spatial network is built, the structure of a clustering tree corresponding to the spatial network is fixed based on the clustering method. Therefore, given a spatial network, we mainly consider the query cost influenced by the position of GPS points.

As a user moves on the sidewalk, the sidewalk segments change. As the user approaches an intersection or move on a relatively short segment, a search window will cover more candidate segments than when the user is moving on a long segment. Query efficiency is worst when a user moves on a boundary between two large-scale clusters. Figure 3.7 shows how candidate segments change with different positions when a user is moving. The red square in each figure shows the received GPS position and the red points on the map show the intersections of candidate segments as the result of searching the clustering tree. Furthermore, by testing the same points in a larger area, (e.g., in Oakland) which covers the University of Pittsburgh campus, it can be seen that the connectivity of the spatial network determines clustered candidate segments, as shown in Figure 3.8. Since the average-linkage clustering approach is based on the average difference of groups, and the same area has a constant spatial network structure, the searching algorithm produces results with certainty. Given a GPS point, an experiment, in

Figure 3.8, shows that selected candidates in the large-scale map are consistent with the search results in small-scale map shown in Figure 3.7.

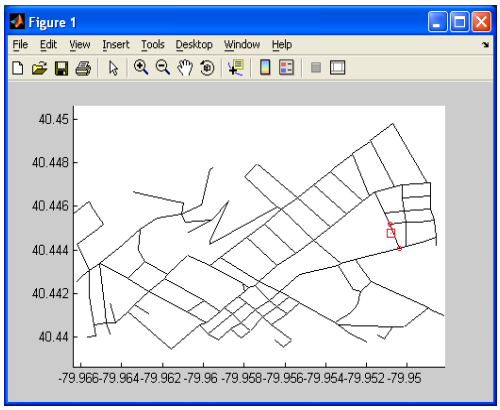


Figure 3.7-1. On a relative long road segment

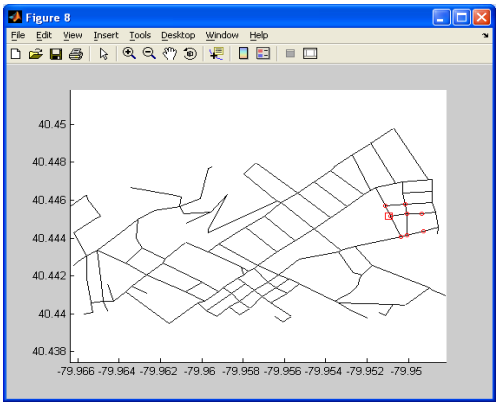


Figure 3.7-2. Approaching to an intersection

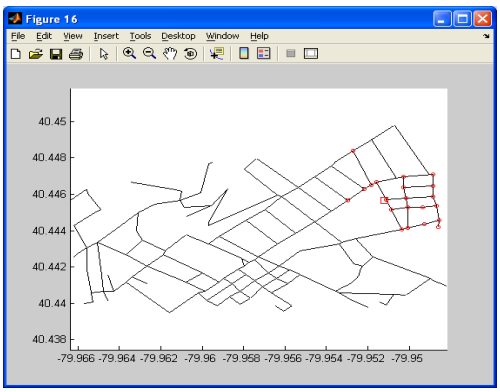


Figure 3.7-3 On another intersection

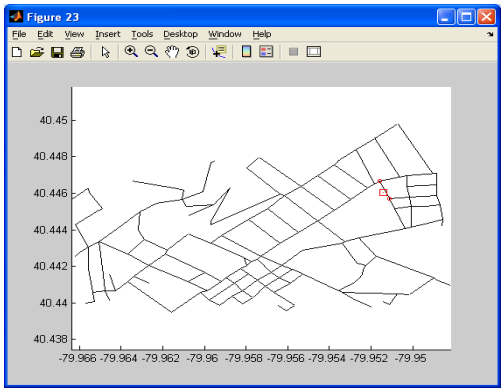


Figure 3.7-4 Middle of a segment

Figure 3.7. Query results changing with scenarios of moving object's positions

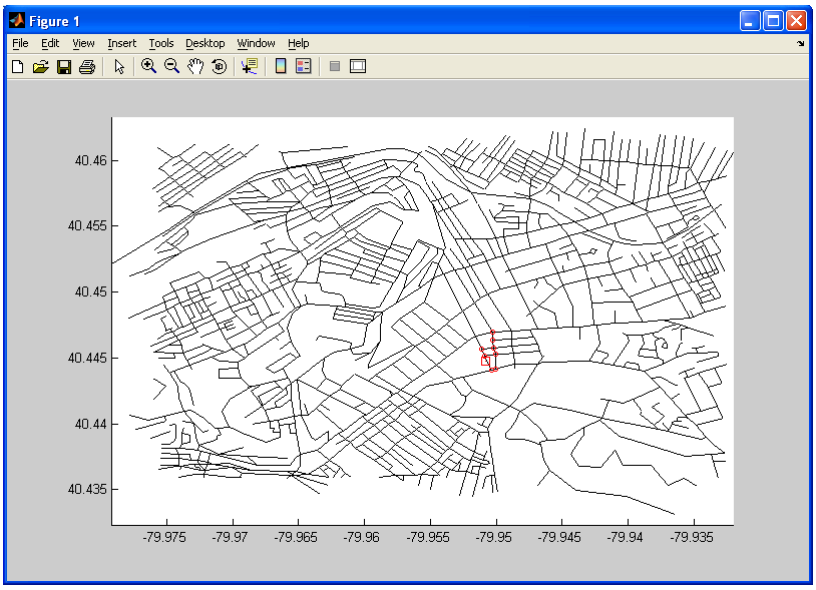


Figure 3.8. A scenario on a large-scale map

As discussed previously, in order to evaluate various map matching situations, experiments were conducted in three scenarios: approaching an intersection; moving on a relatively long segment; and moving around a boundary of two clusters. Table 3.2 shows the average searching cost for the three situations.

Table 3.2. Statistics of the searching cost

Road network	Maximum hierarchical level	Median search depth	Median search depth (Intersection)	Median search depth (on-segment)	Median search depth (boundary)
Pittsburgh campus	14	8	7	10	5
Oakland area	20	9	6	10	6

3.5 SUMMARY

This chapter presented a new algorithm to search for candidate segments given a GPS point for map matching. Rather than fixing a search window, this approach can provide an adaptive window based on obtained GPS positions and a chosen spatial network. Considering that the clustering technique can well present segment connectivity in a spatial network, a hierarchical clustering algorithm is developed to cluster segments. A binary tree is created after building the hierarchical clustering tree. Compared with multi-pass searching an object in indexing techniques, searching on the clustering tree of road segments requires only one pass. From the experiments, it is concluded that this approach can find candidate segments adaptively based on GPS positions and relative changes of relationship between GPS positions and clustered segments. The binary tree was designed to group segments and sped up the search time. As shown in Table 3.2, the algorithm can efficiently find road segments.

4.0 GPS-BASED MAP MATCHING TECHNIQUES

After a set of candidate segments is selected, how to find the correct segment is the key issue for map matching. Typically two main parameters, distance and direction deviation, are coupled to solve this problem.

Distance is defined as the length of the projected line from a GPS position to a link. Let C be the projection of P on a sidewalk segment AB. The distance is defined as equation 4.1 and visualized in Figure 4.1.

$$D(P, AB) = \begin{cases} D(P, C) & \text{if } C \in [AB] \\ \text{Min}\{D(P, A), D(P, B)\} & \text{elsewhere} \end{cases} \quad (4.1)$$

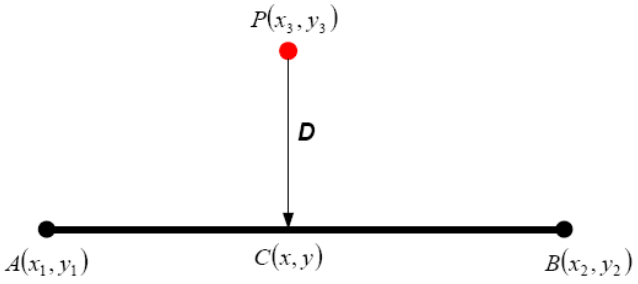


Figure 4.1. Perpendicular distance

Direction deviation is to measure the angular difference between user's trajectory and orientation of candidate segments. Direction of user's trajectory is the measurement of the angle between two or more successive positions.

Based on these two parameters, three advanced map matching algorithms solely using GPS data are presented in this chapter.

4.1 CHAIN-CODE-BASED MAP MATCHING

In this section, a chain-code-based map matching algorithm is designed by considering the unique characteristics in pedestrian/wheelchair navigation. The chain-code-based map matching approach was chosen because instead of computing the precise angle to represent a trend of movement, which is traditionally used, a discrete eight-direction chain code can be considered in order to reduce noise from GPS due to random movements of pedestrians or wheelchair users. Moreover, when pedestrians or wheelchair users move on sidewalks at relatively low speed, GPS data is often plagued with errors that frequently produce inaccurate trajectories. To overcome this problem, the chain-code-based map matching algorithm considers the trajectory of the data rather than merely the current position as in the typical map matching algorithms. Coupled with distance information, map matching decisions are made by comparing the differences between trajectories representing the road segments and GPS tracking data.

Instead of using precise angular, chain codes are defined to represent the direction of sidewalk segments and trajectory of user's movement. A directed straight line segment connecting two adjacent points is called a link, and a chain is defined as an ordered

sequence of links with possible interspersed codes. Chain encoding in this work is based on resolution of the direction to the adjacent links at the intersection.

Chain codes (also known as Freeman's codes) are a common technique to represent a contour of an object in image processing (Freeman and Saghri 1974, Haron et al. 2005). The directions of contour boundaries are coded with integer values k = 0, 1, . . . , K − 1 in the counterclockwise direction starting from the direction of the positive x-axis. A curve or contour is thus a chain of directions. The number of directions K takes is 2(M+1) where M is a positive integer, such as 4, 8. The chain codes where K > 8 are called generalized chain codes, like 16 (Freeman 1978).

4.1.1 Eight-Direction Chain Code

Since the angle between any two adjacent links on most intersections is usually greater than 45 °, we use an eight-direction Chain Code 0, 1, 2, 3, 4, 5, 6, 7 to represent eight direction interval on the counterclockwise direction as shown in Figure 4.2. With this definition, 0 corresponds to an angle between −22.5° to 22.5°, 1 corresponds to an angle between 22.5 °to 67.5 °, and 7 corresponds to angles between −67.5° to −22.5°.

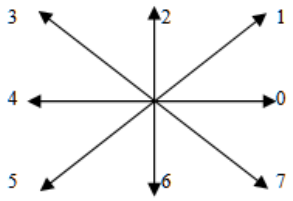


Figure 4.2. 8-Direction chain code

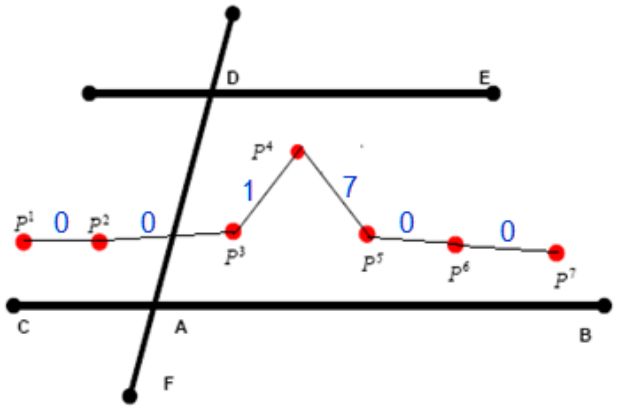


Figure 4.3. Digital map with GPS data

Figure 4.3 shows an example of GPS data and the sidewalk segment on a digital map. In this example, the GPS tracking route has a chain code of 0, 0, 1, 7, 0, 0, and the chain code of the sidewalk link C-A-B is 0,0 and F-A-D is 2,2.

Moreover, in order to find the closet segment to GPS tracking points, we use the difference between sidewalk link chain codes and the trajectory chain code of a user (Dcc) to show the extent of consistency of direction among them. With this, Dcc is defined as follows:

Δ = | Chain-Code (movement of a user) – Chain-Code (sidewalk segments) |

$$Dcc = \begin{cases} \Delta; & \text{if } \Delta<4 \\ (|\Delta - 8|) \bmod 4; & \text{otherwise} \end{cases} \qquad (4.2)$$

With this definition, discrete chain codes take the place of precise angle values; discrete Dcc takes the place of angle differences. This representation not only eliminates noise within short-distance moving, but also is computationally fast for real-time navigation.

Furthermore, as mentioned earlier, GPS position fixes are less reliable at a speed of less than 3.0 m/s. In such cases, in order to reduce the uncertainty of the direction under wheelchair users' control, the algorithm invokes a three-step Dcc between a user's trajectory and sidewalk segments rather than only taking a one-step Dcc.

In Figure 4, P^1, P^2, … , P^7 show the same GPS trajectory as the one in Figure 3. With a GPS data such as P^2, D_i (i = 1, 2, 3) could be calculated as the perpendicular distance to segments. When the first step is complete, the chain code from P^2 to P^3 is calculated, which is 0. Since heading directions are more meaningful than each-step directions, after two steps, the chain code from P^2 to P^4 is calculated, which is 1. Similarly, after three steps, the chain code from P^2 to P^5 is obtained as 0. Therefore, Dcc between P^2P^3 and CA is |0-0|; Dcc between P^2P^4 and CA is |1-0|, and Dcc between P^2P^5 and CA is |0-0|.

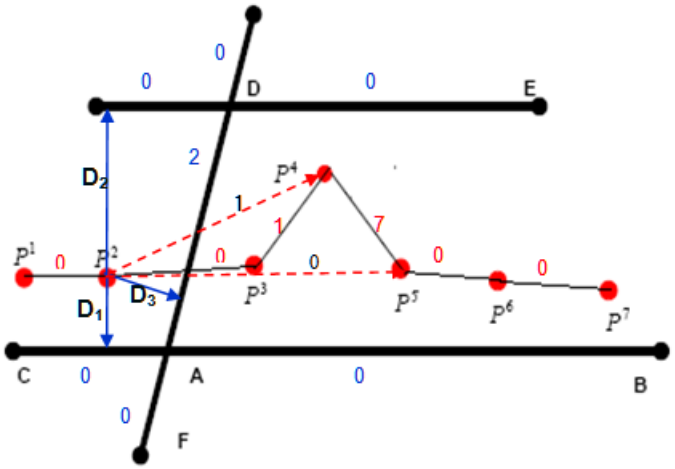


Figure 4.4. Example of chain-code-based map matching

4.1.2 Chain-Code-based Map Matching Technique

In this algorithm, distance and direction are coupled to find the best location of the user on the sidewalk. First, the algorithm uses perpendicular distances from GPS data to each sidewalk segment candidate and direction difference between user's trajectory and each sidewalk segment to select a sidewalk segment. As shown in Figure 4.4, in order to identify which segment a GPS point, such as P^2 in Figure 4.4, is most possibly mapped onto, both distance and direction movement are calculated to be weighted scores. All the segment candidates, close to P^2, are a link set {"CA", "DE", "AD"}; the distances from P^2 to these links are $\{D_1, D_2, D_3\}$, three-step Dcc calculations between trajectory and these segment candidates are taken as described in the previous section. Next, among all the segment candidates, the sidewalk segment with the highest matching evaluation will be chosen. In this case, segment "CA" would be considered the best selection based on map matching evaluation, and as a consequence, P^2 is determined to be projected to "CA". Two approaches are proposed to make map matching decision: linear model and non-linear model.

4.1.2.1 Linear Model

Distance and difference in direction are two determining factors to calculate matching results in order to identify correct segment. Linear model is built on the linear relationship between matching result (M) and evaluation parameters including Dcc and Distance. Figure 4.5 depicts the linear model.

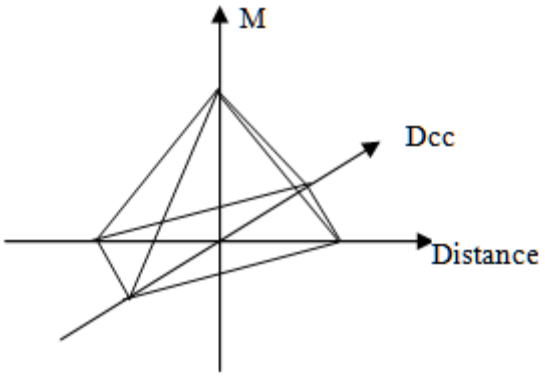


Figure 4.5. Linear model

The linear evaluation equation used in this work is as follows:

$$V_{ij} = W_{ij} * D_{ij} + \sum_{m=1}^{3} W_{ijm} * Dcc(Step[i+m], sidewalk\ segment[j]) \quad (4.3)$$

$$M_{ij}=1/V_{ij} \quad (4.4)$$

where i is the indexing number of GPS points, and j is the indexing number of segment candidates. D_{ij} is the distance from the ith GPS point to the jth segment; Dcc(Step[i+m], sidewalk segment[j]) is the three-step Dcc with m going from 1 to 3; W_{ij} is the weight of D_{ij} , and W_{ijm} is the weight of Dcc. For the ith GPS point, M_{ij} is used to calculate the total weight assigned to the jth candidate link. The link with the highest M_{ij} is selected as the correct link for GPS point i. Therefore, the larger M_{ij}, the smaller is V_{ij}. In Equation (4.3), the total weighting score can then be obtained by summing up the individual scores, including weighting distance and three-step weighting Dcc as shown in Figure 4.5.

4.1.2.2 Non-Linear Model

Based on its definition in Section 4.1.1, Dcc is a discrete value ranging from 0 to 4; whereas, the absolute value of distance is a continuous real number from 0 to a large

number. Non-linear model provides an alternative means to fit the non-linear relationship between the combination of these two input parameters and matching result to make map matching decision, which is depicted in Figure 4.6.

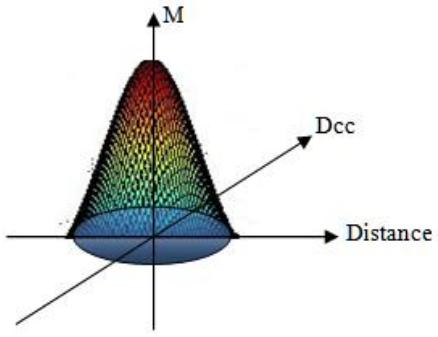


Figure 4.6. Non-linear model

In this non-linear model, evaluation estimation is a fitting curve procedure between response variable (matching result) and a list of input parameters (three-step Dcc and Distance). Among the most common nonlinear models, neural network is a widely used approach. This approach attempts to find a relationship, i.e., a function, between the inputs, and the provided output(s), in order for the network to find a correct answer for the new inputs when network is provided with unseen inputs. As one of various structures in neural network family, radial basis function (RBF) networks (Howlett et al. 2001) have static Gaussian function as the nonlinearity for the hidden layer processing elements, so the network could provide a good non-linear transformation in this map matching algorithm for each input vector, distance and three-step Dcc, to obtain non-linear map matching result (M). The Gaussian function responds only to a small region of the input

space where the Gaussian is centered. Therefore, in this map matching algorithm, the link which provides the highest output of Gaussian function, i.e., M_{ij}, is chosen as the correct link for that positioning fix.

4.1.2.2.1 Radial Basis Functional Neural Network

The structure of RBF networks usually has three layers. Each hidden unit in the network has two parameters: a center u_j and a width σ_j associated with it. The output of each hidden unit depends only on the radial distance between the input vector and the center parameter for that hidden unit. The response of each hidden unit is scaled by its connecting weights W_{kj} to the output units and then summed to produce the overall network output:

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} W_{kj}\theta_j(\mathbf{x}) + W_{ko} \qquad (4.5)$$

where the Gaussian activation function for RBF networks is given by:

$$\theta_j(\mathbf{x}) = \exp\left(-\frac{\| \mathbf{x} - \mathbf{u}_j \|^2}{2\sigma_j^2}\right) \qquad (4.6)$$

and x is the d dimensional input vector with elements x_i, and u_j is the vector determining the center of the basis function; $y_k(x)$ is the output of RBF neural network.

There are two steps to train a RBF neural network: (1) determine the parameters of the basis functions through unsupervised training using only the input data set and (2) determine weights W_{kj} using both input and output data (hidden units are activated using an input pattern and the weights to the output layer are then modified to produce the desired output for the given input). Once all the parameters are produced by training in a RBF network, the neural network model could be used to compute the output based on new input data.

4.1.2.2.2 Design of RBF Neural Network

In this research, the RBF Neural Network is designed as a supervised network. The input data has four features: distance and the three-step Dcc, so the input layer has four neurons, the output layer has one neuron which is used to evaluate the extent to which a GPS point is close to candidate segments. The first step is to train the neural network. The training data consists of many pairs of input and output. Many pairs of Distance and three-step Dcc are calculated as input vectors, and output values are designated as 0 or 1, depending on whether the calculated candidate segment is the correct link or not. Next, the trained neural network is used to perform the non-linear map matching evaluation. Given a 4-d input vector with distance and three-step Dcc, the segment with the smallest output value is the selected segment, where the output of evaluation, $M_{ij,}$ obtains the largest value. Based on the size of sample data in experiments, the hidden layer is designed as a 132-neuron layer. The structure of the RBF neural network for this map matching algorithm is shown in Figure 4.7.

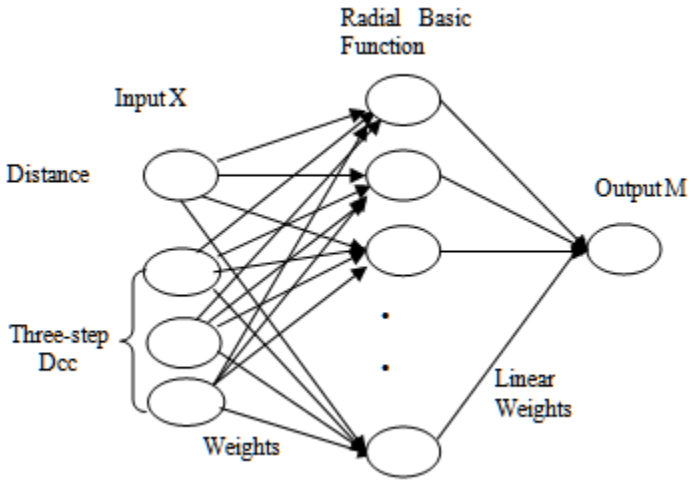


Figure 4.7. RBF neural network for map matching evaluation

By definition in Equations (4.5) and (4.6), the corresponding non-linear evaluation equation used in this work is defined as follows:

$$M_{ij} = \sum_{k=0}^{n} \alpha_k * \exp\left(-\frac{\beta \| x_{ij} - u_k \|^2}{2\sigma_k^2}\right) \quad (4.7)$$

$$V_{ij} = 1 / M_{ij}$$

$$x_{ij} = \begin{bmatrix} D_{ij} \\ Dcc(Step[i+1], sidewalk\ segment[j]) \\ Dcc(Step[i+2], sidewalk\ segment[j]) \\ Dcc(Step[i+3], sidewalk\ segment[j]) \end{bmatrix} \quad (4.8)$$

where i is the indexing number of GPS points, and j is the indexing number of segment candidates. X_{ij} is input, in which D_{ij} represents distance from the ith GPS point to the jth segment, and Dcc(Step[i+m], sidewalk segment[j]) represents three-step Dcc with m going from 1 to 3. α_k is the weight of the kth output value in hidden layer. β is the weight matrix of input vector in hidden layer. As the output, M_{ij} is used to calculate the total weight assigned to the jth candidate link for the ith GPS point. The link with the smallest V_{ij} is viewed as the correct link.

4.1.3 Map Matching Process

There are two main modes in the map matching process: (1) turning mode and (2) normal moving mode. In turning mode, the algorithm performs map matching for each GPS tracking point around intersections in order to identify a new segment. Once the user is located on a segment, the process enters normal moving mode, current and previous positions can be used as constraints for the next step in map matching based on the topology of sidewalk networks. Such constraints accelerate the matching process.

Once the correct sidewalk segment is determined, i.e., the first step of map matching, finding an estimate of the location of the user on that segment is straightforward.

The flowchart shown in Figure 4.8 describes the process of chain-code-based map matching, including pre-processing to initialize data structures, evaluation based on three-step Dcc and distance between sidewalk segments and trajectory made of GPS points, and a constraint on a selected sidewalk segment.

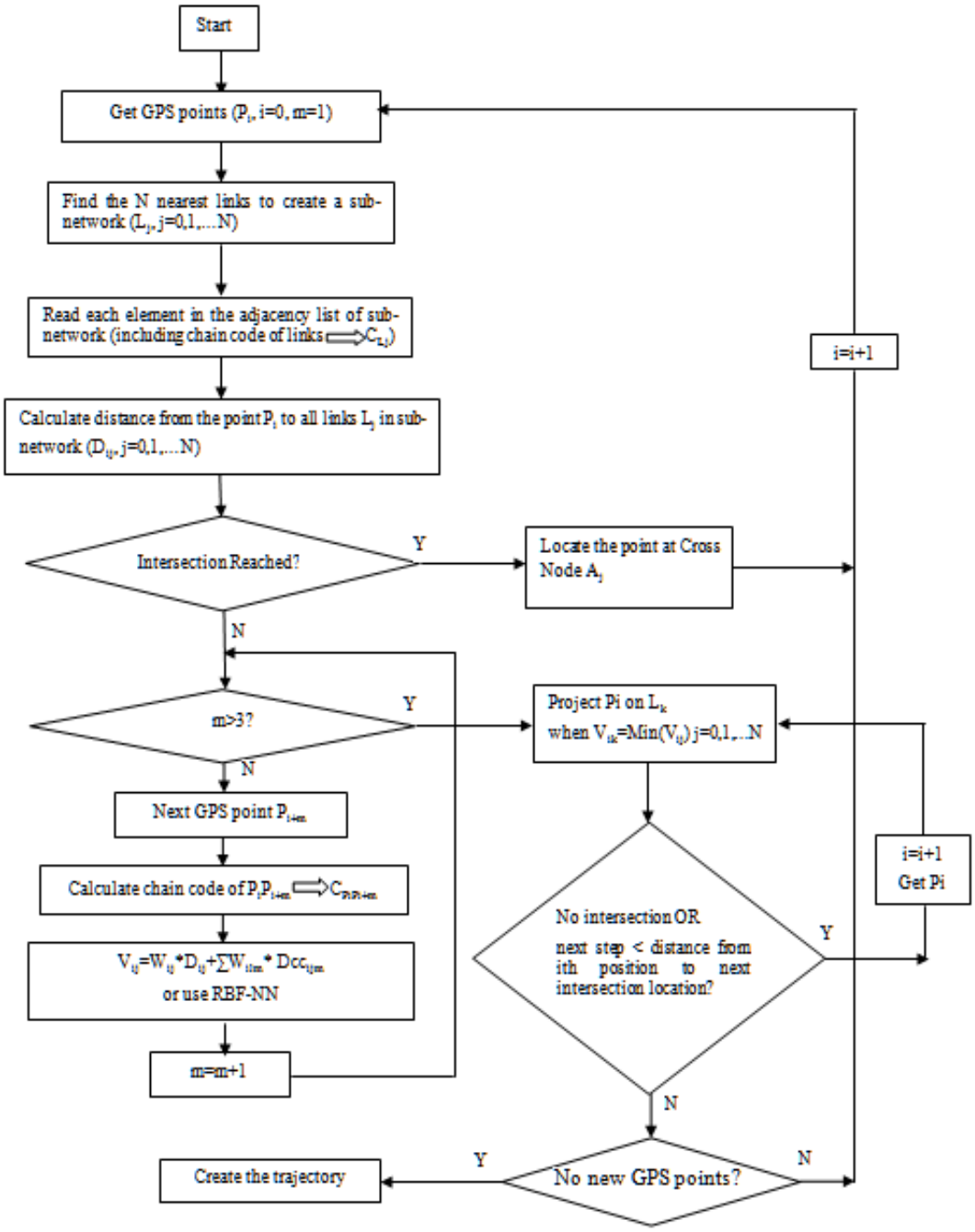


Figure 4.8. Flowchart of chain-code-based map-matching algorithm

4.1.4 Validation

To validate the chain-code-based map matching algorithm, the process of chain-code-based map matching is implemented and tested on a sidewalk network.

4.1.4.1 Test Environment

The sidewalk data along with associated parameters on the University of Pittsburgh campus area were digitized and utilized by scale 1:2500. The sidewalk database, consisting of the sidewalk network, buildings, landmarks, and accessibility information, are built for wheelchair navigation in order to assist wheelchair users' outdoor traveling (Karimi et. al 2006). In the following testing, GPS points in three routes were collected by walking and using a stand-alone GPS receiver, Trimble GeoExplorer 3, and map matched on the established sidewalk network. Fully considering the various types of sidewalks on the different areas, we used three selected routes to test map matching algorithms. Route 1 covered the most main sidewalks on campus; Route 2 included the sidewalks around tall buildings; Route 3 included a loop and some small paths.

A snapshot of the digitized sidewalk map overlaid with Google map shows the campus of the University of Pittsburgh in Figure 4.9. The computing platform used was a PC machine with "Intel Core 2 1.4G Hz" CPU. The software for the fuzzy logic map matching algorithm was written in JAVA in an open source GIS tool called Geotools (www.geotools.org).

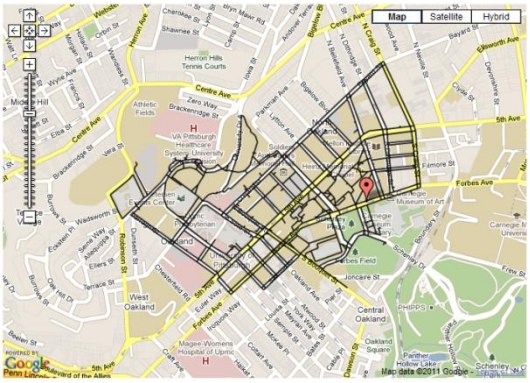


Figure 4.9. University of Pittsburgh's campus

4.1.4.2 Evaluation of Linear Map Matching Models

For evaluation purposes, we employed different forms of linear and non-linear models in matching evaluation.

As discussed in Section 3.3.1, the distance, three Dcc after step one, two, three are the four influencing factors in matching evaluation equation. Their linear relationship was formulated as Equation (4.9). After normalization for these four variables, four estimated weight parameters meet the condition as follows:

$$W_{ij} + \sum_{m=1}^{3} W_{ilm} = 1 \qquad (4.9)$$

After testing different combinations, the four weight parameters are given as:

$$\begin{cases} W_{ij} = 0.5; \\ W_{ij1} = 0.1; \\ W_{ij2} = 0.2; \\ W_{ij3} = 0.2; \end{cases} \qquad (4.10)$$

4.1.4.3 Evaluation of Non-Linear Map Matching Models

We used RBF neural network to build a non-linear evaluation model. As mentioned in Section 3.3.2, the RBF neural network considered in this work has a three-layer structure, including 4-d input layer, 132-d hidden layer and 1-d output layer. In the training stage, 132 pairs of parameters, with input vectors and output values, are calculated and normalized as a sample data set for training RBF neural network. Figure 4.10 shows the training performance.

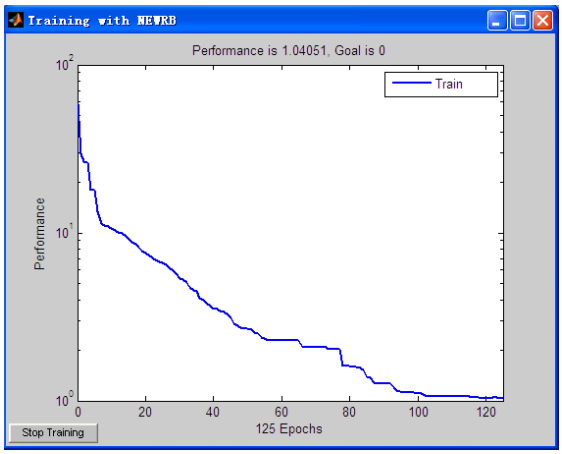


Figure 4.10. Training with RBF neural network

The curve in Figure 10 shows that training is completed after several iterations, when error gets close to 0. This means that we could use this trained model to perform map matching evaluation.

Table 4.1 shows the trained network structure. Input vector is defined as a 4-d vector including distance, Dcc (after step1), Dcc (step2), and Dcc (step3). Output of the

trained neural network is the evaluation result. The candidate segment with the minimum output value is viewed as the matched segment. Table 4.2 shows different output values according to different input vectors for a GPS point and all relevant segment candidates. It could be drawn that the first pair of input parameters results in the smallest output value in the list, which means that the corresponding segment provides the closest match to the GPS data, and thus will be identified as a selected link for the GPS data.

Table 4.1. RBF neural network structure

Inputs	**hidden layer**	**output layer**	**Weights from input layer to hidden layer**	**Weights from hidden layer to output layer**
size: 4	size: 132	Size: 1	4*132	132 *1

Table 4.2. Map matching evaluation using RBF neural network

Input[0] Distance	**Input[1] Dcc[step1]**	**Input[2] Dcc[step2]**	**Input[3] Dcc[step3]**	**output**
0.0116309625	0	0	0	0.0068
0.1534199589	0	0	0	0.3967
0.0482950975	2	2	2	1.3588
0.2902582818	1	2	2	13.221
0.4500096318	3	4	4	542.71
1.0987441382	1	1	1	874.33

4.1.4.4 Performance Analysis

Two groups of tests were conducted to evaluate the performance of the chain-code-based map matching algorithm.

First, linear map matching approaches were performed to compare results with

constraints and without constraints. The linear model is used to test the map matching algorithm on three routes. The map matching performances are presented in Table 4.3 where five statistic values are listed on the three chosen routes. The results of map matching with constraints, compared with GPS raw data, are shown in Figures 4.11 - 4.13.

Table 4.3. Linear-model map matching results

Testing Environment	Total Number of GPS Points	Map Matching Without Constraints		Map Matching With Constraints	
		Correct Link Identification (%)	Average Time/Point (sec)	Correct Link Identification (%)	Average Time/Point (sec)
Route 1 in Urban Area	682	93.2%	0.025	93.4%	0.0044
Route 2 in Urban Area	1517	94.6%	0.026	95.4%	0.0043
Route 3 in Urban Area	933	89.6%	0.026	89.8%	0.0046

a. GPS data before map matching

b. The result of map matching with constraints using the linear model

Figure 4.11. Route 1 comparing map-matching result with GPS raw data on campus sidewalk map

a. GPS data before map matching

b. The result of map matching with constraints using the linear model

Figure 4.12. Route 2 comparing map-matching result with GPS raw data on campus sidewalk map

a. GPS data before map matching

b. The result of map matching with constraints using the linear model

Figure 4.13. Route 3 comparing map-matching result with GPS raw data on campus sidewalk map

Second, linear models and non-linear models are separately applied to analyze their performances in making map matching decisions. As Table 4.4 shows, the result leads to the conclusion that linear evaluation is more advantageous to implement than non-linear evaluation in terms of time performance. Compared to the linear model, map matching results in the non-linear model showed slightly lower correct link identification. Therefore, in order to meet the need of real-time navigation, linear model is preferred model for map matching decision.

Table 4.4. Comparing the linear model and the non-linear model

Scenarios	The Linear Model With Constraints		The Non-Linear Model With Constraints	
	Correct Link Identification (%)	Average Time/Point (sec)	Correct Link Identification (%)	Average Time/Point (sec)
Route 1	93.4%	0.0044	93.3%	0.017
Route 2	95.4%	0.0043	94.9%	0.02
Route 3	89.8%	0.0046	88.5%	0.019

In summary, experimental results show that most mismatched points in Route 3 occur when a user moved on paths with no corresponding segments on the digital map. In the case of Route 2, most mismatched points occur on sidewalks of narrow roads due to GPS errors, where two sides of some narrow roads cannot be distinguished since their distances are below GPS error range generally announced as 10m radiuses or over.

4.2 HMM-BASED MAP MATCHING ALGORITHM

The Hidden Markov Model is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden (unknown) parameters from the observable parameters (Wikipedia, Yariv 2002). The HMM has been used in temporal recognition applications such as text and speech recognition. We argue that map matching is also a temporal recognition application susceptible to a Markov process where the aim is to find actual paths and

actual locations, i.e., the hidden information, using GPS data as observed measurements. Map matching, as a time-series problem, resembles temporal pattern recognition applications, such as speech, handwriting, gesture recognition and bioinformatics, where the hidden Markov model is applied. With these characteristics, the HMM-based map matching approach , where finding the correct sidewalk segment out of all candidate sidewalk segments given a GPS trajectory, was chosen.

The Viterbi Algorithm (Forney 1973) is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process. Many problems can be cast in this form. We applied the Viterbi algorithm to estimating the sidewalk segments based on observed GPS positions. The key innovation using HMM in this algorithm for wheelchair navigation is matching sidewalk segments based not only on the geometry of the location readings, but additionally on the topology of the segments.

4.2.1 Hidden Markov Model

The HMM is represented by a finite set of states, each of which is associated with a probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state, that is visible to an external observer and therefore states are ``hidden" to the outside; hence the name Hidden Markov Model (Rabiner 1998).

The general architecture of a hidden Markov model is shown as Figure 4.14.

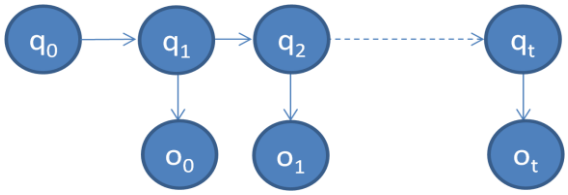


Figure 4.14. Architecture of a HMM

The architecture has two layers: $\{o_t\}$ represents the observable layer and o_t corresponds to an observation value at time t. $\{q_t\}$ represents the hidden layer, and q_t, at time t, comes from one state in a state space.

In order to model a hidden Markov process, the following elements are needed:

- The number of states in the model, n.
- The number of observations, m. If the observations are continuous then m is infinite.
- A set of state transition probabilities. $A=\{a_{ij}\}$

$$a_{ij} = p_r \{ q_{t+1} = j \mid q_t = i \}, 1 \leq i, j \leq n \quad (4.11)$$

where q_t denotes the state at time t.

- An observation probability distribution in each of the states, $B=\{b_j(k)\}$.

$$b_j(k) = p_r \{ o_t = o_k \mid q_t = j \}, 1 \leq j \leq n, 1 \leq k \leq m \quad (4.12)$$

where o_t is the observation at time t and o_k denotes the k_{th} observation.

- The initial state distribution, $\pi = \{\pi_i\}$, where,

$$\pi_i = p_r \{ q_1 = i \}, 1 \leq i \leq n \quad (4.13)$$

With these, $\lambda = (\pi, A, B)$ can be used to denote an HMM with probability distributions.

4.2.2 A Hidden Markov Model for Map Matching

In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible observations. Therefore, the sequence of observations provides some information about the sequence of states by means of a HMM (Olivier 2005).

Given the parameters of the model, the Viterbi algorithm can solve the problem of how to find the most likely sequence of hidden states that could have generated by using a given observed sequence. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states, called the Viterbi path.

In map matching for wheelchair navigation, observed GPS points are the visible observation layer and correct sidewalk segments are the invisible state layer.

Let $Pt \in \{p_1, p_2, \ldots, p_m\}$ denote the observation (i.e., a GPS data point) obtained every second t for $1 \leq t \leq m$.

Let $Rt \in \{ r_1, r_2, \ldots, r_n \}$ denote the actual location (i.e., the correct sidewalk segment) at time t.

Suppose we obtain a series of GPS observations within the time period m, so we could obtain m GPS points as an observation sequence from time t_1 to time t_m. In the state space, there are n states, which represent n candidate segments. The transition probability from any time i to the next time j represents the probability of a user's moving from one segment to another segment. The model could be structured as shown in Figure 4.15. The goal is to find the sidewalk segment sequence that has maximum probability given the

observations. That is finding a sequence of actual locations, R1 . . .Rt, such that Pr(R1R2 . . .Rt|P1P2 . . .Pt) is maximized.

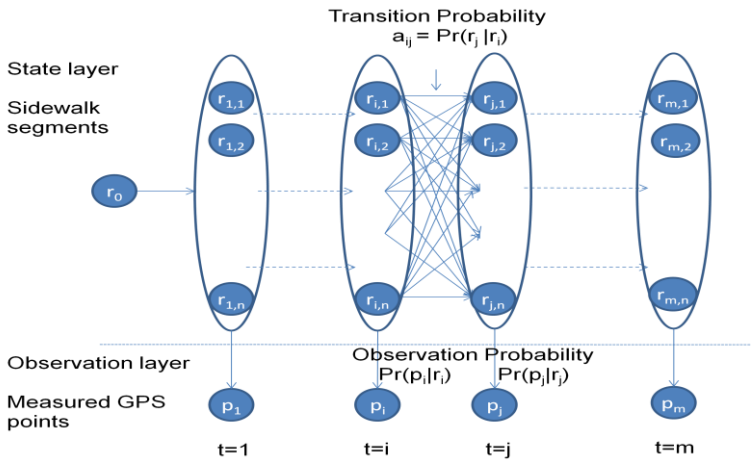


Figure 4.15. The hidden Markov model for map matching

Based on conditional probabilities from basic probability theory, for any sequence $R_1 \ldots R_t$ of actual locations we have:

$$\Pr(R_1R_2 \ldots R_t | P_1P_2 \ldots P_t) = \frac{\Pr(R_1R_2 \ldots R_tP_1P_2 \ldots P_t)}{\Pr(P_1P_2 \ldots P_t)}$$

(4.14)

Given the observations, the denominator of this expression is determined (the exact value is unknown, but that value only depends on the observations, not on the path $R_1 \ldots R_t$.). So the problem is equivalent to finding $R_1 \ldots R_t$ such that $\Pr(R_1R_2 \ldots R_tP_1P_2 \ldots P_t)$ is maximized.

From the basic identities of probability theory, for any events A,B,C we have, $Pr(ABC) = Pr(A)Pr(B|A)Pr(C|AB)$. Let's use this to decompose the complicated event:

$R_1R_2 \ldots R_tP_1P_2 \ldots P_t$ as a product ABC. We define $A = R_1 \ldots R_{t-1}P_1 \ldots P_{t-1}$, $B = R_t$, $C = P_t$,

By applying the above formula, we obtain:

$Pr(R_1R_2 \ldots R_tP_1P_2 \ldots P_t)=Pr(R_1R_2 \ldots R_{t-1}P_1P_2 \ldots P_{t-1})Pr(R_t|R_1 \ldots R_{t-1}P_1 \ldots P_{t-1})Pr(P_t|R_1 \ldots R_{t-1}P_1 \ldots P_{t-1}R_t)$.

Furthermore, we obtain:

$$Pr(R_1 \ldots R_tP_1 \ldots P_t) = Pr(R_1 \ldots R_{t-1}P_1 \ldots P_{t-1})Pr(R_{t-1} \rightarrow R_t)Pr(R_t \rightarrow P_t)$$

$$= Pr(R_0) \prod_{t=0}^{T} Pr(R_t|R_{t-1}) \prod_{t=0}^{T} Pr(P_t|R_t) . \qquad (4.15)$$

We assume each probability in the state transition matrix and in the observation probability matrix in the HMM is time independent. Therefore, given prior probability $Pr(R_0)$, observation probability $Pr(P_t \mid R_t)$ and state transition probability $Pr(R_t \mid R_{t-1})$, we can use the Viterbi algorithm to find the path through the states that maximizes the probability of a sequence of sidewalk segments. The Viterbi algorithm uses dynamic programming methods to efficiently accomplish this, so that the actual path consisting of a sequence of sidewalk segments can be identified.

In Equation (4.15), in order to apply the Viterbi algorithm, we need to know prior probability, observation probability and state transition probability in the HMM. Prior probability $Pr(R_0)$ is $Pr(r_j)$, when $j=1,\ldots,n$, which is simply computed by 1/n as a uniform distribution reflecting the fact that we have no known bias about which is the correct sidewalk segment. Hence, how to compute observation probability and state transition probability becomes the key point.

First, we compute the observation probability, which is the probability of the measured location p_i given r_j. We can compute this with the Bayesian rule:

$$P_r(p_i|r_j) = \frac{P_r(r_j|p_i)Pr(p_i)}{\sum_{k=1}^{n} Pr(r_k|p_i)Pr(p_i)} \qquad i=1,2\ldots,m; \quad j=1,2\ldots,n \qquad (4.16)$$

We presume that$Pr(pi)$, a prior probability in Equation (4.16), follows a uniform distribution. Therefore, equation (4.16) could be further simplified as:

$$P_r(p_i|r_j) = \frac{P_r(r_j|p_i)}{\sum_{k=1}^{n} P_r(r_k|p_i)} \qquad i=1,2\ldots,m; \quad j=1,2\ldots,n \qquad (4.17)$$

$Pr(r_j|p_i)$ is the probability that rj is the correct sidewalk segment out of the candidate sidewalk segments given that measured location is Pi. We computed this by assuming that, for most of the GPS points, the closer a sidewalk segment is to the observed point, the higher the probabilities that it is the correct segment. This is borne out by our informal observations of nearest segment matching. Considering the relationship of distance and observation probability as an inverse proportion, we first compute the probability of the perpendicular distance from GPS point pi to the segment r_j over the summation of the distances from p_i to all the candidate segments, and then use reciprocal relation of the probability based on distances to approximate observation probability. This leads to:

$$P_r(r_j|p_i) = \frac{Expected\ \{number\ of\ times\ in\ segment\ r_j\ |GPS\ measured\ location\ p_i\}}{Expected\ \{all\ the\ times\ in\ all\ possible\ segments\ r_1,\ldots,r_n\ |GPS\ measured\ location\ p_i\}}$$

$$= 1 \Big/ \frac{\text{Distance from } p_i \text{ to } r_j}{\sum_{k=1}^{n} \text{Distance from } p_i \text{ to } r_k} \qquad (4.18)$$

In pedestrian/wheelchair navigation, users either move on the same segment, or they make a turn at a junction such as an intersection, exit, or entrance. Therefore, we need to compute the transition probability a_{ij}, which represents the probability of the user moving from one sidewalk segment corresponding to a measured point to another

sidewalk segment corresponding to another measured point. For this, in this algorithm we use topological relationship to compute the transition probability. We only consider three topological relationships: same segment, connected segment and unconnected segment. It is impossible for a wheelchair user to move from a segment to an unconnected segment in consecutive time windows. Therefore, the transition probability from time i to the next time j=i+1, a_{ij} would be zero where the two segments are not connected. If two sidewalk segments are connected, this transition probability should be higher than if two sidewalk segments are unconnected, since wheelchair users would travel on the same segment most of the time except at an intersection or junction. Thus, the transition probability of moving on the same segment has the highest value. By setting $a_{ij} = e^{-rij}$, we create an exponential curve for this probability distribution, where r_{ij} corresponds to the topological relationship between two segments. By normalization, a_{ij} changes between 0 and 1. The next important step is to build a transition matrix { r_{ij} } and set the value for each element in this matrix. The following set of rules must be followed

(1) If two segments are connected, r_{ij} is set to 1;

(2) If they are unconnected, then r_{ij} is set to ∞.

(3) Otherwise, r_{ij} is 0, when two segments are same segment that is i=j.

Take our measured GPS points on campus as an example, shown in Figure 4.16. First, we model sidewalk segments as a set $\{r_1, r_2, \ldots, r_{12}\}$ as Figure 4.17 shows. Next, we build a matrix $\{r_{ij}\}$, based on the topology of the segments in Figure 4.18. Figure 4.19 shows the map matching results by applying the Viterbi algorithm to the entire sequence of location measurements shown in Figure 4.16.

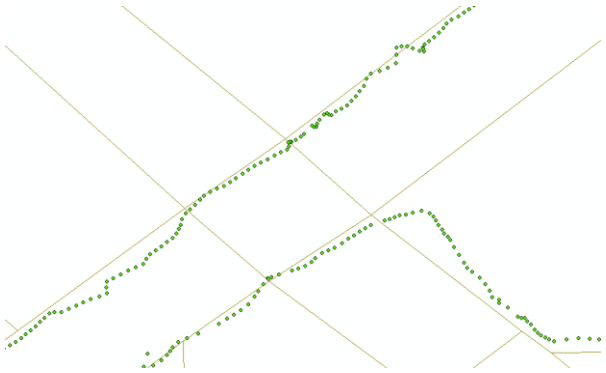


Figure 4.16. An example of GPS points overlaid on sidewalks on campus

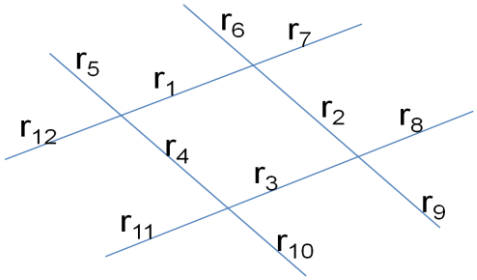


Figure 4.17. An abstracted sidewalk network model

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}
r_1	0	1	∞	1	1	1	1	∞	∞	∞	∞	1
r_2	1	0	1	∞	∞	1	1	1	1	∞	∞	∞
r_3	∞	1	0	1	∞	∞	∞	1	1	1	1	∞
r_4	1	∞	1	0	1	∞	∞	∞	∞	1	1	1
r_5	1	∞	∞	1	0	∞	∞	∞	∞	∞	∞	1
r_6	1	1	∞	∞	∞	0	1	∞	∞	∞	∞	∞
r_7	1	1	∞	∞	∞	1	0	∞	∞	∞	∞	∞
r_8	∞	1	1	∞	∞	∞	∞	0	1	∞	∞	∞
r_9	∞	1	1	∞	∞	∞	∞	1	0	∞	∞	∞
r_{10}	∞	∞	1	1	∞	∞	∞	∞	∞	0	1	∞
r_{11}	∞	∞	1	1	∞	∞	∞	∞	∞	1	0	∞
r_{12}	1	∞	∞	1	1	∞	∞	∞	∞	∞	∞	0

Figure 4.18. State transition matrix

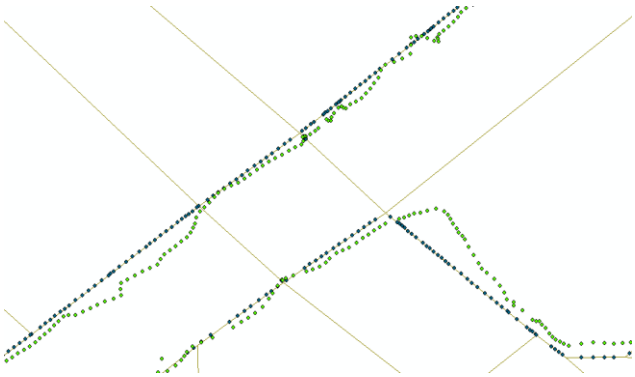


Figure 4.19. Map matching locations versus GPS positions

4.2.3 HMM-based Map Matching Process

For a hidden Markov model, two parameters, n and m, have to be initialized, where m is the size of an observation sequence and n is the state number in a state space. For map matching, the size of the observation sequence is the number of measured GPS points, and the state number is the number of candidate sidewalk segments close to the observed GPS points. In this algorithm, after setting these two values, we take several steps to complete the matching process.

First, a set of nearby candidate sidewalk segments is chosen based on the first GPS data observed in each sequence. Second, the transition matrix on the selected set of nearby candidate sidewalk segments is built (see Figure 4.18). This matrix not only shows the topology of segments but implies two moving modes, which are changing mode and continuing on same segment mode. In the case of continuing on same segment mode, where r_{ij} equals to 0, current and previous positions should be matched on the

same segment. Conversely, if r_{ij} is 1, then the wheelchair is moving in a changing mode, where current and previous positions are on two connected segments. Consequently, we could compute transition probabilities based on the transition matrix. Third, the perpendicular distance from each GPS point to each segment in the set of candidate sidewalk segments is computed, so that observation probabilities for each measured location are calculated. Last, the Viterbi algorithm to the observation probabilities and transition probabilities to compute the maximum probability sequence of sidewalk segments are applied. Once the most likely sidewalk segment is obtained, GPS points are projected to the segments and the map matching result is shown on the map.

Figure 4.20 is the flowchart of HMM-based map matching.

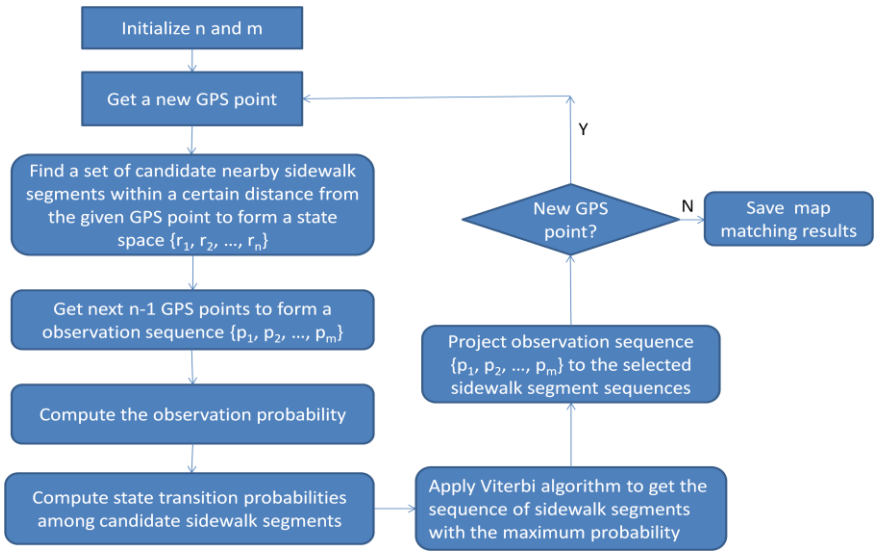


Figure 4.20. Flowchart of HMM-based map matching process

4.2.4 VALIDATION

To validate the HMM-based map matching algorithm for pedestrian/wheelchair navigation, the same GPS data collection on three routes, the same sidewalk map and the testing environment that we described in section 4.1 are applied for evaluation of this algorithm.

4.2.4.1 Performance Analysis

Three groups of data sets, collected on main campus of University of Pittsburgh by GPS receiver, were processed to validate the presented algorithm. the HMM-based map matching algorithm. For contrast, three-route GPS raw data with map matching results were overlapped on campus sidewalk map, shown in Figures 4.21, 4.22, and 4.23.

a. GPS raw data overlapped on campus sidewalk map

b. Projected result data to the matched sidewalk segments on campus sidewalk map

Figure 4.21. Route 1 comparing map-matching result with GPS raw data on campus sidewalk map

a. GPS raw data overlapped on campus sidewalk map

b. Projected result data to the matched sidewalk segments on campus sidewalk map

Figure 4.22. Route 2 comparing map-matching result with GPS raw data on campus sidewalk map

a. GPS raw data overlapped on campus sidewalk map

b. Projected result data to the matched sidewalk segments on campus sidewalk map

Figure 4.23. Route 3 comparing map-matching result with GPS raw data on campus sidewalk map

Like all map matching algorithms, there are mismatched points due to errors in geo-positioning systems and the digital map quality, both affect the performance of the map matching algorithm. We observe that most mismatched points in Route 3 occur when the data collector moved on paths with no corresponding segments on the digital map. Meanwhile, in the case of Route 1 and Route 2, we notice that many mismatched points occur on sidewalks of narrow roads due to GPS errors.

Model parameters, n and m were specified through experiments. Based on the topology extracted from the campus sidewalk data, the size of the state space, i.e., the number of segment candidates in one map matching process, notated by n, was set as twelve. Experiments using 3- to 8-point sequence were conducted to determine the suitable number of points in one sequence. It was realized that for the real-time requirement of map matching a 4-point sequence is appropriate for this HMM-based map matching algorithm. The map matching performances are presented in Table 4.5.

Table 4.5. Performance results

Example	Total Number of GPS Points	Number of Mismatched Points	Correct Link Identification After HMM Correction(%)	Totoal Computation Time (s)	Average Time/4 Points (ms)
Route 1	682	52	92.4%	0.625	3.666
Route 2	1516	70	96%	1.406	3.709
Route 3	933	68	92.7%	0.859	3.682

The average time per 4 points represents the average time taken on one sequence's matching computation. In the offline matching, the total computation time shows the total time to complete the matches of all GPS points in one route. Since correct link identification after applying a map matching algorithm and average computation time are the most important performance parameters for evaluation, statistical data shows that this algorithm performs well and satisfies the requirements of real-time map matching in wheelchair navigation.

4.3 FUZZY-LOGIC-BASED MAP MATCHING ALGORITHM

A Fuzzy-logic map matching algorithm is presented in this section for pedestrian/wheelchair navigation. Fuzzy logic, based on fuzzy reasoning concepts, is one of the most widely used computational methods. It is good at solving problems in many circumstances where uncertainties are difficult to model. Fuzzy logic can take noisy, imprecise input, to yield crisp (i.e. numerically accurate) output. The fuzzy-logic map matching was chosen because GPS data used for localization of pedestrians or wheelchair users contain uncertainties.

4.3.1 Fuzzy Logic Map Matching

Fuzzy logic is a computing approach based on "degrees of truth", a range of values from “true” to “false” that is used in decision making with imprecise data. Fuzzy logic is not any less precise than any other form of logic; it is a mathematical method for handling inherently imprecise concepts.

A typical fuzzy logic inference system starts with the fuzzification of inputs and outputs, and executes rule-based inference, and ends with the defuzzification to obtain its output. In the fuzzification step, the values of the input variables are converted into degrees of membership for the membership functions defined on the variable. The inference of the fuzzy logic system is built by using rule-based fuzzy reasoning. The last step is called the defuzzification process. The input to this process is a fuzzy set obtained

from the output of the aggregation method. The output of the defuzzification process is a single value.

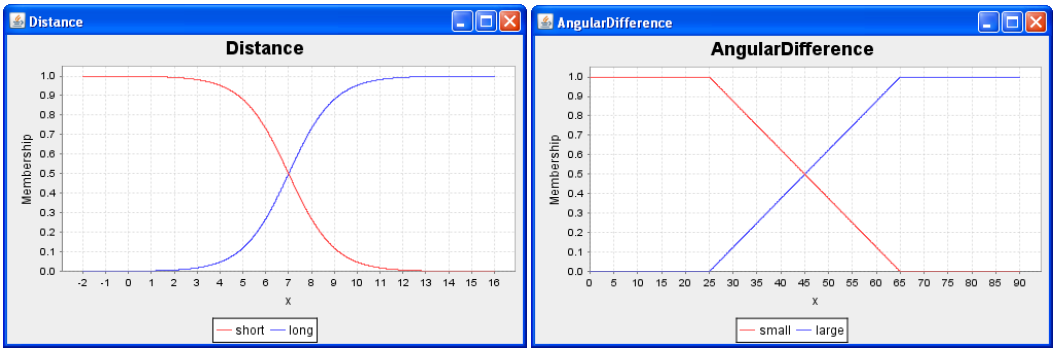
In our fuzzy logic map matching algorithm for pedestrian/wheelchair navigation, we have two input variables. One is distance from a GPS point to a sidewalk segment. The other is the angular difference between the movement trajectory, i.e. a line connecting several GPS points along a segment, and the sidewalk segment. We assign input variable 1 as “distance”, and input variable 2 as “angular difference”. The fuzzy output is the likelihood value to match a GPS point on a segment.

Consider a simple knowledge-based fuzzy rule: “If the distance is short and angular difference is small, then the probability of the GPS point matching on the sidewalk segment is high”. The input variables of this rule are distance and angular difference and the input fuzzy subsets are short and small, respectively. The output variable is the probability of this map matching and the output fuzzy subset is high.

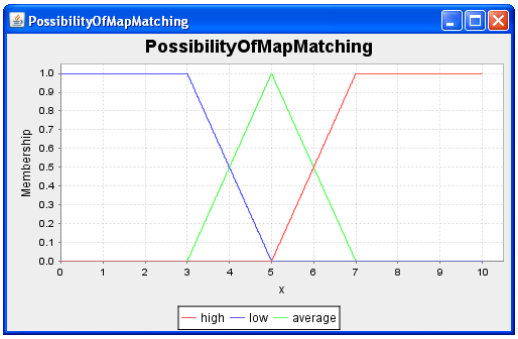
Table 4.6 shows the definition of two inputs, one output. Figure 4.24 graphically presents each parameter defined in the table 4.6. Table 4.7 shows the four rules for this fuzzy logic map matching algorithm.

Table 4.6. Parameters of the fuzzy logic map matching

Parameter	Parameter Name	Fuzzy-logic Graph	Fuzzy-logic Range
Input 1	Distance	Figure 2 (a)	{'short', 'long'}
Input 2	Angular difference	Figure 2 (b)	{ 'small', 'large' }
Output	Possibility of map matching	Figure 2 (c)	{'high', 'average', 'low' }



a. Distance as input parameter 1 b. Angular difference as input parameter 2



c. Possibility of map matching as the output

Figure 4.24. Two inputs and the output in the fuzzy logic map matching

Table 4.7. Rules of the fuzzy logic map matching

RULE 1	IF distance IS short AND angular difference IS small THEN the probability of the GPS point matching onto the sidewalk segment IS high.
RULE 2	IF distance IS long AND angular difference IS large THEN the probability of the GPS point matching onto the sidewalk segment IS low.
RULE 3	IF distance IS short AND angular difference IS large THEN the probability of the GPS point matching onto the sidewalk segment IS average.
RULE 4	IF distance IS long AND angular difference IS small THEN the probability of the GPS point matching onto the sidewalk segment IS average.

Step 1: Fuzzification of inputs and outputs

Since fuzzy subsets describe vague concepts, the truth of any proposition, i.e. the difference is short in fuzzy logic becomes a matter of degree. This is achieved by the fuzzification of the input variable using a membership function (MF). A MF is a curve that defines how each point in the input space, e.g. distance in the above rule, is mapped to a membership value between 0 and 1. Different types of MFs are used, such as triangular, trapezoidal, Z-shaped, S-shaped, Gaussian, generalized bell, and sigmoidal.

In our fuzzy logic map matching, sigmoidal curve is chosen as a MF to approximate the feature value with distance changing. The average horizontal error of the standard positioning service of GPS is 13 m at the 2σ level (ICD-GPS-200C). Therefore, a value

close to this horizontal error can be used as a reference when developing MFs. For instance, the statement "distance is short" is true if the distance is less than or equal to 10 m. The statement is partially true if the distance is greater than 10 m and less than or equal to 20 m. The statement is false if the distance is greater than 20 m. The membership value of the MF for the antecedent "distance is short" can then be defined as $f(x,s,c)=1/(1+e^{s(x-c)})$, where *s* and *c* are the slope and center of the curvature.

In pedestrian/wheelchair navigation, the center of the curvature c is almost half of the width of a road, which is the critical value that differentiates the two sides of a road. In Table 1, the value of c equals 7 after normalization; the slope s equals -1 when the distance is short and equals 1 when the distance is long.

Similarly, we use a piecewise function to represent angular difference. The fuzzy subsets associated with angular difference ($\Delta\psi$) are small and large. The angular difference is fuzzified using the equations

$$\Delta\psi=\begin{cases}1 & 0\le\Delta\psi<25^{\circ}\\ 1.625-0.25\,\Delta\psi & 25^{\circ}\le\Delta\psi<65^{\circ}\\ 0 & 65^{\circ}\le\Delta\psi\le90^{\circ}\end{cases} \tag{4.19}$$

when the membership value of the MF for ($\Delta\psi$) is small, and

$$\Delta\psi=\begin{cases}0 & 0\le\Delta\psi<25^{\circ}\\ 0.25\Delta\psi-0.625 & 25^{\circ}\le\Delta\psi<65^{\circ}\\ 1 & 65^{\circ}\le\Delta\psi\le90^{\circ}\end{cases} \tag{4.20}$$

when the membership value is large.

Step 2: Inference

After the fuzzification of all inputs, fuzzy knowledge-based IF-THEN rules are formulated. In this map matching algorithm we establish four rules connecting input and output variable, listed in Table 4.6. The output of each rule is also a fuzzy set that is achieved by a minimum method (the minimum of all degree of membership values associated with inputs). The output fuzzy sets of each rule are then combined into a single fuzzy set using the 'MAX' aggregation method.

Step 3: Defuzzification

Several methods are available for the defuzzification process, e.g. center of gravity, the largest of maximum, the smallest of maximum, bisector, and weighted average. In order to resolve the potential conflict and to consider all recommended output values based on the strengths of their membership values, this algorithm uses center of gravity (centroid) defuzzification method where X_F represents a combination of the outputs of our four rules. The center of gravity is given by:

$$X_F = \frac{\sum_{r=1}^{4} x_r \, \mu(x_r)}{\sum_{r=1}^{4} \mu(x_r)} \tag{4.21}$$

where x_r is the centroid point in the output range for each rule r, $\mu(x_r)$ is membership value of the output distribution for each x_r and X_F is the final output value.

In summary, the proposed fuzzy logic map matching algorithm performs the aforementioned three steps. By computing perpendicular distance and angular difference and applying fuzzy logic rules, we can obtain the probabilities in which a GPS position is matched onto all the segment candidates. The sidewalk segment that has the highest probability is confirmed as the map matching result.

4.3.2 Fuzzy Logic Map Matching Process

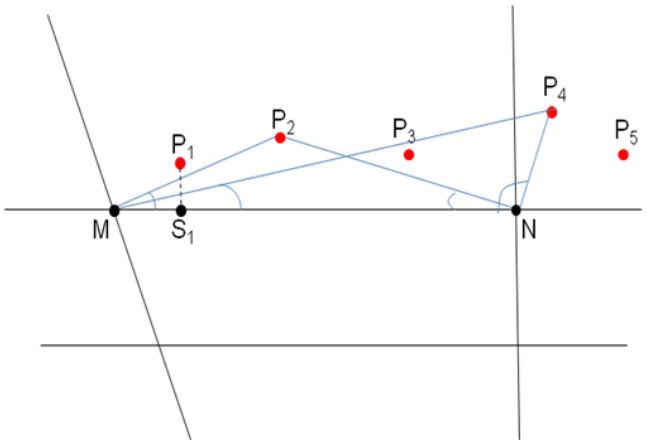
In this algorithm, a three-stage process was used for finding the correct segment. The process has the following steps: (1) the initial map-matching process, (2) the subsequent map-matching process along a segment, and (3) the renewed map matching across an intersection.

Since we need at least two GPS points to form a line and then to compute the angular difference between the movement trajectory and a sidewalk segment, we do the initial map matching after receiving two position fixes. The selected initial link is the link with the highest likelihood value, which is computed as the output value of the fuzzy logic map matching decision system described in the previous section. Then, the user's movement occurs in two ways, either following along an already identified sidewalk segment or entering into a new segment through an intersection. These two cases are defined as two modes: (1) following mode and (2) entering mode.

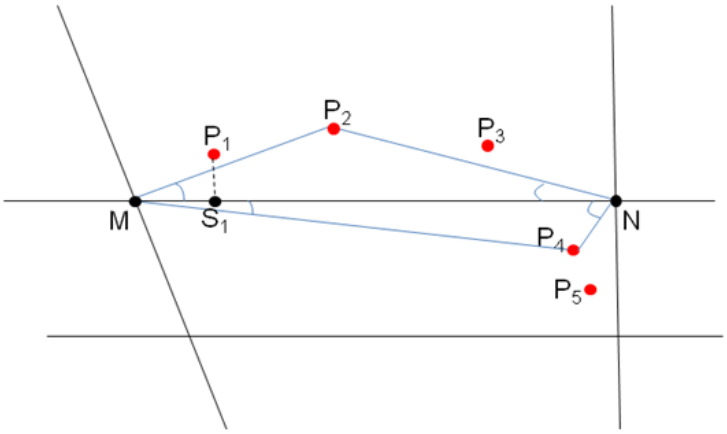
Once a sidewalk segment is ensured as the correct segment, map matching process enters into "following mode", which indicates that users are moving on the same segment so that consequent matches are turn into simply matching the next GPS points onto the underlying segment. However, if an incorrect segment is found, the subsequent steps will continue on the incorrect segment. Therefore, after receiving a few GPS data, we add historical matching data into this process to reduce the mismatch probability. Both distance and angle are computed to track the status of a user. First, we can verify whether the user is on the same segment by comparing the angle between the segment and the line from a GPS point to the segment's starting node and the angle between the segment and

the line from a GPS point to the segment's ending node which must be both less than 90^0. Second, the changing distance from the projected GPS location to each end node of a segment can be used to evaluate whether the user is close to an intersection or not.

In the case of "entering mode", making a turn into a new segment or going straight into a new segment presents their own challenges. The change of distance and angle is the reference to differentiate the two modes. If a wheelchair is traveling straight through an intersection, one of the angles between a segment and the line either drawn from a GPS point to the upstream junction segment or drawn from a GPS point to the downstream junction segment must be exceeding 90^0. However, the change in angles as a condition is not necessarily satisfied when the user just makes a turn. Figure 4.25 provides two examples showing their difference. P_t $\{P_1P_2P_3P_4P_5\}$ presents the recorded GPS point at each epoch, MN is a segment and the wheelchair is moving along its path. Figure 4.25(a) shows the situation where the user entered into a new sidewalk segment at the last two points, P_4 and P_5, whereas Figure 4.25(b) shows the situation where the user made a right turn into a new sidewalk segment starting with point P_4. The first example presents the changes of the angular value of $\angle P_tNM$ from less than 90^0 to greater than 90^0 when passing intersection N from GPS point P_2 through point P_4. Before passing intersection N, $\angle P_tNM$ was an acute angle, but after that, it became an obtuse angle. This pattern does not occur in the second case. In Figure 4.25(b), $\angle P_tNM$ still remains less than 90^0. Therefore, we add distance as an additional reference parameter.



a. An example of going straight



b. An example of making a turn

Figure 4.25. Examples of entering mode

First, we define the angle between segment MN and the line formed by joining GPS point P_t to one end of this segment as α and define the angle between segment MN and the line formed by joining GPS point P_t to the other end of this segment as β. We also

define the distance from the last matched position fix to the upstream junction along the link as d1, and define the distance from the last matched position fix to the downstream junction along the link as d2.

Second, we map match based on the following rules.

1. Following mode:

   Match GPS points on to the underlying segment, where the angles $\alpha < 90^0$ and the $\beta < 90^0$ and MIN(d1, d2) is greater than a threshold σ.

2. Entering mode:

   Renew map matching process where one of two angles, α or β, is greater than 90 , or both angles α and angle β are less than 90 but MIN (d1, d2) is smaller than σ.

The map matching process needs to be renewed when user is entering an intersection. The topological relationship of those segments connecting the same junction help chose candidate segments in the renewed map matching process. The entire process of fuzzy logic map matching is described in Figure 4.26. In this process, the rules mentioned above are to determine whether a user is entering a new segment or not.

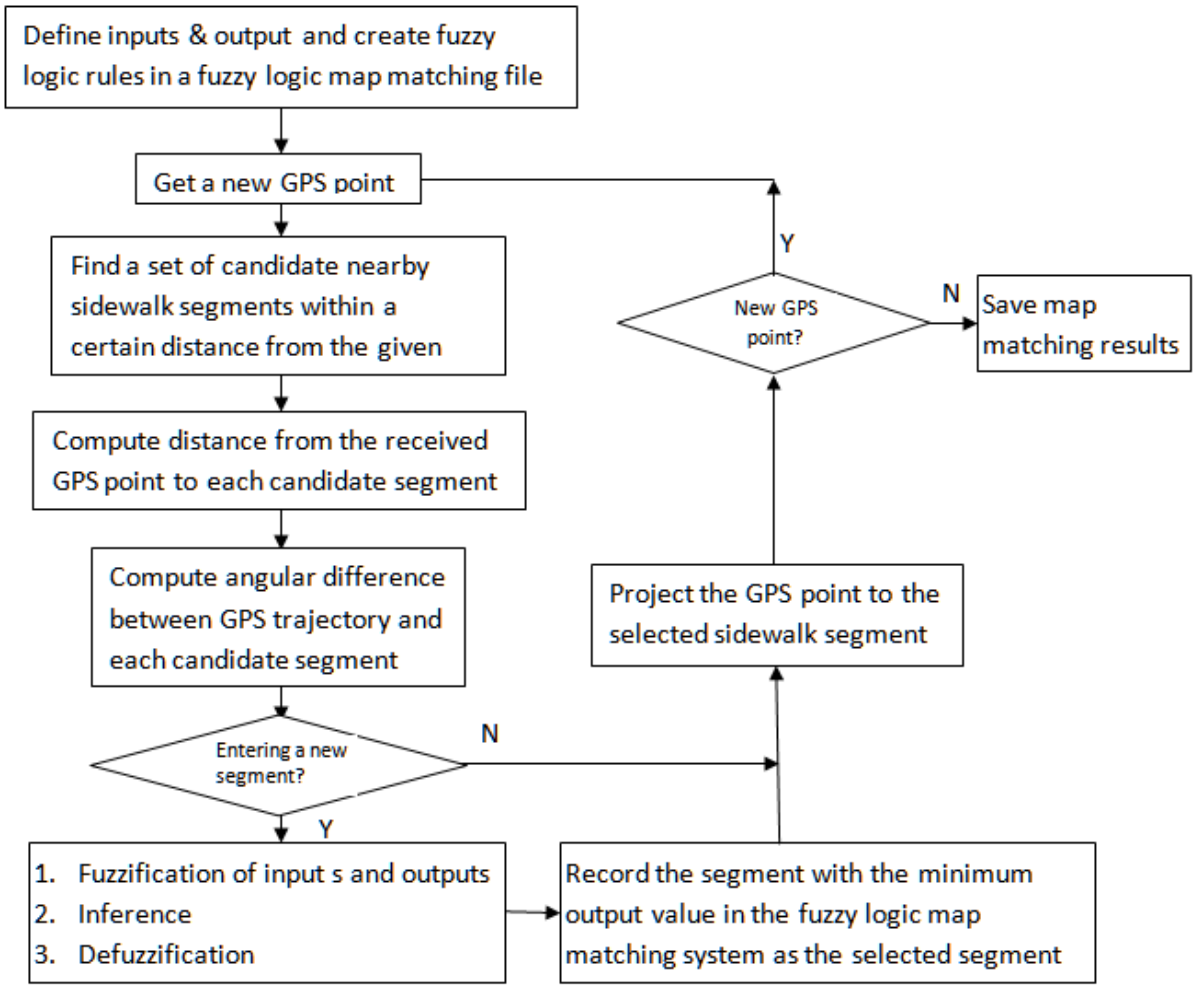
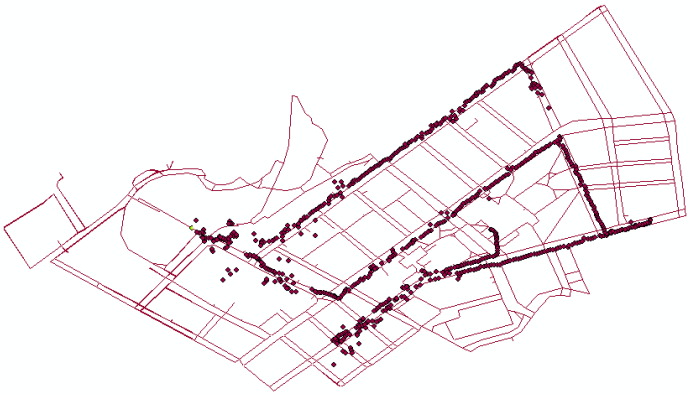


Figure 4.26. Flowchart of the fuzzy logic map matching process

4.3.3 Validation

In order to validate the fuzzy logic map matching algorithm, the process of fuzzy logic map matching is implemented and tested on the same testing environment as shown in section 4.1. After the fuzzy logic map matching algorithm is performed on three-route

GPS raw data, map matching results overlapped on the sidewalk map are shown in Figures 4.27 – 4.29.



a. GPS raw data overlapped on campus sidewalk map

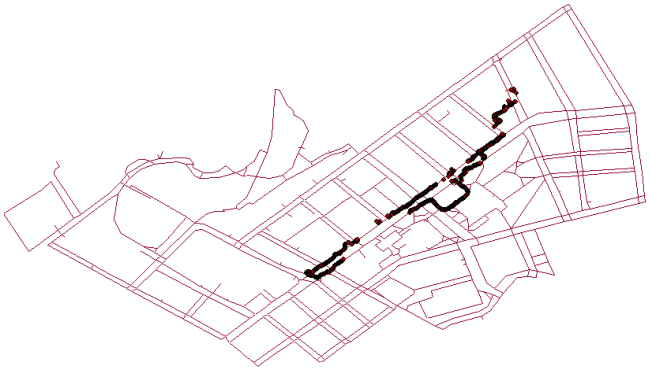
b. Projected result data to the matched sidewalk segments on campus sidewalk map

Figure 4.27. Route 1 comparing map-matching result with GPS raw data on campus sidewalk map

a. GPS raw data overlapped on campus sidewalk map

b. Projected result data to the matched sidewalk segments on campus sidewalk map

Figure 4.28. Route 2 comparing map-matching result with GPS raw data on campus sidewalk map



a. GPS raw data overlapped on campus sidewalk map

b. Projected result data to the matched sidewalk segments on campus sidewalk map

Figure 4.29. Route 3 comparing map-matching result with GPS raw data on campus sidewalk map

The map matching performances on three chosen routes are presented in Table 4.8.

Table 4.8. Performances of Experiments

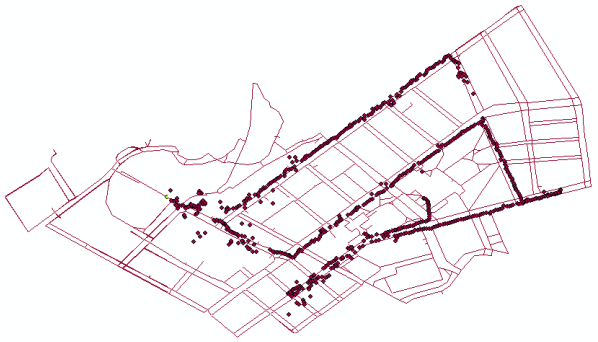
Example	Total Number of GPS Points	Number of mismatched points	Correct Link Identification (%)	Total Computation Time (Sec.)	Average Computation Time/Point (ms)
Route 1	682	85	87.5%	5.485	8.042
Route 2	1516	87	94.3%	12.93	8.529
Route 3	933	53	94.3%	7.828	8.390

The accuracy in the experiment is mainly influenced by the failure in differentiating two sides of a road with stand-alone GPS data. We realize that most mismatched points in our routes occur on sidewalks of narrow roads due to GPS accuracy limitation. With regard to the time performance, this fuzzy logic map matching algorithm performs reasonably well to meet the demand of real-time navigation, based on the average computation time.

4.4 COMPARISON

All the proposed GPS-based map matching algorithms have been developed and evaluated based on three sets of collected GPS data and the sidewalk network on the campus sidewalk network.

Figure 4.30 uses one-route GPS data as an example to show the map matching results after applying each algorithm.



a. GPS raw data overlaid over campus sidewalk map

b. Fuzzy logic map matching result overlaid over campus sidewalk map

c. Chain-code-based map matching result overlaid over campus sidewalk map

d. Hidden Markov Model-based map matching result overlaid over campus sidewalk map

Figure 4.30. Comparison among the three GPS-based map matching algorithms on one route

Experiment results show some issues in common, which are:

1. Poor GPS data lead to mismatching.
2. Many mismatched points occur on sidewalks along the sides of narrow roads, where algorithms have difficulties in distinguishing the two sides of a road within GPS error range.
3. GPS-based map matching does not provide a solution when is no GPS signal.

On the other hand, differences in accuracy, computation time cost and implementation complexity between the three GPS-based map matching algorithms are summarized. Table 4.9 compares the accuracy of the three map matching algorithms on three routes. Table 4.10 shows their differences in time performance and Table 4.11 gives an overall comparison in accuracy, computation time cost and implementation complexity (mainly measured by requirement to skilled developers, length of implementation time and risks taken from implementation errors).

Table 4.9. Accuracy of GPS-based map matching algorithms for pedestrian/wheelchair navigation

Testing Route	Total Number of GPS Points	Correct Segment Identification Rate After Map Matching		
		Fuzzy-Logic MM	HMM-Based MM	Chain-Code-Based MM
Route 1	682	87.5%	92.4%	93.4%
Route 2	1516	94.3%	96%	95.4%
Route 3	933	92.5%	92.7%	89.8%

Table 4.10. Time performance of GPS-based map matching algorithms for pedestrian/wheelchair navigation

Testing Route	Total Number of GPS Points	Average Computation Time per Point in Map Matching (ms)		
		Fuzzy-Logic MM	HMM-Based MM	Chain-Code-Based MM
Route 1	682	8.042	3.666	0.0044
Route 2	1516	8.529	3.709	0.0043
Route 3	933	8.390	3.682	0.0046

Table 4.11. Overall comparison of three GPS-based map matching algorithms

GPS-based Map Matching	Accuracy	Computation Time Cost	Implementation Complexity
Fuzzy-Logic MM	relative low	relative low	medium
HMM-Based MM	relative high	medium	relative high
Chain-Code-Based MM	relative low	relative high	relative low

Shown in Table 4.9, the Hidden Markov Model-based map matching algorithm performs best on the average in terms of correct segment identification. From Table 4.10, it can be seen that all the three map matching algorithms meet the requirement in terms of time performance for real-time pedestrian/wheelchair navigation, when location updates for pedestrian/wheelchair navigation are required to be no longer than 1s, which is normally required in vehicle navigation systems. Comparing accuracy and time performance of the three map matching algorithms, HMM-based map matching algorithm performs better than fuzzy-logic map matching and chain-code-based map matching, but it requires more effort on implementation. In contrast, the chain-code-based map matching algorithm performs relatively poorly in terms of accuracy, but it costs the least in computation with relatively low implementation complexity. Fuzzy-logic map matching is not as good as HMM-based map matching in accuracy but cost most in computation. Therefore, in pedestrian/wheelchair navigation services solely based on GPS data, HMM-based map matching would be the best choice, compared with the other two algorithms, if accuracy is of the highest priority, chain-code-based map matching would be the best choice if computation cost was the most important factor.

5.0 MULTI-SENSOR INTEGRATED MAP MATCHING ALGORITHMS

The experimental results in Chapter 4, which are solely based on GPS data, showed that two major factors influence map matching in pedestrian/wheelchair navigation systems and services. One major factor is that most mismatching occurs in areas with poor GPS signals. The other is that low-end GPS receivers (such as those typically found in mobile devices) do not support a degree of positional accuracy that is high enough to identify the correct side of a narrow road segment in a sidewalk network. Considering these factors, two options are available to enhance positioning accuracy, either by using high-end GPS receivers, adding additional sensors, or both. Since nowadays high-end GPS receivers are still high-priced (Schiller and Voisard, 2004; Theiss et al., 2005), adding additional sensors is considered in this thesis. As a consequence, the performance of map matching can be improved in pedestrian/wheelchair navigation systems and services.

Due to recent advances in computing and mobile device technologies, smartphones, like iPhone and Android phones, are growing in popularity. Navigation services on smartphones can be based on common technologies such as GPS, cameras, accelerometers, compasses, and even gyroscopes. Given the popularity of smartphones and the availability of technologies for navigation services, smartphones are the platform of choice for developing multi-sensor positioning and map matching for pedestrian/wheelchair navigation in this dissertation.

5.1 CLIENT/SERVER ARCHITECTURES FOR MAP MATCHING

There are generally two types of navigation platforms. One works as a standalone system, while the other works on a network with clients and servers. Standalone platforms are referred to as “navigation systems” and network-based navigation platforms are referred to as “navigation services” (Karimi, 2011). Since smartphones have relatively limited memory and computing capabilities, it is difficult to build standalone pedestrian/wheelchair navigation systems on them. Therefore, in this paper, a client/server architecture is discussed to provide multi-sensor map matching for pedestrian/wheelchair navigation services.

A client/server architecture generally involves multiple clients connecting to a central server. In our pedestrian/wheelchair navigation service, the clients are smartphones and the server is responsible for storing large databases, like maps, and performing complex navigation services, such as map matching.

The client/server architecture for the map matching service can be implemented in one of the two approaches: a lightweight client with a heavyweight server, and a heavyweight client with a lightweight server.

5.1.1 Lightweight Client/Heavyweight Server Architecture

In the lightweight client/heavyweight server architecture, the smartphone is responsible for collecting real-time data (positioning data as well as other types of data such as heading data), synchronizing multi-sensor data, requesting map-matched results from the server, and updating the map that is presented to the user. In this approach, the

map data is stored in the server to perform map matching, among other functions. Figure 5.1 illustrates the lightweight client/heavyweight server architecture for map matching.

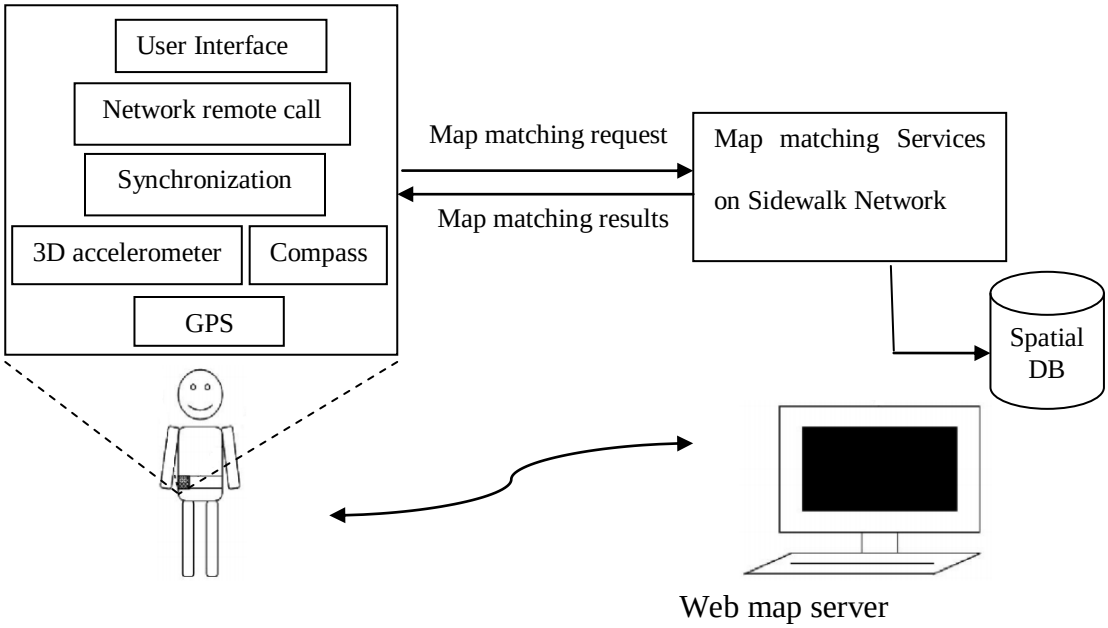


Figure 5.1. Lightweight client/ heavyweight server architecture for map matching

5.1.2 Heavyweight Client/Lightweight Server Architecture

In the heavyweight client/lightweight server architecture, in addition to real-time data collection and multi-sensor data synchronization, the client is responsible for performing map caching and map matching; once a user's current location is given, the relevant map data must be downloaded and cached to the client. With the movement of the user, map-matched locations will be updated and presented to the user in real time. In this approach, all map data (sidewalks) are managed and maintained in the server. For the client to be

able to perform map matching, the server must retrieve the relevant sidewalks and then send them to the client as the user's location changes. Figure 5.2 illustrates the heavyweight client/lightweight architecture for map matching.

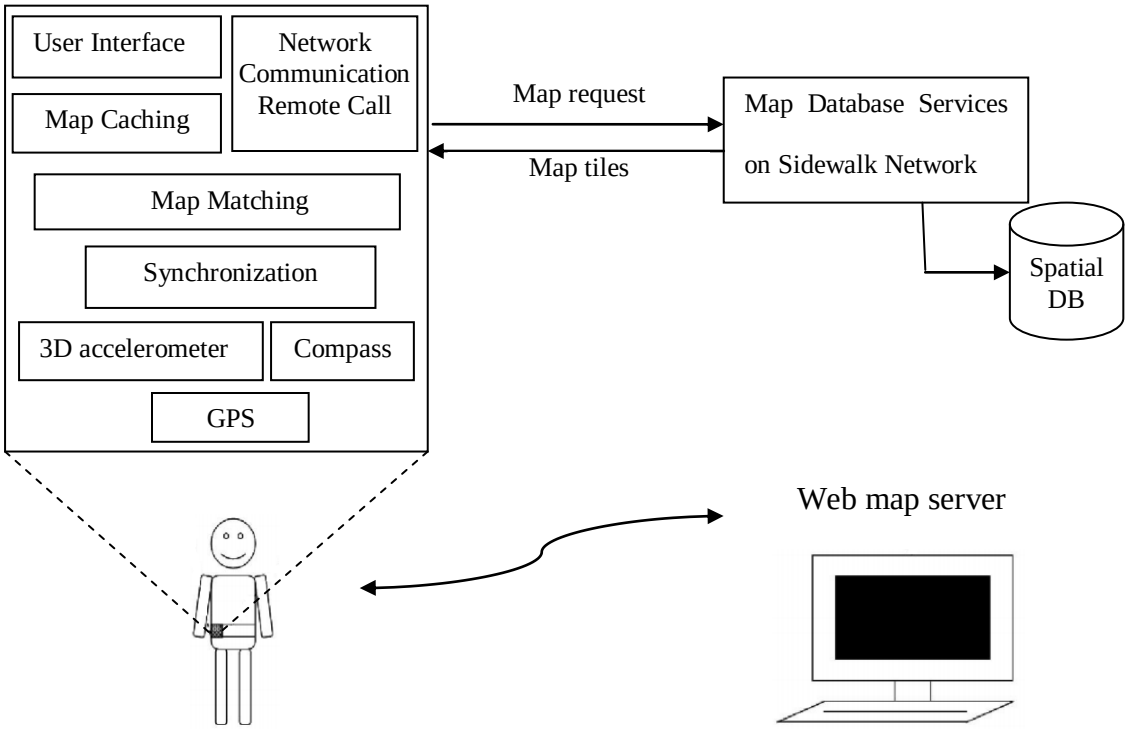


Figure 5.2. Heavyweight client/ lightweight server framework for map matching

5.2 MOVEMENT PATTERN RECOGNITION ASSISTED MAP MATCHING FOR PEDESTRIAN/WHEELCHAIR NAVIGATION

iPhone and Android platforms currently represent the cutting edge of mobile technology and have been widely adopted by people around the world. These two smartphone platforms come with built-in GPS receivers and integrated motion sensors, such as an accelerometer, a compass and even a gyroscope, which can be used for orientation detection, gesture recognition, and image stabilization, among other things.

iPhone and Android platforms are different in terms of GPS location management. iPhone development platform only provides distance-based user-location updating (Arfe et al., 2011). On an Android development platform, the user-location updating function has two modes: distance-based location updating and time-based location updating (Arfe et al., 2011). The distance-based location updating mode updates user's location only when the user travels for a distance that is greater than a pre-determined distance. The time-based location updating mode updates user's location each time a given time interval is reached. Both location updating modes have their own advantages and disadvantages. In the time-based location updating mode, if the time interval is small, frequent user-location updates lead to more awareness of the mobile user's location. However, transmission of too many updates in short time intervals may overload the network. On the other hand, in the distance-based location updating mode, infrequent location updates may cause a lack of awareness about the actual user location but cost less in terms of data transmission compared to the frequent user-location update mode.

Neither of these location updating modes is suitable for pedestrian/wheelchair navigation services. Current GPS technology is unable, due to its accuracy range, to detect movement of pedestrians or wheelchair users who typically move at low speeds. Pedestrians or wheelchair users may move, stop, or make turns at will, which makes presetting a time interval for location updates difficult. As a result, updating a pedestrian's or wheelchair user's location based on time is impractical. Figure 5.3 shows an example of GPS error that can result when a user is stopped at an intersection by a red traffic light.

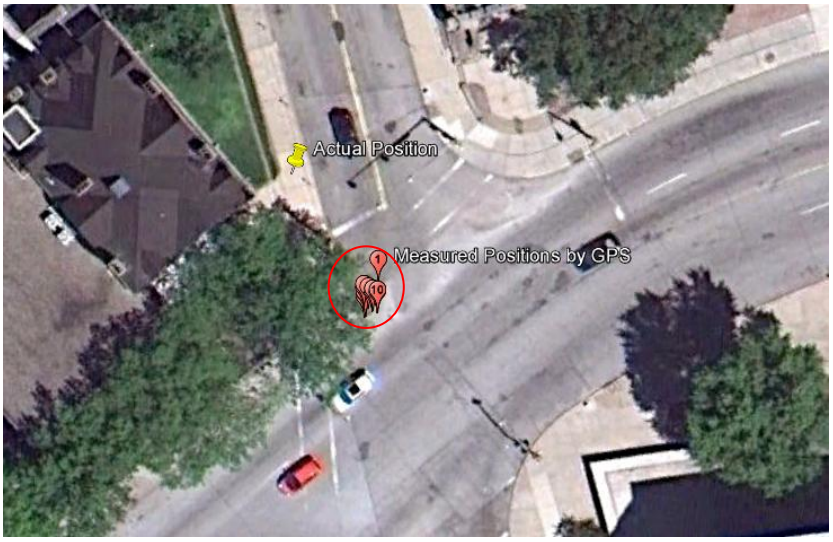


Figure 5.3. An example of GPS error in the scenario in which a user is stopped on a sidewalk.

When there is no movement, GPS keeps updating the same location resulting in multiple positions, circles in Figure 5.3. In this example, with no knowledge about user's movement, the map matching algorithm will treat all the received positions as individual

locations and match them onto the sidewalk. Map matching GPS data when there is no user's movement will not represent user's actual location, since every distinct GPS data within the error circle will be located on a different point of the sidewalk segment.

In order to address the problem of location updating in pedestrian/wheelchair navigation services, an approach that map matches user's locations and updates them based on users' movement behavior is proposed.

Activity recognition from accelerometer data has been a research topic for many years, and is usually formulated as a signal processing and classification problem (Mathie et al. 2004; Ravi, et al., 2005, Sun et al., 2009). Research in activity recognition has focused on identification of physical activities, such as walking, jogging, resting, standing, climbing, or running. Accelerometers have been used as motion detectors (DeVaul &Dunn 2001) as well as for body-position and posture sensing (Foerster, Smeja, & Fahrenberg 1999). Inspired by the accelerometer-related research on activity recognition, this dissertation applies signal processing and pattern recognition techniques to process accelerometer data to recognize user's movement behavior. The following section will present a new map matching algorithm that is assisted by the recognition of user's movement pattern. The algorithm has two major steps. The first step involves using accelerometer data to recognize user's movement behavior and the second step involves performing map matching by using positioning data from GPS, orientation data from a compass, and knowledge of user's movement pattern.

5.2.1 Movement Pattern Recognition

Pedestrian/wheelchair navigation activities that occur outdoors can be grouped into four movement modes: no movement, walking, running, and turning. To detect these four modes of movement, four classes corresponding to these modes are defined in a decision tree classifier. A decision tree classification and recognition was developed and is described below.

Figure 5.4 shows the process of movement pattern classification and recognition. The process has two stages: a training stage and a testing stage. Both training stage and testing stage consist of four steps, three of which are the same in each stage. In the first step, data are collected from multi-sensors. In the second step, the raw data are pre-processed. In the third step, features are extracted from pre-processed data and raw data to create feature vectors. Moreover, the construction process of decision tree is used to feature selection. In the last step of the training stage, a decision-tree classifier is generated which will be used for recognition in the testing stage.

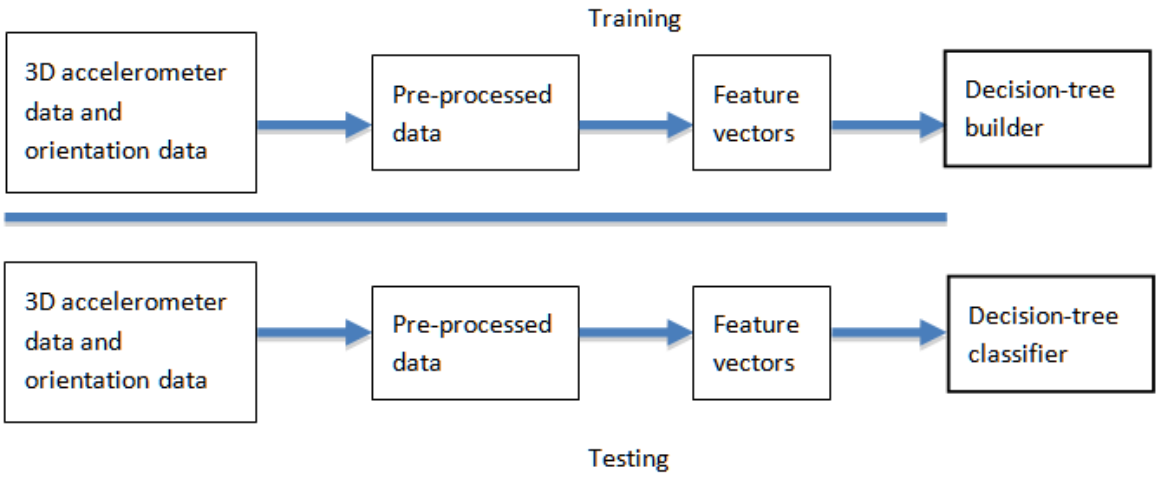


Figure 5.4. Overview of movement pattern recognition

5.2.1.1 Signal Pre-processing

Accelerometers are sensitive to shaking and vibration, while digital compasses are susceptible to noise disturbances in the Earth's magnetic field. The magnetic distortion may vary significantly with time and location due to environmental changes. For this, before accelerations and orientations are measured by an accelerometer and a compass, they must be calibrated in order to reduce the noise disruption of the environment.

Once calibrated, Discrete Fourier Transform (DFT) is used to convert the acceleration data from time-domain values to frequency-domain features. In practice, a Fast Fourier Transform (FFT) algorithm is used to speed up DFT computations. For fast computation of FFT, a window size of 128 is used; this size was chosen as it can provide sufficient data for feature extraction in the next step and can meet the demand of computation in real-time navigation services.

5.2.1.2 Feature Extraction

With noises, if the raw accelerometer data were used directly as inputs to the decision tree classifier, the activity classification would produce poor results. It is possible to extract appropriate features by applying preprocessed data to enhance the quality of classification. In this dissertation, features are extracted from raw accelerometer signals through a sliding window of 128 samples, 64 of which overlap with its predecessor. The reason for utilizing sliding windows with 50% overlap to extract features is explained in the literature (e.g., see Bao and Intille, 2004).

Since a 3D accelerometer, used in most smartphones, can measure acceleration more accurately than a 2D accelerometer can, this dissertation uses typical 3D accelerometers available in smartphones. Figure 5.5 shows a sketch map of a 3D accelerometer, indicating three-axis directions. These three-axis accelerations are measured as a_x, a_y and a_z.

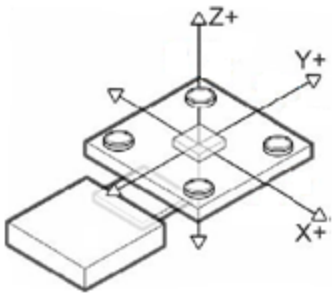


Figure 5.5. 3D accelerometer

It should be noted that it is unnecessary to recognize all types of activities with high accuracy, rather it is sufficient to distinguish between the different modes (i.e., no

movement, walking, running, and turning) for the purpose of navigation. To distinguish between the four modes, four features are extracted from each of the three axes in the accelerometer, giving a total of twelve attributes. The extracted features are mean, standard deviation, energy, and correlation.

Possible range of acceleration data varies with different activities. The energy feature is widely considered in activity measurement and recognition, while correlation is especially useful for differentiating among activities that involve translation of dimensions. No translation in dimensions is produced by the sensor when there is no movement, while walking and running usually involve translation in one dimension. Finally, turning involves translation in more than one dimension. Turning could be making a left turn, making a right turn, or making a U turn. It can be identified by orientation changes measured with a compass.

Taking the x axis of the accelerometer as an example, equations to represent each of the four features are as follows (features in axis y and axis z are computed in the same way).

$m_x = \frac{1}{N}\sum_{i=1}^{N} a_{x_i}$, where a_x is x-axis acceleration, and m_x is the mean of all x-axis accelerations values in sample size N.

$s_x = \sqrt{\frac{\sum(a_x - \overline{a_x})^2}{N-1}}$, where s_x is the standard deviation of x-axis accelerations in sample size N.

$E_x = \frac{\sum_{i=1}^{N} |fx_i|}{N}$, where Ex is energy, f_x is the component produced by FFT, and N represents the length of the sliding window.

$\text{corr}_{xy} = \frac{\text{cov}(a_x, a_y)}{\sigma_{a_x}\sigma_{a_y}}$, where corrxy is correlation between each pair of axes as the ratio of the covariance and the product of the standard deviations.

5.2.1.3 Feature Selection and Classification

After the feature vector is generated, the next step is feature selection and classification. Of the twelve features computed, only eight are considered to be useful to recognize user's movement. For example, when a smartphone is held face-up, its embeded accelerometer is faced up as shown in Figure 5.5. The x direction indicated in the figure is perpendicular to the direction of movement and direction of up-and-down vibration in the movement. Therefore, the features related to the x-axis movement are not useful for distinguishing between the four movement modes. In this case, the features computed based on the x-axis acceleration can be removed from the feature list. Table 5.1 shows the eight features after eliminating the x-axis–related features.

Table 5.1. Selected features

Symbol	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
Feature	m_y	m_z	s_y	s_z	E_y	E_z	$\text{corr}_{xy}(1,2)$	$\text{corr}_{xz}(1,2)$

This eight-feature vector is further compressed by employing a decision tree to eliminate redundant features. In the training stage, a decision tree is constructed based on a training data set. Figure 5.6 shows the movement pattern recognition decision tree after feature

selection. Eventually the selected features are f_1, f_3, f_4, f_6 and f_7, which correspond to m_y, s_y, s_z, E_z and $corr_{xy}(1,2)$, respectively.

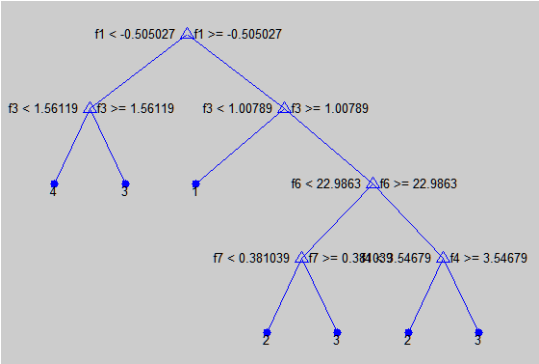


Figure 5.6. Movement recognition decision tree

In Figure 5.6, the leaf nodes 1, 2, 3, and 4 represent the four movement modes, i.e., no movement, walking, running, and turning, respectively. Given the decision tree built in the training stage, to identify user's movement pattern in the testing stage, consecutively collected accelerometer data are processed in real time to compute the five selected features as feature vectors. Through the decision tree, the extracted feature vectors are used to determine the mode to which user's movement belongs. The identified mode will be used for map matching, as described in the next section.

5.2.2 Movement Pattern Recognition Assisted Map Matching

In the lightweight client/heavyweight server architecture, user's movement pattern is recognized in the client by using accelerometer data, where user's location updates are

sent to the server in preset time intervals that vary with the user's movement modes. In the turning mode, with the help of a compass in the smartphone, different turning types, such as left turns, right turns, and U turns can be further distinguished from one another. This orientation data can enhance the accuracy of GPS-based map matching. Given the differences between the three sensors (i.e., accelerometer, compass, GPS), they need to be synchronized in order to ensure that they work effectively in tandem. Figure 5.7 shows the relationship between the three sensors' data, and how they are fused and synchronized for map matching.

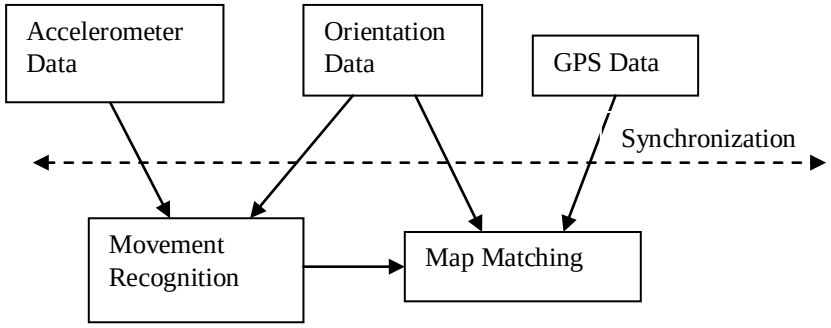
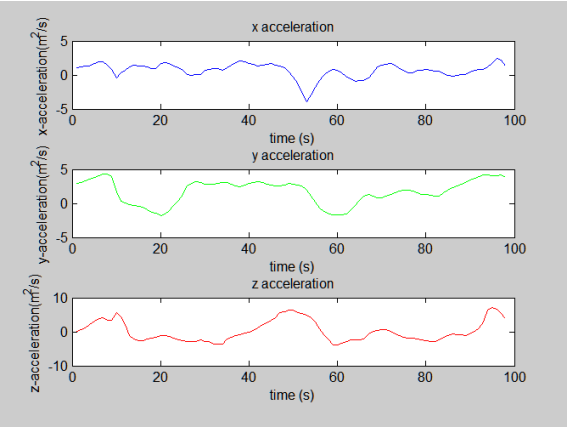


Figure 5.7. Multi-sensor data integrated map matching

In the multi-sensor integration map matching, GPS data are used for absolute positioning in recognizing user's movement. Accelerometer data are used for recognizing four modes of movement, as described earlier. When referring to the North direction, for example, orientation data, obtained from the compass, indicate the orientation of user's movement. This helps in distinguishing between different turning modes. To recognize movement patterns as accurately as possible, accelerometer data are sampled in the highest frequency in order to obtain sufficient samples for FFT processing and feature

extraction. Figure 5.8 shows a snapshot of the 3D accelerometer data. Figure 5.9 shows an example of orientation data relative to the North direction.



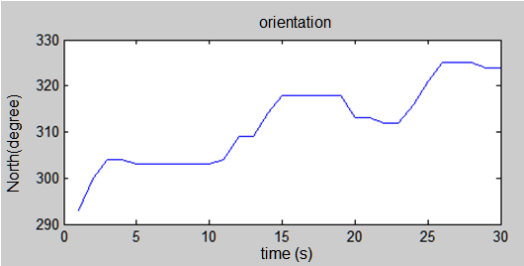


Figure 5.8. Accelerometer Data (acceleration in m^2/s) **Figure 5.9**. Orientation Data (angle in degree)

In the multi-sensor data integration map matching, synchronization is essential to keep all sensor data working in tandem. Data collected from different sensors have different sampling frequencies. With knowing user's current movement mode, the sampling time is determined by the synchronization function. Figure 5.10 shows the user in a walking mode at time t_0 and in a running mode between t_1 and t_2. Once the change in movement pattern is detected, the sampling frequency changes to a suitable interval for sampling data in the running mode. As the user stops between t_4 and t_5, the sampling frequency changes again, since no movement is detected.

Figure 5.10. Timing diagram for synchronization

For map matching, using user's position based only on GPS data, the synchronization timeline starts from the moment when the smartphone begins receiving GPS data. While the map matching algorithm waits for the GPS receiver to provide its first position, this is known as the Time-To-First-Fix (TTFF) problem (Lehtinen et al., 2008), accelerometer and compass data can be obtained and used to detect user's movement behavior. As user's movement mode changes, the time interval of sending GPS data and updating user's location by map matching services will also change. In Figure 5.10, each time point marked on the timeline indicates when all the sensor data are synchronized, given user's movement mode changes.

The flowchart of the movement pattern-recognition-assisted map matching algorithm is shown in Figure 5.11. First, as GPS data and user's heading information provided by the compass are updated in real time, a set of nearest sidewalk segments is chosen as candidates. By comparing heading values of the user in consecutive time, e.g., heading in

time t_{c-1} and heading in time t_c, heading changes above or below a threshold are used to judge whether the user is making a turn or not. By comparing the orientation of a currently map-matched segment with the heading of the user and knowing the current map-matched location of the user, segment candidates can be further limited under different circumstances during the movement. Next, map matching decisions are made by evaluating a weighted combination of distance for positioning data to the candidate segments, and heading differences of the positioning trajectory and segment orientation. Once a position data is map matched, the map matching algorithm will wait for the next set of GPS and orientation data from the client.

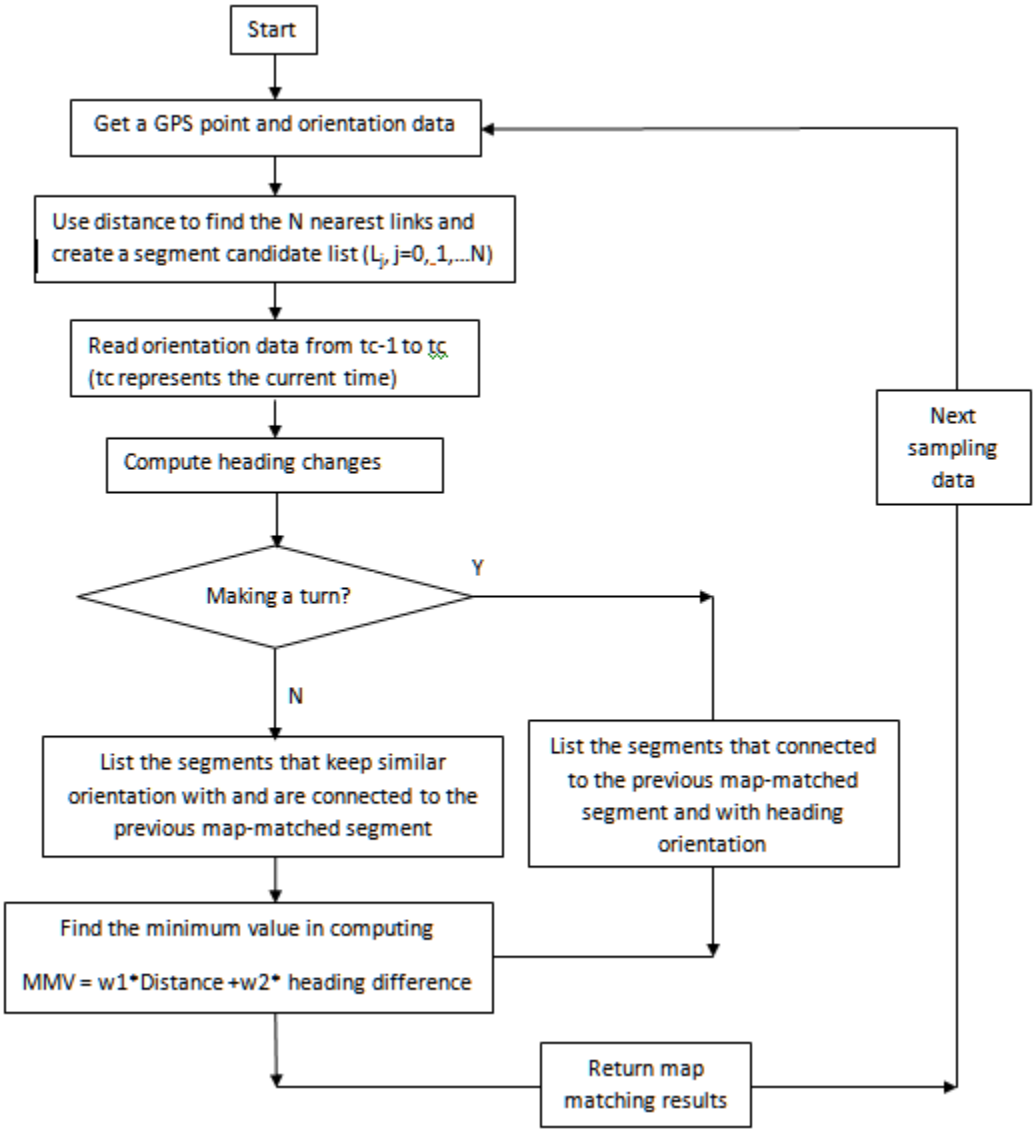


Figure 5.11. Flowchart of the movement pattern-recognition-assisted map matching algorithm

5.2.3 Experiments

To validate the movement-pattern-recognition-assisted map matching algorithm, the sidewalk network of the University of Pittsburgh's main campus was used and GPS points for three routes were collected by walking and using an Android phone (Motorola BackFlip). The server was a PC machine with an "Intel Core 2 2.13G Hz" CPU.

The experiments were performed in two parts. The first part aimed to validate the movement pattern recognition approach. The experimental data contained both training data, which were collected for movement pattern classification, and testing data, which were used to recognize user's movement on real routes. The second part of the experiment aimed to evaluate the map matching performance, as assisted by user's movement pattern recognition.

5.2.3.1 Data Collection and Data Sampling

This section describes the experimental setup to collect sensor data for user's movement pattern recognition and location estimation. All the sensors (GPS, accelerometer, and compass) used in the experiments are available on the Android smartphone. An image of this phone and the direction of its 3D accelerometer are shown in Figure 5.12.

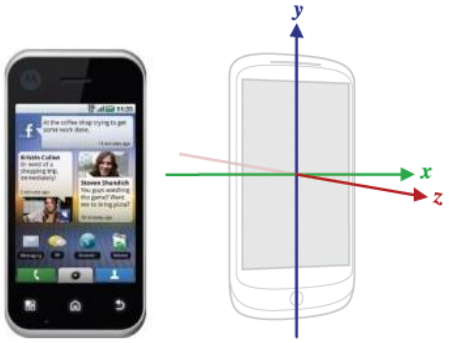


Figure 5.12. Motorola Backflip smartphone and the direction of its 3D accelerometer

Figure 5.13 shows a sample of the recorded data in a log file. The log file includes collected GPS, accelerometer data, and compass data with time stamps. GPS data are tagged by GPS in the log file, which contain longitude, latitude, altitude, accuracy, bearing and speed in order. Accelerometer data are tagged by accelerometer, which are three-axis acceleration data, i.e., acceleration in x-direction, y-direction, and z-direction. Compass data consist of 3-axis orientation data, orientation in x-direction, y-direction, and z-direction.

```
15:20:36:367 ORIENTATION 61.0,-22.0,-23.0
15:20:36:398 ACCELEROMETER -3.7732618,2.8921957,5.707777
15:20:36:405 ACCELEROMETER -2.087744,1.3599066,8.025364
15:20:36:407 ACCELEROMETER -0.21068975,-0.30645782,10.515334
15:20:36:411 ACCELEROMETER 1.340753,-1.4939818,12.737153
15:20:36:439 ACCELEROMETER 2.183512,-1.9536686,14.211981
15:20:36:442 ACCELEROMETER 2.5474305,-2.183512,14.614207
15:20:36:445 ACCELEROMETER 2.5091233,-1.9919758,13.809755
15:20:36:472 ORIENTATION 76.0,4.0,5.0
15:20:36:476 ACCELEROMETER -0.47884035,0.0,10.496181
15:20:36:479 ACCELEROMETER -1.4173675,0.6512229,9.423578
15:20:36:513 ACCELEROMETER -1.5131354,0.5746084,9.308656
15:20:36:516 ACCELEROMETER -1.1683705,0.1340753,9.883265
15:20:36:519 ACCELEROMETER -0.7086837,-0.32561144,10.668563
15:20:36:524 ACCELEROMETER -0.15322891,-0.7278373,11.473015
15:20:36:538 GPS -
79.95231617,40.44751463,26.832815170288086,0.0,283.70001220703
125,0.0
15:20:36:544 ACCELEROMETER 0.34476504,-1.0726024,11.894394
15:20:36:573 ACCELEROMETER 0.6703765,-1.1492168,11.971008
15:20:36:576 ORIENTATION 77.0,4.0,1.0
```

Figure 5.13. A sample of a log file recording GPS, accelerometer, and orientation data

On Android phones, the GPS update frequency is controlled by either setting a minimum time interval (minTime) or setting a minimum distance interval (minDistance). If the value of minTime is greater than 0, the Location Manager in smartphones could stop working for a minTime of milliseconds between location updates to conserve power. If the value of minDistance is greater than 0, locations will only be updated when the device (and thus the user) moves by a distance of minDistance meters. Since GPS receivers on smartphones do provide accurate distance measurement in low-speed movements, distance-based location updates are not appropriate for pedestrian/wheelchair navigation. For this reason, minDistance is set to 0.

In order to save energy and minimize computation time (map matching is potentially a complex task and its response is needed in real time), the following strategy, based on user's movement pattern recognition, is executed to update user's location.

1. Update GPS position every 3 seconds if the user's movement mode is recognized as walking; set minTime to 3s.
2. Update GPS position every 2 seconds if the user is running; set minTime to 2s.
3. Stop updating GPS position if the user is not moving.

The Motorola BackFlip (the smartphone used in these experiments) can provide sampling frequency of, at most, 110Hz on its accelerometer. A sliding window is set, including 128 sampling data which is the same amount of data collected within a time interval of 1.16 s in 110Hz. With 50% overlap between two continuous sliding windows, a three-second interval covers at least four sets of sampling values and a two-second interval covers at least two sets of sampling values. Assuming that the user does not

change movement mode very often, continuous movement in a single mode can provide sufficient sampling data for recognizing pedestrian/wheelchair movement pattern.

5.2.3.2 Training and Testing

To analyze movement pattern, we collected a set of training data by labeling user's behavior, such as walking, no movement, running, and turning. The training data set was used to build a decision tree as the classifier, and is shown in Figure 5.6. We then tested movement recognition on real routes within the study area. Along the testing routes, user's movement pattern in different places is recorded manually as ground truth. By comparing the ground truth data with results of the movement pattern recognition algorithm, the accuracy of recognizing different movement behaviors is shown in Table 5.2.

Table 5.2. Classifier accuracy in identifying four different movement behaviors

	Correct Recognition Accuracy Rate (%)
Walking	92.8%
No movement	97.8%
Running	93.4%
Turning	90.6%

In user's total walking movement, 92.8% were recognized correctly; 1.4% were recognized incorrectly as no movement; 2.6% were recognized incorrectly as running; 3.2% were recognized incorrectly as turning. 97.8% of no movement were recognized

correctly, but 2.2% were recognized incorrectly as walking. Similarly, 93.4% of running movement were recognized correctly, but 6.6% were recognized incorrectly as walking. Turning movement was recognized 90.6% correctly, but 9.4% were recognized incorrectly as running. The confusion matrix of cross-validation on the feature classification of movement behaviors is shown in Table 5.3.

Table 5.3. **Confusion** matrix of cross-validation on feature classification of movement behavior

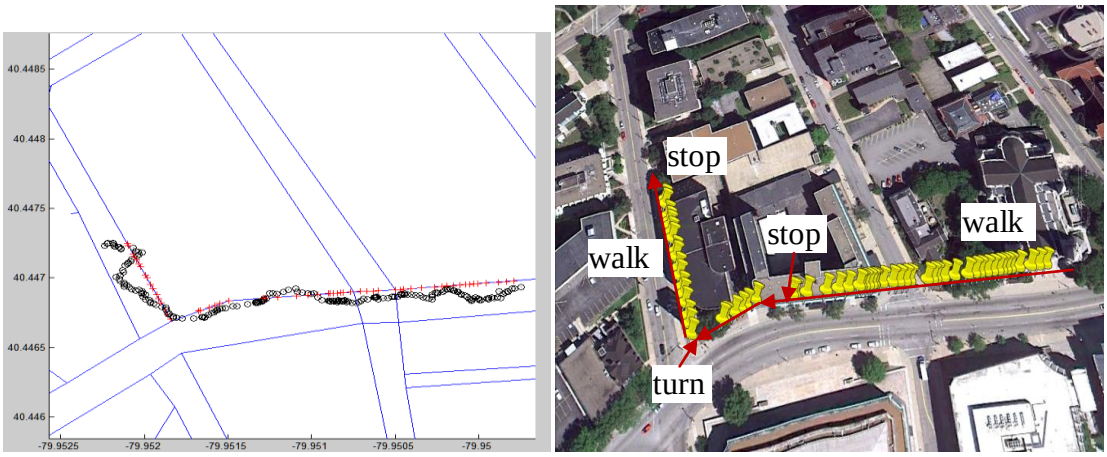
Recognition / Movement Mode	walking	no movement	running	turning
walking	92.8%	1.4%	2.6%	3.2%
no movement	2.2%	97.8%	0	0
running	6.6%	0	93.4%	0
turning	9.4%	0	0	90.6%

Given that over 90% of all movement modes can be correctly recognized, it is feasible to use the movement behavior recognition algorithm to determine actual movement behavior. Based on the movement pattern recognition, map matching is expected to perform more efficiently, as illustrated in the next section.

5.2.3.3 Map Matching Validation

To evaluate the performance of the movement-pattern-recognition-assisted map matching algorithm as outlined in Figure 5.11, we tested the algorithm on a set of routes on the main campus of the University of Pittsburgh. Three routes were chosen in the experiment. Route 1 was selected to represent a short route, where the map matching algorithm was validated in a scenario where the user moved close to buildings. The user started walking along a wide street and then turned into a narrow street. Route 2, as a medium long route, was selected to validate the map matching algorithm in the area with narrow streets and dense buildings. Route 3, the longest route in the three routes, was chosen to validate the map matching algorithm in an area where GPS data are influenced by multipath reflection due to buildings, grasslands, main streets, and small paths. The user's movements on Route 2 and Route 3 include all the four movement modes discussed in the earlier section.

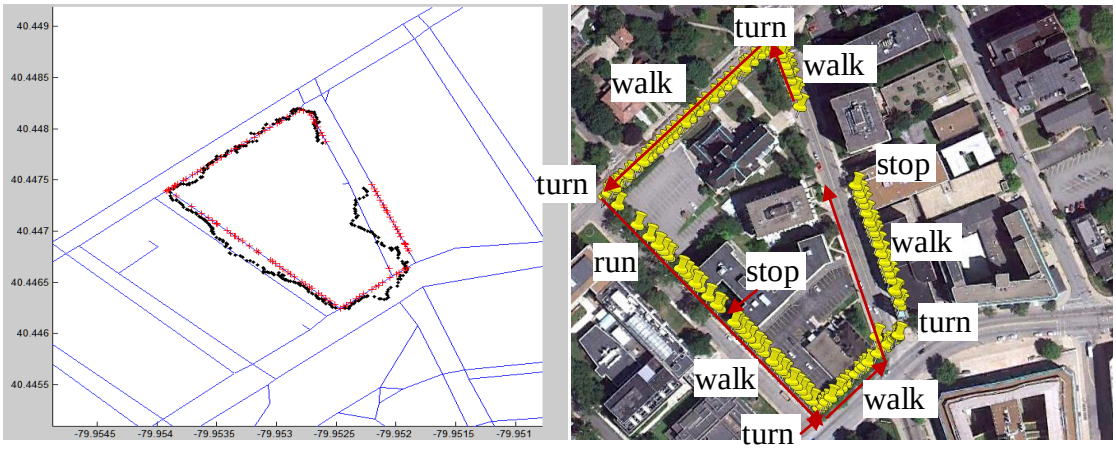
Figures 5.14–5.16 show the comparison of raw GPS positions and map-matched locations in the three routes. In Figures 5.14–5.16 (a), black points indicate raw GPS positions and red points indicate map-matched locations overlapped on the sidewalk network. Figures 5.14-5.16 (b) show map matching results with ground truth labeled by movement modes overlaid on a Google satellite map.



a. Raw GPS data and map matching locati overlaid on campus sidewalk map

b. Map matching locations compared with ground truth overlaid on Google satellite map

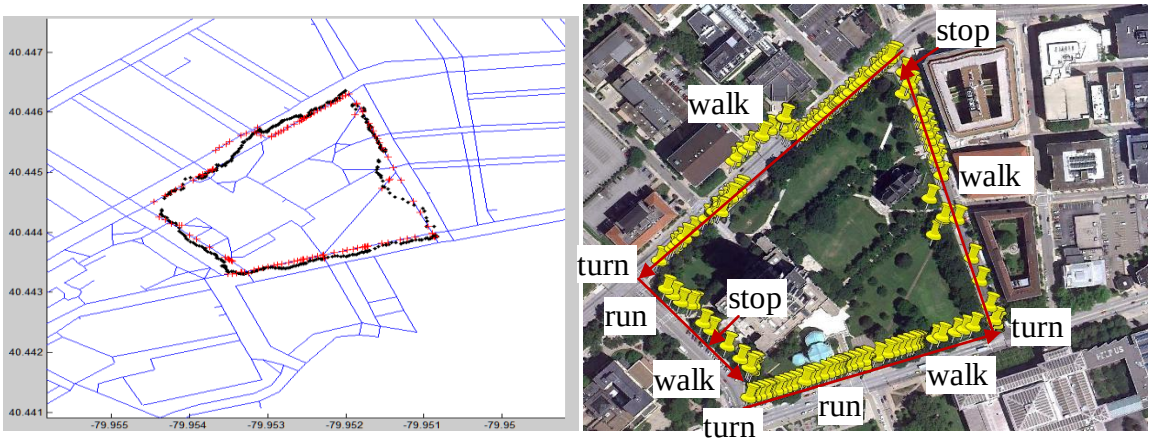
Figure 5.14. Route 1 comparing map matching result with GPS raw data



a. Raw GPS data and map matching locations overlaid on campus sidewalk map

b. Map matching locations compared with ground truth overlaid on Google satellite map

Figure 5.15. Route 2 comparing map matching result with GPS raw data



a. Raw GPS data and map matching locations overlaid on campus sidewalk map

b. Map matching locations compared with ground truth overlaid on Google satellite map

Figure 5.16. Route 3 comparing map matching result with GPS raw data

Table 5.4 shows the result of analyzing the map matching performance in efficiency and accuracy. Taking Route 1 as an example, due to correctly recognizing user's different movement modes, computation for map matching is reduced; 41 out of 133 GPS data were sent for map matching to the server. The visualized results in Figure 5.14 also show that the user's locations still can be continuously and clearly tracked without any influences by the reduction of map matching results. By knowing user's turning behavior, the map matching algorithm was only performed on the sidewalk connected to the previously walked-on sidewalk when the user approached an intersection. This improved the accuracy of the map matching algorithm.

Table 5.4. Map matching performance (efficiency and accuracy)

Testing Environment	Total Number of GPS Points per second	Total Number of GPS Points Sent to Server	Correct Link Identification (%)	User movements
Route 1	133	41	100%	walk->stop->walk-> turn->walk->stop
Route 2	320	100	82%	walk->turn->walk-> tum ->walk->run-> stop -> walk->turn-> walk->stop
Route 3	589	195	87.2%	walk->turn->walk-> run ->stop->walk-> turn ->run->stop

Compared with the high-quality data collected on the campus using a professional-grade GPS receiver in Chapter 4, the segment identification accuracy in our experiments is influenced largely by the poor quality of GPS data collected by a consumer-grade GPS receiver embedded in the smartphone.

Figures 5.14-5.16 show raw GPS data received from the smartphone. It is clear that the GPS data received from the smartphone can be noisy and inaccurate, especially when users are on narrow streets and when users move close to tall buildings. Low GPS data accuracy caused most of the mismatched points in the results. For example, in Route 2, mismatched points occurred during the time the user was turning to a narrow street at the intersection. Even though the turning behavior of the user was recognized, the GPS accuracy is not high enough to distinguish between the two sides of the narrow street. The mismatched points at the intersection led to mismatched projections on the connected segment, as shown in Figure 5.15. This is the reason why the map matching accuracy

shown in Table 5.4 is only 82%, mainly due to the smartphone GPS's inability to distinguish between the two sides of the narrow street. By recognizing user's movement pattern, in Route 2, out of 320 GPS data, 100 were sent to the server for map matching.

In Route 3, the map-matched results also show problems with finding the side of the street on which the user was actually walking. To distinguish between the two sides of a street, high positional accuracy data will be needed in future works. Except for the problem of identifying the side of a street, the experimental results show that the map matching algorithm can correctly estimate user's location in majority of the routes as compared with ground truth shown in Figure 5.16 (b).

Furthermore, the algorithm has low cost of communication and computation. By recognizing user's movement on each route, shown in the last column of Table 5.4, instead of sending user's location, either based on changes of time or changes of distance, movement pattern recognition based on location updates can significantly reduce communication costs between the server and the clients and reduce the calculation costs of map matching. The results shown in Figures 5.14-5.16 also demonstrated that the movement pattern recognition based on location updates can provide location estimation continuously without redundancy. It can be seen that the number of actual GPS data sent to the server, as shown in the second column, is less than one-third that of the number of GPS data received per second, as shown in the first column.

5.3 MULTI-SENSOR MAP MATCHING USING MONOCULAR VISUAL ODOMETRY TECHNIQUE FOR PEDESTRIAN/WHHELCHAIR NAVIGATION

As discussed in Chapter 4, GPS, as the dominant outdoor geo-positioning technology, has been widely used in navigation systems/services. Despite this trend, however, one of the shortcomings of GPS is that its accuracy can be degraded or unavailable in areas with high-rise buildings and obstacles, among other things. Compared to positioning of vehicles, GPS positioning of pedestrians/wheelchair users is more challenging in that pedestrians/wheelchair users move in low speeds and often close to buildings, where multi-path reflections, causing signal degradation, are more frequent.

One possible approach to improve accuracy of geo-positioning, especially in problematic areas, is to integrate GPS with other types of positioning sensors (Karimi 2011; Ahmed et al., 2009). Such integrations can help track users in areas with poor, or even without, GPS signals, in order to fill in extant signal gaps in GPS positioning.

Sensor-integrated geo-positioning estimates positions through the sensory fusion of GPS and other sensors like motion sensors or vision sensors. The additional sensors are used in the measurement of relative movement distance. For instance, data from inertial positioning sensors, like an accelerometer or a gyroscope, can be used to estimate change in position over time; this is called odometry. Odometry is used to interpret data received from the movement of actuators to determine position replacement over time, such as through the use of devices like rotary encoders, which are used to measure wheel rotations (Ohno, et al., 2004). Similarly, visual odometry is the process of estimating traveled distance using sequential camera images (Hagnelius, 2005). However, as all

types of odometry suffer from precision problems, visual odometry must also deal with errors that can occur through the accumulation of data on the continuous motion of subjects. In spite of this, visual odometry can be more accurate compared to classical odometry that relies on motion sensors, (Hagnelius, 2005). For this reason, in this dissertation, visual odometry is explored, by integrating GPS positions with vision-based positioning results, to estimate relative positions of pedestrian/wheelchair users on the sidewalk. Advances in mobile computing technologies have resulted in smartphones that include sensors like GPS, camera, accelerometer, compass, and even gyroscope. GPS provides absolute positioning; camera can be used to record videos or capture images; accelerometer measures acceleration data; and compass data can be used to calculate orientation. Equipped with these sensors, smartphones are seen as suitable platforms for multi-sensor map matching.

The camera pose estimation in visual odometry has various approaches that differ in the number and type of cameras used (Davide, 2008; Kitt et al., 2010). If more than one camera is used, it is possible to recover the scale and scene geometry of the environment through triangulation of the 3D points. Se et al. (2007) and Nogueira et al. (2008) use the binocular vision method to build 3D environment in street view. However, since current smartphones have only one camera, the focus of this dissertation is on the monocular visual odometry. In this section, a multi-sensor map matching algorithm will be presented by integrating visual data, accelerometer data, and GPS data to provide continuous localization of pedestrian/wheelchair navigation.

5.3.1 Multi-Sensor Map Matching Algorithm Using Monocular Visual Odometry

For pedestrian/wheelchair outdoor navigation, map matching decision can be made solely based on GPS data when high-quality GPS data are available. When pedestrians/wheelchair users move into areas with poor or no GPS signals, the monocular visual odometry is one possible approach to continue tracking user's location movement. Whether visual odometry is required for map matching or only GPS data are needed is determined by quality of GPS data. For this reason, the quality of GPS data needs to be detected in real time as users move in order to provide users with continuous map matching service. If GPS accuracy, horizontal accuracy measured in meters, is above a threshold, GPS signals are considered as good quality. In areas with good GPS quality, a GPS-based map matching algorithm is sufficient to estimate user's location. Conversely, in areas with poor GPS quality, when GPS accuracy is above a threshold, vision-based positioning through a visual odometry technique can be used to calculate the relative distance in user's movement. Furthermore, by fusing vision-based positioning results with GPS-based map matching results user's absolute locations can be obtained. In vision-based positioning, image acquisition is required for image matching in motion estimation; this can be accomplished by extracting video frames to reduce redundancy in the obtained images. For this, accelerometer data are utilized to decide when video frames should be extracted by recognizing user's movement (through camera movement). To enhance accuracy of map matching, orientation data from a compass are used during the time the user makes turns. Figure 5.17 shows the flowchart of the algorithm.

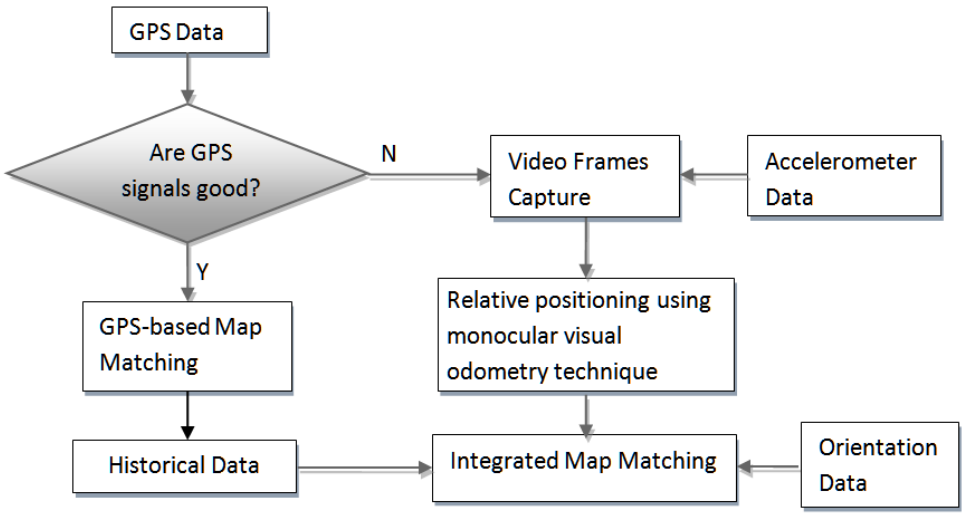


Figure 5.17. Flowchart of multi-sensor map matching algorithm using monocular visual odometry

In Figure 5.17, an accelerometer measures acceleration data to detect user's movements. The user's movement mode determines when images need to be captured from a video stream by following a rule that will be described later. In addition, unlike using a stereo vision, monocular visual odometry has to deal with the ambiguity problem of scale factors from a single image in order to reconstruct the 3D structure of the real world (Hakeem, et al., 2006; Esteban et al., 2010). To address the problem of scale factors in monocular visual odometry, accelerometer data is considered. The accelerometer is utilized to measure the distance between a pair of consecutive frames to calculate the scale factor between them.

5.3.2 Accelerometer-Assisted Monocular Visual Odometry for Motion Estimation

Visual odometry is the process of continuously estimating the position and orientation of a vehicle in robotics research (Davide, 2008). The process of visual odometry includes image acquisition, image analysis, feature extraction and matching, and camera pose recovery from a multiple-view–geometry calculation. Figure 5.18 shows an overview of the visual odometry process.

Figure 5.18. Overview of visual odometry process

The visual odometry process starts by obtaining image data which could come from one or more digital cameras. On captured images, image analysis is applied in order to find interesting points in the images. Interesting points are pixels with distinct intensity compared to those in their neighborhood, and are most likely found as corners or edges in the images. Such interesting points can be extracted and tracked on the overlapped objects captured in consecutive images. Feature vectors are formed after the extraction of features from these interesting points. Work on extracting these feature vectors is followed by feature matching to find the same points in multiple images. The key to measuring distance of movement is the camera pose recovery from multiple view geometry. Intricate geometric relations exist between multiple views of a 3D scene.

These geometric relations are related to camera motion and calibration, as well as to the scene structure. The camera pose recovery is to estimate camera positions where images are taken by computing a rotation matrix and a translation matrix.

Different from previous works on visual odometry (e.g., Davide, 2008; Kitt et al., 2010), in our work only one camera is available for use in smartphones. To capture multiple images, a video stream is taken to track the movement of a user. An image sequence is extracted from the video stream as the input for monocular visual odometry analysis. The next step is to perform image analysis and feature extraction on each image. Adjacent images in the sequence are treated as image pairs and are used for feature matching. The camera pose recovery function takes further responsibility for determining the position and orientation (pose) of the camera. Finally, a final bundle adjustment, based on the Levenberg-Marquardt algorithm (Guerrero et al., 2005; Ke and Kanade, 2003), is used to refine the 3D coordinates and camera positions. As a result, it is possible to estimate the camera trajectory, which corresponds to user’s movement trajectories.

To better understand the relationship between the image plane and the 3D modeling of real world, vision geometry notations are first defined:

(X, Y, Z) are the coordinates of a 3D point in the world coordinate space.

(u, v) are the coordinates of the projection point in pixels.

(x, y, z) are the coordinates of a 3D point in the image coordinate system.

K represents a camera projection matrix, which is a matrix of intrinsic parameters that do not depend on the scene viewed. The matrix represents the quality of each camera, so once K is estimated, it can be re-used as long as the same camera is used.

(cx, cy) is a principal point, which is usually at the image center.

(fx, fy) are the focal lengths expressed in pixels.

[R|T] represents a matrix of extrinsic parameters. This is a joint rotation-translation matrix, where R is the rotation matrix and T is the translation matrix.

Based on the above notations, Equation 5.1 shows the relationship between the image and the 3D scene in the image coordinate system. Equation 5.2 shows the relationship between the image coordinate system and the world coordinate system. Given Equations 5.1 and 5.2, Equation 5.3, which shows the relationship between the image pixels on the image plane and the corresponding 3D points in the world coordinate system, can be calculated.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} fx & s & cx & 0 \\ 0 & fy & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{33} & R_{33} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.2}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \, [I_3 \mid 0_3] \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.3}$$

Set $P = K \, [I_3 \mid 0_3] \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix}$, we have $\begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$.

Given an image represented by $\begin{bmatrix} u \\ v \\ w \end{bmatrix}$, if we have $\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$, which are the coordinates of object points in the real world, it is then possible to calculate matrix P. Matrix P is

mainly made up of two matrices, K and $\begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix}$. K is the intrinsic matrix and $\begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix}$ is the extrinsic matrix. Since all parameters in K are fixed when the same camera is used, if K is known, then the rotation matrix R and translation matrix T can be calculated after P is estimated.

The problem of estimating the trajectory of user's movement can be defined as the trajectory of both the rotation matrix $R_{i-1,i}$ and the translational vector $T_{i-1,i}$ in a given frame, as well as the characterization of the relative movement between two consecutive frames, see Figure 5.19.

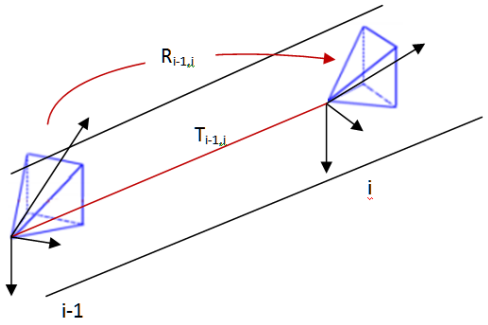


Figure 5.19. Estimation of rotation matrix $R_{i-1,i}$ and translational vector $T_{i-1,i}$ in the motion between video frame Fr_{i-1} and Fr_i

Matrix K must be obtained before the camera pose recovery can proceed. In order to obtain the intrinsic Matrix K, camera calibration must be performed. In this dissertation, since offline camera calibration is more accurate than online camera calibration, we estimate K by using offline camera calibration.

5.3.2.1 Camera Calibration

Camera calibration is to find the essential parameters of the camera that affect the imaging process. Specifically, with the definition of matrix K in Equation 5.1., that is $K = \begin{bmatrix} fx & s & cx & 0 \\ 0 & fy & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, camera calibration is to estimate all parameters in matrix K, which involves calibrating the position of image center, which may not be at the image's true center, estimating the focal length, using different scaling factors for row pixels and column pixels, and accounting for any skew factor and lens distortion (pin-cushion effect). In camera calibration, by taking pictures of a known object and by knowing the coordinates of given object points in the real world, it is possible to obtain internal camera parameters through optimization algorithm.

To implement camera calibration, the camera calibration toolbox (http://www.vision.caltech.edu/bouguetj/calib_doc/) was used.

5.3.2.2 Video Frames Extraction Based on User's Movement Pattern Recognition

Before image analysis and position estimation are performed, frames must be extracted from the video. For extracting frames, the same algorithm discussed in the last section, i.e., pattern recognition of user's movement, is used. A major criterion to perform feature matching between extracted frames is to ensure that there are overlaps between consecutive images. To obtain precise feature matching, dispersedly distributed overlapped features in the images are highly preferred. The objective is to select those frames that will appropriately meet the requirement of image feature extraction and matching, which is needed in a later step for analysis. Moreover, the overlapped features

extraction and matching in consecutive frames will further impact the camera pose estimation.

The key to frame extraction is to decide the points in time at which the extraction should be performed. Recognition of user's movement patterns can aid in this determination. For example, there is no need to extract frames if a user is not moving at a traffic light. Conversely, if a user is making a turn into the next segment of the sidewalk, more frequent frame extraction is required than would be required in the time period when a user is moving straight. This is because turning makes adjacent frames more likely to lose overlapped features. This could cause problems in matching features, and may eventually decrease the accuracy of geometric calculation for the camera pose estimation.

Pedestrians/wheelchair users outdoor activities can be classified into four modes: no movement, walking, running and turning. In the computer vision context, no movement (operating at zero speed), walking (operating at a low speed), running (operating at a relatively high speed), and turning (operating with change in viewpoint of images) require varying frame extraction intervals. In order to extract frames appropriately, with the changes of speed and changes of viewpoint in the movements, frame extraction intervals are set up based on different modes of movements.

A study by Knoblauch et al. (1996) indicates that the mean walking speeds are 1.51 m/sec for younger pedestrians and 1.25 m/sec for older pedestrians. Outdoor powered wheelchairs and mobility scooters have a maximum speed of 1.78 m/sec on paved surfaces. The 1.51 m/sec average speed for walking is used in this dissertation as a baseline to set up frame extraction rates that correspond to different movement behaviors.

The number of standard video frames in one second is 30. Taking walking mode as an example, if the standard pedestrian's walking distance in 1 second is 1.51 m, the frame extraction rate for a 2-m distance interval is (2/1.51)*30, which is about one frame per 40 frames. Since frame extraction rate is inversely proportional to movement speed, using one out of 40 frames as a baseline, the frame-extraction rate in running mode is set as one frame per 30 frames in this work, which corresponds to the relatively higher running speed. In turning mode, in order to keep overlapped features in adjacent frames as much as possible, the frame extraction rate is set as a half of the frame extraction rate in walking mode. In summary, the following rules, corresponding to the four modes of movement, are considered for frame extraction:

1. When a user is walking in a straight path, frame extraction rate is one out of 40 frames.
2. When a user is running, frame extraction rate is one out of 30 frames.
3. When a user is making a turn, frame extraction rate is one out of 20 frames.
4. When a user is not moving, there is no need to extract frames.

To summarize, this section proposes an approach to extract image frames from a video stream by recognizing user's movement mode. When a user is moving on the sidewalk, a smartphone has an accelerometer collecting motion data and has camera taking a live video stream. After matching the user's movement mode with one of the four movement modes, appropriate image frames can be extracted by following the rules described above.

5.3.2.3 Feature Extraction for Map Matching in Wheelchair Navigation

After the images are obtained, feature extraction is the next step in estimating motion. With a prior knowledge about man-made environments on streets, such as rectangular objects with dominant planes (Ohnishia and Imiya, 2006) like buildings, objects matching can make use of some of the special characteristics of street-view images. In a sequence of street-view images, sky and ground are both viewed as backgrounds due to their stable and static characteristics, whereas other objects like buildings and cars are unique or diverse, so they are more helpful in location identification. Unique objects in urban environments include:

- Buildings
- Vehicles, e.g., cars, bikes, strollers
- Pedestrians
- Vegetation, e.g., trees, flowers, bushes
- Urban furniture, e.g., city lights, telephone poles, parking meters, benches
- Signs and banners

Signs and banners can be used to recognize specific locations only when Optical Character Recognition (OCR) Technology is applied to recognize characters in images, while some types of vegetation and urban furniture may appear in multiple locations and images. Vehicles and pedestrians/wheelchair users are moving objects that are not stable in locations, thus they are considered unreliable features to use for feature matching. Buildings are the most stable and distinctive objects for location estimation in pedestrian/wheelchair navigation. Inspired by human cognitive mechanisms that daily navigation strongly relies on landmark information, overlapped landmarks in image sequences are the interesting points (features). As a user moves, viewpoints and objects

in image sequences change. Finally, due to its sensitivity to changes in viewpoints, scales, lighting, and environment, global features, such as color histogram, texture, and edge, are not suitable for location estimation. With change of distance and viewpoint during movements, features with rotation-invariance and scale-invariance are needed. After analyzing various features discussed in the literature (e.g., MOBVIS, 2006; Cipolla, 2004), in this dissertation, local features, both for object recognition and for subsequent location estimation, are used. Of the existing local feature extraction algorithms, Scale Invariant Feature Transform (SIFT) is the most effective algorithm for street view images (Deselaers et al., 2007), described in the next section.

Scale Invariant Feature Transform (SIFT) Descriptor

The SIFT descriptor transforms image data into scale-invariant coordinates that are relative to local features. The SIFT descriptor is a well-known method in computer vision for its capabilities in robust matching to the database records, despite viewpoint, illumination, and scale changes in images. SIFT is suitable for object recognition in urban environments where illumination and scale changes usually degrade performance (Lowe, 1999; 2004).

The following are the major computations that are used to generate SIFT features:

1. Scale-space extrema detection. This is the initial preparation. Of all scale levels and their corresponding image locations, a difference-of-Gaussian function is used to identify potential interest points, which are invariant to scale and orientation.

2. Keypoint localization. At each candidate location, a detailed model is used to determine location and scale. A technique similar to the Harris Corner Detector (Derpanis,

2004) is used in SIFT. Keypoints are selected by eliminating some instable candidates, like edges and low contrast regions in terms of their stability.

3. Orientation assignment. One or more orientations are assigned to each keypoint location, based on local image gradient directions. Image data are processed and transformed relative to the assigned orientation, scale, and location for each feature. This effectively cancels out the effect of transformation.

4. Keypoint descriptor. With scale and rotation invariance in place, local image gradients are measured at the selected scale in the region around each keypoint. This helps identify unique features, allowing for significant levels of local shape distortion and changes in illumination.

5.3.2.4 Monocular Visual Odometry Assisted by Accelerometer for Motion Estimation

Figure 5.20 shows how monocular visual odometry works when a user is moving on a sidewalk.

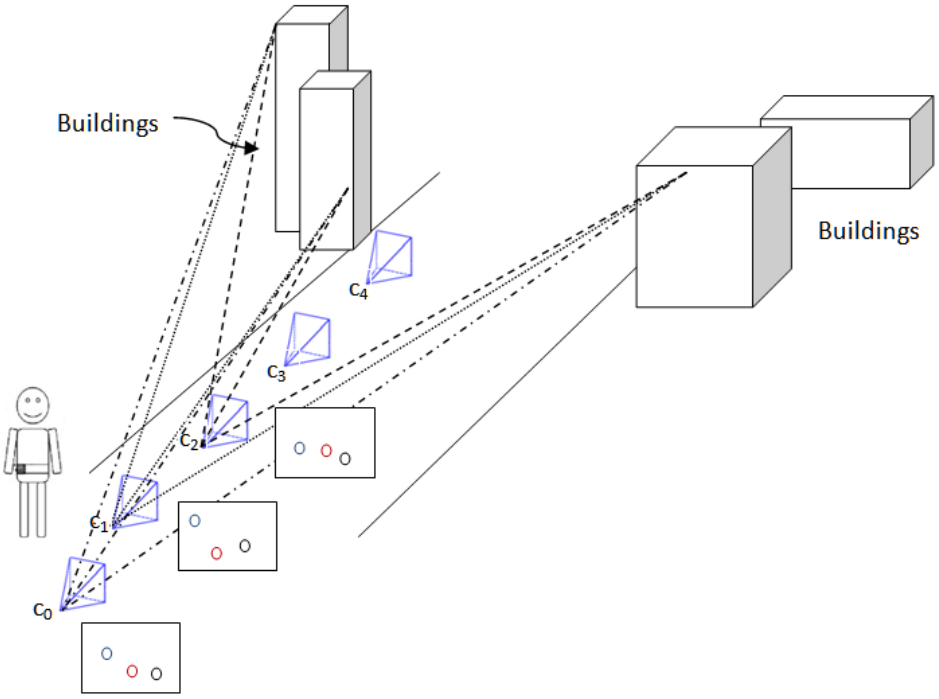


Figure 5.20. Frame-to-frame motion estimation

In Figure 5.20, a video stream is obtained from a camera, and a sequence of images is taken as the user moves. Images are shown on image planes from C_0 to C_4. A different set of image features corresponding to 3D objects is used to compute the motion between consecutive frames. For instance, images of a street scene in the figure are taken in consecutive frames C_0 to C_2, which have overlapping objects, like buildings. Some features of objects are marked as circles on the image planes, and those common features will be used for camera pose estimation in the motion.

The first step of a frame-to-frame motion estimation is to extract a set of salient features that are present in each frame. SIFT is employed to extract local features and build descriptors as feature vectors.

In the second step, the features across consecutive frames are matched using nearest neighbors search and a minimum distance threshold in the SIFT descriptor space, obtaining matches between frames Fr_i and Fr_{i+1}.

In the third step, the normalized 8-point algorithm is employed to compute the frame-to-frame motion, as described by Hartley and Zisserman (2004), due to its computational simplicity. Outliers are then removed between Fr_i and Fr_{i+1} frames using RANdom SAmple Consensus (RANSAC), and the final motion is re-computed using only the set of inliers. This yields a fundamental matrix F that describes camera motion.

In the next step, given that the camera was calibrated beforehand, we already know the calibration matrix K. Therefore, the essential matrix can be obtained by E = K'*F*K.

In the last step, the frame-to-frame rotation matrix $R_{i-1,i}$ and the translational vector $T_{i-1,i}$ are obtained using the method by Horn (1990). This yields four possible solutions, from which the one with more inliers in front of both cameras is selected. Furthermore, a scale factor for each translational vector must be calculated to recover the overall camera pose.

Figure 5.21 shows the visual-based positioning algorithm.

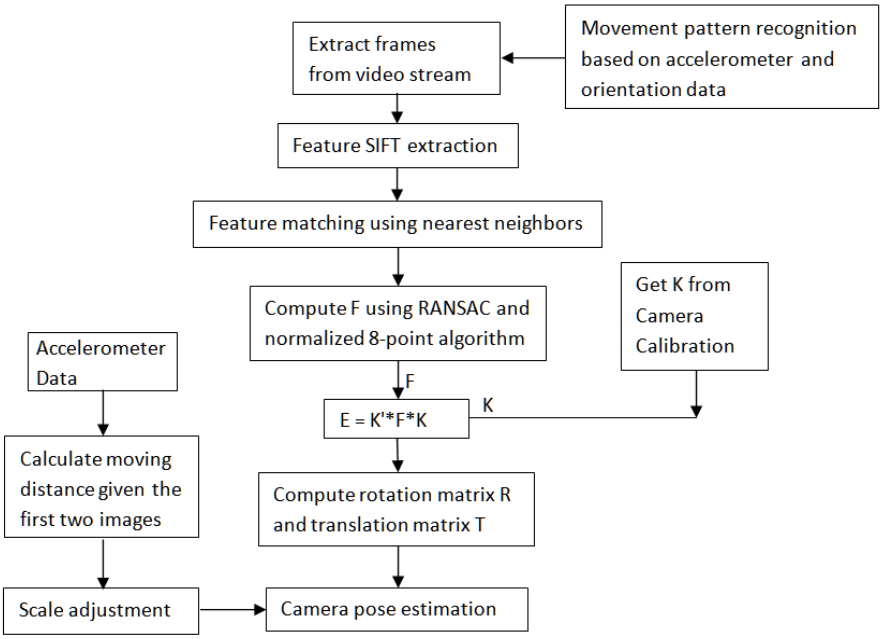


Figure 5.21. Flowchart of vision-based positioning algorithm

Scale Adjustment

With only one camera, the baseline between two instants is unknown and the scale factor of reconstruction is ambiguous. To address this ambiguity problem, this section discusses an approach where an accelerometer is used to assist with estimating the scale factor. Figure 5.22 shows the motion estimation for consecutive frames.

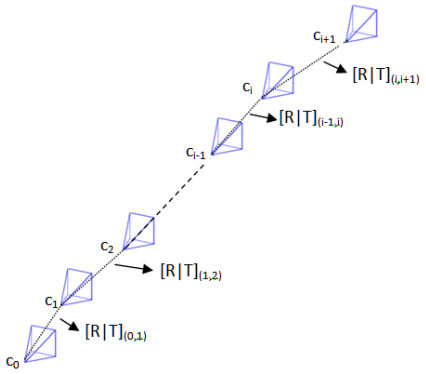


Figure 5.22. Scale adjustment

When the distance between cameras C_0 and C_1 is normalized to 1, the location of the third camera C_2 can be estimated across the translation direction. All scales between consecutive images are adjusted based on the normalized distance $d(C_0, C_1)$. To solve the ambiguity of the scale factor in translational vector, an accelerometer is used to measure the distance between the first frame and the second frame, i.e., $d(C_0, C_1)$.

The integral of acceleration over time from t_0 to t_1 will yield velocity, and the integral of velocity over time results in distance. Since accelerometers return data in units of the gravitational constant, i.e., g, acceleration values need to be multiplied by 9.81 to convert to m/s^2. During this process, errors may accumulate in the integral calculation. As a result, accelerometer is used to only calculate first-step distance, $d(C_0, C_1)$, which measures the movement from the first frame to the second one.

Once the distance between C_0 and C_1 is obtained, the translation is only determined up to the scale ratio between each pair of consecutive frames. The ratio between these distances must be calculated before the camera's pose can be reconstructed.

To calculate this distance ratio, we first calculate the motion between three consecutive frames using frame-to-frame feature matches. This produces two different motion estimations [R|T](i,i+1) and [R|T](i+1,i+2). The quality of this motion estimation is greater than the motion estimation of [R|T](i,i+2), due in part to the larger number of matches. These two motions are translated into two different scale factors, s(i,i+1) and s(i+1,i+2).

Given the motion estimation of the camera and the reconstructed 3D points of 3-frame matches, the following relation is established:

$$K^{-1}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = [R|s_i T]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (5.4)$$

where s_i is the scale ratio that relates the translation between cameras i and i+1 and cameras i+1 and i+2. The ratio s_i = s(i,i+1)/ s(i+1,i+2) is calculated using matches across all 3 frames and a linear system of equations as in the P6P DLT algorithm (Sattler et al., 2011).

5.3.3 Integrated Map Matching

Given the information from the camera pose recovery, the relative displacement combined with GPS historical positions and orientations in the movement are used to perform map matching on a sidewalk network. Figure 5.23 shows the overview of the map matching algorithm, which integrates the camera pose recovery results with the GPS historical trajectory and orientation data on the sidewalk network.

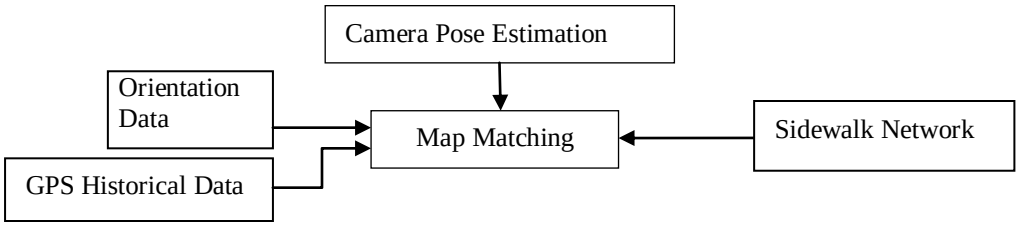


Figure 5.23. Flowchart of map matching approach

In areas with poor GPS signals, the camera is used to capture images for measuring continuous user's movement distance. GPS historical data provides the starting positions at the time when the camera is to be active. Camera pose estimation is performed to obtain the relative displacements between consecutive image planes. Orientation data, as measured by the compass, are integrated with estimated positions to help map matching as users move about the environment.

5.3.3.1 Coordinate System Conversion for Tracking Data Presentation on Digital Map

In order to integrate data from different sensors in multi-sensor map matching, four coordinate systems are involved. These are a 2D image coordinate system, a 3D camera coordinate system, a world coordinate system, and a map coordinate system. In the integration process, GPS positions and sidewalk map data exist in the world coordinate system. They are presented by longitudes and latitudes in the WGS-84 projection system. In the camera's pose estimation, image sequences are extracted from real-time video streams, and image feature extraction and computation are conducted in the 2D plane coordinate system. Image features are further reconstructed in the 3D camera coordinate

system. Therefore, 2D image plane coordinates are transformed and presented in the 3D camera coordinate system by Equation 5.1, and are further translated to a 3D world coordinate system by Equation 5.3. Finally, all the positioning data and map matching results are transformed into the map coordinate system. This requires a conversion between the 3D world coordinate system and the 2D map coordinate system, from WGS-84 to Universal Transverse Mercator (UTM) (Grewal et al., 2002), in order to track user's locations on the 2D map.

5.3.4 Experiments and Analysis

To validate the multi-sensor map matching algorithm, experiments, on the sidewalk segments of the main campus of the University of Pittsburgh were conducted. Multi-sensor data including video, accelerometer, compass, and GPS data were collected by an Android phone. The computing platform was a PC machine with an "Intel Core 2 2.13G Hz" CPU.

5.3.4.1 Data Collection on Smartphone

GPS, camera, accelerometer, and compass data were collected by a Samsung GT I9000 Galaxy S smartphone.

The accelerometer in this experiment has two roles. On the one hand, it is used to identify user's movement pattern for video frame extraction. On the other hand, the acceleration data collected between the first extracted frame and the second extracted frame are used to calculate the distance of movement for scale factor estimation that were discussed earlier.

5.3.4.2 Camera Calibration

The camera was calibrated beforehand by applying "Camera Calibration Toolbox for Matlab". Twenty 720x480 photos of a black and white checkerboard were taken from different angles by the Samsung phone. Figure 5.24 shows one of these photos, on which each corner of the grid on the checkerboard is selected as the featured point and is marked with a red cross.

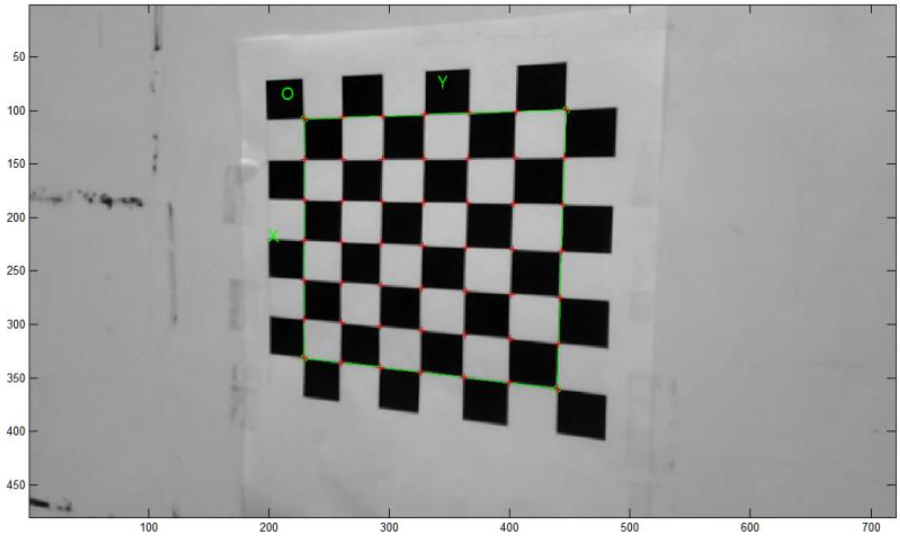


Figure 5.24. A checkerboard to calibrate camera

Camera internal parameters were estimated in the camera calibration. K is the intrinsic matrix, as shown in Equation 5.5.

$$K = \begin{bmatrix} 686.646920000000 & 0 & 359.500000000000 \\ 0 & 687.467270000000 & 239.500000000000 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.5)$$

5.3.4.3 Experimental Results

Several video clips were taken on the main campus of the University of Pittsburgh. Buildings, trees, pedestrians, cars, urban furniture, and signs were the most common objects captured in these clips.

The video stream was recorded as a user was walking on the sidewalk in front of the School of Information Sciences at University of Pittsburgh. A sequence of frames was captured in the video stream. No particular attention was given to the distance between frames, since frames are automatically extracted based on user's movement pattern. These captured frames were saved as images for further image processing and feature extraction. Figure 5.25 shows a sequence of images extracted from a video that was collected in motion.

Figure 5.25. A sequence of images extracted from a video

1. SIFT Feature Extraction and Feature Matching

The SIFT algorithm is first applied to extract features from images. Figure 5.26 gives an example of SIFT feature extraction from one image taken on the campus. In the figure, the length of the arrow represents the scale of the extracted SIFT features and the arrow's direction represents the extracted features' dominant direction. Figure 5.27 shows two image frames that were extracted from the video and with their SIFT features extracted. The lines that link two features in the two images show the correspondences in the feature matching. Table 5.5 presents the number of feature points extracted from each image and the number of matched points in both images.

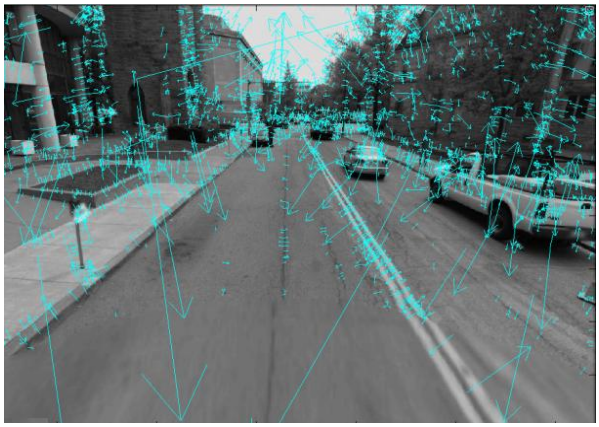


Figure 5.26. SIFT features of a street view image

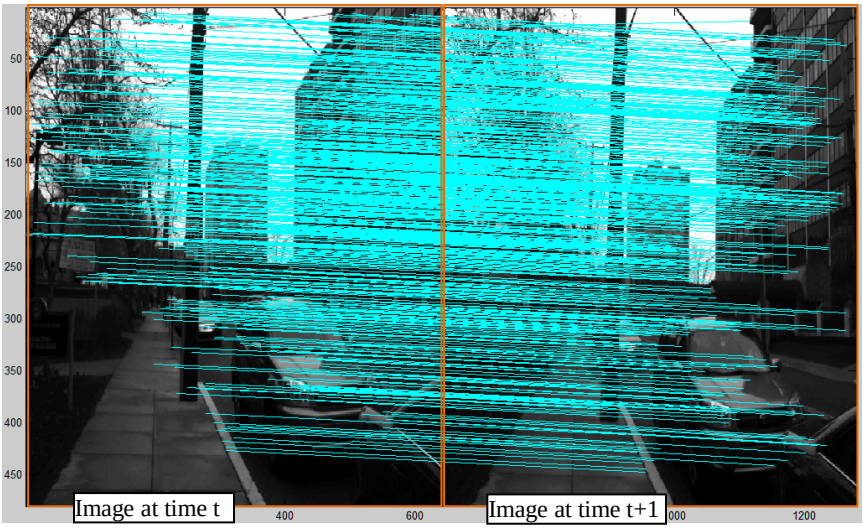


Figure 5.27. Matched SIFT feature points

Table 5.5. Key points and matched points

Image Sequence	Feature Points	Matches
Image 1	2575	737
Image 2	2500	737

In Table 5.5, an image pair in one image sequence is taken as an example, 2575 SIFT feature points are extracted from Image 1 and 2500 SIFT feature points are extracted from Image 2. After feature matching, both images have 737 feature points in common corresponding to the same feature points on objects in the real world. Similarly, feature extraction and feature matching are implemented between all the continuous image pairs

taken in the experiment. Taking 100 images as samples in the experiment, 87.5% features in the images are matched correctly. Since images are taken with changing viewpoints, the high quality of feature extraction and accuracy in feature matching indicate that SIFT is insensitive to changes of viewpoints, which is appropriate in our vision-based geo-positioning.

2. Fundamental Matrix Calculation

After SIFT features are extracted, the fundamental matrix F is calculated, given correspondences in an image sequence. In Figure 5.28, given a calculated matrix F, the marked corresponding feature points are shown in Image 1, while the epipolar lines going through the matches are shown in Image 2.

Figure 5.28. Feature points in image 1 vs. Epipolar lines in image 2

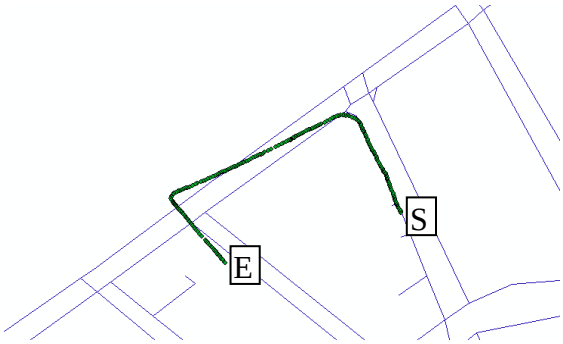
3. Monocular Visual Odometry and Vision-Based Map Matching

After the fundamental matrix F is obtained, the essential matrix E can be calculated as discussed earlier. Therefore, camera positions and poses can be estimated, which provide user's locations. The video was taken starting in the front of the School of Information Sciences building, and Figure 5.29 shows the positioning results, as overlaid on Google Earth. When compared to the video that was taken and shown in Figure 5.25, the results show that locations are estimated quite precisely compared to the actual trajectory that are recorded by the collector.

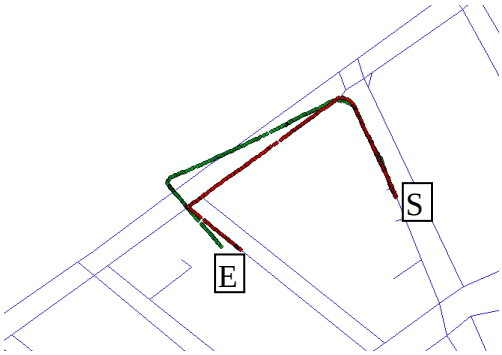
Figure 5.29. **Geo**-positioning results by using visual odometry, top view (left) and street view (right)

However, like other odometry techniques, the visual odometry must overcome the problem of accumulating position errors (Davide, 2008; Kitt et al., 2010). This problem also occurred in our experiment and errors were accumulated in a relatively long distance using the monocular visual odometry. Starting from the same origin and continuing the same route, as shown in Figure 5.29, a video stream is recorded as the user is walking for 332 m in around 4 minute. The experiment is shown in Figure 5.30, where the user started walking from a point marked as S and stopped walking at another point marked as

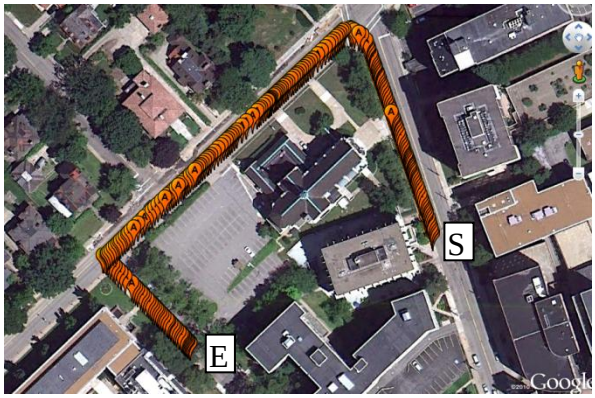
E. All the objects listed in Section 1.4.1.3, such as buildings, trees, cars, pedestrians, and signs, appear on the captured images along roads. Figure 5.30-a shows that the estimated locations drifted over time when only the monocular visual odometry technique was applied. To mitigate this problem, we use geometrical and topological information in the sidewalk map to constrain user's location in every map matching step on the sidewalk in order to reduce positional error accumulation. Figure 5.30-b shows the comparison of location estimation, both before and after map matching. Finally, Figure 5.30-c shows the map matching results overlaid on Google Maps.



a. Monocular visual odometry results in one route before map matching



b. Estimated locations on one route before map matching and after map matching



C. Map matching results overlaid on Google Maps

Figure 5.30. Map matching results overlaid on Google Maps

Since vision-based map matching is only needed in places where GPS signals either are not available or have poor quality, the accuracy of GPS signals is used as the criterion to determine when to start vision-based map matching. Figure 5.31 shows a sample log file from data collection. As described earlier in Section 1.2.3.1, the log file includes GPS, accelerometer, and orientation data, and each GPS data point has recorded longitude, latitude, accuracy, bearing, altitude, and speed, in order.

```
15:20:45:463 ACCELEROMETER 0.47884035,1.9919758,9.059659
15:20:45:466 ACCELEROMETER 0.7278373,1.8579005,9.212888
15:20:45:469 ACCELEROMETER 0.9768343,1.685518,9.595961
15:20:45:474 ACCELEROMETER 1.1109096,1.4939818,10.093954
15:20:45:477 ORIENTATION 54.0,-8.0,6.0
15:20:45:513 ACCELEROMETER 1.3790601,1.3215994,10.572795
15:20:45:516 ACCELEROMETER 1.340753,1.4173675,10.649409
15:20:45:520 ACCELEROMETER 1.4939818,1.6472107,10.70687
15:20:45:525 ACCELEROMETER 1.283292,1.8770541,10.821792
15:20:45:533 ACCELEROMETER 1.091756,1.8387469,10.783484
15:20:45:536 ACCELEROMETER 1.3790601,1.4939818,10.515334
15:20:45:563 ACCELEROMETER 1.8770541,1.2449849,10.802638
15:20:45:577 GPS -
79.95218644,40.44750061,8.9442720413208,0.0,286.70001220703125
,0.0
15:20:45:582 ACCELEROMETER 2.2984335,0.8810662,11.473015
15:20:45:585 ORIENTATION 54.0,-2.0,10.0
15:20:45:610 ACCELEROMETER 0.842759,0.019153614,12.852075
15:20:45:617 ACCELEROMETER 0.2873042,0.019153614,12.852075
15:20:45:621 ACCELEROMETER -0.038307227,-0.019153614,12.469003
```

a. A sample log file recording GPS, accelerometer and orientation data and a highlighted GPS position with accuracy of 8.94 m

```
15:20:38:636 GPS -
79.95222581,40.44747017,17.8885440826416,0.0,286.3999938964844
,0.0
```

b. A highlighted GPS position with accuracy of 17.89 m

```
15:22:47:553 GPS -
79.95297402,40.44828781,10.0,208.6999969482422,263.70001220703
125,1.0
```

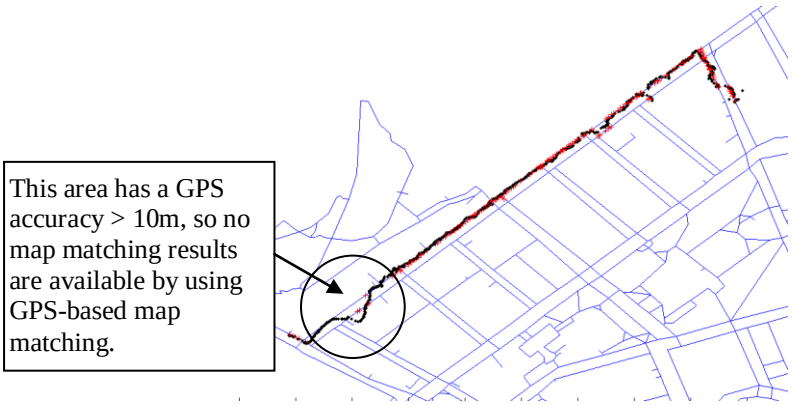
c. A highlighted GPS position with accuracy of 10.0 m

Figure 5.31. A sample log file to compare accuracy of GPS positions

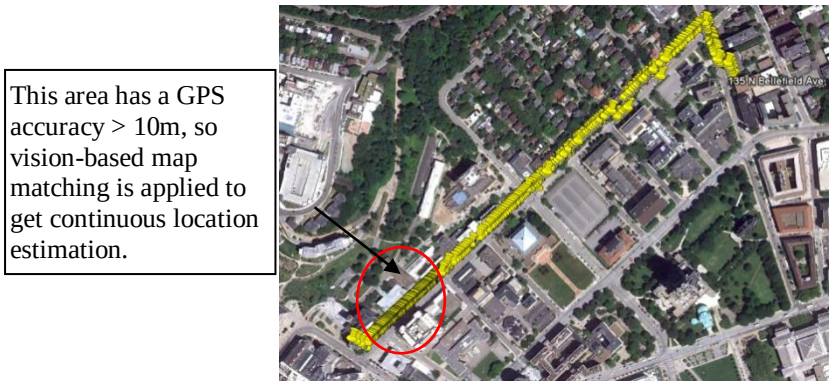
As Figure 5.31 shows, the collected GPS data have different accuracies in different positions at different times. The accuracies of three GPS positions are 8.94 m, 17.89 m, and 10.0 m. Our tests show that, in the multi-sensor map matching algorithm, if GPS accuracy is equal to or better than 10 m, then the quality of GPS data is considered to be acceptable and map matching can be performed by using only GPS data. But if GPS accuracy is worse than 10 m, vision-based map matching is needed to fill in the localization gap. In short, the accelerometer and orientation data are used to help video frame extraction, assist map matching in movement pattern recognition, and improve the overall efficiency of map matching.

Figure 5.32 shows the experimental results that compare the GPS-based map matching results with multi-sensor map matching results. In Figures 5.32 a and b, black points represent a GPS trajectory and red points show the GPS-based map matching results. Figure 5.32-a shows the map matching results of all the GPS raw data, while Figure 5.32-b shows the map matching results based on those GPS data with accuracy <=10 m. In Figure 5.32-b, because a section of sidewalk (along O'Hara St) has GPS accuracy worse than 10 m due to poor GPS signals, GPS-based map matching is not appropriate; this is where the vision-based map matching is performed to fill in the signal gap from GPS. The final map matching results, which are obtained by integrating GPS and vision data, are shown in Figure 5.32-c. These final results prove that using monocular visual odometry the multi-sensor map matching algorithm can provide users with continuous location estimation, regardless of changes in quality of GPS data.

a. GPS-based map matching results, as compared with raw GPS data overlaid on the sidewalk map



b. GPS-based map matching results in GPS accuracy<=10m, compared with raw GPS data overlaid on the sidewalk map



c. Multi-sensor integrated map matching results overlaid on Google Maps

Figure 5.32. Comparison of GPS-based map matching results with multi-sensor map matching results

In terms of time performance, vision-based map matching is computationally-intensive, which requires high CPU and memory usage. This is one reason why in the experiments the lightweight client/heavyweight server architecture was used as the platform to implement the vision-based map matching algorithm. In this architecture, the server is responsible for major computations in vision-based map matching, including SIFT feature extraction and feature matching and camera pose estimation. Clients (smartphones) are responsible for capturing video streams and extracting frames from captured video streams. Frame extraction from video streams takes about 0.1 second in average. Since each image is 720*480 pixels and each pixel requires 8 bits of storage, data size of each image is 720*480*8 bits=2,764,800 bits. To perform feature extraction and feature matching on the server, each image needs to be uploaded to the server, which cost about 1.53s with an average of 1.8Mbps data upload speed on the 3G networking. With this, the total time (computation and communication) is less than 1.7s on the client side. On the server side, computation of vision-based map matching, which involves

SIFT feature extraction and feature matching, is the major cost. In the experiments, all the images extracted from video streams have 720*480 pixels. In Matlab running environment, our experiments showed that the average time of SIFT feature extraction and feature matching between two images is about 1 second. Furthermore, the camera pose estimation process takes about 0.5 second. Therefore, the computation on the server side cost 1.7 second. After adding the response time, from clients to the server, the total time is 3.2s. For this, we set up time intervals of 3.2s to update the map matching results. The average speed of pedestrians is 1.51 m/s, so the distance moved in 3.2 seconds is below 5 m. For pedestrian/wheelchair navigation, 5 m location updates are reasonable. The time performances on clients, over networks, and on servers, indicate that the proposed multi-sensor map matching approach is suitable for pedestrian/wheelchair navigation applications.

In summary, this section presented a multi-sensor integrated map matching algorithm using monocular visual odometry. The experiments showed vision as a complementary sensor making up for the shortfalls of GPS capabilities, and vision-based map matching is supplemental to GPS-based map matching. The integration of GPS and vision can provide users with seamless map matching service.

6.0 SUMMARY, CONCLUSIONS, CONTRIBUTIONS AND FUTURE RESEARCH

6.1 SUMMARY

This dissertation first investigated existing map matching algorithms in car navigation systems. It also discussed the unique characteristics of the pedestrian/wheelchair navigation application and the challenges in map matching for the pedestrian/wheelchair navigation. Based on the study, some advanced map matching algorithms for pedestrian/wheelchair navigation systems/services were designed and developed. A summary of the algorithms developed in this dissertation is as follows.

To address the problem of finding the correct segment (road or sidewalk) efficiently, which is the first step of map matching, an adaptive candidate segment selection algorithm was developed.

In the case of using GPS as the only positioning sensor, three GPS-based map matching algorithms were developed. These are chain-code–based map matching, HMM-based map matching, and fuzzy-logic–based map matching.

To solve the issue of tracking pedestrians or wheelchair users in places that have poor or no GPS signals, two multi-sensor integrated map matching algorithms were developed. In the first algorithm, an accelerometer and a compass were utilized to recognize the user's movement pattern in order to integrate with a GPS and help improve

overall map matching. In the second algorithm, a vision-based map matching algorithm was developed to calculate relative displacement in the movement on the sidewalks in the absence of GPS data or of high accuracy GPS data. Furthermore, a map matching algorithm based on the integration of vision, accelerometer, compass, and GPS sensors can provide users with uninterrupted map matching services. Both multi-sensor-based map matching algorithms were designed, developed, and tested on a client (smartphone)/server architecture for pedestrian/wheelchair navigation.

In this dissertation, experiments were conducted to evaluate the developed algorithms by navigating on the sidewalk network of the main campus of the University of Pittsburgh. The algorithms were evaluated both for accuracy and time performance.

6.2 CONCLUSIONS

In this dissertation, a set of advanced map matching algorithms were designed and developed for pedestrian or wheelchair navigation. The following conclusions can be drawn, based on the results of several experiments using the developed algorithms.

The adaptive candidate segment selection algorithm can perform efficiently by dynamically selecting candidate segments when given updated GPS positions, as well as relative changes between GPS positions and clustered segments.

The three advanced GPS-based map matching algorithms can provide users with high quality of location estimations on sidewalk networks in terms of both accuracy and computation time. However, GPS-based map matching suffers from two problems: it cannot provide high quality solutions in places with poor GPS signals and it does not

provide solutions at all in places with fully blocked GPS signals. Another problem in sidewalk GPS-based map matching is that it cannot distinguish between the two sides of narrow streets, when the distance between the two sides a street is less than the positioning accuracy range of the GPS unit.

The visual odometry multi-sensor integrated map matching algorithm presented in this dissertation supplements some of the drawbacks of GPS-based map matching, especially providing location estimations in places without GPS signals. The experimental results showed that the multi-sensor integrated map matching algorithm is both feasible and practical in providing uninterrupted location estimations when navigation in outdoors. However, identifying the correct sidewalk from parallel sidewalks on both sides of a street still remains a challenge because the GPS accuracy on sidewalks is often not high enough. Additionally, the user movement pattern recognition algorithm, which integrates accelerometer, compass, and GPS data, can greatly improve the efficiency of map matching on the smartphone/server architecture.

The experimental results showed that all the developed algorithms perform fairly well to achieve the goals of the project. Furthermore, these map matching algorithms are feasibile and practical and they are potential to be utilized in different pedestrian/wheelchair navigation applications. With the popularity of smartphones and by building on the smartphone/server architecture, pedestrian/wheelchair navigation services can be widely and quickly accepted by current smartphone users for their mobility use. By providing continuous and precise location estimation for tracking people, the multi-sensor integrated map matching algorithms are particularly beneficial to some groups of people, such as senior citizens or children, who may require uninterrupted tracking

services in various situations. The results of this dissertation can benefit other research areas such as automated wheelchair navigation and walking robots, where ensuring uninterrupted localization by map matching is one of the critical factors necessary to plan routes and achieve automatic location guidance.

6.3 CONTRIBUTION

Map matching is an essential component of and plays a major role in navigation systems/services. By analyzing the requirements of pedestrian/wheelchair navigation, this research concentrated on developing map matching algorithms specifically designed for pedestrian/wheelchair navigation services. We can summarize the novelty of the developed algorithms by examining their four distinct contributions.

First, a hierarchical clustering technique was applied to transportation networks and a binary tree was specially built for indexing segments. This data structure technique was adopted for handling the first step of map matching algorithms by developing a segment candidate selection algorithm based on an adaptive searching scheme.

Second, three GPS-based map matching algorithms were developed for pedestrian/wheelchair navigation. The experimental results demonstrated acceptable map matching accuracy and time performance.

Third, by using multiple types of sensors, a movement pattern recognition algorithm was developed for identifying user's movement behaviors in order to assist with map matching in pedestrian/wheelchair navigation systems/services.

Fourth, the visual odemetry, another multi-sensor integrated map matching algorithm, was developed to provide continuous localization services. This algorithm used the monocular visual odometry technique to estimate the relative displacement in motion when GPS signals are poor or unavailable. To solve the scale factor problem, the accelerometer was used for measuring the camera's displacement between the first two image frames. Furthermore, in order to reduce positional error accumulation over time, which is inherent in the visual odometry technique, geometrical and topological information of sidewalks were used as constraints to match relative displacements onto the map. Vision, accelerometer, and compass data were integrated with GPS data to match user's locations onto sidewalks without interruption.

The multi-sensor-integrated map matching algorithms make significant contributions to the realization of pedestrian/wheelchair navigation services on smartphones. With advances in technology of sensor-embedded smartphones, this research can potentially serve for many user's mobility applications, such as tourist guidance, geo-fencing for children, in which parents are notified when a child leaves a designated area, geo-fencing for senior citizens, look-around navigation systems for the visually impaired, among others. This research also can potentially impact building future automatic wheelchair navigation systems and walking robots. The continuity and accuracy in localization provided by the developed map matching algorithms in this dissertation paves the way for implementation of such systems.

It is also important to note that although this research has made significant contributions to the realization of map matching algorithms designed specifically for

pedestrian/wheelchair navigation, further research is still required in the areas discussed in the next section.

6.4 FUTURE RESEARCH

We suggest the following areas for future research:

- *Improve map matching accuracy distinguish between the opposite sides of narrow streets.* To be able to identify the correct side of narrow streets, which is not of concern in car navigation, is a unique challenge in pedestrian/wheelchair navigation.
- *Investigate heavyweight client/lightweight server architecture for pedestrian/wheelchair navigation systems and services.* The implementation of multi-sensor integrated map matching algorithms in our experiments were based on the model of lightweight client/heavyweight server architecture. However, the heavyweight client/lightweight architecture (the other option proposed in Chapter 5) can speed up location updates by running map matching on clients with sufficient capabilities to store partial map data and to perform some computations. Comparing and contrasting the pros and cons of the two architectural approaches will be invaluable for designing and developing practical pedestrian/wheelchair navigation systems/services.
- *Investigate implementation of the developed multi-sensor integrated map matching algorithms directly on smartphones.*

Android and iPhone are currently the two most popular platforms for mobile applications development, both in terms of engineering quality and consumer satisfaction. The proposed multi-sensor integrated map matching algorithms could be implemented directly on smartphone platforms rather than on a server.

- *Incorporate an image database of geo-tagged landmarks with the sidewalk network database.*

  The current vision-based map matching algorithm accumulates relative movement distance to estimate user's location, which can cause positional error accumulation in a long distance movement. One solution to reduce such positional error is to use geo-referenced landmarks in the map matching process. Future research will be needed to investigate the possibility of using geo-referenced landmarks to further increase the positional accuracy of the vision-based map matching algorithm.

REFERENCES

Ahmed Hasan M., Samsudin, Khairulmizam, Ramli, Abd Rahman, Azmir, Raja Syamsul and Salam A. Ismaeel, A Review of Navigation Systems (Integration and Algorithms), Australian Journal of Basic and Applied Sciences, 3(2): 943-959, 2009.

Alborzi, Houman, Samet, Hanan. Execution time analysis of a top-down R-tree construction algorithm, Information Processing Letters 101, 6–12, 2007.

Arfe A., Deguy P., Guillot L., Guilly T. L. and Louge R., Android Application for Aalborg University, Project report, 2011.

Anousaki, G., Gikas, V. and Kyriakopoulos K., INS-Aided Odometry and Laser Scanning Data Integration for Real Time Positioning and Map-Building of Skid-Steered Vehicles, 5th Int. Symposium on Mobile Mapping Technology Conference, ISPRS , Padua, Italy, May 28-31, 2007.

Badalia, A. P., Zhanga, Y., Carra, P., Thomas, J. P., Hornseya, R. I., Scale factor in digital camera, Laser Florence 2004.

Bao, Ling and Intille, Stephen S., Activity recognition from user annotated acceleration data. In Proceedings of the 2nd International Conference on Pervasive Computing, pages 1–17, 2004

Bay, Herbert, Tuytelaars, Tinne and Gool, Luc Van, SURF: Speeded Up Robust Features, Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008.

Bell, D.A., Borenstein, J., Levine, S. P., Koren, Y., Jaros, L., An assistive navigation system for wheelchairs based upon mobile robot obstacle avoidance, Proceedings of the IEEE Conference on Robotics and Automation, pp. 2018-2022. 1994.

Bercken, J.V.d., Seeger, B., An evaluation of generic bulk loading techniques, in: International Conference on Very Large Databases (VLDB), pp. 461–470, 2001.

Bernstein, D. and Kornhauser, A., (1998), Map matching for personal navigation assistants, 77th Annual meeting, The Transport Research Board, Jan 11-15,

Washington, D.C. Bounding Rectangles. 9th Annual International Conference, Map India, 2006.

Betke, M, Haritaoglu, E. and Davis, L. S. Multiple vehicle detection and tracking in hard real time. Technical Report CS-TR-3667, University of Maryland, College Park, 1997.

Bileschi, Stanley M., Leung, Brian and Rifkin, Ryan M., Towards component-based car detection. In ECCV Workshop on Statistical Learning and Computer Vision, 2004.

Castro, A. P. A., Demisi, J., Silva,S.D., Sim.P.O., Image based autonomous navigation with fuzzy logic control, Neural Networks, Proceedings. IJCNN '01. International Joint Conference on, vol.3, pp. 2200-2205, 2001.

C. F. Olson et al. Stereo ego-motion improvements for robust rover navigation. In ICRA'01, v. 2, pp. 1099–1104, 2001.

Chen, Jidong, Meng, Xiaofeng, Guo, Yanyan and Xiao, Zhen., Update-efficient Indexing of Moving Objects in Road Networks. In Proceedings of the Third Workshop on Spatio-Temporal Database Management, Seoul, Korea, September 11, 2006.

Chen, Tianen and Shibasaki, Ryosuke. Development of a Vision-Based Positioning System for High Density Urban Areas, GISDEVELOPER, 1999.

Chum, Ondrej. Two-View Geometry Estimation by Random Sample and Consensus, PhD Thesis, 2005.

Cipolla J. W., R., Zha H., Image-based Localization and Pose Recovery Using Scale Invariant Features, Robotics and Biomimetics, 2004.

Davide, S., Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles, IEEE Transactions on robotics, vol. 24., No.5, Oct, 2008.

Derpanis, K. G., The Harris Corner Detector, 2004.

DeVaul, R., and Dunn, S., Real-time motion classification for wearable computing applications. Technical report, MIT Media Laboratory, 2001.

Deselaers, Thomas, Keysers, Daniel, and Hermann Ney, Features for Image Retrieval: An Experimental Comparison, DAGM-Symposium 2004: 228-2362004.

Ding, D., Bambang, P., Karimi, H.A., Roongpiboonsopit, D., Kasemsuppakorn, P., Conahan, T., & Pramana, G. Design Considerations for a Personalized Wheelchair Navigation System. The 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Lyon, France, August 23-26 (2007).

Esteban, I., Dijk, J and Groen, F. Automatic 3D modeling of the urban landscape, International congress on ultra telecommunications and control systems and workshops, 2010.

Foerster, F., Smeja, M., and Fahrenberg, J., Detection of posture and motion by accelerometry: a validation in ambulatory monitoring. Computers in Human Behavior 571-583, 1999.

Forney, G. D., The Viterbi algorithm. Proceedings of the IEEE 61(3): 268–278., 1973.

Freeman, H and Saghri, A Generalized chain codes for planar curves. In Proceedings of the 4th International Joint Conference on Pattern Recognition, Kyoto, Japan, 1978.

Freeman, H, Computer processing of line-drawing images, Computing Surveys 6(1): 57-97, 1974.

Fritz, G., Seifert, C. and Paletta, L., A mobile vision system for urban detection with informative local descriptors, Computer Vision Systems, 2006 ICVS '06. 2006.

Garcia, R.G., Sotelo, M.A., Parra, I., Fernandez, D. and Gavilan, M. 2D Visual odometry method for global positioning measurement, Intelligent Signal Processing, 2007.

Geotools, The Java GIS Toolkit. http://sourceforge.net/projects/geotools, 2007.

Grewal, M. S., Weill, L. R., Andrews A. P., Global positioning systems, inertial navigation, and integration, Wiley, 2002.

Guerrero, J. J., Martinez-Cantin, R., and Sagues, C., Visual map-less navigation based on homographies, J. Robot. Syst., vol. 22, no. 10, pp. 569–581, 2005.

Guttman, A. R-tree: A dynamic index structure for spatial searching. In SIGMOD ’84, Proceedings of the ACM SIGMOD Conference. ACM Press, 1984.

Hagnelius, A., Visual Odometry, Master’s Thesis, 2005.

Hakeem, A., Vezzani, R., Shah, M. Cucchiara, R., Estimating geospatial trajectory of a moving camera, ICPR, 2006.

Harris, C. and Stephens, M.J. A combined corner and edge detector. In Alvey Vision Conference, pages 147–152, 1988.

Haron H, Shamsuddin S M and Mohamed D, A new corner detection algorithm for chain code representation. International Journal of Computer Mathematics 82: 941–950, 2005.

Hartley, R. and Zisserman, A., Multiple view geometry in computer vision, Cambridge University Press, March 2004.

He, Z.Y. and Jin, L.W., Activity recognition from acceleration data based on discrete consine transform and SVM, Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, USA, 2009

Henlich, Oliver. VISION-BASED POSITIONING, 1997.

Hidden Markov model. Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Hidden_Markov_model.

Horn, B., Relative orientation, International Journal of Computer Vision, vol. 4, no. 1, pp. 59-78, January 1990.

Howlett, R J. and Jain, L C (Eds.) Radial basis function networks 2: new advances in design series (Studies in Fuzziness and Soft Computing). New York, Springer, 2001.

Huang, Z. Extensions to the k-means algorithm for clustering large datasets with categorical Values. Data Mining and Knowledge Discovery, 2, p. 283-304. 1998.

Imamura, M., Tomitaka, R., Miyazaki, Y., Kobayashi, K. and Watanabe, K., Outdoor waypoint navigation for an intelligent wheelchair using differential GPS and INS," in SICE Annual Conference, pp.2193-2196, 2004.

Jagadeesh, G. R., Srikanthan, T. and Zhang, X. D., A map matching method for GPS based real-time vehicle location, Journal Of Navigation, 57, 429–440, 2004.

Karimi, Hassan A, Conahan, T. and Roongpiboonsopit, D. A., Methodology for predicting performances of map-matching algorithms, W2GIS 202-213, 2006.

Karimi, Hassan A. Universal Navigation on Smartphones, Spring 2011.

Kalashnikov, Dmitri V., Prabhakar, Sunil, Hambrusch, Susanne and Aref., Walid, Efficient evaluation of continuous range queries on moving objects. In DEXA, 2002.

Kasemsuppakorn, P. and Karimi, H. A., Data requirements and spatial database for personalized wheelchair navigation, 2nd International Convention on Rehabilitation Engineering & Assistive Technology, 2008.

Ke, Q. and Kanade, T., Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and groundlayer detection, in Proc. CVPR Jun. 18–20, 2003, vol. 1, pp. I-390–I-397.

Kitching, Ian D. GPS and cellular radio measurement integration, Journal of Navigation, 2000.

Kitt, B., Andreas, G. and Lategahn, H., Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme, IEEE Intelligent Vehicles Symposium, 2010

Knoblauch, Richard, L., Pietrucha, Martin T. and Nitzbur, Marsha, Field studies of pedestrian walking speed and start-up time, Journal of the Transportation Research Board, Volume 1538, p.27-38, 1996.

Koller, Dieter, Klinker, Gudrun, Rose, Eric, Breen, David, Whitaker, Ross and Tuceryan, Mihran, Real-time vision-based camera tracking for augmented reality applications, In Proceedings of the Symposium on Virtual Reality Software and Technology (VRST-97)

Kotsiantis, S., Pintelas, P. Recent advances in clustering: a brief survey, WSEAS Transactions on Information Science and Applications, Vol 1, No 1 (73-81), 2004.

Krakiwsky, E.J, 1993, The diversity among IVHS navigation systems worldwide, IEEEIEE Vehicle Navigation and Information Systems Conference, Ottawa, 433-436.

Krumm, John, Letchner, Julie and Horvitz, Eric, Map matching with travel time constraints, SAE 2007 World Congress, April 16-19, 2007.

Lankton, Shawn , Sonenblum, Sharon Eve , Sprigle, Stephen , Wolf, Jean , Oliveira, Marcelo, Use of GPS and Sensor-based Instrumentation as a Supplement to Self-Report in Studies of Activity and Participation, Presented at theRESNA Annual Meeting. 2005.

LaMarca, Anthony and Lara, Eyal de, Location Systems: An introduction to the technology behind location awareness, A Publication in the Morgan & Claypool Publishers series, 2008.

Lehtinen, M., Happonen, A., Ikonen, J., Accuracy and time to first fix using consumer-grade GPS receivers, Software, Telecommunications and Computer Networks, 2008.

Levin, A. Szeliski, R. Visual odometry and map correlation. In CVPR 2004, v. 1, pp. 611-618, 2004.

Levine, S. P., Bell, D., Jaros, L., Simpson, R., Koren, Y. and Borenstein, J. The navchair assistive wheelchair navigation system, IEEE Transactionson Rehabilitation Engineering, vol. 7, pp. 443-451, 1999.

Lin, Hung-Yi, Efficient and compact indexing structure for processing of spatial queries in line-based databases, Data & Knowledge Engineering Volume 64, Issue 1, Pages 365-380., January 2008.

Lin, Hung-Yi, Using B+-trees for processing of line segments in large spatial databases, Journal of Intelligent Information Systems, Volume 31, Pages 35-52, 2008.

Lowe, D. G., "Object recognition from local scale-invariant features", International Conference on Computer Vision, Corfu, Greece, September 1999.

Lowe, D.G., Local feature view clustering for 3D object recognition. IEEE Conference on Computer Vision and Pattern Recognition,Kauai, Hawaii, 2001, pp. 682-688.

Lowe, David G. Distinctive image features from scale invariant features, International Journal of Computer Vision, Vol. 60, No. 2, pp. 91-110, 2004.

Luley, Patrick, Paletta, Lucas, Almer, Alexander, Schardt, Mathias, Geo-services and computer vision for object awareness in mobile system application, Location Based Services and TeleCartography, Springer Berlin Heidelberg, 2007.

Malis, Ezio. Survey of vision-based robot control, European Naval Ship Design, Captain Computer IV Forum, ENSIETA, Brest, France, April, 2002.

Matas, J., Chum, O., Urban, M. and Pajdla, T., Robust wide baseline stereo from maximally stable extremal regions. In Proc. of British Machine Vision Conference, pages 384–396, 2002.

Mathie, M.J., Celler, B. G., Lovell, N.H. and Coster, A.C.F., Classification of basic daily movements using a triaxial accelerometer, Medical & Biological Engineering & Computing, Vol. 42, 2004.

Meng, Y. Improved positioning of land vehicle in ITS using digital map and other accessory information, PhD Thesis, Department of Land Surveying and Geoinformatics, Hong Kong Polytechnic University, 2006.

Mikolajczyk, K., Schmid, C., An affine invariant interest point detector, in: Proc. Seventh European Conference on Computer Vision, vol. 2350 of Lecture Notes in Computer Science, Springer Verlag, Berlin, Copenhagen, Denmark, 2002.

Mikolajczyk, K. and Schmid C., Scale and affine invariant interest point detectors. IJC V 60(1):63-86, 2004.

Mikolajczyk, K. and Schmid, C., A performance evaluation of local descriptors, IEEE Transactions on Pattern Analysis and Machine Intelligence, 10, 27, pp 1615-1630, 2005.

Motoko Oe, Tomokazu Sato and Naokazu Yokoya, Estimating camera position and posture by using feature landmark database, Proc.SCIA,pp. 171–181, 2005.

MOBVIS, Software prototype and report on global and local informative features, information society technologies, 2006.

Murtagh, Fionn. Comments on "parallel algorithms for hierarchical clustering and cluster validity", IEEE Transactions on PA analysis and machine machine intelligence, VOL. 14, NO. 10, 1992.

Nogueira, S., Ruichek, Y. and Charpillet, F., A Self Navigation Technique using Stereovision Analysis, in Stereo Vision, pp. 287–298, In Tech Education and Publishing, 2008.

Ochieng, W. Y., Quddus, M. A. and Noland, R. B., Map-matching in complex urban road networks, Brazilian Journal of Cartography (Revista Brasileira de Cartografia) 55 (2), 1–18, 2004.

Oe, M., Sato, T. and Yokoya, N. Estimating camera position and posture by using feature landmark database,

Ohnishia N. and Imiya A., Dominant plane detection from optical flow for robot navigation, Pattern Recognition Letters, Volume 27, Issue 9, 1 July 2006

Ohno, K., Tsubouchi, T., Shigematsu B. and Yuta, S., Differential GPS and odometry-based outdoor navigation of a mobile robot, Advanced Robotics, Vol. 18, No. 6, pp. 611 – 635, 2004.

Olivier, Cappé, Eric Moulines, Tobias Rydén. Inference in hidden markov models, Published by Springer, 2005.

Parra, I. and Sotelo, M. A., Llorca D. F. and Fernandez, C. and Llamazares, A., Visual odometry and map fusion for GPS navigation assistance,

Province of Victoria (Australia) Handbook for GPS Data Collection for Integration with GIS Standards, Specifications and Best Practice Field Guide, http://www.land.vic.gov.au/CA256F310024B628/0/311F3E48EE0204AFCA257110001EFCDE/$File/GPS+Handbook+v7.2.pdf, 2006.

Pires, G., Honório, N., Lopes, C., Nunes, U., Almeida, A. T. Autonomous wheelchair for disabled people, Proc. IEEE Int. Symposium on Industrial Electronics (ISIE97), Guimarães, 797-801.

Quddus, M. A., Ochieng, W. Y., Zhao, L., Noland R. B., A general map-matching algorithm for transport telematics applications, GPS Solutions 7 (3), 157–167, 2003.

Quddus, M. A., Noland, R. B., Ochieng, W. Y., Validation of map-matching algorithm using high precision positioning with GPS, Journal of Navigation 58, 257–271, 2004.

Quddus, M. A., High integrity map matching algorithms for advanced transport telematics Applications, A thesis of the University of London, 2006.

Quddus, M. A., Ochieng, W. Y. and Noland, R. B., Current map-matching algorithms for transport applications: State-of-the art and future research directions, Transportation Research Part C 15, pp. 312-328., 2007.

Rabiner, Lawrence R., A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77 (2), p. 257–286, 1989.

Ravi, N., Dandekar, N., Mysore, P. and Littman, M. L., Activity recognition from accelerometer data, Proceedings of the Seventeenth Innovative Applications of Artificial Intelligence Conference, 11--18, 2005

Ren, Ming, Karimi, H. A. A Chain-code-based map matching algorithm for wheelchair navigation. Transactions in GIS, 2009.

Ren, M. and Karimi, H. A., A hidden Markov model map matching algorithm for wheelchair navigation. Journal of Navigation, Vol. 62, No. 3, pp. 383-395, 2009.

Retscher, Günther and Thienelt, Michael. NAVIO - A navigation service for pedestrains, Journal of Space Communication, Issue No. 9, 2006.

Rizos, C., Trands in Geopositioning for LBS, Navigation and Mapping, 2005.

SagYaraj, F., P.THAMBIDURAI, B.S.BHARADWAJ, G.N.BALAGEI & N.HEMANT, An Improved and Efficient Storage Technique for GIS Geometric Primitives Based on Minimum Bounding Rectangles. 9th Annual International Conference, Map India 2006.

Sander, Jörg, Qin, Xuejie, Lu, Zhiyong, Niu, Nan, Kovarsky, Alex, Automatic extraction of clusters from hierarchical clustering representations, Advances in Knowledge Discovery and Data Mining, Vol. 2637, Springer, 2003.

Sattler, T., Leibe, B., Kobbelt, L., Fast image-based localization using direct 2D-to-3D matching, ICCV 2011.

Se, S. and Jasiobedzki, P., Stereo-vision based 3D modeling for unmanned ground vehicles, in Unmanned Systems Technology, vol. 6561 of Proceedings of SPIE, Orlando, Fla, USA, 2007.

Serre, T., Wolf, L. and Poggio, T., Object recognition with features inspired by visual cortex, in Proc. IEEE Comput. Soc. Conf. Computer Vision Pattern Recognition, 2005, pp. 994–1000.

Steinhoff, U., Omer cevi, D., Perko, R., Schiele, B. and Leonardis, A., How computer vision can help in outdoor positioning, LNCS 4794, pp. 124 – 141, 2007.

Shao, H, Svoboda, T, Tuytelaars, T, Gool L V. HPAT indexing for fast object/scene recognition based on local appearance. In: Conference on Image and Video

Retrieval. vol. 2728 of LNCS. Urbana-Champaign, IL: Springer Verlag; p. 71–80. 2003.

Shao, H, Svoboda T, Van-Gool, L. ZuBuD – Zurich buildings database for image based recognition. Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland. Zurich, Switzerland; 2003.

Simpson, R., LoPresti, E. Hayashi, S., Nourbakhsh, I. and Miller, D. The smart wheelchair component system, Journal of Rehabilitation Research & Development, vol. 41, pp. 429-442, 2004.

Sun, Z., Mao, X., Tian, W. and Zhang, X., Activity classification and dead reckoning for pedestrian navigation with wearable sensors, Measurement science and technology, 2009.

Schiller, J. H. and Voisard, A., Location-based services, Elsevier, Apr 30, 2004.

Tao, Y., Skubic, M., Han, T., Xia,Y. and Chi X., Performance Evaluation of SIFT-Based Descriptors for Object Recognition, Proceedings of the International MultiConference of Engineers and Computer Scientists, 2010.

Tardif, J., Pavlidis, Y., Daniilidis, K.: Monocular visual odometry in urban environments using an omnidirectional camera. In: IEEE IROS'08., 2008.

Taylor, G., Blewitt, G., Steup, D., Corbett, S., Car, A., Road reduction filtering for GPS-GIS navigation, Transactions in GIS, ISSN 1361-1682, 5(3), 193–207, 2001.

Taylor, G., Brunsdon C., Li J., Olden A., Steup D., Winter M., GPS accuracy estimation using map-matching techniques: Applied to vehicle positioning and odometer calibration, Computers, Environments, and Urban Systems 30, 757–772, 2006.

Tele Atlas, Tele Atlas Launches Powerful, Feature-rich MultiNav digital map database, June 2008.

Theiss, A., Yen D. C. and Ku C. Y., Global Positioning Systems: an analysis of applications, current development and future implementations, Computer Standards & InterfacesVolume 27, Issue 2, January 2005.

Tolerico, M.L., Ding, D, Cooper R.A., Spaeth, D.M., Fitzgerald, S.G., Cooper, R., Kelleher, A., Boninger, M.L., Assessing mobility characteristics and activity levels of manual wheelchair users. J Rehabil Res Dev 44(4): 561-72, 2007.

Tuytelaars T. and Mikolajczyk, K., Local invariant feature detectors - Survey. CVG, 3(1):1-110, 2008.

Wu, Dongdong, Zhu, Tongyu, Lv, Weifeng, Gao, Xin, A Heuristic Map-Matching Algorithm by Using Vector-Based Recognition, Computing in the Global Information Technology, 2007. ICCGI 2007.

Wu, Y. H., Lu, B. Y., Chen, H. Y., and Ou-Yang, Y. The development of M3S-Based GPS NavChair and tele-monitor system, in 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, pp.4052-4055, 2005.

White C.E., Bernstein D. and A.L. Kornhauser, Some map matching algorithms for personal navigation assistants. Transportation Research Part C 8: 91-108, 2000.

Venkatraman, K., Karthick, N., Naren, J., Amutha, B., Sensor-Based Dead Reckoning for land vehicle navigation system, International Journal of Recent Trends in Engineering, Vol 2, No. 4, November 2009

Yang, J. S. and Kang, S. P., The map matching algorithm of GPS data with relatively long polling time intervals, Journal of the Eastern Asia Society for Transportation Studies, Vol. 6, pp. 2561 -2573, 2005.

Yariv, Ephraim and Neri, Merhav, Hidden Markov processes, IEEE Trans. Inform. Theory, vol. 48, pp. 1518-1569, 2002.

Zhao, Yilin. Vehicle location and navigation systems, Artech House Inc., 1997.

Zhao, J. Leon, and Cheng, Hsing Kenneth. Graph indexing for spatial data traversal in road map databases. Computers & Operations Research, Volume 28, Issue 3, Pages 223-241. March, 2001.

Zhao, Z.T., Chen, Y.Q., Liu, J.F., Shen, Z.Q. and Liu, M.J., Cross-people mobile-phone based activity recognition, Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 2011.