

大阪市立大学大学院創造都市研究科  
博士学位論文

Development of a Framework for Post-Disaster Road  
Traversability Mapping: An Integrated Approach Using GPS  
Track Sharing and Map-matching

(災害後の経路地図作成フレームワークの開発：GPS 軌跡情報の  
共有とマップマッチングを用いた統合的アプローチ)

2017 年 09 月  
大阪市立大学大学院創造都市研究科  
創造都市専攻都市情報環境研究領域  
D13UD502  
于 文龍 (Yu Wenlong)

Development of a Framework for Post-Disaster Road  
Traversability Mapping: An Integrated Approach Using GPS  
Track Sharing and Map-matching

(災害後の経路地図作成フレームワークの開発：GPS 軌跡情報の  
共有とマップマッチングを用いた統合的アプローチ)

2017 年 09 月  
大阪市立大学大学院創造都市研究科  
創造都市専攻都市情報環境研究領域  
D13UD502  
于 文龍 (Yu Wenlong)

---

キーワード：位置情報サービス, 全地球測位サービス, 遅延耐性ネットワーク, マップマッチング, モバイル端末, 防災

Keywords: Location Based Service (LBS), Global Positioning System (GPS), Delay Tolerant Network (DTN), Map-matching, Mobile device, Disaster prevention

# Abstract

The main motivation in undertaking this research was to address some of the practical issues that were widely reported during major disaster situations such as the 11 March, 2011 Tohoku earthquake in Japan. Apart to great loss of life and property in the near vicinity of the earthquake and the subsequent tsunami event, metropolitan areas such as Tokyo experienced disruption of train services, electrical outages and inaccessibility to Internet services. As a result a vast number of commuters were stranded and compelled to proceed to their destinations using unfamiliar routes and transportation modes. Supporting stranded commuters during and soon after earthquake disasters or other emergency situations have been considered as one of the most important issues for ensuring safety of citizens. It is, therefore, necessary to develop a near real-time traversability mapping service that can be available during or soon after the disaster.

In this research, a workflow for road traversability mapping service was implemented and its performance was evaluated. The traversability mapping workflow is basically designed to perform GPS data collection and sharing between nearby devices when Internet connectivity is unavailable. Mobile Ad-hoc Network (MANET) and Delay Tolerant Network (DTN) were investigated using Android application developed for sharing GPS tracks. The application was tested using 6 Android devices in urban and semi-urban areas in Osaka to evaluate feasibility and efficacy. In order to evaluate a performance with large number of devices, a simulation experiment using the NS-2 simulator was carried out. The field and simulation experiment demonstrate the effectiveness of using a using a combination of MANET and DTN for sharing and aggregation of GPS tracks. The simulation experiment also revealed that in densely populated areas, around 2000 devices could provide complete coverage of entire road network in 60 minutes.

In the next step, pre-processing of GPS tracks collected by individual devices was considered, in order to minimize volume of data transfer over Ad-hoc networks where bandwidth is limited. An Android application for filtering GPS tracks, sharing between mobile devices and aggregation of data was developed. GPS filtering was implemented based on standard parameters such as number of GPS satellites used for positioning, GPS accuracy and Horizontal Dilution of Precision (HDOP) to eliminate low accuracy GPS data. The application also allows for uploading the aggregated data to the server for further processing when the Internet connection becomes available.

A post-processing workflow was implemented for generating updated road traversability map using the aggregated GPS tracks. The post-processing consists of a line simplification based on Douglas-Peucker algorithm and map-matching with existing road network using the Hausdorff distance algorithm. Line simplification was applied reduce the data in further processing and map-matching to remove outliers and generate a navigable traversability road network. Performance of map-matching algorithm was evaluated using data collected in two field campaigns in Yamate-cho, Suita City and Sumiyoshi-ward, Osaka City, Japan. Combined use of line simplification and map-matching algorithm was found to provide desired results to achieve the final objective.

Subsequently, as and when new GPS tracks are loaded to the server, the traversability maps are automatically updated by running the post-processing program on the server. The updated road traversability maps are published as Web Map Service (WMS) using GeoServer and available as a raster layer on any mobile device or computer that is connected to Internet. The traversability road network is stored in a PostgreSQL/PostGIS spatial database and routing functionality is implemented using the pgRouting library to facilitate in guiding users from their present location to target destination.

Map tiles are also generated automatically to support offline use. The road traversability map tiles can be either downloaded when Internet connectivity is available or the tiles can be

share among devices connected through the Ad-Hoc network. As a future work, it is planned to incorporate off-line routing functionality using an embedded database. Lastly, to cater to users who may not have access to smart phone, a Raspberry Pi based digital signage system is deployed to facilitate display the traversability map for public viewing.

The data processing workflow implemented as a part of this research helps overcome several of the limitations of previous research. Some salient features that distinguish this research from previous works are a) Usability by stranded commuters using public transportation rather than private automobile. This is an important factor especially in urban areas in Japan where public transportation is more widely used than private automobiles. b) Most of the functionality is availability both in online as well as offline mode and the system is usable even when Internet connectivity is unavailable. c) The entire workflow is implemented using Open Source software and libraries and, therefore more amenable for future enhancements and customization.

As a future work, it is necessary to consider ways and means to incorporate additional contextual information to the traversability maps to make them more intuitive. Further, additional field experiments or simulation with large number of field devices need to be undertaken to evaluate scalability and robustness of the proposed system. Lastly, it will also be useful to consider low-cost gateway accepting the inbound data sent by field devices and outbound communication to a process running in server and further enhance operability in offline mode.

# Acknowledgements

My profound gratitude goes to all the people who assisted and support me during my doctoral course. First of all, I would like to express the sincere gratitude and appreciation to my supervisor, Prof. Venkatesh Raghavan for his guidance, advice and moral support. His untiring attention made it possible for me do this work as a part of my doctoral course and also helped me mature as researcher during my tenure at the Graduate School for Creative Cities, Osaka City University.

I sincerely appreciate Professor Hiroyuki Ebara for his critical advice on thesis composition and consistent support. I acknowledge Associate Prof. Go Yonezawa and Associate Prof. Daisuke Yoshida for providing constant supports and a lot of encouragements throughout my doctoral course. I would also like to thank Professor Xianfeng Song for many supports, especially on map-matching algorithms. I could not have attained doctoral research objectives if their consistent help was not there.

I sincerely appreciate the Nishimura International Scholarship Foundation for granting me Nishimura scholarship to facilitate my doctoral course study at Osaka City University. In addition, I would like to thank the Ohtsuki Memorial Scholarship Trust Fund for Asian and African Students for granting me Ohtsuki Memorial scholarship to facilitate my master course study at Kansai University.

Thank you all to OSGeo Japan team for providing me a lot of great opportunities, ideas on projects such as Geopaparazzi. I think I have been grown up with their community, many thanks to Mr. Hirofumi Hayashi for supporting Geopaparazzi Cloud codes. I sincerely thank Mr. Naoki Ueda for his invaluable help in preparing the journal manuscript in language and writing style.

I would like to thank also my colleagues Dr. Poliyapram Vinayaraj, Dr. Tran Thi An and Dr. Raito Matsuzaki who always reviewing my work on suggesting several improvements. Thanks also to all members of GIS laboratory at the Osaka City University and AL laboratory at Kansai University for their supports and encouragements.

Finally, my special thanks go to my family for getting me through this period and always supporting me. Especially, my wife Mrs. Shuang Sun for encouraging and support me any time, anywhere with deep affection. I would like to thank all my Japanese friends who made my life in Japan easy and very pleasant that keep me in good shape mentally and physically.

# Contents

<b>Abstract .....</b>	<b>i</b>
<b>Acknowledgements .....</b>	<b>iv</b>
<b>Contents .....</b>	<b>vi</b>
<b>List of Figures .....</b>	<b>viii</b>
<b>List of Tables .....</b>	<b>x</b>
<b>List of Abbreviations .....</b>	<b>xi</b>
<b>Chapter 1 Introduction .....</b>	<b>1</b>
1.1 Review of Related Research .....	3
1.2 Research Objectives .....	5
1.3 Thesis Outline .....	6
<b>Chapter 2 Ad-hoc Network for GPS Track Sharing .....</b>	<b>8</b>
2.1 Introduction .....	8
2.2 Mobile Ad-hoc Network and Delay Tolerant Network .....	8
2.2.1 Mobile Ad-hoc Network .....	9
2.2.2 Delay Tolerant Network .....	9
2.3 Sharing Location Information .....	10
2.4 Field Experiment .....	11
2.4.1 Environment for Field Experiment .....	12
2.4.2 Methods for Field Experiment .....	12
2.4.3 Results of Field Experiment .....	13
2.5 Simulation Experiment .....	13
2.5.1 Environment for Simulation Experiment .....	14
2.5.2 Methodology of Simulation Experiment .....	16
2.5.3 Results of Simulation Experiment .....	18
<b>Chapter 3 Data Processing for Road Traversability Map .....</b>	<b>21</b>
3.1 Introduction .....	21
3.2 Acquiring and Sharing Location Data using Android application .....	22
3.3 OpenStreetMap Data .....	24



3.3.1 Import OSM Road Network Data into Database .....	25
3.4 Line Generalization Using Douglas-Peucker Algorithm .....	26
3.5 Map-matching Using Hausdorff Distance Algorithm .....	28
3.6 Field Experiment .....	30
3.6.1 Environment of Field Experiment .....	30
3.6.2 Results of Field Experiment .....	31
<b>Chapter 4 Map Services for Stranded Commuters .....</b>	<b>34</b>
4.1 Map Services for Stranded Commuters .....	34
4.1.1 Publishing Post-Disaster Road Traversability Map using GeoServer WMS .....	34
4.1.2 Digital Signage Service for Stranded Commuters .....	36
4.2 Sharing Recorded GPS Tracks using Geopap-cloud Web Service .....	37
4.3 Deploying Routing using Road Traversability Map .....	38
4.4 Publishing Offline Road Traversability Map using Tile Map .....	40
4.5 Offline Tile Map Sharing using MANET .....	41
<b>Chapter 5 Summary and Discussions .....</b>	<b>43</b>
<b>Chapter 6 Conclusions and Future Work .....</b>	<b>47</b>
<b>References .....</b>	<b>51</b>
<b>Appendix A .....</b>	<b>82</b>
<b>Appendix B .....</b>	<b>92</b>
<b>Appendix C .....</b>	<b>115</b>

# List of Figures

## Chapter 1

Figure 1.1: The framework of post-disaster road traversability mapping system .....	58
Figure 1.2: Flowchart showing post-disaster road traversability workflow .....	59

## Chapter 2

Figure 2.1: Assimilating map data from multiple users .....	60
Figure 2.2: Map of study area around the Asahi Ward, Osaka City.....	60
Figure 2.3: Simulated GPS tracks obtained for Asahi Ward, Osaka City .....	61
Figure 2.4: Completion of mapping related to time (2000nodes~3000nodes) .....	61
Figure 2.5: GPS data transfer time .....	62
Figure 2.6: Packet reception rate .....	63

## Chapter 3

Figure 3.1: Screenshots of Android application shows data collection, filtering and sharing to server. (a) Location data information (b) NMEA data store on a device (c) Automatic network (d) Update location data to server construction .....	63
Figure 3.2: OSM data around Sumiyoshi-ward, Osaka City displayed in QGIS .....	64
Figure 3.3: Road network superimposed over OSM in QGIS .....	64
Figure 3.4: Line simplification steps in Douglas-Peucker algorithm .....	65
Figure 3.5: Results of Douglas-Peucker line simplification in Osaka City area .....	65
Figure 3.6: Study area (a) Yamate-cho, Suita City and (b) Sumiyoshi-ward, Osaka city in (c) Osaka Fu .....	66
Figure 3.7: GPS tracks collected using six mobile devices in (a) Yamate-cho, and (b) Sumiyoshi-ward .....	67
Figure 3.8: Roads networks and buffered GPS tracks before map-matching (a) Yamate-cho, and (b) Sumiyoshi-ward.....	68
Figure 3.9: Results of map matching (a) Yamate-cho, and (b) Sumiyoshi-ward based map-matching .....	69

## Chapter 4

Figure 4.1: GeoServer connecting to PostGIS database .....	70
Figure 4.2: Road traversability map displayed as WMS layer in GeoServer .....	70
Figure 4.3: Illustration of Raspberry Pi based digital signage system .....	71
Figure 4.4: Sharing GeoServer WMS layer using Concerto signage system .....	72
Figure 4.5: View of road traversability map in Concerto signage system.....	72
Figure 4.6: Sharing recorded GPS tracks using Geopap-Cloud .....	73
Figure 4.7: GPS tracks are displayed in Geopap-Cloud .....	73
Figure 4.8: Shortest path using pgRouting between point A and B inside the traversability road map area a) Normal situation b) Using traversability road map .....	74
Figure 4.9: Shortest path using pgRouting between point A and B outside the traversability road map area a) Normal situation b) Using traversability road map .....	75
Figure 4.10: Illustration of tile map.....	76
Figure 4.11: Map tiles of traversability road map .....	76
Figure 4.12: Tiled traversability road map is displayed in Geopaparazzi .....	77

# List of Tables

## Chapter 2

Table 2.1: Parameters used for field experiments .....	78
Table 2.2: Parameters used for simulation experiments .....	78

## Chapter 3

Table 3.1: Sample of acquired location data .....	79
Table 3.2: Road network attribute data used for road traversability mapping system .....	80
Table 3.3: Parameters used for field experiments .....	80
Table 3.4: Results of field experiments .....	81

# List of Abbreviations

API	Application Programming Interface
DBMS	Database Management System
DOP	Dilution of Precision
DTN	Delay Tolerant Network
FANET	Flying Ad-hoc Network
FOSS4G	Free and Open Source Software for Geospatial
GIS	Geographical Information System
GPS	Global Positioning System
GPX	GPS eXchange Format
GSI	Geographical Survey Institute
HDOP	Horizontal Dilution of Precision
LBS	LBS Location Based Service
OGC	Open Geospatial Consortium
OSM	OpenStreetMap
MANET	Mobile Ad-hoc Network
NMEA	The National Marine Electronics Association
NS-2	Network Simulator Version 2
TTL	Time to Live

UAV	Unmanned Aerial Vehicle
VANET	Vehicular Ad-hoc Network
WMS	Web Map Service
WMTS	Web Map Tile Service

# Chapter 1

## Introduction

In recent years, with the development of mobile communication technology and Geographic Information System (GIS), location-based services are become extensively important in day-to-day life. Nowadays, geospatial technology is widely used for disaster management and prevention endeavors. Location information based mapping tools are developing rapidly and can be used widely in the field of disaster relief.

By the factor of geographical location, natural disasters are inevitable in Japan. Recently, Great Hanshin-Awaji Earthquake Disaster in 1995 and The Tohoku Earthquake in 2011 caused huge damages<sup>1</sup>. The Cabinet Office of the Government of Japan has reported that the probability of occurrence of the Nankai Megathrust Earthquakes with in the next 30 years reaches 70%<sup>2</sup>. Apart from great loss of life and property in the near vicinity of the earthquake and the subsequent tsunami event, metropolitan areas such as Tokyo experienced disruption of train services, electrical outages and inaccessibility to Internet services. As a result a vast number of commuters were stranded and compelled to proceed to their destinations using unfamiliar routes and transportation modes. Supporting stranded commuters during and soon after earthquake disasters or

---

<sup>1</sup> <http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h23/pdf/n0010000.pdf>

<sup>2</sup> [http://www.bousai.go.jp/jishin/nankai/tyosabukai\\_wg/pdf/h281013shiryo-03.pdf](http://www.bousai.go.jp/jishin/nankai/tyosabukai_wg/pdf/h281013shiryo-03.pdf)

other emergency situations have been considered as one of the most important issues for ensuring safety of citizens. It is, therefore, necessary to develop a near real-time traversability mapping service that can be available during or soon after the disaster.

Recently, location based mapping tools have been developing rapidly and can be applied widely in the field of disaster relief. For instance, during The Tohoku Earthquake in 2011, an automobile manufacturing company demonstrated the use of near real-time Floating Cellular Data to provide usable routes and shared them to drivers<sup>3</sup>. When road conditions are altered by natural disaster, distributing information of blocked road to emergency vehicle is very important. However, it is difficult to do so if telephone or the Internet is down or busy. It is, therefore, necessary to develop a real time information sharing system under off-the-network situation.

The impact of the disaster can be significantly reduced if a post-disaster road traversability map is provided to the stranded commuters. In order to address post-disaster navigation problems that may occur not only in Japan but also in any parts of the world, we propose a system for post-disaster road traversability mapping. This will support the stranded commuters using Mobile Ad-hoc Network (MANET)<sup>4</sup>, Delay Tolerant Network (DTN)<sup>5</sup>, GPS data sharing and map-matching.

---

<sup>3</sup> [http://www.drs.dpri.kyoto-u.ac.jp/projects/jitsumusha/18/09\\_amano.pdf](http://www.drs.dpri.kyoto-u.ac.jp/projects/jitsumusha/18/09_amano.pdf).

<sup>4</sup> [http://itl.nist.gov/div892/www/wahn\\_mahn.shtml](http://itl.nist.gov/div892/www/wahn_mahn.shtml)



## 1.1 Review of Related Research

This section describes the previous research attempts to attain efficient post-disaster road traversability map, merits and demerits of the available systems. Recently, several researchers have used GIS for better disaster relief support. Previous researches on post-disaster traversability mapping were mainly based on Internet availability (Weiser *et al.*, 2007). Further on, researches were focused on developing a disaster evacuation guidance method even when Internet is down or busy (Fujihara *et al.*, 2013) using mobile gaming device. Nonetheless, the previous research attempts were not to develop a near real-time post traversability mapping. The improvements made in the present research as compared with previous researches are described below.

- a) Estimating disaster situation: Minamimoto *et al.*, (2010) used location information and data of buildings based on communication log among mobile devices and GPS log of devices. The objective of their research was to estimate geographic information and building shape after disaster. However, the limitation in this research is that it is unsuitable for large areas due to time constraints. Their simulation result shows the time required to collect data logs with consistent estimation quality increases logarithmically with increase in geographical extent of the target. Therefore, this research was taken up considering the need to maintain estimation quality with in a given time constraint.

---

<sup>5</sup> [http://ipnsig.org/wp-content/uploads/2012/07/DTN\\_Tutorial\\_v2.03.pdf](http://ipnsig.org/wp-content/uploads/2012/07/DTN_Tutorial_v2.03.pdf)

- b) Disaster evacuation guidance method: Data used for Fujihara *et al.*, (2013) is passing-by-communication for mobile device, which is popularly used in mobile game devices. The objective of the research was to map the near real-time routes to evacuation centres after disaster and also examine the condition of the evacuation centres. Evacuation staff would collect information about damaged place, and share by passing-by-communication mechanism. Device uses Bluetooth communication and shows evacuating route that avoid damaged places. The limitation of this system is that the user needs to download map and location of evacuation places before using the application. Therefore, if disaster occurs, when people are not at their usual commuting route, they may not be able to use system.
- c) Real-time urban traffic monitoring: Shi *et al.*, (2010) collecting GPS traced data from GPS enabled vehicles on road networks, and update traffic condition such as traffic jams in near real-time. The proposed method utilizes GPS track data to successfully distinguish between traffic signal and jam. Also, experiment was done with GPS taxi scheduling data of Shanghai, China, and evaluate efficacy of this system. The lacuna in this system is that data is based on vehicle's movement, in case of traffic jam and many vehicles are stationary, the system may not be able to collect adequate data.

The major improvements made in this research compared to above mentioned approaches are that the proposed workflow can create post-disaster road traversability map for wide area by using GPS enabled mobile devices, without the need to download data. Communication is carried out on MANET and DTN,

which is not dependent on public network, so this research is applicable for disaster situation even when network connectivity is unavailable. In addition, this research also proposes a method to collect data using pedestrian stranded commuters and construct road information where people can walk through, even under traffic jam, and provide the post-disaster road traversability map during disaster situation that affect relatively large areas.

## **1.2 Research Objectives**

With the aim of providing a viable solution for supporting stranded commuters and resolving some of the limitations in existing solutions, this research is taken up with the following main objectives:

- a) Development of mechanism for exchange of GPS tracks over MANET and DTN and evaluate the efficacy and performance for field data aggregation in urban and semi-urban areas.
- b) Establish a workflow for processing of aggregated GPS data and enable the generation navigable road traversability maps with due consideration of scenarios during and after disaster or emergency situations.
- c) Providing multiple conduits for information delivery to address typical post-disaster issues such as power outages, disruption in network connectivity and lack of available devices to access information.
- d) Facilitating online and offline usability of traversability maps and value addition through incorporation of routing services.

- e) Deploying a comprehensive framework for data gathering, processing and sharing of results based on Open Source software, Open Data and Open Standards.

The framework and data processing workflow of the proposed post-disaster road traversability mapping system is shown in Figure 1.1 and Figure 1.2 respectively.

### **1.3 Thesis Outline**

This dissertation is divided into six Chapters and organized as follows:

- Chapter 1, the present Chapter provides an introduction to the background of the post-disaster road traversability mapping system, objective of the study and thesis outline.
- Chapter 2 explains the implementation of networks and communication characteristics of MANET and DTN using the NS-2 simulator. Further in order to demonstrate the effectiveness of the proposed networks, field evaluation and simulation experiment were carried out at two study areas in Osaka City, Japan.
- Chapter 3 explains some data filtering functions to mitigate positioning errors. Further, the implementation of map-matching algorithms for post-disaster road traversability mapping is presented and the map-matched results are also demonstrated.

- Chapter 4 outlines the application examples of post-disaster road traversability mapping system. It also describes deployment of interoperable Web services for publishing created post-disaster road traversability map and sharing recorded GPS tracks.
- Chapter 5 discussed several advantages and problems related to the system framework. Here, the functions and workflow of the system are explained in detail.
- Chapter 6 enumerates the salient features, discusses the main conclusions that can be drawn from this research and also elucidates some suggestions for future works.

# **Chapter 2**

## **Ad-hoc Network for GPS Track Sharing**

### **2.1 Introduction**

Chapter 1 presented the problem to be addressed, aims of this study and the outline of the proposed solution. This Chapter explains the implementation of network, communication characteristics of MANET and DTN. To demonstrate the effectiveness of the proposed method, a field evaluation and simulation experiment has been performed. The efficacy and performance of the system was difficult to demonstrate using large numbers of mobile devices in the field experiment. Therefore, simulation experiment was carried out with large numbers of simulated devices to evaluate the network performance.

### **2.2 Mobile Ad-hoc Network and Delay Tolerant Network**

In this research, data sharing has been performed using MANET and DTN. Even when Internet connection is unavailable, the proposed system can get data from nearby devices connected through Ad-hoc networks. Further, DTN used to transfer data even in the situation of network interruption or disconnection, such as in the situation of earthquake disaster event. In the present case, we propose a Hybrid DTN-MANET routing protocol (HYMAD) (Whitbeck *et al.*, 2010) that can integrate and use both MANET and DTN. In addition, as shown in Figure 2.1,

combining each GPS tracks logged in individual devices can help create post-disaster road traversability map.

### **2.2.1 Mobile Ad-hoc Network**

MANET is a type of Ad-hoc network that can configure itself and change location on the fly (Joshi, 2010). Ad-hoc network can be built anytime, anywhere using any wireless enabled devices and connect various wireless networks to provide end-to-end connectivity. A number of Ad-hoc networks have been proposed by several authors, for instances, a Vehicular Ad-hoc network (VANET) (Wang and Li, 2009) consists of smart vehicles on the road and provides communication services among nearby vehicles or with roadside infrastructure. It is envisioned to provide numerous interesting services. A Flying Ad-Hoc Network (FANET) (Bekmezci *et al.*, 2013) has been developed for creating Ad-hoc network connecting for UAV (Unmanned Aerial Vehicle). One of the most important design problems for multi-UAV system is the communication between them. However, mobile devices based Ad-hoc is more reliable in terms communication.

### **2.2.2 Delay Tolerant Network**

DTN was originally proposed to contact satellites and transfer data between satellites and station in earth. Further, DTN has been extensively used to solve communication issues between mobile devices. DTNs were designed to overcome issues about connectivity, such as mobility, poor infrastructure, and the short-range of radio communications. In fact, there is an experimental project

such as TIER<sup>6</sup> proposed to provide an Internet connection in sparsely populated areas without communicational infrastructures. However, network throughput will decrease with increase in number of devices and there is a potential to high network delays. Thus, there are a few proposed methods to increase contact opportunities. For example, using Throwboxes in DTN (Zhao *et al.*, 2006), have proposed ways to enhance network capacity in mobile DTN. Throwboxes are inexpensive wireless storage devices, which act as intermediate nodes between source and destination in wireless networks.

DTN was designed to enable communications between mobile devices and allows long communication delay. Such situations are expected after large disaster when network infrastructure has been damaged and communication is disabled or other devices do not exist within communication range, DTN network can still transmit location data once the communication retained.

## **2.3 Sharing Location Information**

In the present case, we apply HYMAD for creating network, which uses both MANET and DTN. Firstly, each mobile device is grouped with other devices available in neighborhood, to establish mobile Ad-hoc network connection. Secondly, DTN network would be propagated to multiple groups and flooding routing protocol used to transmit data to all devices. Although flooding routing can transmit information fast, it may result to network overhead and packet loss and decrease communication quality. This issue can be solved by a defined limit

---

<sup>6</sup> <http://tier.cs.berkeley.edu/drupal/>



for number of times of transmission using Time-To-Live (TTL)<sup>7</sup>. TTL is a mechanism that limits the lifespan of the transmitted data in network. TTL provides information to the network router on whether the transmitted data has resided in the network for long time and should be discarded. TTL controls how many times the information should be transmitted from one device to the others.

The workflow of the algorithm, used to transmit data is described below:

- a) Each mobile device transmits GPS track information to all devices within the communication range.
- b) Screening of device is carried out each time it enters into the communication range and checked whether device has already received data from other devices or not by using device ID.
- c) If data has already received from a particular device ID, time of data receipt would be checked before considering the GPS track information. Therefore, the same device transmits GPS track information from different places, which will be considered as a new data.

## 2.4 Field Experiment

In this section, the performance of near real-time mapping that proposed in this thesis has been evaluated. Study demonstrates the experiment result using mobile device (Nexus7). Several field evaluations have been performed in order to demonstrate the effectiveness of the proposed method.

---

<sup>7</sup> [https://docs.datastax.com/en/cql/3.3/cql/cql\\_using/useExpire.html](https://docs.datastax.com/en/cql/3.3/cql/cql_using/useExpire.html)

### **2.4.1 Environment for Field Experiment**

Field experiment was carried with four mobile devices. Mobile devices are connected via an Ad-hoc network using Wi-Fi function without using Internet connection. To evaluate radio wave environment effect, experiment is carried out at two study areas. The first one is residential area Sugimoto-cho area in Osaka City and the second one is downtown Umeda area in Osaka City, Japan.

Prior to the experiment, confirmation about presence of wireless signals has been carried out. Field investigation using mobile device has been confirmed that there are few Wi-Fi access point in Sugimoto-cho area and interference from wireless networks is limited. On the other hand, many Wi-Fi access points exist in Umeda area. In terms of wireless signal interference it has been observed that, Sugimoto-cho is generally similar to the areas of recent disaster events.

### **2.4.2 Methods for Field Experiment**

In the field experiment, each mobile device is assigned with a fixed IP address. Four mobile devices have been used for field experiment and each mobile device receives 15 GPX files from other neighborhood devices. Further, data transfer rate and transfer time are recorder for each mobile device. GPX files used in experiment are real data logs that are recorded by mobile device for 15 minutes. Location data size is about 13KB, on an average, after removing invalid GPX data. Stable communication range is fixed as 30m.

The complete process of sending data to form a particular device to all other devices is defined as a cycle. Field experiments are carried out for three cycles. Characteristics of parameters of field experiment are shown in Table 2.1.

### **2.4.3 Results of Field Experiment**

In Sugimoto-cho area, the average time of sending one GPX file from one device is calculated as 547.38ms (communication time) at 100% communication rate. In this case, the communication rate is the percentages of amount of successfully transmitted GPX files. In case of Umeda area communication time was 726.19ms and the communication rate was 97.96%. Based on the result, good wireless signal condition area like Sugimoto-cho, communication rate is assured and can be exchange data in short time. Also in downtown area like Umeda, where there are many smart phone users and access points, the experiment shows communication rate of 97.96%.

In light of the above, it suggested that the proposed method is suitable to area like Sugimoto-cho that is similar with disaster areas in-terms of availability of wireless access points.

## **2.5 Simulation Experiment**

Only field experiment is not enough to predict the efficacy of the system because difficulty in carrying out field experiment with large number of devices. Therefore, in addition to field experiment a simulation experiment is carried out to evaluate the network performance. In this simulation the degree of perfection

of map and the number of mobile devices and time taken to create map are considered. The output file of the simulation is packet arrival factor, and transmission time of location data. DTN network has been used to transmit the data in the simulation experiment.

In case of connection between sender and receiver devices has interrupted, relay device stores the data and resume transmission when connection with receiving device is re-established.

### **2.5.1 Environment for Simulation Experiment**

The simulation experiments were carried out of Asahi Ward, Osaka City (Figure 2.2) as a target area. The base map used to assist simulation experiments indicating available route in Asahi Ward, Osaka City was obtained from Geographical Survey Institute (GSI)<sup>8</sup>, Japan. Where roads are denoted by black color, building or houses are denoted as grey colour. River, pond and water place are denoted as blue color. Park and place with vegetation are denoted as green color. The simulation program is configured such that the mobile devices move only on the road that can be traversed and road map that can be used for transmit are created.

Details of the parameters used in simulation experiment are shown in Table 2.2. Simulation experiment carried out is explained below in detail.

a) Number of mobile devices

---

<sup>8</sup> <http://www.gsi.go.jp/tizu-kutyu.html>

The number of stranded commuters in Tokyo during the Great East Japan Earthquake was reported as 5,150,000 persons, while the total population in Tokyo is 37,000,000 persons<sup>9</sup>. Total population in the simulation target area (14km<sup>2</sup>) is estimated to be about 202,600 persons according to the population data of Asahi Ward, Osaka City<sup>10</sup>. Therefore the stranded commuters in experiment area can be estimated to be about 28,000 persons using the same ratio in the case of Tokyo. Assuming 10% of them has mobile device and use the proposed system for GPS data sharing. The number of mobile devices is assumed between 2000 and 3000. A simulation run with the number of devices is 2000, 2200, 2400, 2600, 2800, and 3000.

b) Setting of moving speed

Moving speed of people is set to 1.2m/s according to human's average moving speed. Daytime population in Asahi Ward, Osaka City consists of 22% of floating population and 78% of residential population<sup>11</sup>. In case of floating population, we assume their home is outside of experiment area, and model movement towards their home direction until they reach outside limit of experiment area. In case of residential population we assume that their home is inside experiment area, and follows random waypoint (Matsuzaki *et al.*, 2013). Random waypoint is a model to select destination randomly, and move toward them. In order to have fair estimate

---

<sup>9</sup> <http://www.metro.tokyo.jp/INET/KEIKAKU/2012/11/DATA/70mbd101.pdf>

<sup>10</sup> <http://www.city.osaka.lg.jp/contents/wdu020/asahi/english/gaiyo/index.html>

<sup>11</sup> <http://www.city.osaka.lg.jp/contents/wdu020/english/>

of destination selection, each residential population is assigned to random destination initially and the system intended to show user the shortest way to their home. Note that, normally random way point model may have dead-ends by limiting movable area by obstacles such as wall. So in this simulation, solve this issue by moving along with obstacles when it cannot cross the obstacles.

c) Communication range

In this study, a Two-ray Ground Reflection Model is assumed. Communication range is set to 30meters based on real measurement. Communication protocol between devices is IEEE802.11n.

d) Limiting the number of times of data transmission

In this study, limit the transmitting rate using TTL to reduce network load. TTL is set to 50 . As long as TTL is not 0, device will transmit to proximal devices.

## **2.5.2 Methodology of Simulation Experiment**

The experiment is carried out to evaluate system performance by simulated movement of stranded commuters. Using modelled result, GPS trace logs of device movement on map are recorded, and network simulator is run for evaluating communication status. Network Simulator Version 2, widely known as NS-2<sup>12</sup>, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols can be done using NS-2.

---

<sup>12</sup> <http://www.isi.edu/nsnam/ns/>

Using this data, each road link is numbered by road-width-priority search. Road link is considered as a line between one intersection and the next. The application evaluates the status of the road whether the road is able to travel or not by a mobile device that traversed the road initially. Meaning that the road is usable for transportation if one mobile device has already went through the particular road. Any two devices close in communication range, share information, merge GPS track data, and update the map. *Comp*, denoting the completion of map is calculated as shown below:

$$Comp = \min \left( \frac{EdgeNum_{node_i}}{EdgeNum_{total}} \right) \times 100 \quad (1)$$

Here,  $EdgeNum_{node_i}$  is total number of road tracks that each mobile device acquires.  $EdgeNum_{total}$  shows the number of all road tracks on map. Completion rate of map is the minimum value of road tracks in device divided by all road links in map area. Equation for computing packet reception rate  $Packet_{rate}$  of network is shown below:

$$Packet_{rate} = \frac{Packet_{rece}}{Packet_{send}} \times 100 \quad (2)$$

Where  $Packet_{rece}$  is total amount of received packet and  $Packet_{send}$  is total packet sent. Even within same communication range (30m), packet reception

rate will decrease by increasing the number of devices. This is because increasing the number of devices increases the potential to high network load and bandwidth limitation. Simulation experiments using NS-2 configuration files and codes can see in Appendix A.

### **2.5.3 Results of Simulation Experiment**

Figure 2.3 shows the result of simulation experiment of Asahi Ward, Osaka City after 3600 seconds. Red line in figure shows moving path. Each mobile device starts with no map and the time to complete of map and communication status is evaluated.

Figure 2.4 shows graph of completion of mapping related with time for 2000, 2200, 2400, 2600, 2800 and 3000 when mobile devices are used for simulation. The result is calculated by averaging 10 results at each case. In order to evaluate minimum number of devices, the simulation run with device number of 100, 400, 700, and 1000. Based on the experimental result for 2000, 2200 and 2400 devices take 50 minutes to reach 100% map completion rate. A number of 2600, 2800 and 3000 devices need 40 minutes to reach 100% completion rate and each mobile device could share completed maps. Focus on the number of mobile devices, 1000 devices needs 60 minutes, and 3000 devices need 40 minutes to accomplish 100% completion rate. Therefore, the usage of more devices is potential to reduce the completion time significantly and improve the quality of the map in a given time duration.



Further, evaluation of data transfer time has been carried in terms of number of devices used for simulation experiment. In Figure 2.5, X-axis shows the number of devices (nodes) and Y-axis denotes the data transfer time. According to experiment result, it is noted that while number of devices are increased data transmission time is decreased. It is observed that increased data transfer time was required in case of experiments that carried out with relatively less number of devices. In case of less number of devices, more time is required to reach the communication range due to the increased distance between the proximal devices. Meanwhile in the case of relatively higher number of devices being used, less data transfer time is need due to shorter distance between the adjacent devices.

In addition to the previous evaluation performance, average road density is calculated in order to estimate the reasonable number of devices is required to complete the map with in particular time. Average road density (total road length in 1 km<sup>2</sup>) is 28.7km/km<sup>2</sup>, according to data from GSI, Japan<sup>13</sup>. Simulation result shows, study area at Asahi Ward, Osaka City (14km<sup>2</sup>), 2000 mobile devices can reach 100% completion rate within 50 minutes

Packet reception rate within communication range of 30m is shown in Figure 2.6. In case of Sumiyoshi Ward (9.4km<sup>2</sup>) that has a road density of 27.1km/km<sup>2</sup>, it is estimated that 1400 mobile devices can make 100% completion map within 50 minutes. In case of Tokyo Shibuya Ward (15.1km<sup>2</sup>), average road density is higher

---

<sup>13</sup> <http://nlftp.mlit.go.jp/ksj-e/gml/datalist/KsjTmplt-N04.html>

(30.9km/km<sup>2</sup>), 2400 devices will be needed to complete 100% map within 50 minutes.

Two field experiments were carried out at Osaka City and Suita City to evaluate the performance of the system. The detailed explanation for the field experiments has given in the following data processing for road traversability map Chapter 3.

# **Chapter 3**

## **Data Processing for Road Traversability Map**

### **3.1 Introduction**

Chapter 2 focused on acquiring GPS location data, MANET and DTN to implement post-disaster road traversability mapping system. The field experiment and the simulation experiment were also demonstrated. Results show that there is some issues exist with GPS location and navigation system, such as insufficient positional accuracy. Therefore, this Chapter discusses about the improvements made to address these issues.

This Chapter describes methodology and data processing workflow for sharing GPS tracks data and filtering method use to remove low accuracy GPS data. Location information is filtered using HDOP parameter of NMEA format in order to overcome accuracy problems. Further, GPS track data is uploaded to the server and data generalization using Douglas-Peucker algorithm was carried out to reduce the amount of location data and speed up processing. The system match the stranded commuter's GPS track data to the nearest street node and users are able to find the traversability path to reach their destination using post-disaster road traversability map. The post-disaster road traversability mapping created does not include the data like road intersection, buildings, etc. OpenStreetMap (OSM) offline map has been used as background map to enhance the utility for navigation.

Further, the performance of post-disaster road traversability mapping that originally proposed by this research has been evaluated. Study demonstrates the experiment result using mobile device (Nexus7). Field experiments were carried out at Sumiyoshi-ward, Osaka City and Yamate-cho, Suita City, Japan to evaluate the performance of map-matching algorithm.

### 3.2 Acquiring and Sharing Location Data using Android application

In this study, an Android<sup>14</sup> application was implemented to acquire and share location data between mobile devices. The user-friendly Android application for mobile device is developed in Java language<sup>15</sup> using Java Development Kit (JDK)<sup>16</sup>, Java Software Development Kit (SDK)<sup>17</sup>.

Collected GPS tracks are filtered using parameters such as satellite numbers, GPS accuracy and HDOP parameters. The horizontal positioning can be determined with minimum of three satellites available. In practice, a minimum of four satellites is needed to determine altitude value. Therefore, we filter GPS data, acquired more than four satellites using the function *gpsStatus.getSatellites()*<sup>18</sup>.

---

<sup>14</sup> <https://www.android.com>

<sup>15</sup> <https://www.java.com/>

<sup>16</sup> <http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html>

<sup>17</sup> <https://developer.android.com/studio/index.html>

<sup>18</sup> [https://developer.android.com/reference/android/location/GpsStatus.html#getMaxSatellites\(\)](https://developer.android.com/reference/android/location/GpsStatus.html#getMaxSatellites())

Secondly, is selected according to the size of the road in the study area. In this study, 24m<sup>19</sup> is assigned as the maximum road size, defined in Japan's Road Traffic Act. The GPS accuracy of the location data is estimated by using Android function called *location.getAccuracy()*<sup>20</sup>. Finally, HDOP represents the accuracy of GPS in horizontal space and time. HDOP is a key factor in determining the relative accuracy of a horizontal position. The position with a lower HDOP value means a potentially high positioning accuracy, although the reliability should be cross checked with ground truth data. Therefore, one can expect a low position accuracy if the HDOP value is higher. However, HDOP can be effectively used to remove deviant position points from the GPS location information using *NmeaGpogg* function. Figure 3.1(a) shows the filtered GPS data interface on single mobile device using satellite numbers, GPS accuracy and HDOP parameters. Figure 3.1(b) shows NMEA data stored on a device.

Further, filtered location data are saved as customized file format. The format of the saved location data is shows as below:

*\$-- ID/ yyyy-mm-dd/ hhmmss/ Device model/ HDOP/ Accuracy/ Satellite numbers/ Latitude/ Longitude*

Table 3.1 shows an example of saved location data.

As explained in Section 2.3, the Android application shares location data between nearby devices that are connected through MANET. The mobile devices

---

<sup>19</sup> [http://www.japaneselawtranslation.go.jp/law/detail\\_main?re=&vm=&id=2724](http://www.japaneselawtranslation.go.jp/law/detail_main?re=&vm=&id=2724)

<sup>20</sup> [http://developer.android.com/reference/android/location/Location.html#getAccuracy\(\)](http://developer.android.com/reference/android/location/Location.html#getAccuracy())

automatically recognize and connect to each other when they are within communication range. Not only the Android application shares data between closely surrounding devices but also data transfer to other devices using MANET through surrounding devices. More devices in the MANET, the greater will be the communication range. Figure 3.1(c) shows status of network connection.

Finally, the system allows the Android application to upload the collected location data to server using relative path. A PHP script is executed on the server to receive location data and transfer the location data to PostgreSQL<sup>21</sup>/PostGIS<sup>22</sup> database. Figure 3.1(d) shows the interface to upload the location data to server. Acquiring and sharing location data using Android application configuration files and codes show in Appendix B.

The next section discusses in detail the post-processing carried out in the server such as line generalization and map-matching. Freely available OSM data has used as reference map for map-matching.

### **3.3 OpenStreetMap Data**

OSM is an open project to create a free editable map of the world. OSM's data is free to use by anyone, for any purpose. OSM is also a good data source to use for map-matching, as it is freely available and has no technical restrictions in terms

---

<sup>21</sup> <https://www.postgresql.org>

<sup>22</sup> <http://www.postgis.net>

of processing the data. In this study, collected location data using mobile devices overlay with road network data collected from OSM.

OSM data around Yamate-cho, Suita City and Sumiyoshi-ward, Osaka City, Japan were downloaded from OSM web site<sup>23</sup>. OSM data used in QGIS<sup>24</sup> is shown in Figure 3.2. The Coordinate Reference System (CRS) used in the Android application is Mercator projection (EPSG: 900913)<sup>25</sup>, therefore, the following parameter is needed to add to PROJ4 Open Source cartographic projections libraries in order to synchronize the CRS GPS data with OSM data (Yoshida, 2007).

```
<900913> +proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0  
+x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +wktext +no_defs <>
```

### 3.3.1 Import OSM Road Network Data into Database

OSM road network data was obtained in ESRI Shape file format<sup>26</sup>. PostGIS<sup>27</sup> command (shape2pgsql)<sup>28</sup> is used to upload into database. Feature types and road

---

<sup>23</sup> <http://download.geofabrik.de>

<sup>24</sup> <http://www.qgis.com>

<sup>25</sup> <http://crschmidt.net/blog/archives/243/google-projection-900913/>

<sup>26</sup> <http://www.digitalpreservation.gov/formats/fdd/fdd000280.shtml>

<sup>27</sup> <http://www.postgis.net>

<sup>28</sup> [http://www.bostongis.com/pgsql2shp\\_shp2pgsql\\_quickguide.bqg](http://www.bostongis.com/pgsql2shp_shp2pgsql_quickguide.bqg)

classes were automatically stored in the spatial database. Table 3.2 shows basic road network attribute data used for post-disaster road traversability mapping. *GID* is a code assigned to the road and *OSM\_id* is used to identify the road segment. *The\_geom* is the geometry type of the road. That means calculations on geometries (areas, distances, lengths, intersections, etc.) can be done using Cartesian mathematics and straight-line vectors<sup>29</sup>. Figure 3.3 shows OSM road network data.

The up-to-date global OSM road network data can be extracted from the OSM database. Therefore, the framework for post-disaster road traversability mapping suggested by this study can be applied not only in this study area but also in any parts of the world.

In the next section, the implementation of line generalization using Douglas-Peucker algorithm is discussed. This reduces the amount of GPS track data to minimize system load and data communication overheads.

### **3.4 Line Generalization Using Douglas-Peucker Algorithm**

Filtering location information data on mobile devices can reduce the number of GPS track points. However, some queries extract a large volume of data and need a bit of time to display the result, in addition, processing gets slow if the extracted data is large. In order to reduce the amount of GPS track points, a line generalization function was implemented using the Douglas-Peucker algorithm

---

<sup>29</sup> [http://postgis.net/docs/using\\_postgis\\_dbmanagement.html](http://postgis.net/docs/using_postgis_dbmanagement.html)



(Douglas and Peucker, 1973). The algorithm could produce the closest approximating results possible by a human being manually simplifying a line. It recursively splits the approximating poly-line at the vertex of furthest distance, keeping those vertices under a given error bound or tolerance (Figure 3.4).

Song *et al.*, (2010) and Yoshida *et al.*, (2010) also adopted this algorithm to produce generalized lines using accumulated GPS track logs. The application of the Douglas-Peucker algorithm for line generalization is effective and efficient in eliminating redundant points. Additionally, the algorithm also eliminates extraneous position points and can speed up the performance at the next processing stage.

The function simplifies points based on a given tolerance parameter, and it is also effective for the points in which mobile device record the same location information positions. This function not only achieves efficient data management, but also reduces noise. Figure 3.5(a) displays filtered GPS track points, which were shown as green circles. Figure 3.5(b) shows generalization results as green line. The tolerance parameter was at 5m in Figure 3.5(b), after the generalization, emphasizing how the number of points is successfully reduced. Finally the results with OSM background map show in Figure 3.5(c), which closes resembles the actual tracks collected.

This section introduced a line generalization using Douglas-Peucker algorithm. The results are reliable and reduce the number of GPS points to be used in map-matching algorithm.

### 3.5 Map-matching Using Hausdorff Distance Algorithm

A common problem observed in traversability mapping is spatial misalignment of GPS location information with road network data due to the low GPS horizontal accuracy. In order to overcome this problem, map-matching is carried out by using the Hausdorff distance algorithm. The Hausdorff distance algorithm was originally developed to measure the similarity of two point sets and is now widely used for curve matching. Unlike most shape comparison methods that build a one-to-one correspondence between a GPS track and a road, Hausdorff distance can be calculated without explicit point correspondence (Gao, 2005). The Hausdorff distance for map-matching is more tolerant to perturbations in the locations of points than the other map-matching algorithm since it measures proximity rather than exact superposition (Huttenlocher *et al.*, 1993).

The Hausdorff distance algorithm based matching selects the roads that match a vehicle trajectory and uses the length-weighted distance to calculate the similarity between the GPS track and candidate roads. The roads with a short distance measurement are selected for matching.

The given two finite point sets  $T = \{t_1, t_2, \dots, t_m\}$  (representing a GPS track in the database) and  $R = \{r_1, r_2, \dots, r_n\}$  (representing a road from road networks), the length-weighted Hausdorff distance from  $T$  to  $R$ ,  $H(T, R)$ , is calculated as follows:

$$(1) \quad H(T, R) = \max(h(T_i, R_j), h(R_j, T_i)), \quad i = 1, 2, \dots, m-1 \quad j = 1, 2, \dots, n-1$$

where

$$h(T, R) = \max_{t_i \in T} \min_{r_j \in R} ||t_i - r_j||, \quad i = 1, 2, \dots, m-1 \quad j = 1, 2, \dots, n-1$$

(2)

The  $||t_i - r_j||$  is the shortest distance from points of  $T$  and  $R$ . The function  $h(T, R)$  is called the directed Hausdorff distance from  $T$  to  $R$ . It identifies the point  $t_i \in T$  that is the farthest from any point of  $R$ . It measures the distance from  $t_i$  to its nearest neighbor in  $R$ . The Hausdorff distance  $H(T, R)$  is the maximum of  $h(T_i, R_j)$  and  $h(R_j, T_i)$ . Thus, it measures the degree of mismatch between two sets by measuring the distance of the point of  $T$  that is farthest from any point of  $R$  and vice versa. The steps involved in the map-matching process are as follows.

- a) Selecting candidate roads from road networks: GPS tracking buffer roads are generated by applying a threshold buffer size assigned as 24m, which is almost half of the minimum distance between roads in study area. In order to reduce the map-matching time, overlapping GPS track buffer roads are used as candidate roads.
- b) Finding the counterpart of the candidate road on the GPS track: By computing the shortest distance from the candidate road to the GPS track, the counterparts on the GPS track are found. A correspondence line is determined accordingly for the GPS track segment being matched to the candidate road.
- c) Computing the length-weighted Hausdorff distance: Calculating the length-weighted Hausdorff distances from the candidates to their counterpart GPS track segments.

- d) Determining map-matched roads: Selecting the map-matched roads using a distance less than a specific threshold. The threshold is determined by considering the positioning accuracy and the road width from the field experiment.

Map-matching using Hausdorff distance algorithm codes can see in Appendix C.

### **3.6 Field Experiment**

In this section, we evaluate performance of traversability mapping that proposed in this research. Study demonstrates the experiment result using mobile device (Nexus7). In order to demonstrate the effectiveness of the proposed map-matching method, we performed field evaluation.

#### **3.6.1 Environment of Field Experiment**

In order to evaluate performance of the proposed post-disaster road traversability mapping, two field experiments were carried out in Yamate-cho, Suita City and Sumiyoshi-ward, Osaka City, Japan. Study area shown in Figure 3.6(a), geographically stretches from latitude  $34^{\circ}46'01''\text{N}$  to  $34^{\circ}46'57''\text{N}$  and longitude  $135^{\circ}30'02''\text{E}$  to  $135^{\circ}31'23''\text{E}$  covers about 2km x 1.5km area. The second study area shown in Figure 3.6(b), geographically stretches from latitude  $34^{\circ}35'06''\text{N}$  to  $34^{\circ}36'02''\text{N}$  and longitude  $135^{\circ}29'40''\text{E}$  -  $135^{\circ}31'01''\text{E}$  covers about 2km x 1.5km area.

In this field experiment six mobile devices were used to collect GPS track points from both study areas Figure 3.6(a) and Figure 3.6(b). Characteristics of parameters used for field experiment are shown in Table 3.3. GPS tracks were collected using mobile devices in field experiments are shown in Figure 3.7. More than 20,000 position points with GPS tracks in each area were collected.

### **3.6.2 Results of Field Experiment**

The GPS track data were collected using mobile devices and buffered in order to overlay with road network data collected from OSM. The Figure 3.8 demonstrates the overlay map of buffered GPS tracks and road network for both study areas Figure 3.6(a) and Figure 3.6(b). The GPS tracks were processed, and the roads that overlap the buffer area of the GPS tracks are preliminarily selected as candidates for matching. The results of matched segments and running time are described in Table 3.4. In study area Figure 3.6(a), an average of 3,543 track points was collected using six devices. Around 10% of low accuracy track points were removed by filtering and an approximately 3,187 track points were finally derived. Further, duration of average 1.491s was taken to run Hausdorff distance map-matching algorithm for 3,187 GPS track points. Around 198 candidate segments were overlapped with the area of GPS tracks. In study area Figure 3.6(b), an average of 3,340 track points was collected using six devices. From these, around 14% of low accuracy track points were removed by filtering that an average of 2,880 track points was derived. Further, an average of 1.279s was taken as time to run Hausdorff distance map-matching algorithm for a number of 3,340 GPS track points. Around 176 candidate segments were overlapped the with

GPS tracks of the area. Also the detailed result of filtering for each device is shown in Table 3.4. In the field experiments it was found that using the Hausdorff distance algorithm, more than 2,000 segments were map-matched with in 1s.

The length-weighted Hausdorff distances from candidate roads to the GPS tracks were computed. Finally, the roads with a distance less than a specific threshold value were chosen as matched roads. The result of road map-matching depends on the selection of threshold value. The threshold value is an experimental value related to the positional accuracy of GPS and the environment of data collecting. In this research, it was set to 20m by comparing all the map-matching results. Map-matching with 20m Hausdorff distance threshold provides decent balance with real road network data. Map-matching with specific threshold was used to determine if the GPS tracks collected by using mobile devices had a adequate accuracy or not. Consequently, only GPS traces which lie inside road network buffers were used to create the road traversability map.

The system evaluates the status of the road whether the road is usable for travel or not by using GPS tracks collected by stranded commuters. A road is considered usable for transportation, if one stranded commuter has already travelled through a particular road. Finally, the Hausdorff distance of the roads measured to be less than a specified threshold value, were chosen as traversability roads. Traversability maps based map-matching for two areas are shown in Figure 3.9. Different colors are used to indicate the traversability of the road, which is determined by the pass frequency. Only two colors were used to differentiate traversability of the roads due to limitation in the number of mobile

devices used for filed experiments. The roads, which were traversed only once by the commuters are shown in red color, and roads, which were traversed twice or more are shown in green color in Figure 3.9.

In this Chapter we described methodology and data processing workflow for managing GPS tracks data and map-matching method for post-disaster road traversability mapping system. The next Chapter discusses demonstrate the efficacy of post-disaster road traversability mapping system. Further, Chapter 4 considers the post-disaster road traversability map publishing services using GeoServer<sup>30</sup> WMS. In addition, GPS tracks sharing using Geopap-cloud<sup>31</sup> web service also discussed.

---

<sup>30</sup> <http://geoserver.org>

<sup>31</sup> <https://github.com/geopaparazzi/geopaparazzi/wiki>

# **Chapter 4**

## **Map Services for Stranded Commuters**

This Chapter describes the methodology of generating post-disaster traversability map and making them available to standard commuters. In addition, a cloud service has been deployed to publicly share the collected GPS track for other purposes such as evaluating mobility behaviors.

### **4.1 Map Services for Stranded Commuters**

The post-disaster traversability map is disseminated as WMS using GeoServer. Therefore, the post-disaster traversability map can be instantaneously viewed on device that connected to the Internet. Moreover, a digital signage system powered by single Raspberry Pi mini computer is deployed to display the post-disaster traversability map using available monitors to standard commuters who do not carry a mobile device.

#### **4.1.1 Publishing Post-Disaster Road Traversability Map using GeoServer WMS**

This section, discusses ways for making post-disaster road traversability maps available in the event of a natural disaster. Road traversability information may change during and soon after disaster event. Therefore, the road traversability map must be generated based on the real-time information obtained from the field.



In this study, WMS protocol was used to publish traversability map using GeoServer, which is an open source server for managing and sharing geospatial data. OpenLayers<sup>32</sup>, a JavaScript library, is integrated into GeoServer, making map rendering quick and easy. Deploying of maps using the WMS standard enables interoperability with different software applications. GeoServer also supports connection to PostgreSQL/PostGIS spatial database facilitating spatial queries and routing.

As explained in Chapter 3, GPS tracks data have been collected and two field experiments were carried out in and around Yamate-cho, Suita City and Sumiyoshi-ward, Osaka City, Japan. The Web-application was created to facilitate publishing the post-disaster road traversability map using data stored in PostgreSQL/PostGIS database. Figure 4.1 shows GeoServer connect to PostgreSQL/PostGIS database. The OpenLayers Javascript API was used to construct Web-GIS functionalities in this application. Since the Web application is executed on the server side users need not to have software installed on client devices and the application can be used on Web browsers for mobile devices.

The post-disaster road traversability map was developed by considering real-time road condition during and soon after disaster events with updated information. GPS tracks data collected by Android devices were automatically stored in the spatial database. Newly collected GPS tracks data are automatically updated to the road network database and published as traversability road map.

---

<sup>32</sup> <http://openlayers.org>

Figure 4.2 shows the Web interface of publishing post-disaster road traversability map using GeoServer. Different colors are used to represent different levels of traversability. The roads, which passed only once by the standard commuter is shown in red color, and roads, which passed twice or more are denoted in green color.

#### **4.1.2 Digital Signage Service for Stranded Commuters**

This section describes how to publish post-disaster road traversability map using the digital signage services. Since many people do not carry a mobile device or devices may not be operable due to lack of electricity during the disaster. Therefore, the created post-disaster road traversability map needs to be made available to the standard commuters who do not carry the mobile device.

In this study, digital signage powered by low-cost single mini computer (Raspberry Pi 2 Model B V1.1<sup>33</sup>, Figure 4.3) is be used to display the created post-disaster road travesability map through available monitors that can be set up at vantage points. The Raspberry Pi offers a suite of small single-board computers developed by the Raspberry Pi Foundation. Raspberry Pi has low power requirement, can be run on solar energy or running on a battery power source. The signage display is managed using Concerto<sup>34</sup>, to easily engage standard commuters through display monitors that maybe available or set-up at vantage points. Concerto is a web-based digital signage application available

---

<sup>33</sup> <https://www.raspberrypi.org>

<sup>34</sup> <http://www.concerto-signage.org>

under the Open Source<sup>35</sup> License. It uses of screens to broadcast specific messages about events, services, and other noteworthy items. Figure 4.4 shows sharing GeoServer WMS using Concerto digital signage.

The signage system connects to GeoServer to access the created post-disaster road traversability map automatically and display them on available monitors. Since the digital signage services are controlled remotely and the results displayed on the monitors, users does no require a mobile device to access the information. Figure 4.5 shows the interface of publishing post-disaster road traversability map using Concerto digital signage system.

## **4.2 Sharing Recorded GPS Tracks using Geopap-cloud Web Service**

This section describes sharing recorded GPS tracks using Geopap-cloud service for managing GPS tracks data and sharing them to people in need.

Geopaparazzi<sup>36</sup> is a tool developed to do fast qualitative technical and scientific surveys. Even if the main aim is for field survey, it contains tools that can be of use for a variety of purposes. In this study, a method has been used to share recorded GPS tracks which using Geopap-cloud<sup>37</sup> tool of Geopaparazzi. Mobile device users can connect to Geopap-cloud server during only set server URL in preferences. To server URL the relative path upload is attached and sent

---

<sup>35</sup> <http://httpd.apache.org>

<sup>36</sup> <http://www.geopaparazzi.eu>

<sup>37</sup> <https://github.com/geopaparazzi/geopaparazzi/wiki>

to that URL via http POST. Mobile device users would be able to export a project to the configured server. There have a main page listing the uploaded projects. The sharing recorded GPS tracks using Geopap-Cloud service show in Figure 4.6

Figure 4.7 shows the characteristics of GPS tracks data displayed in the Geopap-Cloud service. The summary of the data can be viewed using the project view. It is also possible to see the profiles of the traces and download the points and lines as shape files.

### **4.3 Deploying Routing using Road Traversability Map**

This section, discusses the implementation of dynamic routing using created road traversability map, which may be required by stranded commuters during and soon after disasters.

Acquired GPS tracks already stored in the server as PostGIS database. In this study, open source software pgRouting<sup>38</sup> is used to carry out the routing service using post-processed GPS tracks. The pgRouting can compute a route where the cost can be either distance or time duration. Hence, the routing service can be effectively used to find the optimal route for commuters.

This study uses Dijkstra's algorithm<sup>39</sup> readily available in pgRouting (Kastl and Junod, 2011) to calculate the shortest route. Figure 4.8(a) demonstrates an

---

<sup>38</sup> <http://pgrouting.org>

<sup>39</sup> <http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/>

example of shortest path result as map from source point to target point using the pgRouting function.

Firstly, length and topology for all the road segments were calculated using functions *ST\_Length()*<sup>40</sup> and *pgr\_createTopology()*<sup>41</sup>. Secondly, SQL command used to calculate the route by using the road length as cost. In this study, stranded commuters can update the location information during disaster into database using Android application. As explained in Chapter 3, the traversability road condition can be created using map-matching. The roads traversed by stranded commuters are assigned lower cost in database using SQL command. Based on an updated traversability condition, pgRouting algorithm can provide dynamic result using changed road cost and send to stranded commuters. Figure 4.8(b) shows demonstrate and example of traversability path result as map from source point to target point using the pgRouting inside the traversability road map area.

Actually, not in all of the cases, the source point may inside the traversability road map area. For example, as demonstrated in shortest path result (Figure 4.9(a)), the source point is outside the traversability road map area. In that case, the shortest path from source point to traversability road is calculated and using traversability road to target point. Figure 4.9(b) demonstrates a traversability path result as map where source point outside the traversability road map area.

---

<sup>40</sup> [https://postgis.net/docs/ST\\_Length.html](https://postgis.net/docs/ST_Length.html)

<sup>41</sup> [http://docs.pgrouting.org/2.2/en/src/topology/doc/pgr\\_createTopology.html](http://docs.pgrouting.org/2.2/en/src/topology/doc/pgr_createTopology.html)

## 4.4 Publishing Offline Road Traversability Map using Tile Map

As mentioned in previous section, the road traversability map has been created and publishing using GeoServer WMS. Further, acquired GPS tracks stored in the server as PostGIS database can be effectively used to provide a routing service using pgRouting. Access to GeoServer and PostGIS database require Internet connectivity. This section, discusses a method to publish offline road traversability map using Web Map Tiling Service (WMTS)<sup>42</sup>.

One of the objectives of this study is to support stranded commuters during and soon after disaster events even when Internet connection is unavailable. Therefore, WMTS was used to share the post-disaster traversability map even when Internet is unavailable. Offline map allows stranded commuters to view created road traversability map when they are disconnected from the Internet. Once they are reconnected, they can get map updates from WMTS provided by GeoServer.

In this study, GeoServer support for generating tiles map is used. To make tile map as fast and responsive as possible, traversability road map has been set several zoom levels to generate map tiles. Tiled traversability road map takes the form of a pyramid where the map is drawn at a progressive series of zoom levels, with the smallest zoom level using fewer tiles. Figure 4.10 shows a sample of tile map sets. These tiles can be served with the WMTS and stranded commuters can directly manipulate to pan and zoom.

---

<sup>42</sup> [http://wiki.osgeo.org/wiki/Tile\\_Map\\_Service\\_Specification](http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification)

Tiled maps can reasonably work for serving traversability road map to Android devices user. Stranded commuters are able to find the traversability path to reach their destination using offline tile map. Figure 4.11 shows tiles based on traversability road map.

However, stranded commuters cannot share tile maps to the other people who have not installed appropriate Android application. The solution of this problem will be discusses in next section.

## **4.5 Offline Tile Map Sharing using MANET**

Previous section focused on publishing offline road traversability map using WMTS. This section describes how to share created offline tile map using MANET.

As discussed in Chapter 2, GPS location data sharing has been performed using MANET. Even when Internet connection is unavailable, the proposed system can get data from nearby devices connected through Ad-hoc network. Therefore, offline tile map created also can be shared through Ad-hoc network. Any mobile devices connect the Ad-hoc network, if at least one device receive the post-disaster road traversability map either physically or through Internet, it can transmit offline tile map to all mobile devices.

Many applications for Android, iOS, and other mobile devices can use offline tile map data to display a maps offline. For example, offline tile map based on traversability road map can be displayed using Geopaparazzi application. Offline

tile map to be used in Geopaparazzi can be created in several ways<sup>43</sup>. In this study, stranded commuters only need export shared mapurls<sup>44</sup> and mbtiles<sup>45</sup> files to the device map folder. The tiled traversability road map will be automatically load and display on mobile device. Figure 4.12 shows the tiled traversability road map is displayed in Geopaparazzi.

Further, the study recommends another method to share the post-disaster road traversability map using SQLite database<sup>46</sup>. SQLite database is an open source SQL database that stores data to a text file on mobile device. Through created MANET, SQLite database file can be transmit to all mobile devices. Therefore, server can export PostGIS database data to SQLite database file using ogr2ogr<sup>47</sup>.

---

<sup>43</sup> <https://github.com/geopaparazzi/geopaparazzi/wiki/>

<sup>44</sup> <https://github.com/geopaparazzi/geopaparazzi/wiki/.mapurl-parameters>

<sup>45</sup> <http://wiki.openstreetmap.org/wiki/MBTiles>

<sup>46</sup> <https://www.sqlite.org>

<sup>47</sup> <http://www.gdal.org/ogr2ogr.html>



# **Chapter 5**

## **Summary and Discussions**

The main motivation in undertaking this research was to address some of the practical issues that were widely reported during major disaster situations such as the 11 March, 2011 Tohoku earthquake in Japan. Apart to great loss of life and property in the near vicinity of the earthquake and the subsequent tsunami event, metropolitan areas such as Tokyo experienced disruption of train services, electrical outages and inaccessibility to Internet services. As a result a vast number of commuters were stranded and compelled to proceed to their destinations using unfamiliar routes and transportation modes. Supporting stranded commuters during and soon after earthquake disasters or other emergency situations have been considered as one of the most important issues for ensuring safety of citizens. It is, therefore, necessary to develop a near real-time traversability mapping service that can be available during or soon after the disaster.

The post-disaster road traversability mapping system developed as a part of this research facilitates an effective and timely response for emergency scenarios. The offers a significant improvement compared to existing online map services such that may be rendered unusable during and soon after disasters. Supporting stranded commuters during and soon after earthquake disasters have been considered as one of the most important issues in ensuring safety.

Although several previous studies were focused on navigation by GPS locations and location based services for disaster evacuation (Magnusson, 2012; Fujihara *et al.*, 2013), most of them using map data that have been collected in the past and require Internet services.

In this research, a workflow for road traversability mapping service was implemented and evaluated for viability and performance. The traversability mapping services are basically designed to perform GPS data collection and sharing between nearby devices even when Internet connectivity is unavailable. MANET and DTN were investigated using a mobile application for sharing GPS tracks. The application was tested using 6 Android devices in urban and semi-urban areas in Osaka to evaluate feasibility and performance. In order to evaluate scenario with large number of devices, a simulation experiment using the NS-2 simulator was carried out. The field experiment and simulation experiment elucidates the efficacy of GPS data sharing and aggregation using a combination of MANET and DTN. The simulation experiment also demonstrates that in densely populated areas, around 2000 devices could provide complete coverage of entire road network in 60 minutes.

In the next step, pre-processing of GPS tracks collected by individual devices was considered to minimize volume of data transfer considering limited bandwidth available in Ad-hoc networks. An Android application for filtering GPS tracks, sharing between mobile devices and aggregation of data was developed. GPS filtering is implemented based on standard parameters such as number of GPS satellites used for positioning, GPS accuracy and HDOP in order to eliminate

low accuracy GPS data. The application also allows for uploading the aggregated data to the server for further processing when the Internet connection is available.

A post-processing workflow was implemented for generating updated road traversability map using the aggregated GPS tracks. The post-processing consists of a line simplification based on Douglas-Peucker algorithm and map-matching with existing road network using the Hausdorff distance algorithm. Line simplification was applied reduce the data and map-matching to remove outliers and generate a navigable traversability road network. Performance of map-matching algorithm was evaluated with data collected in two field campaigns in Yamate-cho, Suita City and Sumiyoshi-ward in Osaka City, Japan. Line simplification and map-matching algorithm was found to give desired results to achieve the final objective.

As a final step, as and when new GPS tracks are loaded to the server, the traversability maps are automatically updated by running the post-processing program in the server. The updated road i maps are published as WMS using GeoServer and available as a raster layer to any device connected to Internet. The traversability road network is stored in a PostgreSQL/PostGIS spatial database and routing functionality is implemented using the pgRouting library to facilitate in guiding users from their present location to target destination.

Map tiles are also generated automatically to support offline use. The road traversability map tiles can be either downloaded when Internet connectivity is available or the tiles can be shared among devices connected though the Ad-Hoc

network. As a future work, it is planned to incorporate off-line routing functionality using an embedded database. Lastly, cater to users who may not have access to smart phone, a Raspberry Pi based digital signage system is deployed to facilitate display the traversability map for public viewing.

The framework was originally designed to be use during and soon after disaster. Since the system can be used in an offline as well as online mode, it can used to update the road map during other situations such as disaster drills. Such use would greatly help in disaster preparedness and mitigate loss of life and property in future disasters.

# **Chapter 6**

## **Conclusions and Future Work**

As discussed in Chapter 5, stranded commuters require post-disaster road traversability mapping system. Post-disaster traversability mapping in a near real-time scale is always been a challenging task especially due to the non-availability of Internet connection. Potential obstacles that can be occurring during disaster events include fire outbreaks, unexpected stoppages in public transportation, road damage, traffic signal damage and Internet disconnection. Hence, the traversability mapping system needs to consider these issues. The traversability mapping system developed as a part of the present research, is effective in addressing several of these issues and provides a workflow for generation a near real-time road traversability map even when no Internet connection is available.

The data processing workflow implemented as a part of this research helps overcome several of the limitations of previous research. Some salient feature that distinguish this research from previous works are below:

- a) Usability by stranded commuters using public transportation rather than private automobile. This is an important factor especially in urban areas in Japan where public transportation is more widely used than private automobile. Although a large automobile maker demonstrated the use of near

real-time Floating Cellular Data to provide usable routes and shared them with drivers. It is difficult for stranded passengers using public transportation to utilize information provided for automobile users. This thesis described the workflow of post-disaster road traversability mapping system that can be used supporting stranded commuters during and soon after disasters in urban areas. The system is made available to mainly support standard commuters carrying mobile device.

- b) Most of the functionality is availability both in online as well as offline mode and the system is usable even when Internet connectivity is unavailable. In this study, acquired GPS location data from several devices were assimilated to create a post-disaster road traversability map. Even when Internet connection is unavailable, the proposed system can get GPS location data from nearby devices connected through created Ad-hoc networks. After post-processing, map tiles are also generated automatically to support offline use. The road traversability map tiles can be either downloaded when Internet connectivity is available or the tiles can be share among devices connected though Ad-hoc networks.
- c) The entire workflow is implemented using Open Source software and libraries and, therefore more amenable for future enhancements and customization. Here, the post-disaster road traversability map generated is made available to mobile devices as well as digital signage. The source codes developed post-disaster road traversability mapping will be uploaded to

GitHub<sup>48</sup> repository. Therefore, the source code can be access by one who wishes to carry out further improvement in the developed system.

Based on the features described above, it can be concluded that the proposed framework and data processing workflow are effective in accomplishing the objectives envisaged for the current research.

As a future work, it is necessary to consider ways and means to incorporate additional contextual information to the traversability maps to make them more intuitive. Another challenge would be to enhance the system functionality by considering realistic situations affecting crowd dynamics and behavioural patterns as regards mobility.

Additional field experiments or simulation with large number of field devices need to be undertaken to evaluate scalability and robustness of the proposed system. The present study focused on a small experiment area (5km<sup>2</sup>) and only limited number of mobile devices used to evaluate the system. Moreover, further work is required to evaluate effectiveness of the proposed system in situations where civil infrastructure like bridges may have been rendered unusable. Therefore, in the future investigations will need to focus on experiments including scenarios of collapse of civic infrastructures and other disruptions that may occur in real disaster situations.

---

<sup>48</sup> <https://github.com>

Lastly, it will also be useful to consider low-cost gateway<sup>49</sup> accepting the inbound data sent by field devices and outbound communication to a process running in server and further enhance operability in offline mode.

---

<sup>49</sup> <https://thenewstack.io/tutorial-prototyping-a-sensor-node-and-iot-gateway-with-arduino-and-raspberry-pi-part-1/>



## References

- Amirian, P. and Alesheikh, A. (2008). A Service Oriented Framework for Dissemination Geospatial data to Mobile, Desktop and Web Clients. *World Applied Sciences journal*, vol.3, no.1, pp.140-153.
- Anagnostopoulos, C. and Hadjiefthymiades, S. (2012). Optimal, quality-aware scheduling of data consumption in mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, vol.72, no.10, pp.1269-1279.
- Capkun, S., Hamdi, M. and Hubaux, J. (2002). GPS-free positioning in mobile ad hoc networks. *Cluster Computing*, vol.5, no.2, pp.157-167.
- Chenji, H., Zhang, W., Stoleru, R. and Arnett, C. (2003). DistressNet: A disaster response system providing constant availability cloud-like services. *Ad Hoc Networks*, vol.11, no.8, pp.2440-2460.
- Choosumrong, S., Raghavan, V. and Realini, E. (2010). Implementation of Dynamic Cost Based Routing for Navigation Under Real Road Conditions Using FOSS4G and OpenStreetMap. *Proceedings of GIS-IDEAS 2010*, No.5, pp.53-58.
- Choosumrong, S., Raghavan, V. and Bozon, N. (2012). Multi-Criteria Emergency Route Planning Based on Analytical Hierarchy Process and pgRouting. *Geoinformatics*, vol.23, no.4, pp.159-168.
- Choosumrong, S. (2014). Development of Algorithm for Multi-Criteria, Multi-Purpose Kinetic Routing Service Using Free and Open Source Software. *Osaka City University, Department of Graduate School for Creative Cities, Ph.D Dissertation*, 105p.  
[http://dlisv03.media.osaka-cu.ac.jp/il/meta\\_pub/G0000438repository\\_6036](http://dlisv03.media.osaka-cu.ac.jp/il/meta_pub/G0000438repository_6036)

(Accessed: 2017/06/22)

- Dockstader, S.L. and Tekalp, A.M. (2001). Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE*, vol.89, no.10, pp.1441-1455.
- Fujihara, A. and Miwa, H. (2013). Disaster evacuation guidance method using opportunistic communication (in Japanese). *IEICE*, vol.96, no.6, pp.580-588.
- Gao, Y. (2005). Fast Screening in Large Face Databases Using Merit-Based Dominant Points. *Multimedia Modelling Conference, MMM 2005. Proceedings of the 11th International*, pp.284-290.
- George, S.M., Zhou, W., Chenji, H., Won, M., Lee, Y.O., Pazarloglou, A., Stoleru, R. and Barooah, P. (2010). DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response. *Communications Magazine, IEEE*, vol.48, no.3, pp.128-136.
- Haerri, J., Fiore, M., Filali, F., Bonnet, C., Chiasserini, C. and Casetti, C. (2005). A realistic mobility simulator for vehicular ad hoc networks. *Eurecom Technical Report*, pp.1-12.
- Huttenlocher, D.P., Rucklidge, W.J. and Klanderman, G.A. (1993). Comparing images using the Hausdorff distance under translation. *IEEE Computer Society Conference*.
- Jain, S., Fall, K. and Patra, R. (2004). Routing in a delay tolerant network. *SIGCOMM Comput. Commun.Re*, vol.34, no.4, pp.145-158.
- Joshi, P. (2011). Security issues in routing protocols in MANETs at network layer. *Procedia Computer Science*, vol.3, pp.954-960.
- Kastl, D. and Junod, F. (2011). pgRouting Workshop Manual. <http://workshop.pgrouting.org/>  
(Accessed: 2017/06/22)

- Kim, Y.P., Nakano, K., Miyakita, K., Sengoku, M. and Park, Y.J. (2012). A routing protocol for considering the time variant mobility model in delay tolerant network. *IEICE Transactions on Information and Systems*, vol.E95-D, no.2, pp.451-461.
- Lorincz, K., Malan, D.J., Fulford-Jones, T.R., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Welsh, M. and Moulton, S. (2004). Sensor networks for emergency response: challenges and opportunities. *Pervasive Computing, IEEE*, vol.3, no.4, pp.16-23.
- Lwin, K. and Murayama, Y. (2011). Web-based GIS system for real-time field data collection using a personal mobile phone. *Journal of Geographic Information System*, vol.3, no.4, pp.382-389.
- Magnusson, C., Rassmus-Grohn, K. and Szymczak, D. (2012). Navigation by pointing to GPS locations. *Personal and Ubiquitous Computing*, vol.16, no.8, pp.959-971.
- Mashhadi, A., Mokhtar, S. and Capra, L. Fair content dissemination in participatory DTNs. *Ad Hoc Networks*, vol.10, no.8, pp.1633-1645.
- Matsuzaki, R. and Ebara, H. (2013). Ad-hoc networks using smart homes in an Earthquake disaster delivery of rescue request map data for buried victims. *IPSJ*, vol.6, no.1, pp.1-15.
- Matsuzaki, R., Ebara, H. and Muranaka, N. (2015). Rescue support system with DTN for earthquake disasters. *IEICE Transactions on Communications*, vol.E98-B, no.9, pp.1832-1847.
- Minamimoto, S., Fujii, S., Yamaguchi, H. and Higashino, T. (2010). Estimating disaster situation using mobile nodes' position and wireless link information (in Japanese). *IPSJ*, vol.51, no.12, pp.2169-2183.

- Patel, S.N., Reynolds, M.S. and Abowd, G.D. (2008). Detecting human movement by differential air pressure sensing in HVAC system ductwork: An exploration in infrastructure mediated sensing. *Pervasive Computing, Springer*, vol.5013, pp.1-18.
- Sammou, E.M. and Abdali, A. (2011). Routing in delay tolerant networks (DTN). *Network and System Sciences*, vol.4, no.1, pp.53-58.
- Schroedl, E., Wagstaff, K., Rogers, S., Langley, P. and Wilson, C. (2004). Mining GPS traces for map refinement. *Data Mining and Knowledge Discovery*, vol.9, no.1, pp.59-87.
- Seok-Kap, K., Hakjeon, B., Kyungran, K. and Chang-Soo, P. (2012). Quasi fair forwarding strategy for delay tolerant networks. *IEICE Transactions on Communications*, vol.95, no.11, pp.3585-3589.
- Shah, S., Bashir, A., Chauhdary, S., Jiehui, C. and Park, M. (2009). Mobile ad hoc computational grid for low constraint devices. *Future Computer and Communication*, pp.416-420.
- Shi, W. and Liu, Y. (2010). Real-time urban traffic monitoring with global positioning system equipped vehicles. *Intelligent Transport Systems*, vol.4, no.2, pp.113-120.
- Song, X., Raghavan, V. and Yoshida, D. (2010). Matching of vehicle GPS traces with urban road networks. *Current Science*, vol.98, no.12, pp.1592-1598.
- Song, X., Raghavan, V. and Yoshida, D. (2009). Open Web Processing Services for Improving Accuracy of GPS tracks using Filtering and Map-Matching, *Proceedings of International Conference FOSS4G2009*, <http://download.osgeo.org/osgeo/foss4g/2009/SPREP/1Wed/Parkside%20GO4/1100/wed%20g04%201100%20song.ppt>  
(Accessed: 2017/06/22)

- Spyropoulos, T., Psounis, K. and Raghavendra, C. (2008). Efficient routing in intermittently connected mobile networks: the multiplecopy case. *Networking, IEEE/ACM Transactions on*, vol.16, no.1, pp.77-90.
- Sun, J., Zhu, X., Zhang, C. and Fang, Y. (2011). RescueMe: location-based secure and dependable VANETs for disaster rescue. *Selected Areas in Communications*, vol.29, no.3, pp.659-669.
- Sun, W., Ishimaru, Y., Yasumoto, K. and Ito, M. (2010). Data routing for DTN environments according to data size and deadline (in Japanese). *IPSSJ SIG*, vol.2010, no.80, pp.1-6.
- Tachiki, Y., Yoshimura, T., Hasegawa, H., Mita, T., Sakai, T. and Nakamura, F. (2005). Effects of polyline simplification of dynamic GPS data under forest canopy on area and perimeter estimations. *J. Forest Res.*, vol.10, no.6, pp.419-427.
- Tracy Camp, J.B. and Davies, V. (2002). A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, vol.2, no.5, pp.483-502.
- Weiser, A. and Zipf, A. (2007). Web Service Orchestration of OGC Web Services for Disaster Management. *Geomatics Solutions for Disaster Management*, pp.239-254.
- Whitbeck, J. and Conan, V. (2010). Hymad: Hybrid dtn-manet routing for dense and highly dynamic wireless networks. *Computer Communications*, vol.33, no.13, pp.1483-1492.
- Wilson, D.H. and Atkeson, C. (2005). Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors. *Pervasive computing, Springer*, vol.3468, pp.62–79.
- Wren, C.R. and Tapia, E.M. (2006). Toward scalable activity recognition for sensor networks. *Location and context-awareness, Springer*, vol.3987, pp.168–185.

- Yamamoto, T. (2013). Development of Indoor Navigation System Using Visible Light Communication and FOSS4G. *Osaka City University, Department of Graduate School for Creative Cities, Master Thesis*, 49p.
- Yoon, J., Liu, M. and Noble, B. (2006). A general framework to construct stationary mobility models for the simulation of mobile networks. *IEEE Transactions on Mobile Computing*, vol.5, no.7, pp.860-871.
- Yoshida, D. (2007). Implementation of System for Sharing and Managing Spatial Data Using Open Standards. *Osaka City University, Department of Graduate School for Creative Cities, Master Thesis*, 97p.
- Yoshida, D., Song, X. and Raghavan, V. (2010). Development of track log and point of interest management system using free and open source software. *Applied Geomatics*, vol.2, no.3, pp.123-135.
- Yoshida, D. (2010). Development of Processing Services for G Enhancement by Data Filtering and Kinematic Relative Positioning. *Osaka City University, Department of Graduate School for Creative Cities, Ph.D Dissertation*, 100p.  
[http://dlisv03.media.osaka-cu.ac.jp/il/meta\\_pub/G0000438repository\\_111TD0000171](http://dlisv03.media.osaka-cu.ac.jp/il/meta_pub/G0000438repository_111TD0000171)  
 (Accessed: 2017/06/22)
- Yu, W., Ebara, H., Matsuzaki, R. and Raghavan, V. (2013). Real-time mapping system using GPS for stranded commuter. *Mobile Computing and Ubiquitous Communications*, vol.2013, no.35, pp223-228.
- Yu, W., Ebara, H., Matsuzaki, R., Raghavan, V. and Yoshida, D. (2014). A revisit of: real-time mapping system using GPS for stranded commuters ~A simulation by network simulator 2~. *Mobile Computing and Ubiquitous Communications* vol.2014, no.40, pp131-118.

- Yu, W., Ebara, H., Matsuzaki, R., Raghavan, V. and Yoshida, D. (2014). Real-time support system for stranded commuters considering traffic conditions. *INFORMS 2014*, San Francisco, USA, November 2014.
- Yu, W., Raghavan, V., Yoshida, D., Ebara, H. and Matsuzaki, R. (2014). GEOMANET: A Post Disaster Location Information Service Using Mobile Ad-hoc Networks. *FOSS4G-ASIA 2014*, Bangkok, Thailand, December 2014.
- Yu, W., Song, X., Raghavan, V., Yoshida, D. and Ebara, H. (2016). Post disaster road traversability mapping using mobile user generated GPS traces and OpenStreetMap based on map-matching. *GEOINFORUM-2016*, vol.27, no.2, pp.116-117.
- Zhao, L., Ochieng, WY., Quddus, MA. and Noland, RB. (2002). An extended Kalman filter algorithm for Integrating GPS and low-cost dead reckoning system data for vehicle performance and emissions monitoring. *J Navig*, vol.56, no.2, pp.257-275.

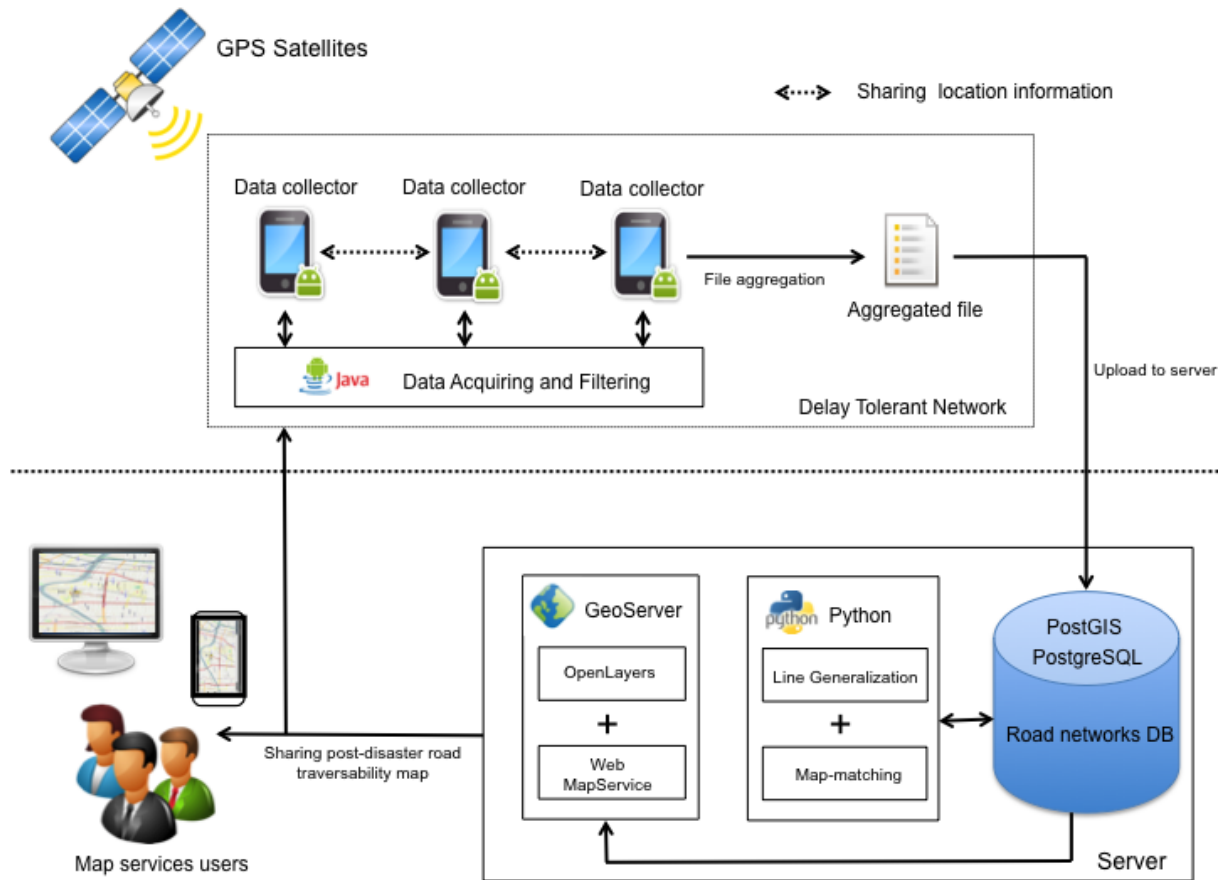


Figure 1.1: The framework of post-disaster road traversability mapping system



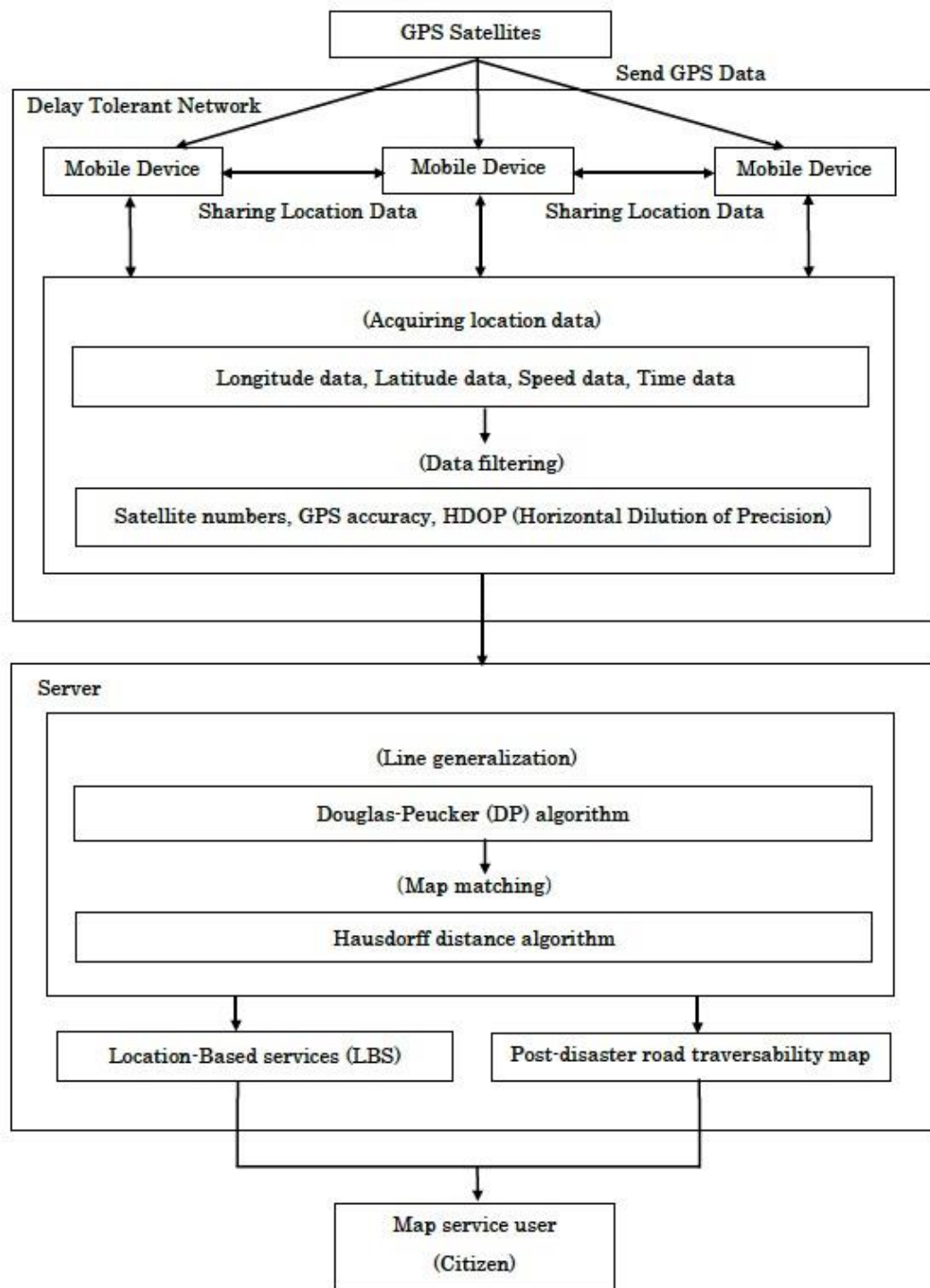


Figure 1.2: Flowchart showing post-disaster road traversability workflow

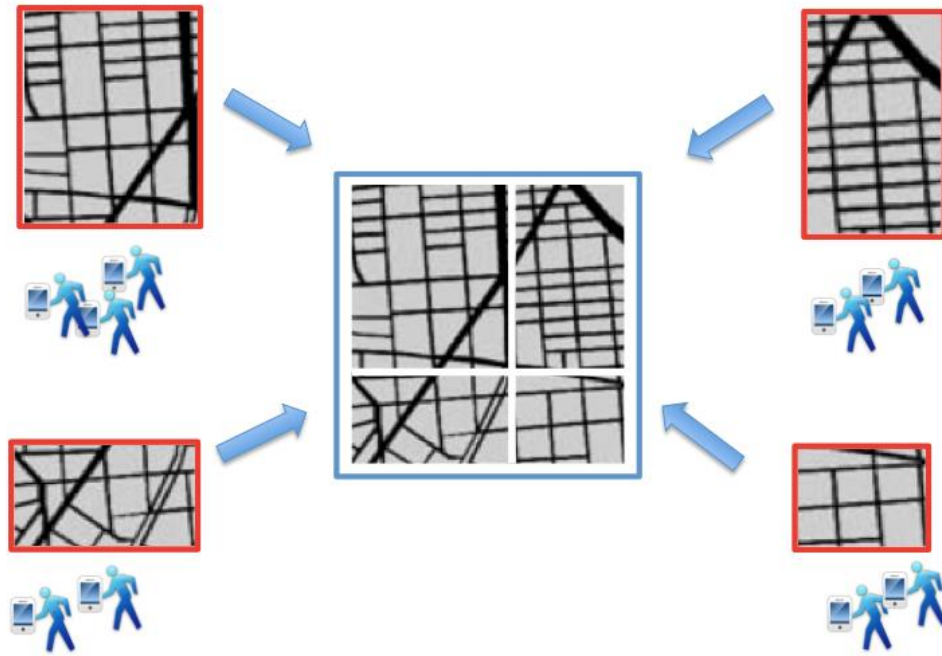


Figure 2.1: Assimilating map data from multiple users (after Yu *et al.*, 2015)



Figure 2.2: Map of study area around the Asahi Ward, Osaka City (after Yu *et al.*, 2015)



Figure 2.3: Simulated GPS tracks obtained for Asahi Ward, Osaka City  
(after Yu *et al.*, 2015)

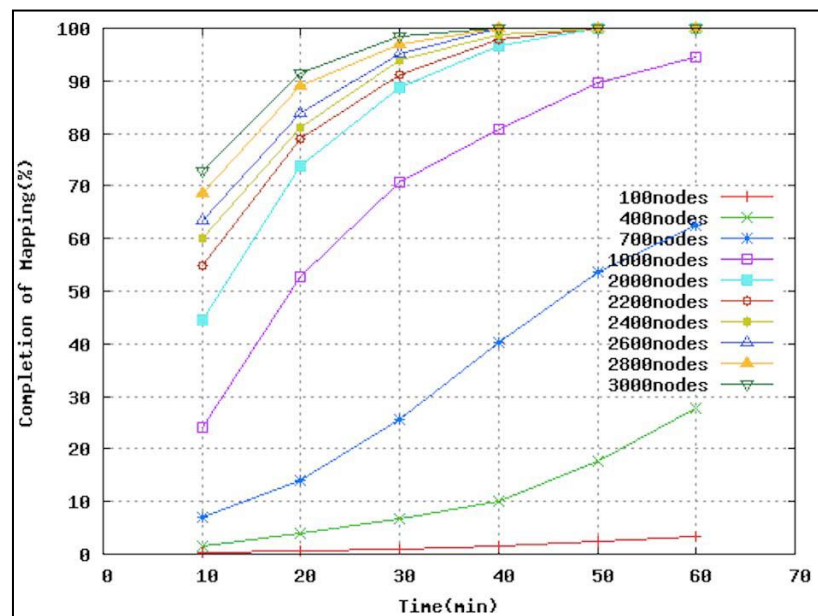


Figure 2.4: Completion of mapping related to time (2000nodes~3000nodes)  
(after Yu *et al.*, 2015)

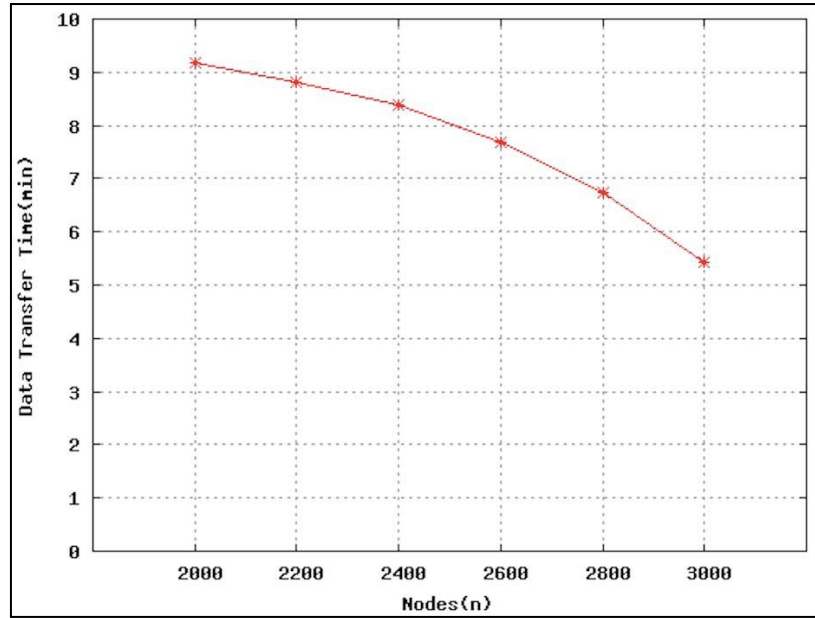


Figure 2.5: GPS data transfer time (after Yu *et al.*, 2015)

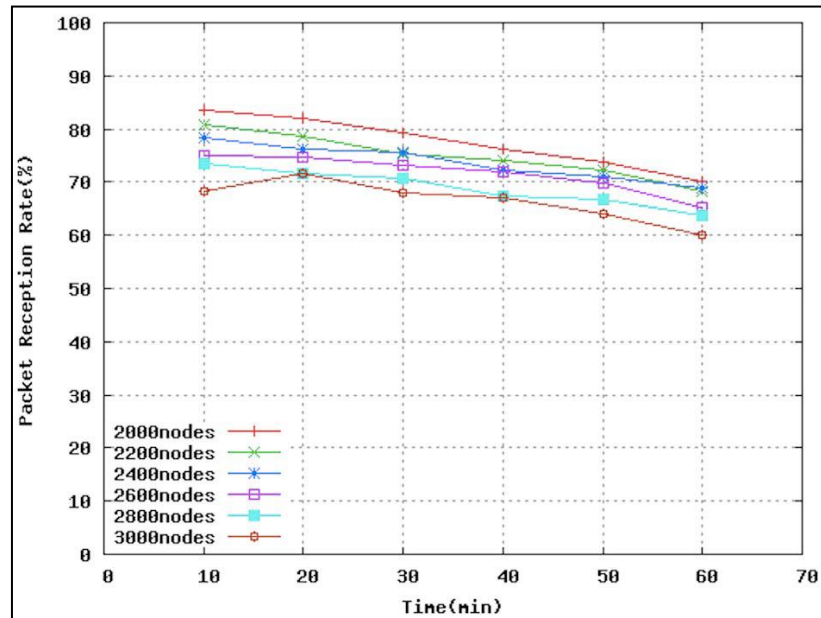


Figure 2.6: Packet reception rate (after Yu *et al.*, 2015)





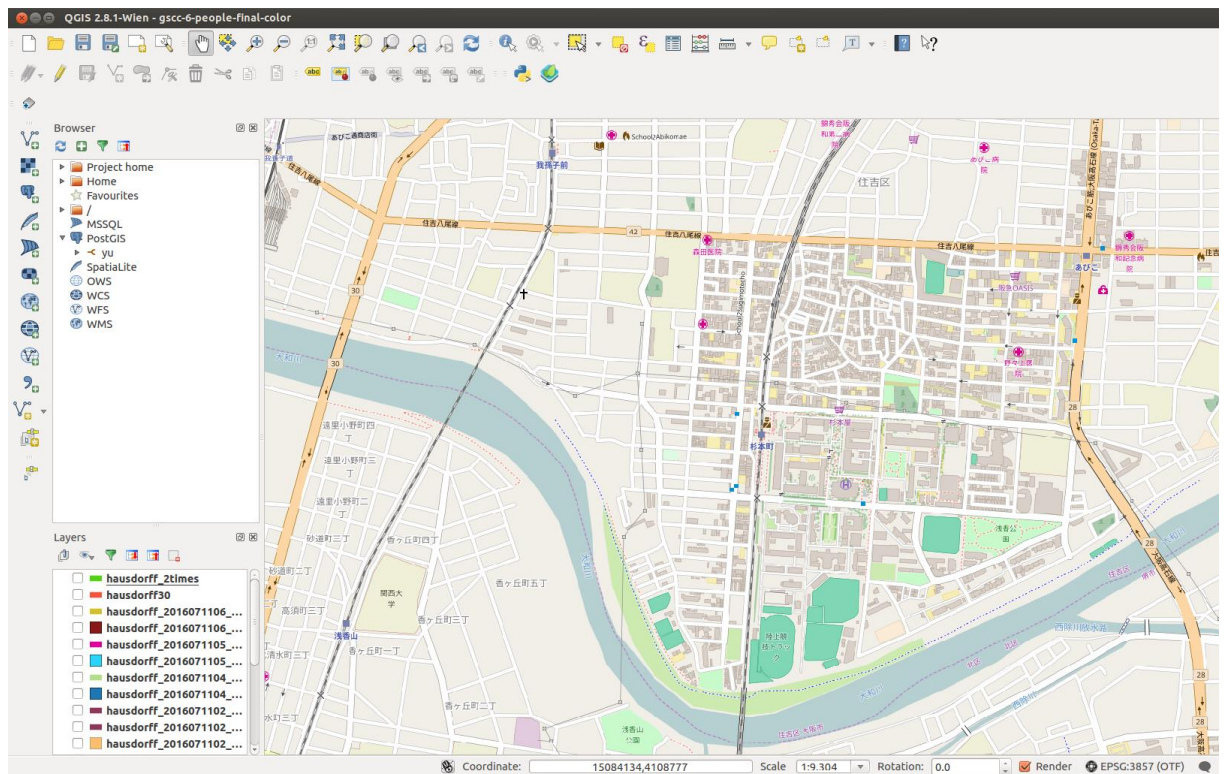


Figure 3.2: OSM data around Sumiyoshi-ward, Osaka City displayed in QGIS

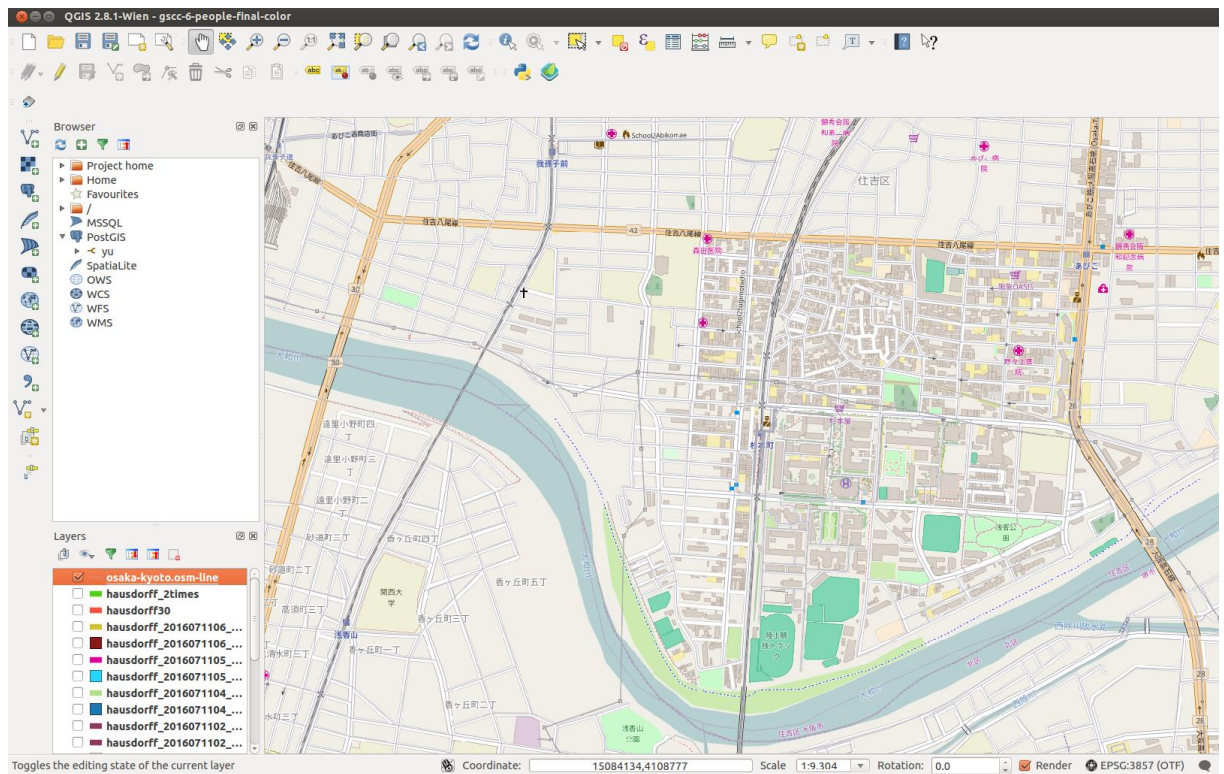


Figure 3.3: Road network superimposed over OSM in QGIS

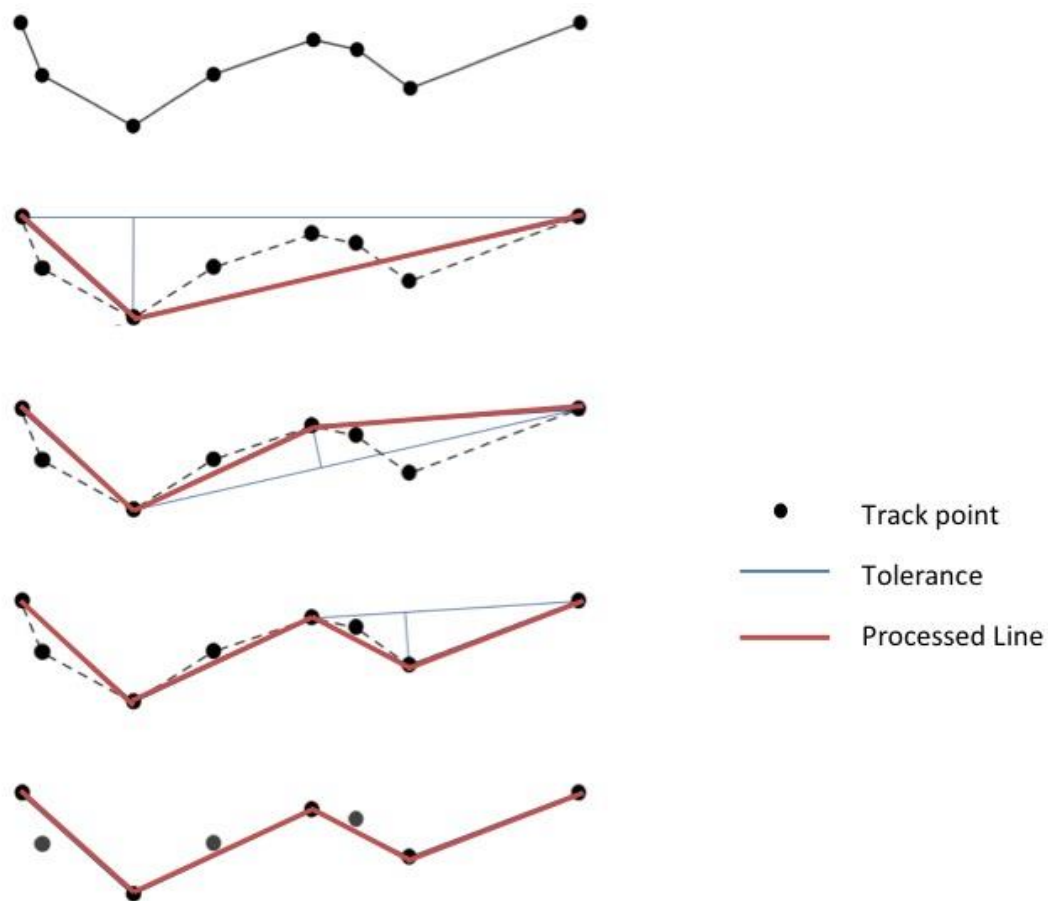
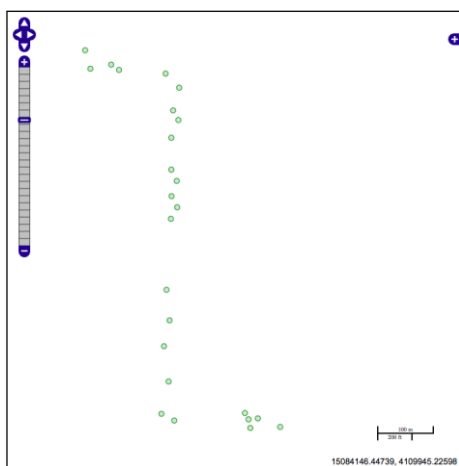
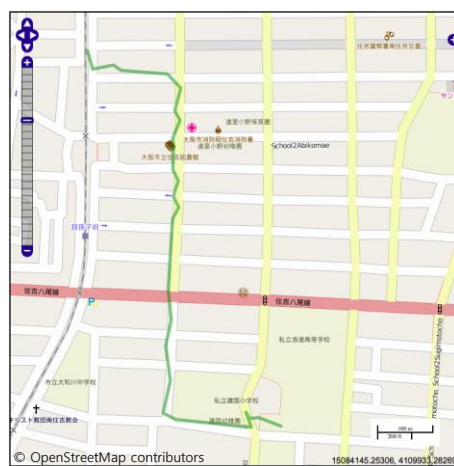


Figure 3.4: Line simplification steps in Douglas-Peucker algorithm



(a) Before generalization



(b) Result with background map

Figure 3.5: Results of Douglas-Peucker line simplification in Osaka City area

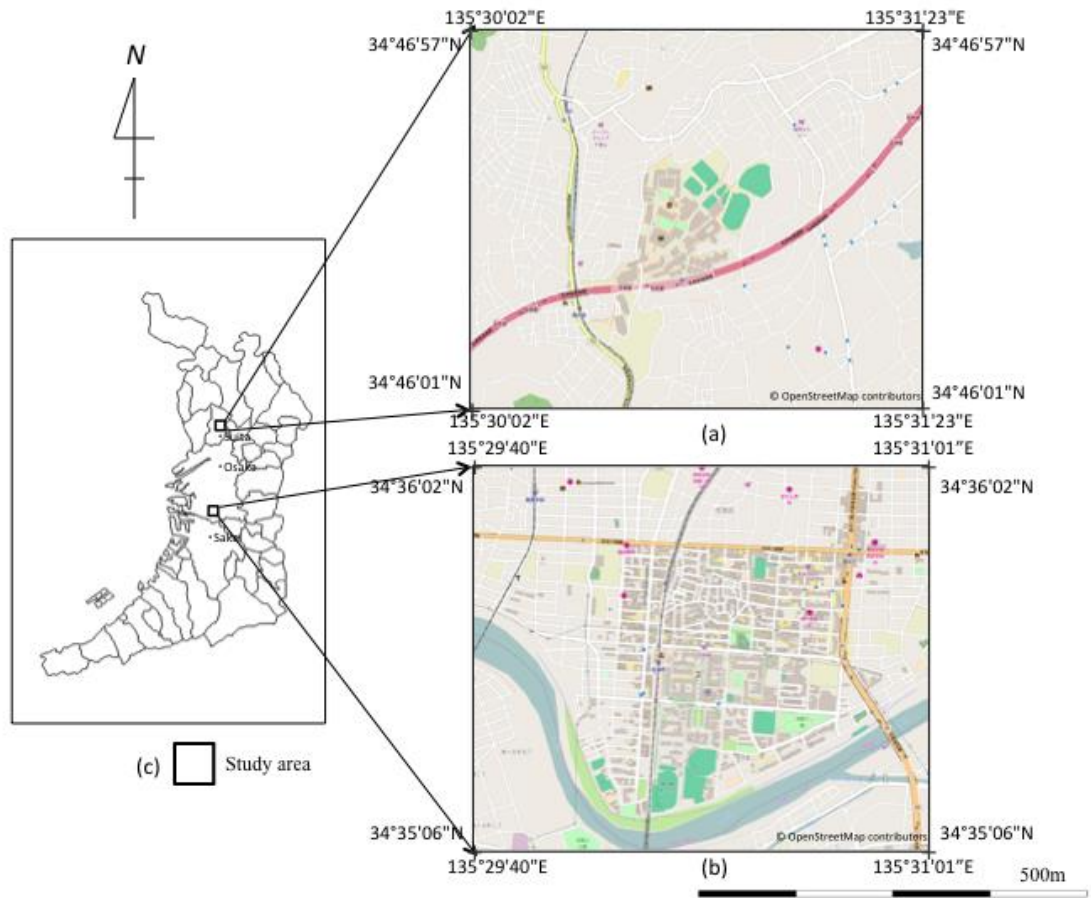


Figure 3.6: Study area (a) Yamate-cho, Suita City and (b) Sumiyoshi-ward, Osaka city in (c)

Osaka Fu



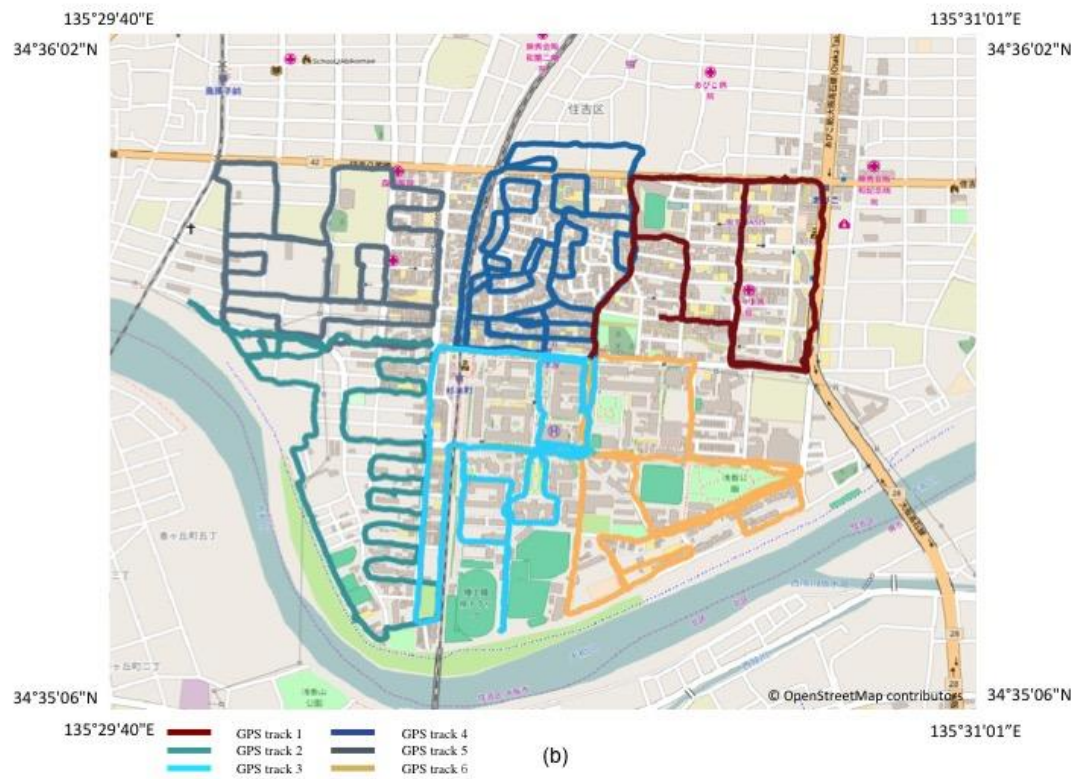
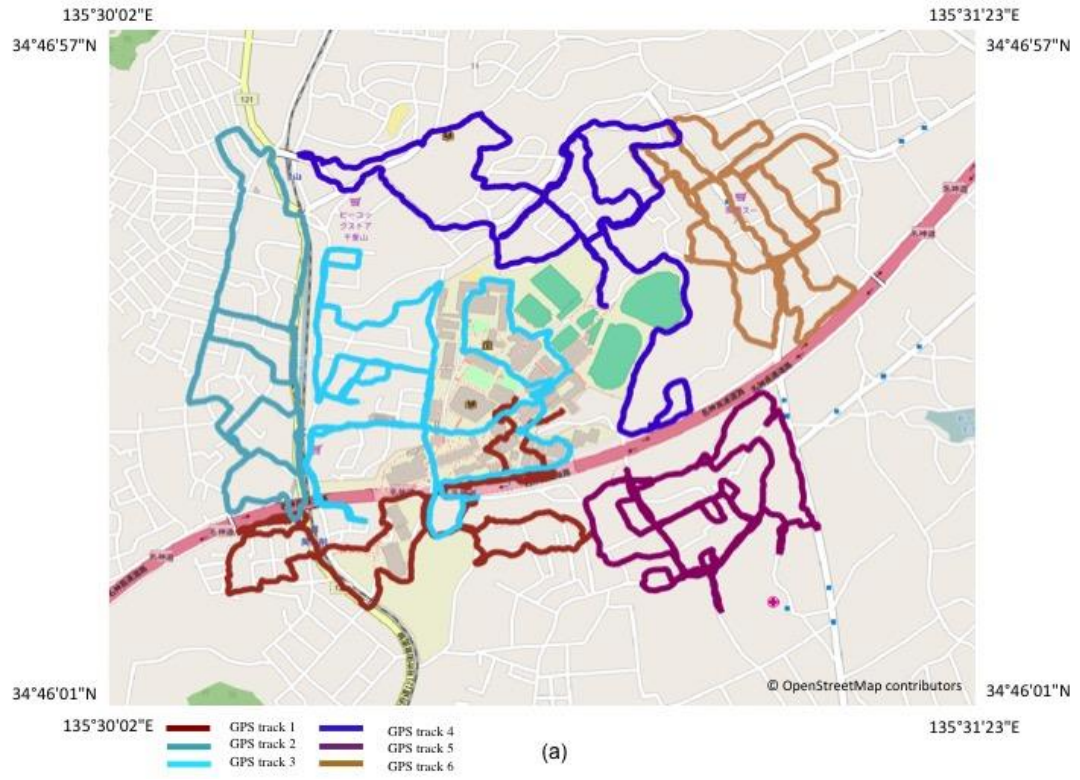


Figure 3.7: GPS tracks collected using six mobile devices in (a) Yamate-cho, and (b) Sumiyoshi-ward

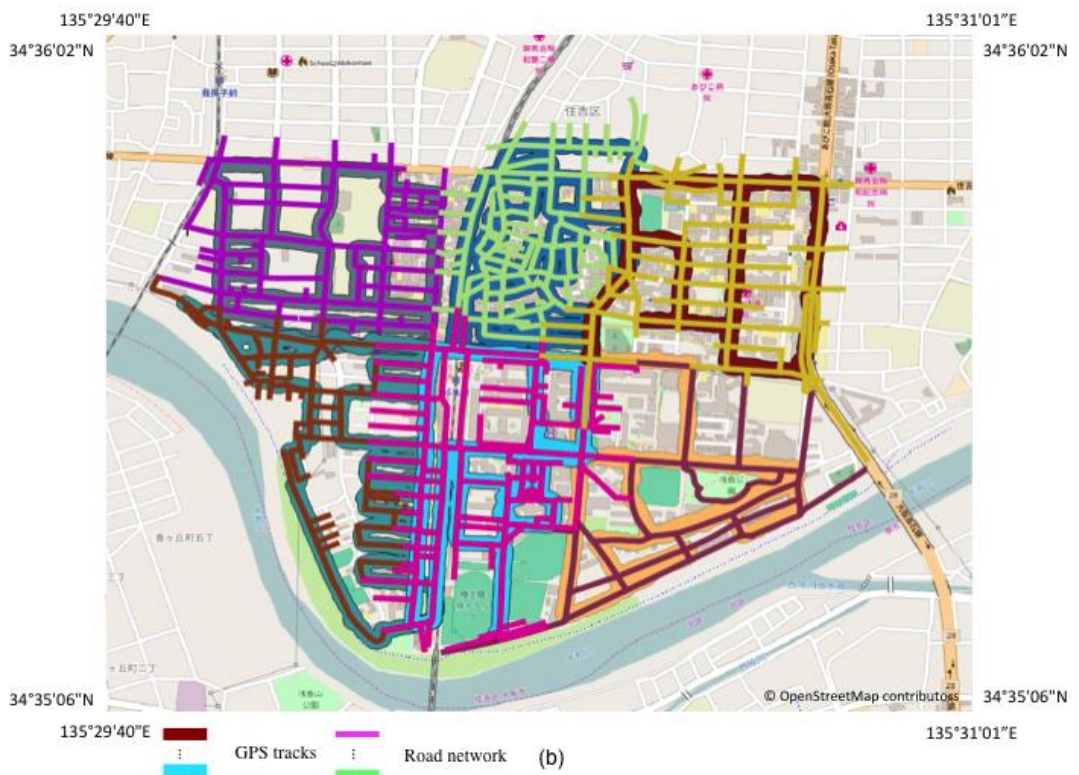
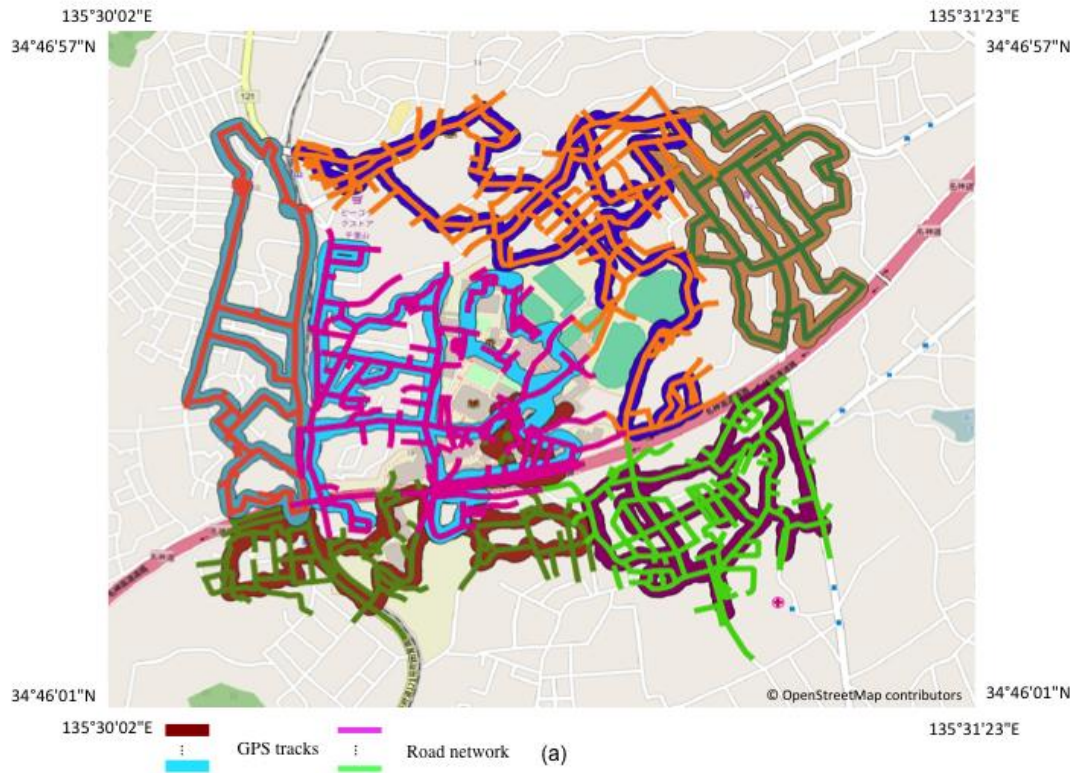


Figure 3.8: Road network and buffered GPS tracks before map-matching (a) Yamate-cho, and (b) Sumiyoshi-ward



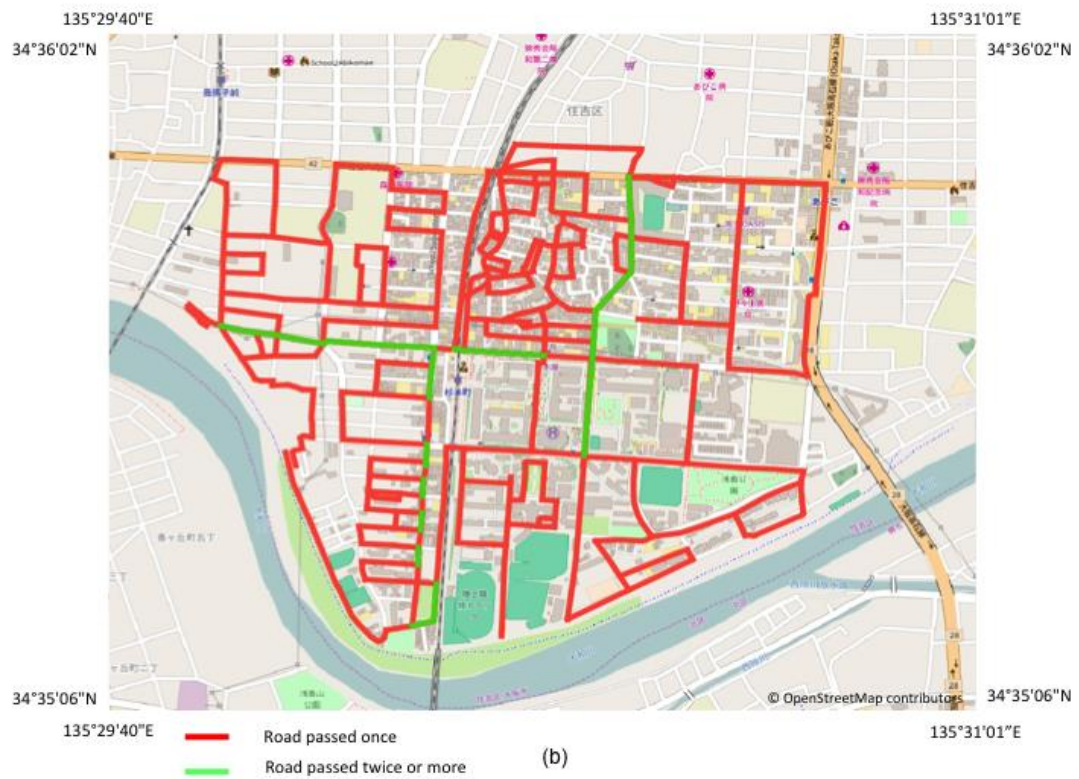
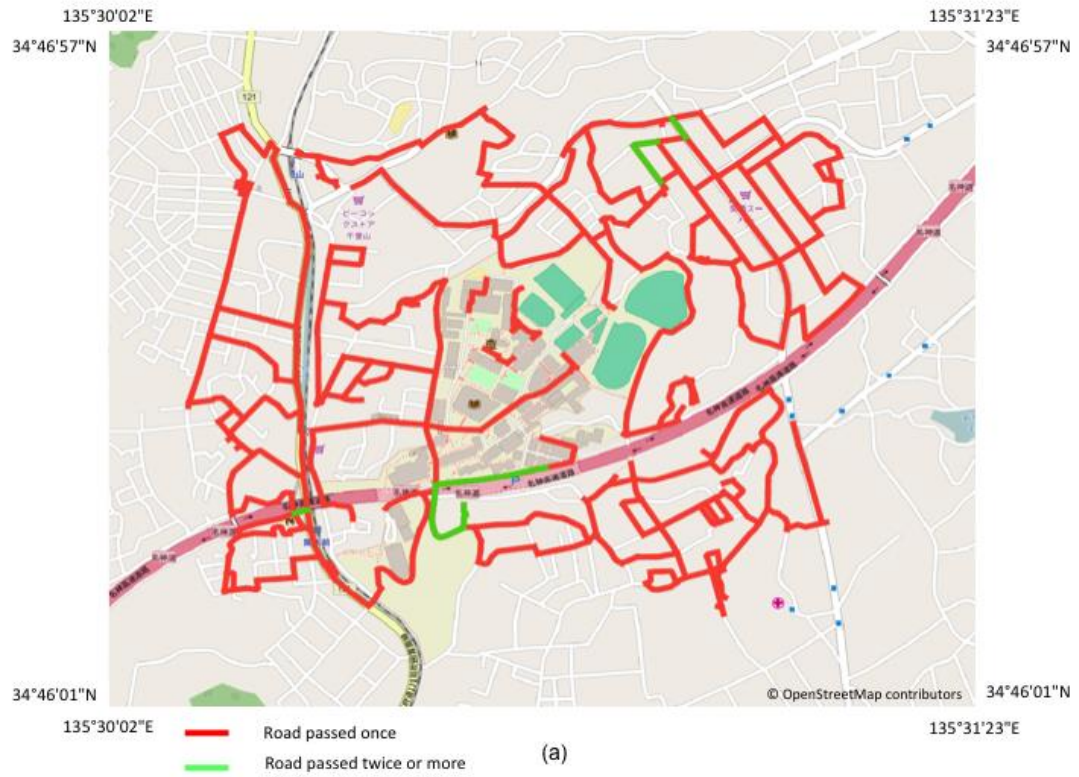


Figure 3.9: Results of map matching (a) Yamate-cho, and (b) Sumiyoshi-ward based map-matching

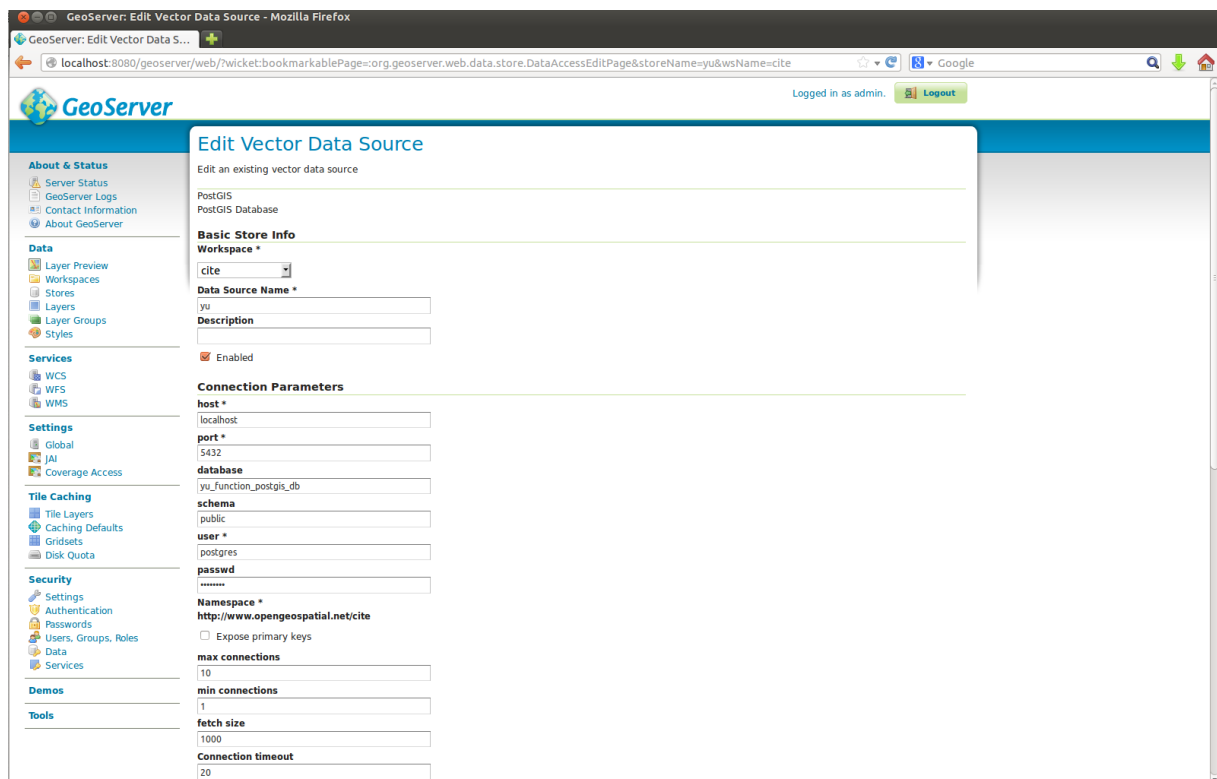


Figure 4.1: GeoServer connecting to PostGIS database

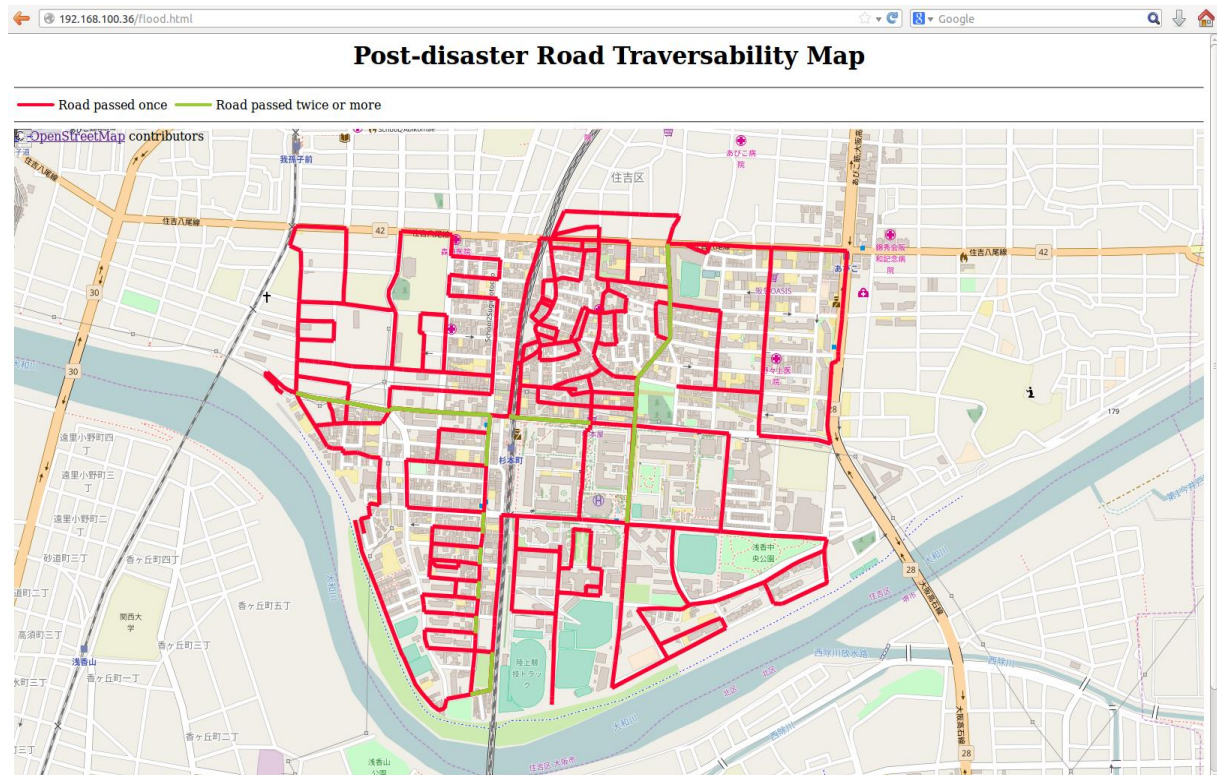


Figure 4.2: Road traversability map displayed as WMS layer in GeoServer

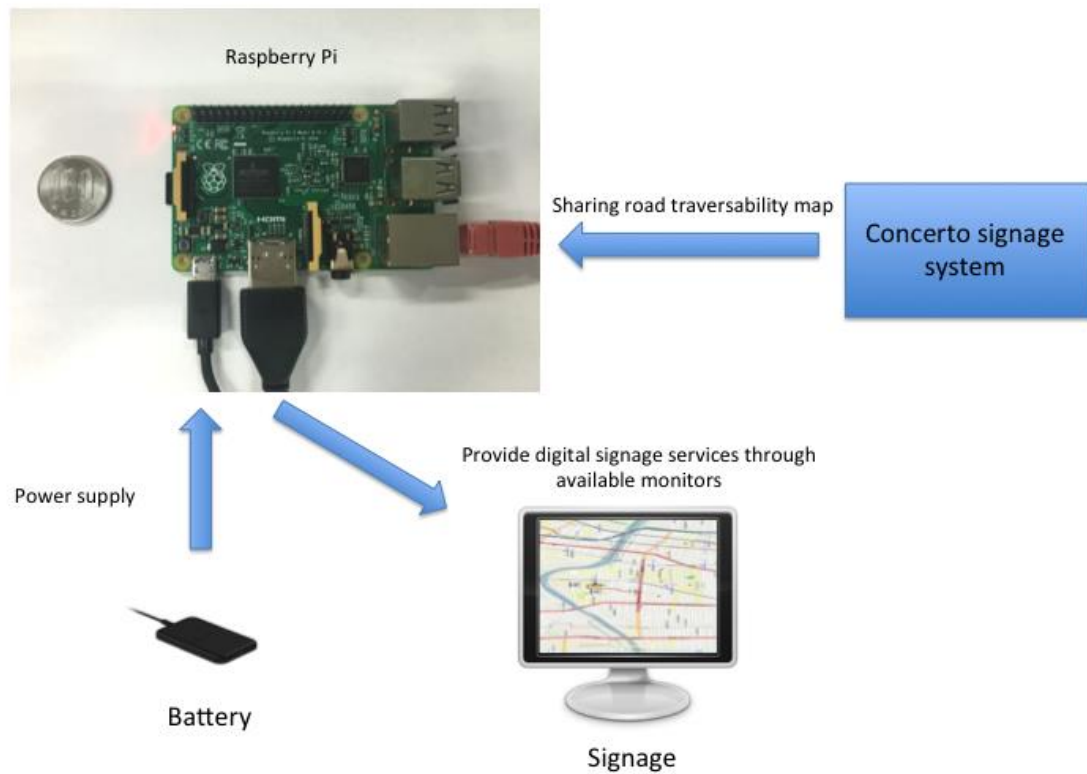


Figure 4.3: Illustration of Raspberry Pi based digital signage system



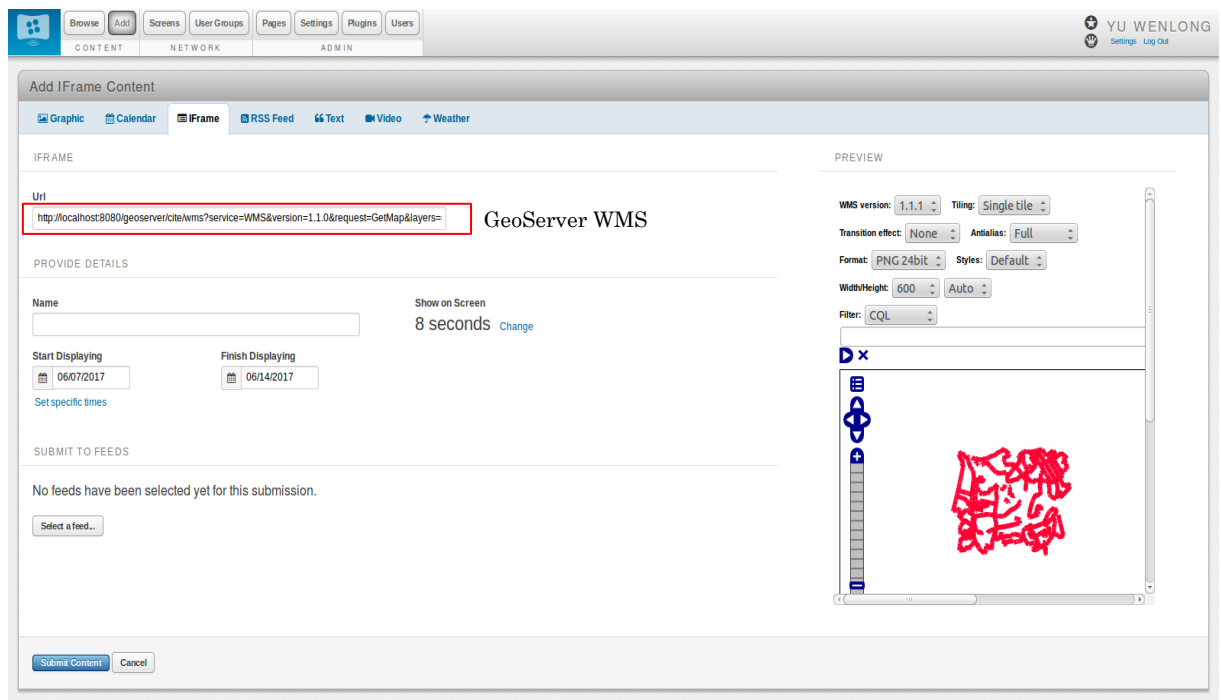


Figure 4.4: Sharing GeoServer WMS layer using Concerto signage system

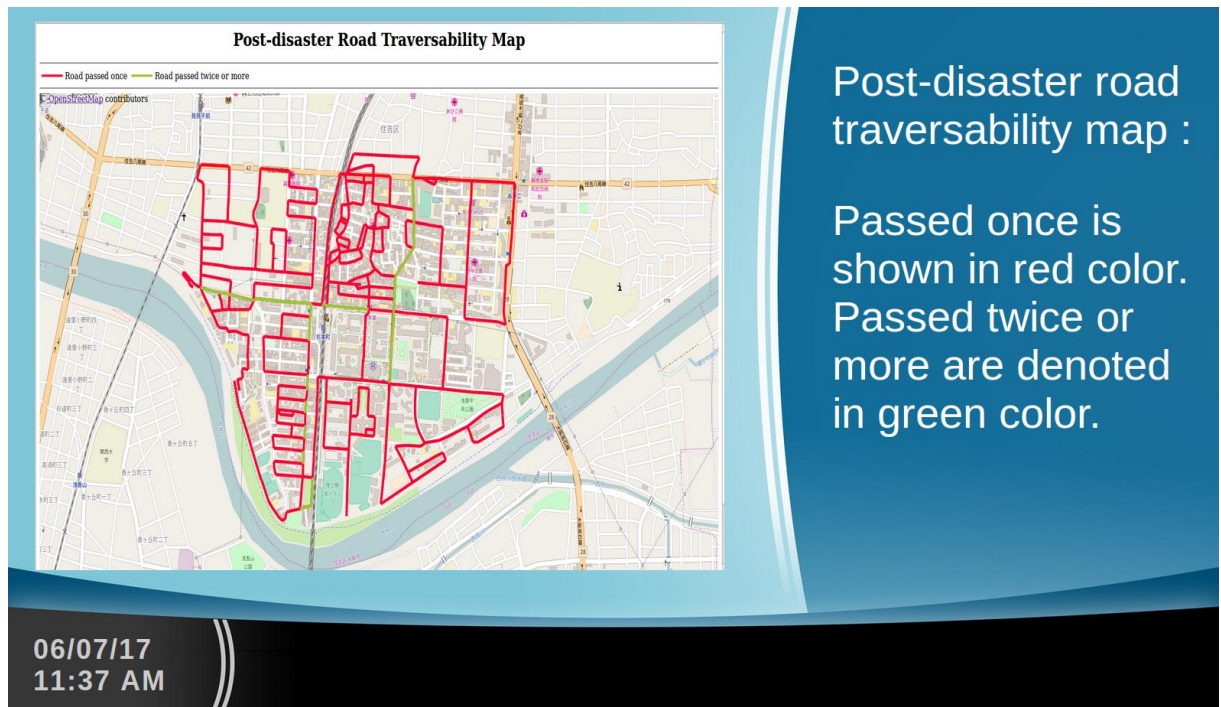



Figure 4.5: View of road traversability map in Concerto signage system



# Geopap-browser

A simple Geopaparazzi Project Browser implementation



Home Project upload Geopaparazzi

## Available projects

Filter by name, description, user or date

Name	Description	Date	User	Delete	Show
geopaparazzi_giovanni	test	2012-04-27 10:06:27	moovida	<a href="#">delete</a>	<a href="#">view</a>
geopaparazzi_giovanni_20120427102124	rrr	2012-04-27 10:21:24	moovida	<a href="#">delete</a>	<a href="#">view</a>
geopaparazzi	ttt	2012-04-27 11:41:23	moovida	<a href="#">delete</a>	<a href="#">view</a>
media0420	111	2014-01-10 16:09:53	yu	<a href="#">delete</a>	<a href="#">view</a>

Figure 4.6: Sharing recorded GPS tracks using Geopap-Cloud

## Gps Logs

[download as shapefile](#)






Description	Start date	End date	Profile chart
yu1120itmenkou	2013-11-20 19:38:19	2013-11-20 19:40:20	
yu1120lowson	2013-11-20 19:44:51	2013-11-20 19:47:22	
yu1121damen	2013-11-21 15:21:34	2013-11-21 15:24:10	
yu1121shitang	2013-11-21 19:03:07	2013-11-21 19:05:18	
yu1121dongbeijiao	2013-11-21 19:05:47	2013-11-21 19:08:43	

Figure 4.7: GPS tracks are displayed in Geopap-Cloud



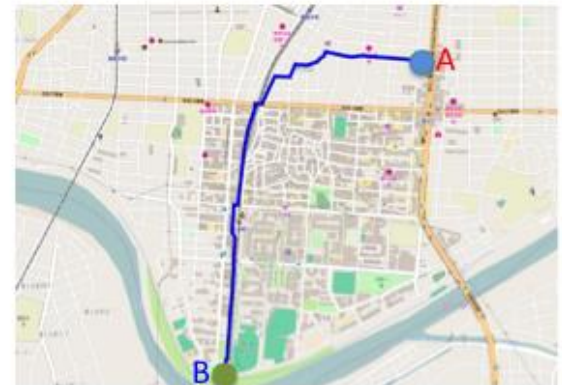
(a) Shortest distance result from location A to B



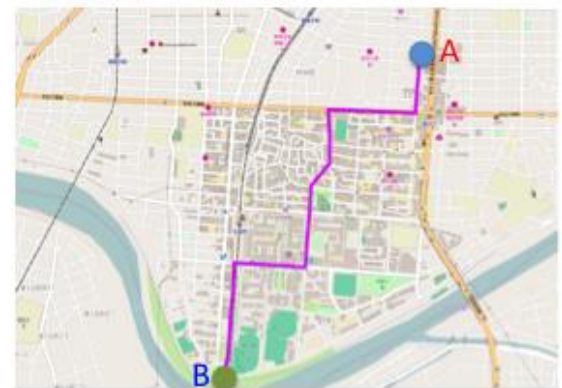
(b) The new result from A to B using road traversability map

Figure 4.8: Shortest path using pgRouting between point A and B inside the traversability road map area a) Normal situation b) Using traversability road map.





(a) Shortest distance result from location A to B



(b) The new result from A to B using road traversability map

Figure 4.9: Shortest path using pgRouting between point A and B outside the traversability road map area a) Normal situation b) Using traversability road map



Figure 4.10: Illustration of tile map.

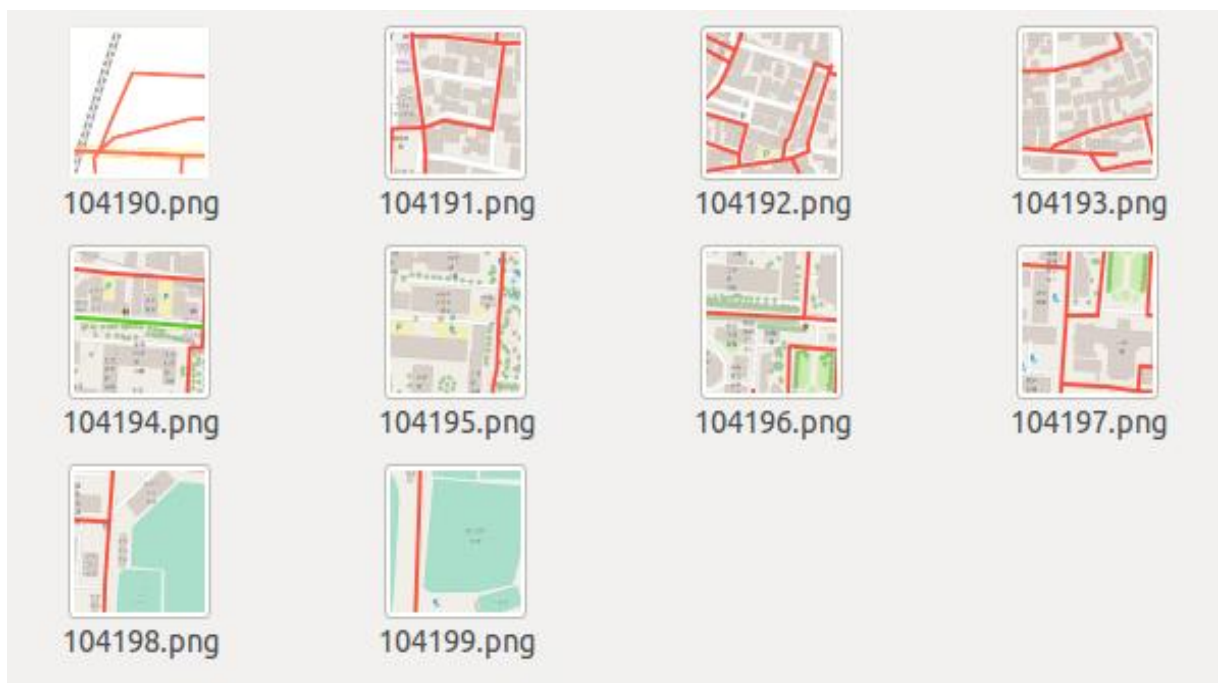


Figure 4.11: Map tiles of traversability road map.

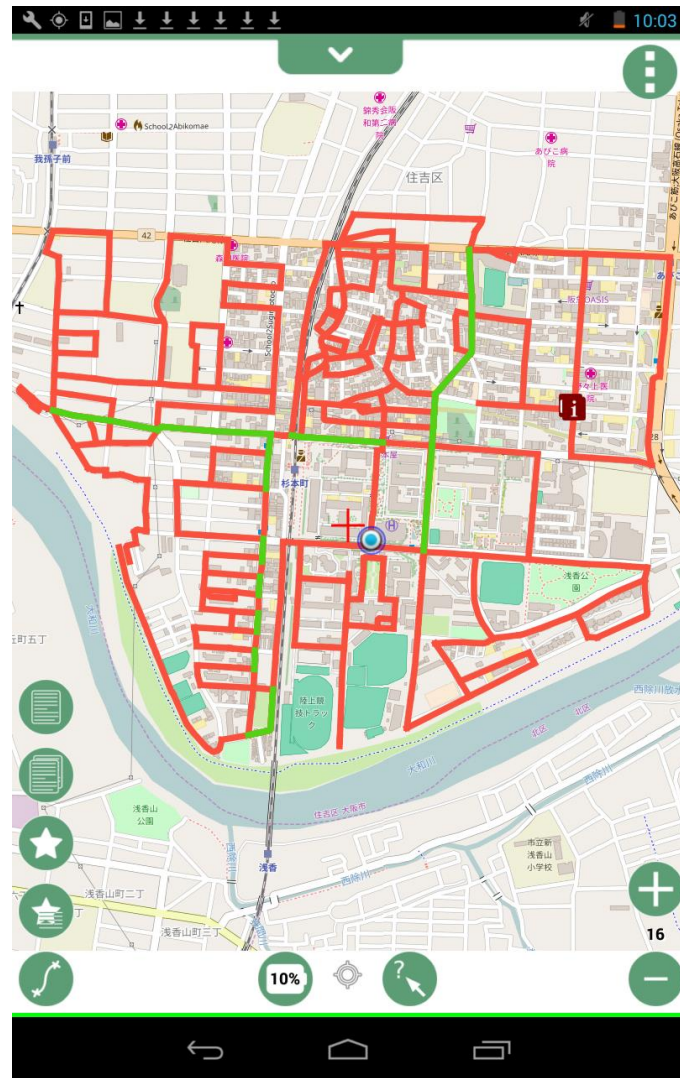


Figure 4.12: Tiled traversability road map is displayed in Geopaparazzi

Table 2.1: Parameters used for field experiments

Items	Parameters
The number of devices	4
GPX file	13 KB
Communication range	30 m
The number of communications	$15 \times 3$ (One device)
Experiment time	15 minutes
The number of cycles	3

Table 2.2: Parameters used for simulation experiments

Items	Parameters
Simulation area	14 km <sup>2</sup>
The number of mobile devices	2000~3000
Moving speed	1.2 m/s
Simulation range	30 m
Communication interval	1 s
Experiment time	3600 s
The number of trials	10
TTL	50

Table 3.1: Sample of acquired location data

ID	Date	Time	Device model	HDOP	Accuracy	Satellites numbers	Latitude	Longitude
0	2016-07-11	14:21:13	Nexus7	1.2	8.0	6.0	34.594456	135.50232
1	2016-07-11	14:21:14	Nexus7	0.8	8.0	6.0	34.59445	135.50224
2	2016-07-11	14:21:15	Nexus7	0.8	8.0	7.0	34.59445	135.50224
3	2016-07-11	14:21:16	Nexus7	0.8	8.0	8.0	34.594448	135.50221
4	2016-07-11	14:21:17	Nexus7	0.8	8.0	8.0	34.594357	135.50203
5	2016-07-11	14:21:18	Nexus7	1.2	8.0	8.0	34.594357	135.50203
6	2016-07-11	14:21:19	Nexus7	0.8	8.0	8.0	34.594327	135.50203
7	2016-07-11	14:21:20	Nexus7	0.8	8.0	8.0	34.594322	135.50192
8	2016-07-11	14:21:21	Nexus7	0.8	8.0	8.0	34.594322	135.5019
9	2016-07-11	14:21:22	Nexus7	0.8	8.0	8.0	34.594322	135.50189
10	2016-07-11	14:21:23	Nexus7	0.8	8.0	8.0	34.594322	135.50188
11	2016-07-11	14:21:24	Nexus7	0.8	8.0	8.0	34.594322	135.50186
12	2016-07-11	14:21:25	Nexus7	0.6	6.0	8.0	34.594322	135.50185
13	2016-07-11	14:21:26	Nexus7	0.6	6.0	8.0	34.59432	135.50183
14	2016-07-11	14:21:27	Nexus7	0.8	8.0	8.0	34.59432	135.50183
15	2016-07-11	14:21:28	Nexus7	0.8	8.0	8.0	34.59432	135.5018
...	...	...	...	...	...	...	...	...

Table 3.2: Road network attribute data used for road traversability mapping system

<b>GID</b>	<b>OSM_id</b>	<b>The_geom</b>
1	196468596	0102000000020000006B0DCA...
2	138932771	010200000004000000DD8993...
3	138949151	01020000000110000007C8F6B...
4	199316349	0102000000090000007C8F6B...
5	198028347	01020000000200000027AA4E...
6	117152095	0102000000020000008D95FF...
7	83417778	01020000000500000043CBF8...
...	...	...

Table 3.3: Parameters used for field experiments

<b>Study area</b>	<b>Yamate-cho, Suita City</b>	<b>Sumiyoshi-ward, Osaka City</b>
The model of devices	Nexus7	Nexus7
The number of devices	6	6
Area covered	2 x 1.5 km	2 x 1.5 km
Experiment time	1 hour	1 hour

### 3.4: Results of field experiments

Yamate-cho, Suita City					Sumiyoshi-ward, Osaka City			
Device number	Number of points	Number of filtered points	Number of matched segments	Running time (s)	Number of points	Number of filtered points	Number of matched segments	Running time (s)
Device1	3525	3363	238	1.125	4157	3208	180	2.273
Device2	3315	3179	197	0.983	3275	3057	153	0.868
Device3	3743	3620	248	1.332	2550	2335	156	0.989
Device4	3818	3099	161	2.743	4158	3975	219	1.703
Device5	3387	3008	201	1.135	2767	2419	188	0.634
Device6	3467	2857	144	1.628	3131	2288	157	1.212

# Appendix A

As mentioned in the Chapter 2 simulation experiments used NS-2 configuration files. Two NS-2 configuration files were used in this study namely aodv.tcl and cbrgen.tcl. In order to run the simulation experiment, parameters such as simulation area, number of devices to be used for simulation, moving speed, simulation range, communication interval, experiment time and number of trials should be mentioned in a separate file ("scene-n1000-m12-t1-x4568-y2988"), which would consider for the simulation. In addition, another file should named "cbr\_n1000\_m100\_r1" is used to specify the network event settings.

- aodv.tcl : make communication between nodes using AODV routing protocol and CBR traffic
- cbrgen.tcl : create connections between the nodes , one can specify the maximum number of connections to be made for all the nodes in the network

Location data transfer time, packet loss and reception rate parameter are derived as the output of the simulation. Further, these parameters are supplied to awk files in order to compute network load, communication frequency and network delay using functions such as load.awk, frequency.awk and delay.awk respectively. Finally, function average.awk is used to calculate the average of the results.

- load.awk : calculate the load of the network
- frequency.awk : calculate the frequency of the communication
- delay.awk : calculate the delay of the network
- average.awk : calculate the average of the results



```

=====
# aodv.tcl
=====

#=====
# Define options
#=====

set val(chan)          Channel/WirelessChannel ;# channel type
set val(prop)          Propagation/TwoRayGround ;# radio-propagation model
set val(netif)         Phy/WirelessPhy ;
set val(mac)           Mac/802_11 ;
set val(ifq)           Queue/DropTail/PriQueue ;
set val(ll)            LL ;
set val(ant)           Antenna/OmniAntenna ;
set val(ifqlen)        50;
set val(nn)            1000;
set val(rp)            AODV;# routing protocol
set opt(cp)            "cbr_n1000_m100_r1" ;
set opt(sc)            "scene-n1000-m12-t1-x4568-y2988" ;

#=====

# Main Program
#=====

set ns_ [new Simulator]
set tracefd [open aodv.tr w]
$ns_ trace-all $tracefd
$ns_ use-newtrace
set namtracefd [open aodv.nam w]
$ns_ namtrace-all-wireless $namtracefd 4568 2988
set topo [new Topography]
$topo load_flatgrid 4568 2988
set god_ [new God]
create-god $val(nn)
$ns_ node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \

```

```

        -ifqLen $val(ifqlen) ¥
        -antType $val(ant) ¥
        -propType $val(prop) ¥
        -phyType $val(netif) ¥
        -channelType $val(chan) ¥
        -topoInstance $topo ¥
        -agentTrace ON ¥
        -routerTrace ON ¥
        -macTrace OFF ¥
        -movementTrace OFF ¥
for {set i 0} {$i < $val(nn)} {incr i}
{
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0
}
source $opt(cp)
source $opt(sc)
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ at 1.1 "$node_($i) reset";
}
$ns_ at 1.2 "stop"
$ns_ at 1.3 "puts ¥"NS exiting...-¥"; $ns_ halt"
proc stop {} {
    global ns_ tracefd namtracefd
    $ns_ flush-trace
    close $tracefd
    close $namtracefd
    exit 0
}
$ns_ run

```

```

=====
# cbrgen.tcl
=====

```

```

# Traffic source generator from CMU's mobile code.
# $Header: /cvsroot/nsnam/ns-2/indep-utils/cmu-scen-gen/cbrgen.tcl,v 1.4 tomh Exp $#

```

```

#=====
# Default Script Options#
#=====

set opt(nn)          0          ;# Number of Nodes
set opt(seed)        0.0
set opt(mc)          0
set opt(pktsize)     512
set opt(rate)        0
set opt(interval)    0.0        ;# inverse of rate
set opt(type)        ""

#=====

proc usage {} {
    global argv0

    puts "¥usage: $argv0 ¥[-type cbr|tcp¥] ¥[-nn nodes¥] ¥[-seed seed¥] ¥[-mc
connections¥] ¥[-rate rate¥]¥n"
}

proc getopt {argc argv} {
    global opt
    lappend optlist nn seed mc rate type
    for {set i 0} {$i < $argc} {incr i}
    {
        set arg [lindex $argv $i]
        if {[string range $arg 0 0] != "-"} continue
        set name [string range $arg 1 end]
        set opt($name) [lindex $argv [expr $i+1]]
    }
}

# create-cbr-connection
proc create-cbr-connection {src dst}
{
    global rng cbr_cnt opt
    set stime [$rng uniform 0.0 360.0]
    set stime1 [expr 20+$stime]
    puts "#¥n# $src connecting to $dst at time $stime¥n#"
    puts "set udp_($cbr_cnt) ¥[new Agent/UDP¥]"
    puts "¥$ns_ attach-agent ¥$node_($src) ¥$udp_($cbr_cnt)"
}

```

```

    puts "set null_($cbr_cnt) ¥[new Agent/Null¥]"
    puts "¥$ns_ attach-agent ¥$node_($dst) ¥$null_($cbr_cnt)"
    puts "set cbr_($cbr_cnt) ¥[new Application/Traffic/CBR¥]"
    puts "¥$cbr_($cbr_cnt) set packetSize_ $opt(pktsize)"
    puts "¥$cbr_($cbr_cnt) set interval_ $opt(interval)"
    puts "¥$cbr_($cbr_cnt) set random_ 1"
    puts "¥$cbr_($cbr_cnt) set maxpkts_ 10000"
    puts "¥$cbr_($cbr_cnt) attach-agent ¥$udp_($cbr_cnt)"
    puts "¥$ns_ connect ¥$udp_($cbr_cnt) ¥$null_($cbr_cnt)"
    puts "¥$ns_ at $stime ¥"¥$cbr_($cbr_cnt) start¥"
    puts "¥$ns_ at $stime1 ¥"¥$cbr_($cbr_cnt) stop¥"
    incr cbr_cnt
}

# create-tcp-connection
proc create-tcp-connection { src dst }
{
    global rng cbr_cnt opt
    set stime [$rng uniform 0.0 180.0]
    puts "#¥n# $src connecting to $dst at time $stime¥n#"
    puts "set tcp_($cbr_cnt) ¥[¥$ns_ create-connection ¥
    TCP ¥$node_($src) TCPSink ¥$node_($dst) 0¥]";
    puts "¥$tcp_($cbr_cnt) set window_ 32"
    puts "¥$tcp_($cbr_cnt) set packetSize_ $opt(pktsize)"
    puts "set ftp_($cbr_cnt) ¥[¥$tcp_($cbr_cnt) attach-source FTP¥]"
    puts "¥$ns_ at $stime ¥"¥$ftp_($cbr_cnt) start¥"
    incr cbr_cnt
}

#=====

getopt $argc $argv
if { $opt(type) == "" } {
    usage
    exit
} elseif { $opt(type) == "cbr" } {
    if { $opt(nn) == 0 || $opt(seed) == 0.0 || $opt(mc) == 0 || $opt(rate) == 0 } {
        usage
        exit
    }
}

```

```

    }
    set opt(interval) [expr 1 / $opt(rate)]
    if { $opt(interval) <= 0.0 } {
        puts "¥ninvalid sending rate $opt(rate)¥n"
        exit
    }
}

puts "#¥n# nodes: $opt(nn), max conn: $opt(mc), send rate: $opt(interval), seed: $opt(seed)¥n#"

set rng [new RNG]
$rng seed $opt(seed)
set u [new RandomVariable/Uniform]
$u set min_ 0
$u set max_ 100
$u use-rng $rng
set cbr_cnt 0
set src_cnt 0
for {set i 0} {$i < $opt(nn)} {incr i}
{
    set x [$u value]
    if {$x < 50} {continue;}
    incr src_cnt
    set dst [expr ($i+1) % [expr $opt(nn) + 1] ]
    if { $opt(type) == "cbr" } {
        create-cbr-connection $i $dst
    } else {
        create-tcp-connection $i $dst
    }
    if { $cbr_cnt == $opt(mc) } {
        break
    }
    if {$x < 75} {continue;}
    set dst [expr ($i+2) % [expr $opt(nn) + 1] ]
    if { $opt(type) == "cbr" } {
        create-cbr-connection $i $dst
    } else {

```

```

        create-tcp-connection $i $dst
    }
    if { $cbr_cnt == $opt(mc) } {
        break
    }
}
puts "#¥n#Total sources/connections: $src_cnt/$cbr_cnt¥n#"

```

```
=====
# delay.awk
=====
```

```

BEGIN{
    highest_packet_id=0;
    duration_total=0;
}
$0 ~/^r.*AGT/{
    receives++;
}
{
    time=$3;
    packet_id=$41;
    if(($1=="s") && ($19=="AGT") && (start_time[packet_id]==0)) {
        start_time[packet_id]=time;
        if(packet_id>highest_packet_id)
            highest_packet_id=packet_id;
    }
    if(($1=="r") && ($19=="AGT") && (end_tim[packet_id]==0)) {
        end_time[packet_id]=time;
    }
    if($1=="d"){
        end_time[packet_id]=-1;
    }
}
END{
    for(packet_id = 0; packet_id <= highest_packet_id; packet_id++) {
        start=start_time[packet_id];

```

```

        end=end_time[packet_id];
        if(end != -1 && start < end) {
            packet_duration=end - start;
            duration_total = duration_total + packet_duration;
        }
    }
    printf("%.4f¥n",duration_total/receives);
}

=====

# frequency.awk
=====

BEGIN{
    requests=0;
    time=$3;
    id=$5;
    source_ip=$57;
    frequency=0;
}
{
    if(($1=="s") && ($61=="REQUEST") && (id==source_ip))
        {requests++;}
}
END{
    frequency=requests/300;
    printf("%.4f¥n",frequency);
}

=====

# getRatio.awk
=====

BEGIN {
    sendLine = 0;
    recvLine = 0;
    fowardLine = 0;
}
$0 ~/^s.* AGT/ {
    sendLine ++ ;

```

```

}
$0 ~/^r.* AGT/ {
    recvLine ++ ;
}
$0 ~/^f.* RTR/ {
    fowardLine ++ ;
}
END {
    printf "cbr  s:%d  r:%d,  r/s  Ratio:%.4f,  f:%d  ¥n",  sendLine,  recvLine,
(recvLine/sendLine),fowardLine;
    printf "%d %.4f¥n", scr, (recvLine/sendLine) >> outfile;
}

=====

# load.awk

=====

BEGIN{
    recvLine=0;
    load=0;
    Normalized_load=0;
}
{
    if(($1=="r") && ($19=="AGT") && ($35=="cbr"))
    {
        recvLine++;
    }
    if(((($1=="s") || ($1=="f")) && ($19=="RTR") &&
(($35=="AODV") || ($35=="message"))))
    {
        load++;
    }
}
END{
    Normalized_load=load/recvLine;
    printf("l:%d,nload:%.4f¥n",load,Normalized_load)

=====

```



```
# average.awk
```

```
=====
BEGIN
{
    average=0.0;
    count=0;
    count_N=10;
}
{
    for(n=0; n<10; n++)
    {
        if($1=="n")
        {
            average=average + $n;
        }
    }
}
END {
    printf "%d %.4f ¥n",time, (average/count_N) >> outfile;
}
```

## Appendix B

As mentioned in the Chapter 3, an Android application used in this research was developed by Java language. Four functions such as MainActivity.java, GPSActivity.java, SocketManager.java and TransmitForServer.java have been compiled to process GPS data transfer and sharing. Workflow, output and other parameters used for each function are explained in detail below.

In order to run developed Android application a function named MainActivity.java has been defined, which can call the other functions and show location data information. Secondly, GPSActivity.java function saving GPS location information that filtered using parameters such as satellite numbers, GPS accuracy and HDOP parameters. Further, as explained in Section 2.3, the SocketManager.java shares location data between nearby devices, which connected through MANET. Finally, TransmitForServer.java upload the collected location data to server.

- MainActivity.java: Show location data information
- GPSActivity.java: Acquiring, filtering and saving location data
- SocketManager.java: Sharing location data
- TransmitForServer.java: Transmit the location data to server.

```
=====
# MainActivity.java
=====
```

```
package com.ssydiai.filetransmit;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

```

import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.app.Activity;
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.text.SimpleDateFormat;
import java.util.Iterator;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.location.Criteria;
import android.location.GpsSatellite;
import android.location.GpsStatus;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
    private TextView tvMsg;
    private EditText txtIP, txtPort, txtEt;
    private Button btnSend, button1, button2, button3;
    private Handler handler;

```

```

private SocketManager socketManager;
private EditText editText;
private LocationManager lm;
public static float GPSAccuracy;
public static float GPSAltitude;
public static float GPSSpeed;
private static final String TAG="GpsActivity";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tvMsg = (TextView)findViewById(R.id.tvMsg);
    txtIP = (EditText)findViewById(R.id.txtIP);
    txtPort = (EditText)findViewById(R.id.txtPort);
    txtEt = (EditText)findViewById(R.id.et);
    button1 = (Button)findViewById(R.id.traserver);
    button1.setOnClickListener(new OnClickListener()

{
@Override
    public void onClick(View v){
        Intent intent = new Intent();
        intent.setClass(MainActivity.this, TransmitForServer.class);
        startActivity(intent);
    }
});
    button2 = (Button)findViewById(R.id.nmea);
    button2.setOnClickListener(new OnClickListener()

{
@Override
    public void onClick(View v){
        Intent intent = new Intent();
        intent.setClass(MainActivity.this, NMEAActivity.class);
        startActivity(intent);
    }
}

```

```

    });
    button3 = (Button)findViewById(R.id.osmmap);
    button3.setOnClickListener(new OnClickListener()
    {
@Override
        public void onClick(View v){
            Intent intent = new Intent();
            intent.setClass(MainActivity.this, OSMMMapActivity.class);
            startActivity(intent);
        }
    });
    btnSend = (Button)findViewById(R.id.btnSend);
    btnSend.setOnClickListener(new OnClickListener(){
@Override
        public void onClick(View v) {
            Intent intent = new Intent(getApplicationContext(),FilesViewActivity.class);
            startActivityForResult(intent, 0);
        }
    });
    handler = new Handler(){
@Override
        public void handleMessage(Message msg) {
            switch(msg.what){
                case 0:
                    SimpleDateFormat format = new SimpleDateFormat("hh:mm:ss:SS");
                    txtEt.append("\n[" + format.format(new Date()) + "]" + msg.obj.toString());
                    break;
                case 1:
                    tvMsg.setText("IP:" + GetIpAddress() + "  port:" + msg.obj.toString());
                    break;
                case 2:
                    Toast.makeText(getApplicationContext(),msg.obj.toString(),
                    Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    }
}

```

```

    };

    socketManager = new SocketManager(handler);
    editText=(EditText)findViewById(R.id.editText);
    lm=(LocationManager)getSystemService(Context.LOCATION_SERVICE);
    if(!lm.isProviderEnabled(LocationManager.GPS_PROVIDER)){
        Toast.makeText(this, "open GPS...", Toast.LENGTH_SHORT).show();
        Intent intent=new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
        startActivity(intent);
    }

    # Acquiring GPS location data
    startActivityForResult(intent,0);
    return;
}

String bestProvider = lm.getBestProvider(getCriteria(), true);
LocationManager locationManager = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);
Location location= lm.getLastKnownLocation(bestProvider);
updateView(location);
lm.addGpsStatusListener(listener);

# If location information has been changed get the new location information
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 1,
    locationListener);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK){
        final ArrayList<String> fileName =
            data.getStringArrayListExtra("filename");
        final ArrayList<String> safeFileName =
            data.getStringArrayListExtra("safeFileName");

        final String ipAddress = txtIP.getText().toString();
        final int port = Integer.parseInt(txtPort.getText().toString());
        Message.obtain(handler, 0, "It's sending" + ipAddress + ":" + port)
            .sendToTarget();

        Thread sendThread = new Thread(new Runnable(){
            @Override

```

```

        public void run() {
            socketManager.SendFile(fileName, safeFileName, ipAddress, port);
        }
    });
    sendThread.start();
}

}

public String GetIpAddress() {
    WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
    WifiInfo wifiInfo = wifiManager.getConnectionInfo();
    int i = wifiInfo.getIpAddress();
    return (192 + "." + 168 + "." + 1 + "." + 201 );
}

@Override
# Showing the GPS location information
protected void onDestroy() {
    super.onDestroy();
}

private LocationListener locationListener=new LocationListener() {
public void onLocationChanged(Location location) {
    updateView(location);
    Log.i(TAG, "時間 : "+location.getTime());
    Log.i(TAG, "経度 : "+location.getLongitude());
    Log.i(TAG, "緯度 : "+location.getLatitude());
    Log.i(TAG, "海拔 : "+location.getAltitude()+"m");
    Log.i(TAG, "精度 : "+location.getAccuracy()+"m");
    Log.i(TAG, "速度 : "+location.getSpeed()+"m/s");
}

public void onStatusChanged(String provider, int status, Bundle extras) {
    switch (status) {
        case LocationProvider.AVAILABLE:
            Log.i(TAG, "今の状態");
            break;
        case LocationProvider.OUT_OF_SERVICE:
            Log.i(TAG, "今の GPS 状況");

```

```

        break;
    case LocationProvider.TEMPORARILY_UNAVAILABLE:
        Log.i(TAG, "停止原因");
        break;
    }
}

public void onProviderEnabled(String provider) {
    Location location=lm.getLastKnownLocation(provider);
    updateView(location);
}

public void onProviderDisabled(String provider) {
    updateView(null);
}

};

private List<GpsSatellite> numSatelliteList = new
ArrayList<GpsSatellite>();

    GpsStatus.Listener listener = new GpsStatus.Listener() {
    public void onGpsStatusChanged(int event) {
        switch (event) {
        case GpsStatus.GPS_EVENT_FIRST_FIX:
            Log.i(TAG, "初めて定位");
            break;
        case GpsStatus.GPS_EVENT_SATELLITE_STATUS:
            Log.i(TAG, "衛星状態変更");
            GpsStatus gpsStatus=lm.getGpsStatus(null);
            int maxSatellites = gpsStatus.getMaxSatellites();
            Iterator<GpsSatellite> iters = gpsStatus.getSatellites().iterator();
            numSatelliteList.clear();
            int count = 0;
            while (iters.hasNext() && count <= maxSatellites) {
                GpsSatellite s = iters.next();
                numSatelliteList.add(s);
                count++;
            }
            System.out.println(count+"衛星");
            break;

```



```

        case GpsStatus.GPS_EVENT_STARTED:
            Log.i(TAG, "定位開始");
            break;
        case GpsStatus.GPS_EVENT_STOPPED:
            Log.i(TAG, "定位終了");
            break;
    }
};

};

* @param location
    */

private void updateView(Location location){
    if(location!=null){
        GPSAccuracy = Float.valueOf(location.getAccuracy());
        GPSSpeed = Float.valueOf(location.getSpeed());
        editText.setText("Location information¥n¥n" + "Satellite numbers : " +
numSatelliteList.size() + "¥nLongitude : ");
        editText.append(String.valueOf(location.getLongitude()));
        editText.append("¥nLatitude : ");
        editText.append(String.valueOf(location.getLatitude()));
        editText.append("¥nAccuracy : ");
        editText.append(String.valueOf(location.getAccuracy()));
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        editText.append("¥nSave Time : ");
        editText.append(sdf.format(location.getTime()));
        editText.append("¥nSpeed : ");
        editText.append(String.valueOf(location.getSpeed()));
        editText.append("¥nk! ");
        editText.append(String.valueOf(GPSAccuracy));
    }else{
        editText.setEditableText().clear();
    }
}

public static float upAccuracy(float str2){
    float str1 = GPSAccuracy;

```

```

        return str1;
    }

    public static float upAltitude(float str3){
        float str1 = GPSAltitude;
        return str1;
    }

    public static float upSpeed(float str4){
        float str1 = GPSSpeed;
        return str1;
    }

    private Criteria getCriteria(){
        Criteria criteria=new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setSpeedRequired(false);
        criteria.setCostAllowed(false);
        criteria.setBearingRequired(false);

        criteria.setAltitudeRequired(false);
        criteria.setPowerRequirement(Criteria.POWER_LOW);
        return criteria;
    }
}

```

```
=====
# GPSActivity.java
=====
```

```

package com.ssydiai.filetransmit;
import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import java.util.Calendar;
import android.os.Bundle;
import android.os.Handler;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;

```

```

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import nmea.NmeaCommon;
import nmea.NmeaFactory;
import nmea.NmeaGpgga;
import nmea.NmeaGpgsv;
import nmea.NmeaManager;
import nmea.NmeaSatellite;
import android.location.GpsStatus;
import android.location.Location;
import android.location.LocationManager;
import android.util.Log;
import android.util.Pair;
import android.widget.ScrollView;
import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.Iterator;
import android.app.Activity;
import android.content.Context;

```

```

import android.content.Intent;
import android.location.Criteria;
import android.location.GpsSatellite;
import android.location.GpsStatus;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.widget.EditText;
import android.widget.Toast;

public class NMEAActivity extends Activity{
private static final String TAG = "GpsStatus";
private static final String LF = "\n";
private NmeaManager mNmeaManager;
private NmeaFactory mNmeaFactory;
private NmeaGpgga mNmeaGpgga;
private GPSActivity mGPSActivity;
private float mNumSatellites ;
public long time ;
public long judge;
public long timeold;
public float latitude ;
public float longitude ;
public static float hdop ;
public float accuracy;
public float altitude;
public float speed;
public int IDNum;
public int LoopNum;
public String location;
private SatelliteView mSatelliteView;
private ScrollView mScrollView;

```

```

private TextView mTextViewResult;
private TextView mShowSaveData;
private LocationManager lm;
private static final String TAG1="GpsActivity";
private HashMap<String,String> mHash = new HashMap<String,String>();
private EditText editText;
public static float GPSAccuracy;
public static float GPSAltitude;
public static float GPSSpeed;
# Setting the location data saving path
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nmea);
        mSatelliteView = (SatelliteView) findViewById( R.id.SatelliteView );
        mScrollView = (ScrollView) findViewById( R.id.ScrollView );
        mTextViewResult = (TextView) findViewById( R.id.TextView1 );
        Button btnRadar = (Button) findViewById( R.id.Button_radar );
        btnRadar.setOnClickListener( new View.OnClickListener() {
@Override
    public void onClick( View view ) {
        }
    });
    Button saveLogging = (Button) findViewById( R.id.LoggingGPS );
    saveLogging.setOnClickListener( new View.OnClickListener() {
@Override
    public void onClick( View view ) {
        }
    });
    Button btnNmea = (Button) findViewById( R.id.Button_nmea );
    btnNmea.setOnClickListener( new View.OnClickListener() {
@Override
    public void onClick( View view ) {
        execNmea();
    }
    });

```

```

mNmeaManager = new NmeaManager( this );
mNmeaFactory = new NmeaFactory();
execRadar();
    }

// Save GPS file
protected void saveGPSLogging() {
try {
if (Environment.MEDIA_MOUNTED.equals(Environment
        .getExternalStorageState())) {
String datetime1 = "";
String datetime2 = "";
SimpleDateFormat tempDate1 = new SimpleDateFormat("yyyy-MM-dd");
SimpleDateFormat tempDate2 = new SimpleDateFormat("hh:mm:ss");
datetime1 = tempDate1.format(new java.util.Date()).toString();
datetime2 = tempDate2.format(new java.util.Date()).toString();
final String PATH = "/sdcard/youGPSLogs/"; // Save Path
final String FILENAME = "/youFullGPSNMEATEST.gpx";
final String FILENAME4 = "/youFilterGPSSongFormatNMEATEST.txt";
final String FILENAME5 = "/youAllDataGPSSongFormat.txt";
File path = new File(PATH);
File f = new File(PATH + FILENAME);
FileWriter fw = null;
FileWriter fw4 = null;
FileWriter fw5 = null;
BufferedWriter bw = null;
BufferedWriter bw4 = null;
BufferedWriter bw5 = null;
try {
        fw = new FileWriter(PATH + FILENAME, true);
        fw4 = new FileWriter(PATH + FILENAME4, true);
        fw5 = new FileWriter(PATH + FILENAME5, true);
        bw = new BufferedWriter(fw);
        bw4 = new BufferedWriter(fw4);
        bw5 = new BufferedWriter(fw5);
# Filter GPS location data and saving on device
String startXML = "<?xml version='1.0' encoding='UTF-8'?">";

```

```

String myreadline = "hdop=" + hdop + "latitude=" + latitude + "longitude=" +
longitude + "NumSatellites=" + mNumSatellites + "accuracyGPS=" + accuracy;

String startGPX = "<gpx creator=¥\"yu!!¥\" version=¥\"1.1¥\">";

String trk = "<trk>¥n<trkseg>";

String trkpt = "<trkpt lat=¥\"\" + latitude + "¥\" lon=¥\"\" + Math.abs(longitude) + "¥\">";

String ele = "<ele>" + altitude + "</ele>";

String GPXTime = "<time>" + datetime1 + "T" + datetime2 + "Z" + "</time>";
String trkptEnd = "</trkpt>";

String endGPX = "</trkseg>¥n</trk>¥n</gpx>";

bw.write(trkpt);
bw.write(GPXTime);
bw.write(ele);
bw.write(trkptEnd);
bw.newLine();
bw.flush();
bw.close();
fw.close();

bw5.write(IDNum + "|" + datetime1 + "|" + datetime2 + "| | | |" + " |
Nexus7 | |" + hdop + "|" + accuracy + "|" + mNumSatellites + "|" + speed + " | | | |" +
latitude + "|" + Math.abs(longitude) );

bw5.newLine();
bw5.flush();
bw5.close();
fw5.close();

if(hdop <= 1.3 & mNumSatellites >= 4 & accuracy <= 30)
{

    bw4.write(IDNum + "|" + datetime1 + "|" + datetime2 + "| | | |" + " |
Nexus7 | |" + hdop + "|" + accuracy + "|" + mNumSatellites + "|" + speed + " | | | |" +
latitude + "|" + Math.abs(longitude) );

    bw4.newLine();
    bw4.flush();
    bw4.close();
    fw4.close();

}

} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();

```

```

        try {
            bw.close();
            fw.close();
        } catch (IOException e1) {
            }
        }

        if (path.exists() && f.exists()) {
            Toast.makeText(NMEAActivity.this, "start save file",
                Toast.LENGTH_SHORT).show();
        } else {
            if (!path.exists()) {
                path.mkdirs();
            }
            if (!f.exists()) {
            }
        }
    } catch (Exception e) {
        Log.d("Fover", "OH no write SDcard faild");
    }
}

@Override
protected void onResume() {
    super.onResume();
    GpsStatus.NmeaListener listener = new GpsStatus.NmeaListener() {
@Override
        public void onNmeaReceived( long timestamp, String nmea ) {
            execNmea( timestamp, nmea );
        }
    };

    mNmeaManager.addNmeaListener( listener );
    mNmeaManager.requestLocationUpdates();
}

@Override
protected void onPause() {
    super.onPause();

```



```

mNmeaManager.removeNmeaListener();
mNmeaManager.removeUpdates();
    }

private void execRadar() {
mSatelliteView.setVisibility( View.VISIBLE );
mScrollView.setVisibility( View.GONE );
    }

private void execNmea() {
mSatelliteView.setVisibility( View.GONE );
mScrollView.setVisibility( View.VISIBLE );
    }

# NEMA parameters
private void execNmea( long timestamp, String nmea ) {
mNmeaFactory.setTimestamp( timestamp );
time = mNmeaFactory.setTimestamp( timestamp );
hdop = NmeaGpgga.ggahdop(hdop);
latitude = NmeaGpgga.ggalatitude(latitude);
longitude = NmeaGpgga.ggalongitude(longitude);
mNumSatellites = NmeaGpgga.num_satellites(mNumSatellites);
System.out.println("HDOP : "+hdop+"!!!");
judge = time - timeold;
System.out.println("time : "+time+"!!!");
System.out.println("timeold : "+timeold+"!!!");
System.out.println("judge : "+judge+"!!!");
if(judge>=5)
    {
        saveGPSLogging();
        timeold = time;
    }

// get the accuracy, location and speed information
accuracy = MainActivity.upAccuracy(accuracy);
altitude = MainActivity.upAltitude(altitude);
speed = MainActivity.upSpeed(speed);
mNmeaFactory.parse( nmea );
mSatelliteView.setSatellites( mNmeaFactory.getAllSatellites() );

```

```

        Pair<String, String> p = mNmeaFactory.getPair();
        mHash.put( p.first, p.second );
        Iterator<String> iterator = mHash.keySet().iterator();
        String msg = "";
        while ( iterator.hasNext() ) {
            String key = (String) iterator.next();
            msg += mHash.get( key ) + LF + LF;
        }

        mTextViewResult.setText( msg );
    }

    @SuppressWarnings("unused")
    private void log_d( String msg ) {
        Log.d( TAG, msg );
    }
}

```

```

=====
# SocketManager.java
=====

```

```

package com.ssydiai.filetransmit;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;

public class SocketManager {
    private ServerSocket server;

```

```

        private Handler handler = null;

# Sharing GPS location using Ad-hoc network
public SocketManager(Handler handler){
    this.handler = handler;
    int port = 9999;
    while(port > 9000){
        try {
            server = new ServerSocket(port);
            break;
        } catch (Exception e) {
            port--;
        }
    }
    SendMessage(1, port);
    Thread receiveFileThread = new Thread(new Runnable(){
@Override
        public void run() {
            while(true){//receive
                ReceiveFile();
            }
        }
    });
    receiveFileThread.start();
}

void SendMessage(int what, Object obj){
    if (handler != null){
        Message.obtain(handler, what, obj).sendToTarget();
    }
}

void ReceiveFile(){
    try{
        Socket name = server.accept();
        InputStream nameStream = name.getInputStream();
        InputStreamReader streamReader = new InputStreamReader
(nameStream);

```

```

        BufferedReader br = new BufferedReader(streamReader);
        String fileName = br.readLine();
        br.close();
        streamReader.close();
        nameStream.close();
        name.close();
        SendMessage(0, "Now Receiving:" + fileName);
        Socket data = server.accept();
        InputStream dataStream = data.getInputStream();
        String savePath = Environment.getExternalStorageDirectory().getPath() +
        "/" + fileName;
        FileOutputStream file = new FileOutputStream(savePath, false);
        byte[] buffer = new byte[1024];
        int size = -1;
        while ((size = dataStream.read(buffer)) != -1){
            file.write(buffer, 0 ,size);
        }
        file.close();
        dataStream.close();
        data.close();
        SendMessage(0, fileName + "Receive Completed");
    }catch(Exception e){
        SendMessage(0, "Receive Error:¥n" + e.getMessage());
    }
}

# Transmit data (device to device)

public void SendFile(ArrayList<String> fileName, ArrayList<String> path, String
ipAddress, int port){
    try {
        for (int i = 0; i < fileName.size(); i++){
            Socket name = new Socket(ipAddress, port);
            OutputStream outputName = name.getOutputStream();
            OutputStreamWriter          outputWriter          =          new
OutputStreamWriter(outputName);
            BufferedWriter bwName = new BufferedWriter(outputWriter);
            bwName.write(fileName.get(i));

```

```

        bwName.close();
        outputWriter.close();
        outputName.close();
        name.close();
        SendMessage(0, "Now Sending" + fileName.get(i));
        Socket data = new Socket(ipAddress, port);
        OutputStream outputData = data.getOutputStream();
        FileInputStream fileInput = new FileInputStream(path.get(i));
        int size = -1;
        byte[] buffer = new byte[1024];
        while((size = fileInput.read(buffer, 0, 1024)) != -1){
            outputData.write(buffer, 0, size);
        }
        outputData.close();
        fileInput.close();
        data.close();
        SendMessage(0, fileName.get(i) + "Send Completed");
    }
    SendMessage(0, "All files Completed");
} catch (Exception e) {
    SendMessage(0, "Send Error:¥n" + e.getMessage());
}
}
}
}

```

```

=====
# TransmitForServer.java
=====

```

```

package com.ssydiai.filetransmit;
import android.app.Activity;
import android.os.Bundle;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.InputStreamReader;

```

```

import java.net.HttpURLConnection;
import java.net.URL;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class TransmitForServer extends Activity{
private String uploadFile = "/storage/emulated/0/youFilterGPSSongFormatNMEATEST.txt ";
private String srcPath = "/storage/emulated/0/youFilterGPSSongFormatNMEATEST.txt ";
private String actionUrl = "http://160.193.95.10/receive_file.php";
private TextView mText1;
private TextView mText2;
private Button mButton;

@Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_transmitoserver);
        mText1 = (TextView) findViewById(R.id.uploadText);
        mText1.setText("File address : ¥n" + uploadFile);
        mText2 = (TextView) findViewById(R.id.uploadIPText);
        mText2.setText("Server address : ¥n" + actionUrl);
        mButton = (Button) findViewById(R.id.uploadButton);
        mButton.setOnClickListener(new View.OnClickListener()
        {
@Override
# Upload GPS location data to Server
        public void onClick(View v)
        {
            uploadFile(actionUrl);
        }
        });
    }
    private void uploadFile(String uploadUrl)

```

```

{
    String end = "¥r¥n";
    String twoHyphens = "--";
    String boundary = "*****";
    try
    {
        URL url = new URL(uploadUrl);
        HttpURLConnection httpURLConnection = (HttpURLConnection) url
            .openConnection();
        httpURLConnection.setDoInput(true);
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setUseCaches(false);
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setRequestProperty("Connection", "Keep-Alive");
        httpURLConnection.setRequestProperty("Charset", "UTF-8");
        httpURLConnection.setRequestProperty("Content-Type",
            "multipart/form-data;boundary=" + boundary);
        DataOutputStream dos = new DataOutputStream(
            httpURLConnection.getOutputStream());
        dos.writeBytes(twoHyphens + boundary + end);
        dos.writeBytes("Content-Disposition: form-data; name=¥uploadedfile¥";
filename=¥" + srcPath.substring(srcPath.lastIndexOf("/") + 1) + "¥" + end);
        dos.writeBytes(end);
        FileInputStream fis = new FileInputStream(srcPath);
        byte[] buffer = new byte[8192]; // 8k
        int count = 0;
        while ((count = fis.read(buffer)) != -1)
        {
            dos.write(buffer, 0, count);
        }
        fis.close();
        dos.writeBytes(end);
        dos.writeBytes(twoHyphens + boundary + twoHyphens + end);
        dos.flush();
        InputStream is = httpURLConnection.getInputStream();
        InputStreamReader isr = new InputStreamReader(is, "utf-8");
    }
}

```

```
        BufferedReader br = new BufferedReader(isr);
        Toast.makeText(this, result, Toast.LENGTH_LONG).show();
        dos.close();
        is.close();
    } catch (Exception e)
    {
        e.printStackTrace();
        setTitle(e.getMessage());
    }
}
```



# Appendix C

As mentioned in the Chapter 3, map-matching was carried out by using Hausdorff distance algorithm which developed using python language. This research modified Hausdorff distance algorithm to develop a new python function (hausdorffYu.py) used for map-matching. GPS tracks and road network data were used as input for map-matching. The hausdorffYu.py describes how to connect to PostgreSQL/PostGIS database, calculate the GPS tracks and roads geography spatially intersect and save results as shape file.

```
=====
# hausdorffYu.py
=====

#!C:\Python24\python.exe -u
#-*- coding:utf-8 -*-

import math
import os
import pgdb
from shapely.geometry import *
from shapely.wkt import loads
from datetime import datetime,timedelta
from osgeo import ogr
class HAUSDORFF:
    def __init__(self,gps,utb,ute):
# connect to database
        self.conn = pgdb.connect(host="localhost:5432", database='*****', user='****',
password='*****')
        self.cur = self.conn.cursor()
        self.gps = gps
        self.utb = utb
        self.ute = ute
        self.attrlist = []
        self.featlist = []
        self.featenvl = []
```

```

def read(self,path):
    sql = "select oid,ST_AsEWKT(the_geom) from trace where path = %i and time >
to_timestamp('%s', 'YYYY-MM-DD HH24:MI:SS') and time < to_timestamp('%s',
'YYYY-MM-DD HH24:MI:SS') order by time" % (path,self.utb,self.ute)

    print sql
    self.cur.execute(sql)
    pathlist = self.cur.fetchall()
    print 'path: ',len(pathlist)
    if(len(pathlist) > 0):
        wkt = 'LINESTRING('
        for pathr in pathlist:
            #print pathr[1][10:]
            p = loads(pathr[1][10:])
            wkt += str(p.x)+' '+str(p.y) + ','
        wkt = wkt[:-1]
        wkt += ')'
        feat = [wkt,path,0,len(pathlist),loads(wkt).length]
        self.featlist += [feat]
        print 'read_finished'
    def read_(self,infilename):
# open shape file
        shpnam = infilename
        ds = ogr.Open( shpnam )
# read shape file
        la = ds.GetLayer( 0 )
        extent = la.GetExtent()
        self.featenvl = [extent[0]-50,extent[1]+50,extent[2]-50,extent[3]+50 ]
        layerdef = la.GetLayerDefn()
        for i in range(layerdef.GetFieldCount()):
            defn = layerdef.GetFieldDefn(i)
            self.attrlist
            [(defn.GetName(),defn.GetWidth(),defn.GetType(),defn.GetPrecision())]
            f = la.GetNextFeature()
            while f is not None:
                attr = []
                for i in range(f.GetFieldCount()):

```

+=

```

        attr += [f.GetField(i)]
    curv = []
    geom = f.GetGeometryRef() # get the first geometry (supposed a polygon)
    wkt = geom.ExportToWkt()
    feat = [wkt] + attr
    self.featlist += [feat]
    f = la.GetNextFeature()
#close shape file
del ds
def fetchroad(self,poly):
    polywkt = poly.to_wkt()
    sql = "select gid as id,st_asText(the_geom) from osaka_roads_900913 where
ST_Intersects('%s'::geometry,the_geom)" % (polywkt)
    self.cur.execute(sql)
    recordlist = self.cur.fetchall()
    if len(recordlist)>0:
        geomlist = []
        for rec in recordlist:
            wkt = rec[1]
            geomlist += [(rec[0],wkt)]
        if len(recordlist) > 0 and len(geomlist) > 0:
            return geomlist
    else:
        return None
def saveassshape(self,geomtype,featlist,outputfile): # save as shape
if geomtype == 0:
    datatype = ogr.wkbPoint
elif geomtype == 1:
    datatype = ogr.wkbLineString
elif geomtype == 2:
    datatype = ogr.wkbPolygon
driver = ogr.GetDriverByName("ESRI Shapefile")
if os.access(outputfile, os.F_OK ):
    driver.DeleteDataSource(outputfile)
# create shape file
ds_new = driver.CreateDataSource(outputfile)

```

```

layer_new = ds_new.CreateLayer(outputfile, None, datatype)
field_id = ogr.FieldDefn("id",ogr.OFTInteger)
field_id.SetWidth(8)
layer_new.CreateField(field_id)
field_max = ogr.FieldDefn("hausdorff",ogr.OFTReal)
field_max.SetWidth(8)
field_max.SetPrecision(2)
layer_new.CreateField(field_max)
field_mean = ogr.FieldDefn("weighed",ogr.OFTReal)
field_mean.SetWidth(8)
field_mean.SetPrecision(2)
layer_new.CreateField(field_mean)
field_min = ogr.FieldDefn("shortest",ogr.OFTReal)
field_min.SetWidth(8)
field_min.SetPrecision(2)
layer_new.CreateField(field_min)
if(len(featlist) > 0):
    for feat in featlist:
        geom = ogr.CreateGeometryFromWkt(feat[0])
        ft_new = ogr.Feature(layer_new.GetLayerDefn())
        ft_new.SetGeometry(geom)
        ft_new.SetField('id', feat[1])
        ft_new.SetField('hausdorff', feat[2])
        ft_new.SetField('weighed', feat[3])
        ft_new.SetField('shortest', feat[4])
        layer_new.CreateFeature(ft_new)
# close shape file
ds_new.Destroy()
def match(self):
    utdict = {}
    for feat in self.featlist:
        time0 = datetime.utcnow()
        wkt = feat[0]
        linb = loads(wkt)
# set the buffsize = 23 because (6.5 x 2 + 10)
buffsize = 23

```

```

buff = linb.buffer(buffsize)
self.saveasshape(2,[[buff.to_wkt(),1,0,0,0]],'hausdorff_'+str(feats[1])+'_buffer.shp')
# save as shape file (hausdorff_num_buffer.shp)
print 'b'
linalist = self.fetchroad(buff)
print 'e'
if linalist == None:
    return
ftlist = []
count = 0
for tup in linalist:
    id = tup[0]
    lina = loads(tup[1])
    print count
    count += 1
    dist = self.hausdorff(lina,linb)
    dist[-1] = lina.crosses(buff)
    ft = [lina.to_wkt(),id] + dist
    ftlist += [ft]
time1 = datetime.utcnow()
utdict[feats[1]] = (time1-time0).seconds+(time1-time0).microseconds/1000000.0
if len(ftlist) > 0:
    geomtype = 1
    outfilename = 'hausdorff_'+str(feats[1])+'_match.shp'
    test.saveasshape(geomtype,ftlist,outfilename)
print "-----¥n%8s %12s¥n-----" % ('trace_id','match_time')
for id,ut in utdict.items():
    print "%8i %12.3f" % (id,ut)
print "-----"
#hausdorff distance between linearstring geometry (shapely)
def hd(self,lina,linb):
    return max([self.hausdorff(lina,linb),self.hausdorff(linb,lina)])
def hausdorff(self,lina,linb):
    hab = []
    weighed = 0
    pnta = lina.coords

```

```

sizea = len(pnta)
for i in range(sizea):
    x,y = pnta[i][0:2]
    hab += [Point(x,y).distance(linb)]
return [max(hab),sum(hab)/len(hab),min(hab)]

def hausdorff_(self,lina,linb):
    hab = []
    hba = []
    weighed = 0
    pnta = lina.coords
    pntb = linb.coords
    sizea = len(pnta)
    for i in range(1,sizea):
        pa0 = pnta[i-1]
        pa1 = pnta[i]
        sht0 = self.shortest(pa0,pntb)
        sht1 = self.shortest(pa1,pntb)
        hab += [max(sht0[2],sht1[2])]
        seg = LineString(((pa0[0], pa0[1]), (pa1[0], pa1[1])))
        loc = sht0[0:2] + sht1[0:2]
        loc.sort()
        if loc[0]==loc[1] and loc[2]==loc[3]:
            hba += [0]
        else:
            dist = [0]
            for i in range(loc[1],loc[2]+1):
                pb = pntb[i]
                dist += [Point(pb[0],pb[1]).distance(seg)]
            hba += [max(dist)]
        weighed += max(hab[-1],hba[-1])*seg.length/lina.length
    return [max(max(hab),max(hba)),weighed,min(min(hab),min(hba))]

def shortest(self,pa,pntb):
    size = len(pntb)
    res = None
    for i in range(1,size):
        pb0 = pntb[i-1]

```

```

        pb1 = pntb[i]
        seg = LineString(((pb0[0], pb0[1]), (pb1[0], pb1[1])))
        dist = Point(pa[0],pa[1]).distance(seg)
    if res == None:
        res = [i-1,i,dist]
    else:
        if dist < res[2]:
            res = [i-1,i,dist]

    return res

def hausdorff02(self,lina,linb):
    hab = []
    para = 0
    acum = 0
    pnt0 = None
    pnta = lina.coords
    pntb = linb.coords
    for pa in pnta:
        pnt1 = Point(pa[0],pa[1])
        hab += [pnt1.distance(linb)]
        if pnt0 != None:
            len = pnt1.distance(pnt0)
            para += len
            acum += (hab[-1]+hab[-2])*0.5*len
        pnt0 = pnt1
    return [max(hab),acum/para,min(hab)]

if __name__ == "__main__":
    gps = "
    utb = 'yyyy-mm-dd 00:00:00'
    ute = ' yyyy-mm-dd 23:59:59'
    test=HAUSDORFF(gps,utb,ute)
    infilename = "tmp_allpath.shp"
    test.read(yyymmdd-no)
    test.match()
    print 'finished!'

```