HMM Map Matching for Trajectories in City Areas with Multipath Errors

Zhong, Yu

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Applied Geometric Algorithms Group

HMM Map Matching for Trajectories in City Areas with Multipath Errors

Master Thesis

Yu Zhong

Supervisors:
Marcel Roeloffzen,
Kees Huizing
Kevin Verbeek

Eindhoven, May 2021

Abstract

The map matching is an essential preprocessing step for many location-based or trajectory-based applications since it can improve the accuracy of the trajectories. It matches a series of user location points collected from the Global Positioning System (GPS) to road segments in an existing road network. However, GPS suffers from invisible satellite signals and multipath effects due to tall buildings in urban areas, and this leads to inaccuracy in user location points, which influences the performance of map matching. Although many approaches have been employed to match trajectories to paths in a road network, the idea that combines multipath mitigation and map matching is still not well exploited. Therefore, this study proposed a method in which multipath mitigation is integrated with Hidden Markov Model (HMM) map matching. The proposed algorithm uses 3D building models to detect non-line-of-slight (NLOS) signals, mitigates multipath effects and computes a more reasonable probability distribution among the road segment candidates. Then the HMM map matching is employed on these candidates with improved probabilities to select the globally optimal path.

The proposed method was verified through experimental tests in urban areas in Newcastle upon Tyne, the UK. We collected data in this urban area including 7 trajectories, the 2D building map, elevation and satellite positions, and performed the proposed algorithm. The results show that the proposed method slightly improved the performance of the HMM map matching as it provides a better matching path for 5 trajectories. For the other two trajectories, it degraded the accuracy of the conventional matching only in a few specific places. Regardless of the occasional instability of the proposed method, this study reveals the potential of combining multipath mitigation with the conventional map matching, especially for trajectories in urban canyons.

Preface

As a result of my final project, this thesis marked the end of my master education and my student life. I would like to take this opportunity to express my deep and sincere gratitude to people that have supported me in this final project.

Firstly, I would like to thank my supervisor, Marcel Roeloffzen, for his patience, his continuous guidance, his commitment and countless excellent discussions. His input and feedback helped me shape this complex project and thesis.

Next, I would like to thank my fiance, Chuang Song, for sharing his knowledge about GPS with me, supplying me with surveying equipment and helping me to collect trajectory data.

Lastly, my honest thanks to my family and friends for their listening ear and unwavering support throughout the past six month.

Yu Zhong,
Auguest, 2021

Contents

Chapter 1

Introduction

1.1 Problem Description

Nowadays, the successful construction of Global Navigation Satellite Systems (GNSSs) and wide deployment of GNSS sensors have benefited position fixing which is frequently involved in our daily used services, such as route guidance, fleet management, road user charging, accident and emergency response and other location-based services (LBS) [32]. GNSS sensors determine positions based on satellite signals received from GNSSs and a successful position fixing relies on good quality of signals. For many LBS, the accuracy of the position data is one of the most important prerequisites but the accuracy not only depends on the hardware of sensors. The environment is another factor that should be taken into consideration and some challenging environments lead to bad quality position data. For instance, when surveying in city areas with many skyscrapers, often call an urban canyon, the positioning performance often dramatically degrades due to blocked, reflected and diffracted satellite signals.

There are three possible situations for bad quality signals:

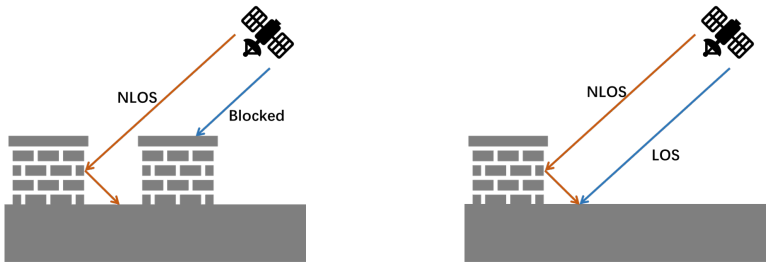
1. When signals are completely blocked, they will not reach a GNSS sensor. And this usually results in loss of lock, which means the sensor cannot decode position data due to fewer visible satellite signals [22].

2. Sometimes GNSS sensors can still receive a blocked signal via a reflected path and these kind of signals are called non-line-of-sight (NLOS) signals (see Figure 1.1a). A NLOS signal takes a longer path from the satellite, resulting in an overestimation of the distance between the satellite and the sensor.

3. Also, interference occurs when both NLOS signals and the direct line-of-sight (LOS) signal for a same satellite reach the sensor. Simply speaking, the NLOS signals interfere the reception of the LOS signal (see Figure 1.1b).

The latter two phenomena are usually referred to as multipath but they require different mitigation techniques [11, 13, 27]. But in some other literature like [20, 28, 4], multipath only refers to the third phenomenon and the second is called NLOS problem. In this project, we follow the definition in [11, 13, 27] but distinguish these two phenomena by NLOS errors and multipath interference errors.

Map-matching algorithms take positioning data as input and supplement this with the road network to provide an enhanced positioning output [32] or to find the user's travel route. Map matching is a preprocessing step for some LBS applications, like navigation, vehicle tracking and traffic surveillance [5]. For noisy positioning data, a map-matching process is usually performed to infer a path on the road network to eliminate the influence of observation errors. However, the drift of GNSS positioning caused by multipath errors can be large sometimes and influences the map matching results, especially in urban areas. Figure 1.2 shows an example trajectory influenced by



(a) The NLOS signal. (b) Multipath interference.

Figure 1.1: Two phenomena of multipath.

multipath errors. Naturally, we hope to mitigate the influence of multipath error and this process is known as multipath mitigation. Therefore, we consider combining multipath mitigation with the map matching process, which means to resolve some potential positions regarding multipath before matching.

Figure 1.2: An example trajectory influenced by multipath errors. The trajectory consists of 6 green dots and the road segment marked in red is where the sensor actually passed.

In this final project, we aim at improving existing map matching techniques by using local building geometry and information of satellites to mitigate potential multipath errors. The rest of paper is structured as follows: in later sections of this chapter, we discuss some existing techniques for multipath mitigation and map matching, as well as introduce the overall design of our project. In Chapter 2 we describe some background knowledge and notations. Then the process and details of data preparation are explained in Chapter 3. In Chapter 4 and 5, we display the principles of the key parts of our project, multipath mitigation and map matching. Later we compare the results in Chapter 6 and draw a conclusion in Chapter 7.

1.2 Literature Survey

There are many different approaches to mitigate multipath errors including NLOS and interference errors. According to [4, 13, 28], multipath mitigation techniques constitute the following high-level solutions:

1. Hardware mitigation. In this category, the advanced features of hardwares are utilized to minimize the multipath effect, including antenna design, receiver-based techniques and using signal quality parameters, such as signal-to-noise ratio (SNR), Doppler frequency and so on, These methods mainly mitigate the effects of multipath interference errors and do little work on NLOS [13].

2. Identification. Except for making use of hardware, the rest of techniques can be subsumed to this category where NLOS signals are identified for further processing. Depending on the way of further processing, these methods can be subdivided into:

   (a) Elimination, which means to eliminate NLOS effects after identifying them. For instance, to modify the receiver's processing strategy to achieve better performance, or to make use of explicit models (e.g. environmental models) to filter and smooth NLOS signals [3].

(b) Exploitation. Instead of mitigating multipath at its root (directly eliminate NLOS signals), they identify NLOS signals and take advantage of these information to improve current results. Methods under this group includes using 3D city models [24, 25, 6], consistency checking which computes positions for different combination of satellite signals, and etc.

3. Physically avoiding NLOS, which means to avoid surveying in challenging areas.

Based on the classification above, the idea of using 3D building models to mitigate multipath errors falls in the category of exploitation. Compared to hardware mitigation and elimination, methods in the category of exploitation are more feasible for us since we can not improve the receiver itself. Many works have been done to mitigate multipath errors by using 3D building models. In [6], Drevelle et al. represented 3D area maps and GPS pseudo range measurements by geometric constraints, turning the positioning problem into a constraint satisfaction problem. Their method has a 95% probability to provide consistent error bounds with 16-meter average radius location zones. In another work of Kumar et al.[19], they proposed a positioning method of predicting NLOS signal delays from those observed pseudo ranges using a 3D city model. Through this method, the errors are bounded by 15 meters. Miura et al.[24, 25] proposed a positioning method by simulating possible paths for multipath signals. They proved that the method provides error bounds with 13.5 meters maximum. Among these methods, we decide to adapt the method of [24] and integrate it with map matching. Our adaption simplifies the original method. After all, our purpose is to find the most possible match for a trajectory sample instead of generating a precise position after correction. Detailed information about the adapted multipath mitigation method is described in Chapter 4. According to [12], 3D building models can have a fair performance in mitigating NLOS errors but do not work for interference errors. Since dealing with interference errors are always related to hardware and waves, it is not something that can be done with data and equipment we currently have. Thus, we decide to focus on mitigating NLOS errors.

Map matching techniques, on the other hand, can be divided into four groups as proposed by the work of Chao et al.[5]:

1. Similarity models. This category is equivalent to geometric and topological analysis models in [32] that returns the road segments that are geometrically or topologically closest to the trajectory. These approaches match points/curves to closest curves where the closeness is defined by different similarity metrics[5].For example, except for Euclidean distance, Fréchet distance and longest common subsequence can also be used to measure geometric similarity.

2. State-transition models. These models return global optima solutions. The rational of this kind of methods is to build a state machine where states refer to the possible routes the user travels at a particular moment and transition between states are labelled by a transition possibility. This category includes Hidden Markov model (HMM), Conditional Random Field (CRF) and Weighted Graph Technique (WGT).

3. Candidate-evolving models. These models hold a candidate set during the whole process and the candidate set is updated by adding new road segment candidates while pruning candidates that are less irrelevant.

4. Scoring models. They build a sequence of candidate sets for each sample in the trajectory and choose a combination of road segments that can maximizes the predefined scoring function.

Among all the methods mentioned above, HMM model is one of the most widely used map matching algorithms in practice as it simulates the road network topology[5] and it well suits the process of finding the most suitable matching road (i.e., hidden state) to each GPS point (i.e., observed state) on the road network [7]. Also, it can be extended to many variants by using different probability functions. The idea of using HMM in map matching was first proposed by Pink et al. in [30] where they increased the robustness of map matching by using an extended Kalman filter and incorporating the road network topology constructed by HMM. Later Newson

et al. [26] proposed a novel map matching algorithm that completely uses HMM to find the most likely path. They proved that this method performs well in high-sampling trajectories and will not significantly degrade when the sampling rate decreases and the measurement noise increases. Then more restrictions and influence factors are considered and integrated into HMM-based algorithms, such as the number of turns [29] and velocity changes[9]. Detailed description for HMM map matching can be seen in Chapter 5.

1.3 Overall Design

In this section, we will discuss the design and structure of our project at a high level. Figure 1.3 shows the common workflow of HMM map matching algorithms. From this figure and the discussion in the previous section, we can see that an HMM map matching algorithm can be briefly divided into two main parts:

- Candidate selection. In this part, some potential positions (or road segments) are selected for each sample in the trajectory. The candidate set is prepared for further processing in the following steps.
- The matching process. The concrete methods in this matching process differ according to map matching algorithms themselves. Generally, road segments are selected from the candidate sets to formulate a route.

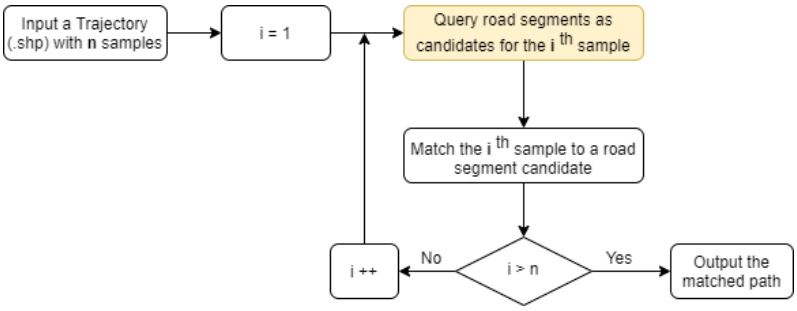


Figure 1.3: The common workflow of HMM map matching algorithms. The process in yellow box is the process of candidate selection.

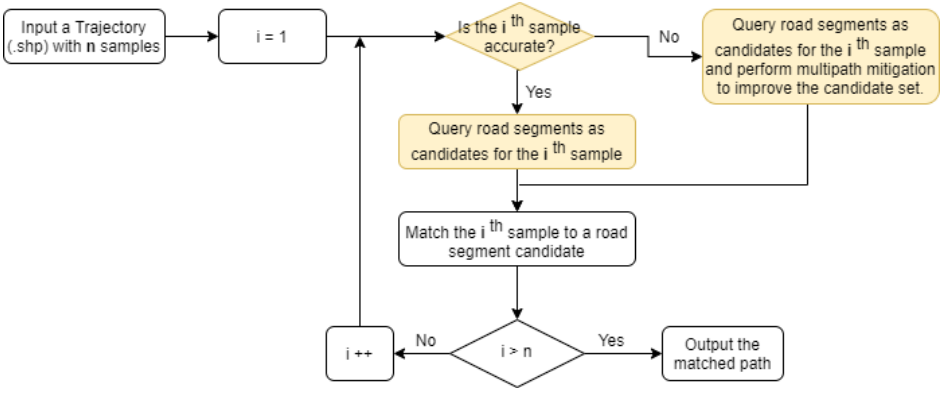


Figure 1.4: The workflow of map matching integrated with multipath mitigation. The steps in yellow boxes are belong to the part of candidate selection.

In our project, we aim to improve the performance of map matching by integrating multipath mitigation when selecting candidates. An enhanced procedure is shown in Figure 1.4. The differ-

ence between this workflow and the previous one is the incorporation with multipath mitigation. For every sample that is not accurate enough in the trajectory, an extra step is performed to mitigate the multipath effect.

For the common procedure of HMM map matching algorithms as shown in Figure 1.3, the details of candidate selection are described in Figure 1.5. Basically, it computes the distance from a road segment candidate to the sample. The smaller the distance is, the higher probability to be selected the candidate has.

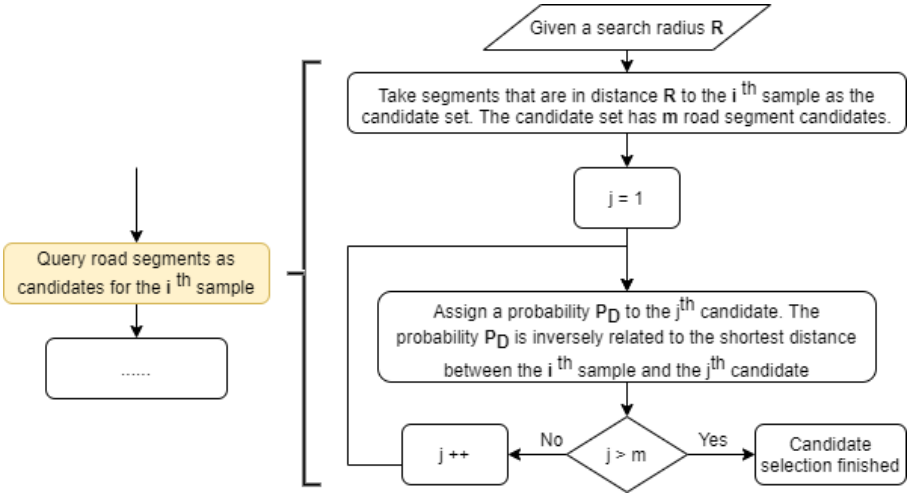


Figure 1.5: The details in candidate selection for common map matching algorithms.

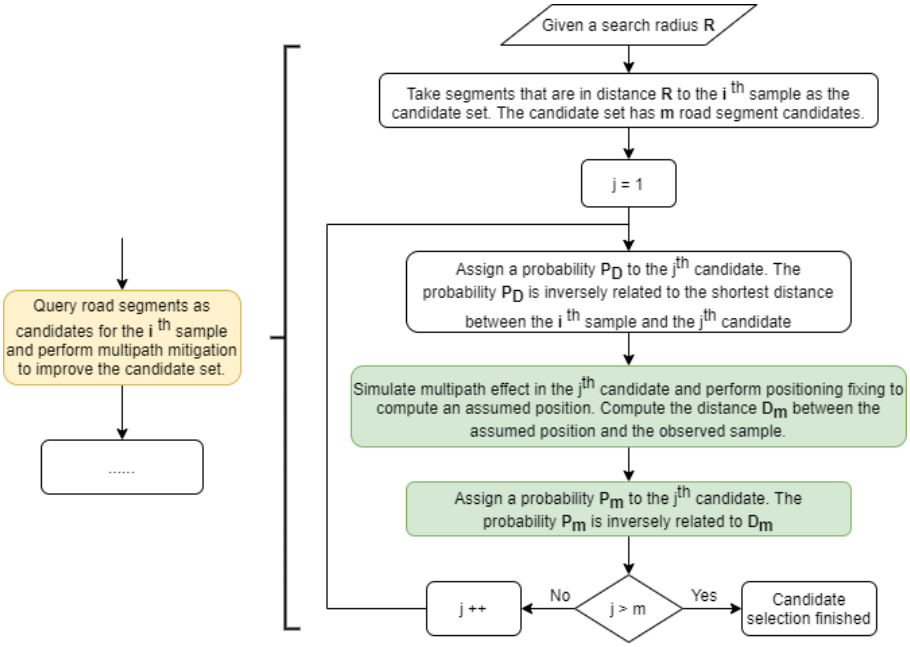


Figure 1.6: The details in candidate selection for map matching integrated with multipath mitigation. The steps in green boxed are extra procedure to mitigate multipath error.

In our project instead, the process of candidate selection becomes more complicated as Figure 1.6. After computing the shortest distance between the sample in trajectory and road segment candidates and P_D, we also need to mitigate multipath errors and provide a probability correction

to candidates. Then we compute the distance D_m between the assumed position and the observed position of the candidate. After obtaining D_m, it is taken as the input of a probability function and output probability P_m. The final probability P of this road segment candidate is obtained from P_D and P_m.

From the discussion above, we can list the data we need: (1) trajectory data; (2) road network of the city to search candidates; (3) satellite positions in order to simulate multipath error; (4) 3D city models to simulate blocked and reflected signals. However, 3D city models are not easy to obtain. An alternative way is to build up 3D building environment using a 2D building map and building heights as proposed in some literature [15, 24, 25, 27].

Chapter 2

Preliminaries

In this chapter, we will introduce some necessary background knowledge. Since we use multiple datasets in different format, we start by stating the notations used in later description in order to clarify their differences.

Definition 1 *(Trajectory) A trajectory T is a sequence of n ordered positions $T : p_0 \rightarrow p_1 \rightarrow ... \rightarrow p_{n-1}$ that records a travel path of a moving object. The element p_i is called a trajectory sample/point/observation and it consists of a coordinate (x_i, y_i, z_i), a timestamp t_i, and a list of visible satellites L_i. That is, $p_i = (x_i, y_i, z_i, t_i, L_i)$.*

Definition 2 *(Road Network) A road network is a undirected graph $G = (V, E)$ where vertices represent the intersection of roads or the end of road and edges represent the road starting from vertex v_i to v_j with $v_i, v_j \in E$. An edge e is called a road segment or a polyline, which goes through a sequence of positions.*

Definition 3 *(Building Map) A building map refers to a 2D map that only contains building location and the outline-shape of buildings. A building B containing n vertices is expressed by a closed chain $B : p_0 \rightarrow p_1 \rightarrow .. \rightarrow p_{n-1} \rightarrow p_0$. And $p_o, p_1, ..., p_{n-1}$ are in clockwise order.*

2.1 Geographic Coordinate System

Generally, a Geographic Coordinate System (GCS) is a coordinate system used to express a position on Earth. In our project, we use three primary GCSs:

1. Ellipsoidal Coordinate System. It defines the position of a point on the surface of an ellipsoid that approximately fits the globe by longitude, latitude and elevation [33].

2. Projected Map Coordinate System. It projects the surface of the Earth to a plane by different projection algorithms so it is a 2D coordinate system.

3. Earth-fixed Cartesian Coordinate System. A 3D coordinate system which takes the center of the ellipsoid as the origin.

The Ellipsoidal Coordinate System mainly refers to World Geodetic System 1984, or WGS84 in short. The coordinate origin of WGS84 falls at the mass centre of the Earth and it approximates the Earth into an oblate spheroid with the equatorial radius $\alpha = 6378.137$ km and the polar radius $\beta = 6356.75231424518$ km [16]. A position on the Earth can then be recognized by the longitude, latitude and ellipsoidal height which is the shortest distance measured from the position to the spheroid surface.

As for the Projected Map Coordinate System, we are using two map projection algorithms in the project. The most general coordinate system we used to render the data and check the

performance of the output on the plane is WGS 1984 Web Mercator. As it has great capability for multiple maps, it is used virtually by all online map providers, including Google Maps, OpenStreetMap and ESRI. By using this projected coordinate system in our project, we can take online maps as references to evaluate our outputs. Another projected coordinate system is British National Grid + OSGB 1936. Projection from 3D to 2D always leads to distortion and different map projections result in deformation in different locations. Thus, British National Grid + OSGB 1936 is used to map the UK accurately. It is a horizontal coordinate system consisting of a traditional geodetic datum using the Airy ellipsoid; a Terrestrial Reference Frames called OSGB 1936 (Ordnance Survey Great Britain 1936); a Transverse Mercator map projection allowing the use of easting and northing coordinates [33].

The Earth-fixed Coordinate System is essential when simulating interactions between a GPS sensor on the Earth and satellites. It is not reasonable to describe the locations of satellites and the GPS sensor by longitude, latitude and height in this case, because we cannot compute the distances between them directly from longitudes and latitudes. A regular Cartesian coordinate system using three perpendicular axes X, Y and Z can make this work. The X and Y axes lie in the equator of the ellipsoid and pass through 0 and 90 longitude respectively. The Z axis is perpendicular to the equator and points toward the North Pole. Also, the data unit in Earth-fixed Coordinate System is kilometers or meters instead of degrees.

Note that transformation between each pair of GCS we discussed above is possible. And this transformation can be performed by some geographic information systems (GISs) easily.

2.2 Elevation System

The elevation of a geographic position is its height above or below a reference surface. Depending on different datum, elevation can be classified into two catalogs: ellipsoidal height and orthometric height. The difference between ellipsoidal and orthometric height can be seen in Figure 2.1. The orthometric height, which is H_o in Figure 2.1, describes the height above or below the mean sea level, or say the geoid in common. The ellipsoidal height H_e is the height above or below an ellipsoid. The ellipsoidal model we consider in the project is the one with an equatorial radius $\alpha = 6378.137$ km and a polar radius $\beta = 6356.75231424518$ km in WGS84. Therefore, the relation between H_o and H_e is

$$H_e = H_o + N \tag{2.1}$$

where N is the geoid height, also called geoidal separation, which is the shortest distance measured from the geoid to the ellipsoid.

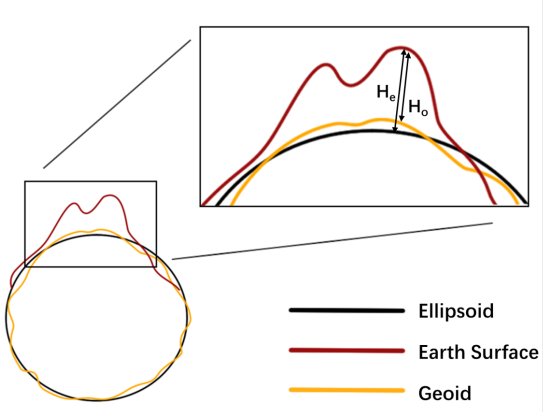


Figure 2.1: The difference between ellipsoidal height and orthometric height.

2.3 Geographic Information System and Arcpy

Geographic information systems (GISs) have become an indispensable tool in terms of urban planning and mapping practice during the past few decades [23]. The data processed by GIS is mostly spatial data which has connectivity to a geographic reference on Earth [34]. Later some GIS techniques analyse this spatial data, construct a geographic model and output some geographic products, like all kinds of maps. Thanks to the rapid development of GIS over the past two decades, multiple enterprise GIS platforms, which are complete software systems that can satisfy most geospatial needs, support our studies and do some basic spatial analysis for us.

One of the most famous and commonly used GIS is ArcGIS maintained by the Environmental Systems Research Institute (ESRI). ArcGIS has been in operation for 21 years and it includes functionalities of data management, geoprocessing, data conversion, spatial analysis, map preview or production. Another contribution of ESRI is the spatial data format shapefile (.shp). Shapefile is a geospatial vector data format for storing the geometric location and attribute information of geographic features. A geometric object in shapefile is called a geometry. According to the shape of the object, there are three types of geometries in definition: point, polyline and polygon. A position can be expressed as a point geometry. A bikeline is a polyline geometry which might contains an array of segments and a building is a polygon geometry. Points, polylines and polygons that are grouped into layers have an associated row in an attribute table which records geographic features, for example, the name of the geometry. Therefore, we can connect the definition of our dataset with their format: each trajectory as Definition 1 is stored as a layer of point geometries; road network in Definition 2 is considered as a set of polyline geometries and a polyline can only intersect with other polylines at its endpoints; the building map in Definition 3 is an array of polygon geometries.

Another GIS used in our study is QGIS. It is also known as a free and open-source GIS platform and it also supports the shapefile format. Compared with ArcGIS, QGIS usually takes less processing time with a better rendering capability. But ArcGIS performs better at spatial topological analysis and has more advanced spatial functionalities in mapping [1].

ArcPy is a Python site package that provides a useful and productive way to perform functionalities as ArcGIS does. All tools in the ArcGIS platform are encapsulated in Arcpy. Since we are using an early version of ArcGIS (10.2), the Arcpy library is consistent with Python 2.7 or lower version. Thus we are intended to use Python 2.7 for our project.

2.4 Global Navigation Satellite System and Positioning

Global Navigation Satellite System (GNSS) refers to a constellation of satellites providing accurate positioning services, such as GPS in America, GLONASS in Russia, Galileo in Europe, Beidou in China and so on. Among these, the Global Positioning System (GPS), consisting of 24 satellites, is the first constructed and most widely used system. Also, we collected trajectory data by a GPS sensor.

Positioning, or say position fixing, is the main purpose and the fundamental feature of a GNSS. As stated by [2] and [10], positioning techniques can be categorized into the following five catalogs: proximity, ranging, angular positioning, Doppler positioning and pattern matching. Before we go deep into each method, we first define the common notation needed: the receiver/sensor position is defined as the measured user position; in contrast, actual/real positions indicate the actual user position; landmark positions are the positions of transmitters, base stations, satellites or other spatial references.

Proximity positioning, as its name implies, is an approximate positioning method, which measures the receiver position by the average landmark position. It is natural to conclude that proximity positioning is more suited to short-range radio signals like indoor positioning by WiFi or Bluetooth

[1] Differences between ArcGIS and QGIS: https://planningtank.com/geographic-information-system/differences-qgis-arcgis

[10]. This method can hardly be extended to a large 3D space since it is too expensive to setup evenly distributed transmitters in such a space.

(a) The sensor receives three radio signals.

(b) The sensor receives one radio signals from landmark A. The receiver position is exactly at the position of Landmark A.

Figure 2.2: Proximity positioning. The triangle shows the actual user position and the star marks the measured receiver position. The dash lines in blue indicate the radio signals received by the receiver.

To fix a position by ranging, we have to measure the range which is actually the distance between the receiver and the spatial references. Take satellites as an example, the distance that signal travels, named pseudo range, is computed for each satellite that the receiver connects to. Each pseudo range for a satellite defines a spherical surface (a circular line of position in 2D, also written as LOP). Therefore, together with the positions of four satellites and their pseudo ranges, the receiver position can then be computed. Figure 2.3 shows the principle of position fixing using ranging.

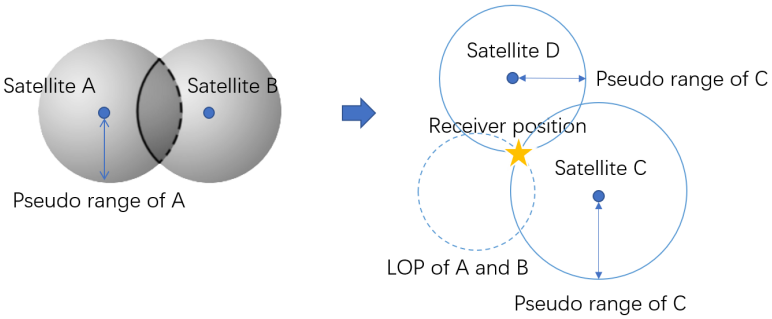


Figure 2.3: The principle of positioning fixing using ranging.

Angular positioning is in fact similar to ranging positioning. Instead of using the distance between the user and spatial references, this method measures the angle between the north direction and the line consisting of the user position as well as the landmark. Each angle of a landmark can determine a straight LOP and two of that can serve the purpose of positioning in 2D space (see Figure 2.4). This method can also be extended to 3D space by adding an elevation angle to a landmark.

Doppler positioning measures the Doppler shift to determine the distance between the user and the transmitter, which requires relative motion between them. The Doppler shift is the change in frequency of a wave in relation to an observer who is moving relative to the wave source [8]. After obtaining the distance between the user and the transmitter, Doppler positioning performs similar steps as ranging positioning to compute a location.

The last positioning method, pattern matching, maintains a database that stores features that

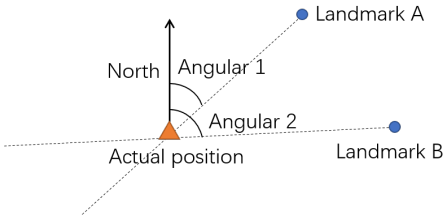


Figure 2.4: The principle of angular positioning.

vary with position, like geometric attributes and signal information [10]. Then the parameters measured from the user position are taken as the input of a matching algorithm to find a receiver position.

Ranging positioning is the most suitable technique for us to use. The reasons are as follows. Proximity positioning is less accurate and can hardly simulate a large space including Earth and satellites in GPS even though it is fast and simple. As for angular positioning, this method is accurate enough but it is easier to compute distance than angular since GPS emits ranging codes from which the distance can be obtained. And in terms of our purpose of multipath mitigation, distance is better to obtain and use in a 3D building model. Also, we do not have raw data to measure Doppler shift not to mention performing Doppler positioning. Finally, it is relatively more expensive to maintain a database containing features of positions spreading all over the city, so pattern matching positioning will not be a better choice.

Chapter 3

Data Preparation

This chapter will introduce the details of data preparation for our project, including by which way we collect the data and how the data is processed for later use. The data we concentrate on in this chapter is listed as follows:

1. Road Network (see Definition 2).
2. 3D building model. Since it is not easily accessed, we construct it by the following data:
   (a) Building map (see Definition 3). In order to detect multi-path error and find possible signal reflections for a position, 3D scene of the surroundings has to be built. Thus, 2D building map is needed.
   (b) Elevation data. As compensation for the building map, elevation data can decide the heights of buildings.
3. Position data for satellites. To simulate a scene of multi-path error, the positions for satellites should be known.
4. Trajectory data (see Definition 1). The input of the project, a set of positions with timestamps recorded by a GPS sensor.

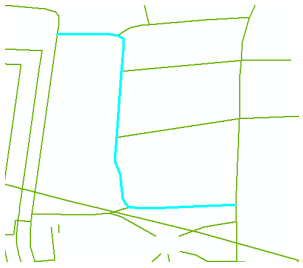
To create a complete accessible dataset for the project, we chose Newcastle upon Tyne, a city of the UK in this project.

3.1 Road Network and Building map

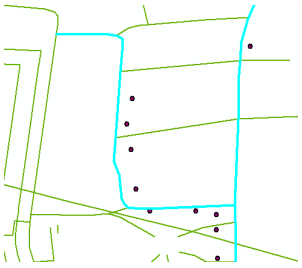
The road network and building map are extracted from OpenStreetMap (OSM), an open-source project including geodata of the world. There are two ways to make use of OSM data: online and offline. For the road network and low-rate-sampling trajectories in a large scale area, such as for the whole country of the UK, it might be a better choice to request partial data online instead of storing geodata locally since the number of segments and buildings may reach the magnitude of millions. However, each round of request and response will cost around 0.5 seconds, which slows down the whole process. In our experiments, trajectories were collected from the southeastern part of Newcastle upon Tyne and thus, it is reasonable to keep all data local. By querying OSM database, we downloaded geodata of Newcastle upon Tyne from OSM database via an API called Overpass[1].

The preliminary road network downloaded contains over 19,000 polylines representing overpasses, motorways, streets, and alleys for the whole city, except small footpaths. The size of the road network shrank to about 3,000 once we eliminated roads out of the area of interest. However, for the purpose of map matching, these roads are not standard enough. Figure 3.1a and 3.1b show

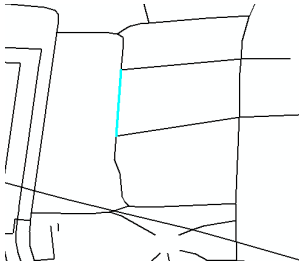
[1]The interface address: http://www.overpass-api.de/query_form.html



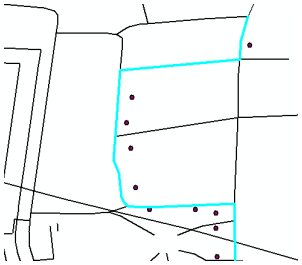
(a) The smallest unit (in blue) before planarizing polylines.



(b) The matched result (in blue) before planarizing polylines.



(c) The smallest unit (in blue) after planarizing polylines.



(d) The matched result (in blue) after planarizing polylines.

Figure 3.1: Planarize polylines in the road network.

what these roads look like and why they are not appropriate. As Figure 3.1a shows, the selected polyline (in blue) is inseparable since polylines are considered as the smallest unit and the matched result of a trajectory is a set of polylines. In such a condition, the matched route of a trajectory is shown in Figure 3.1b where a long polyline would be selected even if the trajectory goes no more than 1/4 of it. A trivial solution for this is to planarize polylines [2]. That is, we break the road segments into smaller ones by splitting them at intersection points with other roads. In this way, we shortened the smallest unit (as Figure 3.1c shows) and allowed flexible turns at intersections in results (see Figure 3.1d for example). After this step, the number of polylines in the road network increased from 3,000 to around 7,000 and the average length of road segments in the road network is 38.9 meters.

To obtain the building map, we downloaded the polygon layer of the city from the OSM server. There are not only buildings but also other facilities in this layer, for instance, squares, parks, district boundaries and so on. There is an attribute "building" associated with the polygon layer stating the characteristic of polygons. According to values of the "building" attribute (shown in Table 3.1), we extracted buildings from this layer [3]. The final dataset includes about 1,200 buildings.

	Values of "building"
is building	Yes, building, school, university, hospital, stadium, terrace, tower, church
is not building	No, parking, square, district

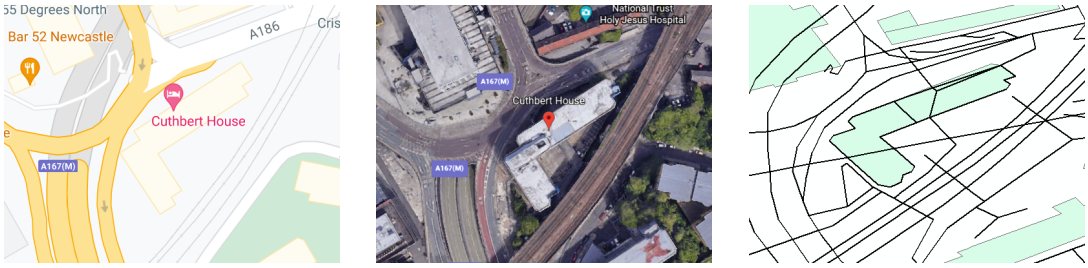
Table 3.1: Values of the attribute "building" in the polygon layer.

However, errors exist in both datasets of the road network and the building map. As a collab-

[2]More details for the tool in https://desktop.arcgis.com/en/arcmap/10.3/manage-data/editing-existing-features/splitting-lines-at-intersections-with-planarize.htm

[3]More details for the tool in https://desktop.arcgis.com/en/arcmap/10.3/map/working-with-layers/using-select-by-attributes.htm

orative project, OSM data is created by individual mappers and sometimes contains topological errors and data redundancy. Figure 3.2 shows some details of the errors in our datasets. Compared with the city layout on Google map (Figure 3.2a), there are some unnecessary segments in the road network. Furthermore, the topological relationship between buildings and roads in our datasets does not completely consistent with that on Google map. Accordingly, we manually edited the road network and the dataset of buildings to tackle the inconsistency.



(a) Google map. (b) Google Earth. (c) Our datasets.

Figure 3.2: Topological errors and data redundancy in road network and the dataset of buildings.

3.2 Elevation Data

To detect and simulate multipath error in a city area, a 3-dimensional representation of city view is required. The 2D layout of buildings we obtained in Chapter 3.1 cannot serve this purpose since the height of buildings is missing. Inspired by the work by Obst et al. [27], we tend to create 3D building models from the 2D building layout and height data of buildings.

The height data for buildings can be extracted from digital terrain models (DTM) and digital element models (DEM) according to [27]. These terms including DTM, DEM and also DSM (digital surface models) are defined as digital representations of elevation data but they differ from each other even if they are often considered to be similar. The term DTM represents the height of the solid base of the earth (as the blue line shows in Figure 3.3) with regards to its natural features, including land forms, terrain features, natural resources and environments [21]. DSM reflects the elevation of the earth surface (the yellow line in Figure 3.3) including all objects, both natural and manufactured, on top of it [36]. The term DEM, on the other hand, has multiple definitions. Some definitions make it equal to DTM [31] or DSM [27] and others use it as a generic term to describe both DTM and DSM [14]. Therefore, we can extract height data from DSM and DTM. That is, the building height

$$h_{building} = h_{DSM} - h_{DTM}. \tag{3.1}$$

We download DTM and DSM for the purpose of computing the height of buildings as the UK government published high-resolution DTM[4] and DSM[5] of Newcastle upon Tyne to the public. Both two are in a resolution of 2 meters. The raw data is originally in *.asc* format and following a tutorial[6] *.asc* files can be transformed into TIFF images. Note that raw data is in a projected coordinate system of British National Grid + OSGB 1936 and these images have to be transformed to images in WGS 84 since our road network, building map and trajectory data are in WGS 84.

[4]The website for DTM data: https://data.gov.uk/dataset/002d24f0-0056-4176-b55e-171ba7f0e0d5/lidar-composite-dtm-2017-2m

[5]The website for DSM data: https://data.gov.uk/dataset/fba12e80-519f-4be2-806f-41be9e26ab96/lidar-composite-dsm-2017-2m.

[6]Tutorial: How to download the LIDAR datasets from the UK Environment Agency website: https://www.roger-pearse.com/weblog/2019/07/08/tutorial-how-to-download-the-lidar-datasets-from-the-uk-environment-agency-website/

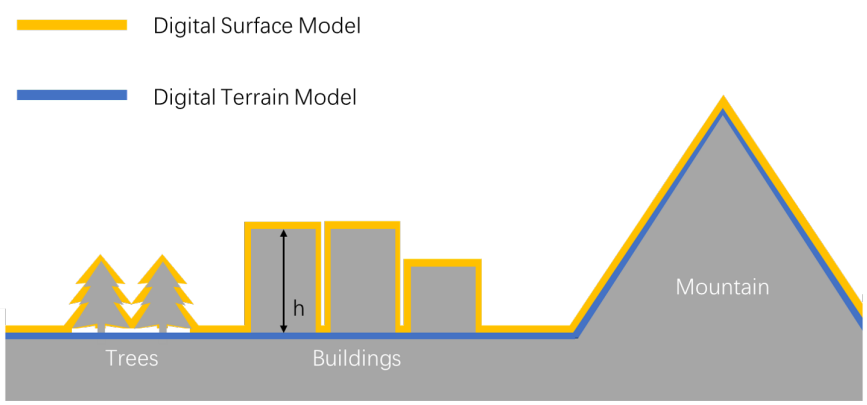


Figure 3.3: The way to extract height data from DTM and DSM.

Figure 3.4 shows parts of DSM and DTM and Table 3.2 lists detail information about DSM and DTM.

(a) DSM (b) DTM

Figure 3.4: DSM and DTM

Property	Value
Image size	5000 (columns), 5056 (rows)
Cell size	3.1421079×10^{-5} , 1.7871655×10^{-5}
Top	55.024138607
Left	-1.689395002
Right	-1.532289605
Bottom	54.933779517

Table 3.2: Detail information of TIFF images for DTM and DSM.

After obtaining DSM and DTM of Newcastle upon Tyne, the next step is to extract elevation data from these two images for a given position. A trivial idea to achieve this goal is that we first transform the position from longitude and latitude to numbers of columns and rows based on the information in Table 3.2. Then we can read the float value of a cell positioning by the new coordinates and this float value is the elevation of the given position.

The elevation data collected via or related to satellites is typically measured above the ellipsoidal model of the earth, whereas DTM and DSM are using an orthometric model. The ellipsoidal height of the given position can also be known once the values (orthometric height) of cells in TIFF images are read. The difference and transformation between ellipsoidal height and ortho-

metric height can be seen in Chapter 3.2. Following the instruction by ArcGIS[7] and the principle in Chapter 3.2, the geoidal separation between these two can be approximated by a constant (+50 meters). Put it simply, after we read orthometric height H_o from DTM/DSM, we store $H_e = H_o + 50$ for later use.

3.3 Satellite Data

Accurate positions of satellites are essential in simulating the formation of multipath errors. The International GNSS Service provides many satellite orbit products that can support precise measurement of satellite positions. These products come in multiple flavors, including Rapid (IGR), Ultra-rapid (IGU) and Final (IGS) products. Table 3.3[8] summarizes the differences among IGR, IGU and IGS in terms of orbit accuracy[9] and satellite clock accuracy (clock accuracy computed by standard deviation (SDEV)), latency and sampling frequency. We can see that IGU and IGS have higher quality but they take longer time to become available compared to IGR.

	IGR	IGU	IGS
Orbit Accuracy	$\leq 5cm$	$\leq 2.5cm$	$\leq 2.5cm$
Clock Accuracy (SDEV)	$\leq 1.5ns$	$\leq 25ps$	$\leq 20ps$
Latency	Realtime or 3-9 hours	17-41 hours	12-18 days
Sample Interval	15 min	15 min	15 min

Table 3.3: Differences among IGR, IGU and IGS products.

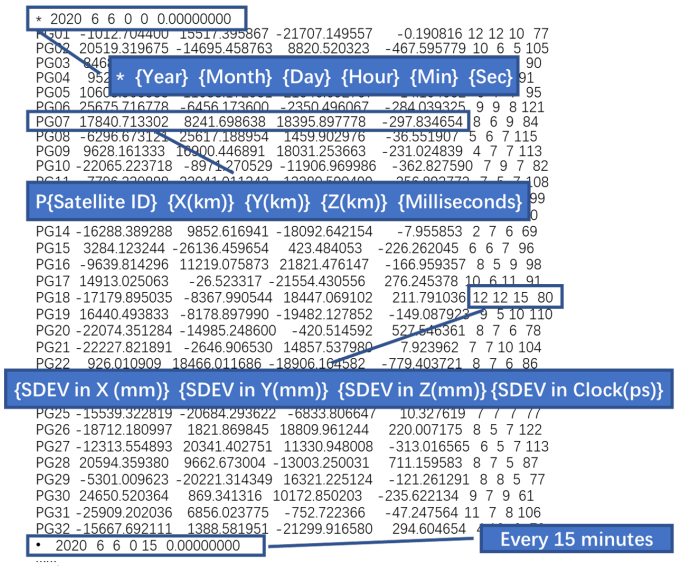


Figure 3.5: The structure of an IGS fragment.

We take advantage of the IGS product for accuracy instead of low latency. Each IGS file includes orbit data for one day, named by GPS time. GPS time is used by satellites for their own timescale, including two factors: the week number and the elapsed number of days within that week. The week number is counted from time zero which is UTC 0 : 00 on January 6^{th}, 1980.

[7]Converting from orthometric to ellipsoidal heights: https://pro.arcgis.com/en/pro-app/latest/help/data/imagery/converting-from-orthometric-to-ellipsoidal-heights-pro.htm

[8]Data collected from https://www.igs.org/products/.

[9]Orbit accuracies are 1D mean root-mean-square values over the three XYZ geocentric components.

Note that trajectories collected before March 28^{th} are in standard UTC time, but the time stamps for trajectories collected after that day are in UTC+1 time since it switches to summer time.

The IGS files store positions of GPS satellites per 15 min. Figure 3.5 explains the structure of an IGS fragment. Given positions for satellites per 15 min, we can predict satellite positions at any time using Lagrangian interpolation. The problem can be simplified as follows:

Definition 4 *(Lagrangian interpolation for satellite positions) Given four sets of $k+1$ values t_0, t_1,..., t_k; x_0, x_1,..., x_k; y_0, y_1,..., y_t; and z_0, z_1,..., z_k. The interpolation polynomials in the Lagrange form are linear combinations*

$$\begin{aligned} L_x(t) &:= \sum_{j=0}^{k} x_j l_j^x(t) \\ L_y(t) &:= \sum_{j=0}^{k} y_j l_j^y(t) \\ L_z(t) &:= \sum_{j=0}^{k} z_j l_j^z(t) \end{aligned} \tag{3.2}$$

of Lagrange basis polynomial

$$l_j^p(t) := \prod_{\substack{0 \le m \le k, m \ne j \\ p \in \{x,y,z\}}} \frac{t - t_m}{t_j - t_m} \tag{3.3}$$

where $0 \le j \le k$. The satellite positions at anytime can be computed by plugging the time stamp into Equations 3.2. Note that all coordinates are in Earth-fixed Coordinate System.

To decide the value of k in Definition 4, we tested for values from 3 to 9, trying to find a balance between accuracy and time cost. However, the time spent on Lagrangian interpolation is negligible so that $k = 9$ is chosen with an accuracy of $5 \times 10^{-5} km$.

3.4 Trajectory Data

Initially, we planned to choose an existing dataset of trajectories in Beijing[10]. Those trajectories are collected from 2006 to 2012 by sorts of transportation such as by cars, buses, subways, trains and on foot. However, many problems arose when we tried to access other data required. The first problem is the road network of Beijing. What we found on OSM and other online maps are always collected in recent years and not consistent with the trajectories. The situation happens that we cannot match a trajectory to any road according to our road network since the road network changed significantly during these years (see Figure 3.6 for examples).

In addition to the road network, the lack of elevation data made the situation even worst. We could not get free access to high-resolution DTM and DSM, not to mention related data in 2008. Also, the Beijing dataset is preprocessed data from which we cannot tell which satellites were visible at that time. Additionally, without knowing the actual routes users passed, it is difficult to evaluate the results of our project. For all the reasons above, we decided to make the trajectory data ourselves.

We used a Leica GS18 T GNSS RTK Rover for positioning and making trajectories. The Leica GS18 T has two modes when collecting data: (1) discrete mode where the device does a location measurement only when the user presses the measure button; (2) continuous mode where the machine records raw GPS data and this data can be processed into positioning data by the Leica Captivate field software afterwards. In the discrete mode, the position and timestamp are given once the button is pressed, whereas in the continuous mode, it records pseudo ranges of satellites

[10]Data from https://www.microsoft.com/en-us/download/details.aspx?id

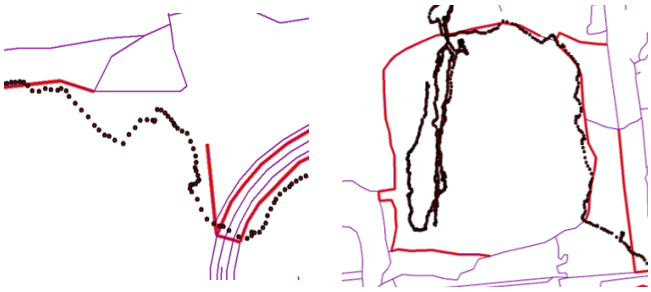


Figure 3.6: Inconsistency between trajectories (consisting of black dots) and the road network in Beijing. Road segments marked in red shows the map matching results for these trajectories.

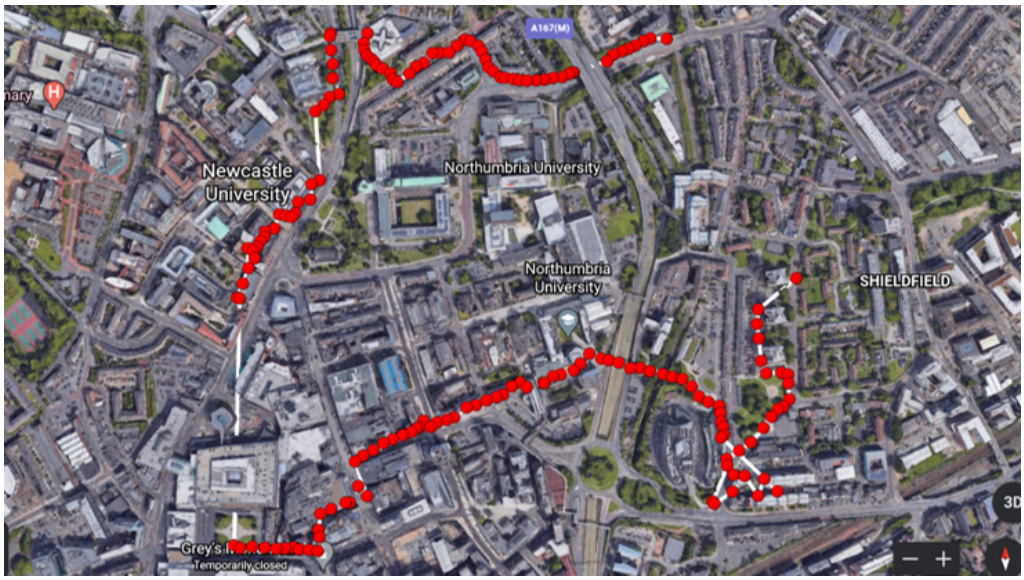
per second, which means it distinguishes different satellite signals. As we could not find a way to customize the parameters we preferred, we decided to run both two modes in parallel. That is, we position locations manually but in the back end, it records detailed signal data per second. Finally, we collected 7 trajectories consisting of 800 points in total. Figure 3.7 shows the machine we used and how we collected data.

(a) Leica GS18 T GNSS RTK Rover.

(b) Use Leica GS18 T to collect position data.

Figure 3.7: Trajectory data collection.

The output includes two parts. The output of the discrete mode is two files including locations (longitude, latitude and height) and timestamps separately. The output of the continuous mode is *.o* and *.n* files. From *.o* files we can extract available satellites for each point in the trajectories according to the timestamps. We integrated these files into a shapefile (shp) product which stores the position, ellipsoidal height, timestamp, available satellites and total signal number for each point in the trajectory as shown in Figure 3.8.



(a) Points in the trajectory.

	FID	Name	Ellp_Hgt	Signal_Num	Time	Avail_Stl
	0	GPS0016	106.612	5	20210325155710	G05, G18, G20, G26, G29
	1	GPS0017	90.743	6	20210325155932	G16, G18, G26, G27, G29, G31
	2	GPS0018	96.463	7	20210325155953	G16, G18, G20, G26, G27, G29, G31
	3	GPS0019	92.334	6	20210325160014	G16, G18, G20, G26, G27, G31
	4	GPS0020	92.312	6	20210325160015	G16, G18, G20, G26, G27, G31
	5	GPS0021	94.635	5	20210325160147	G16, G18, G26, G27, G31
	6	GPS0022	90.826	8	20210325160259	G16, G18, G20, G23, G26, G27, G29, G31
	7	GPS0023	91.67	7	20210325160319	G16, G18, G20, G23, G26, G27, G29
	8	GPS0024	91.457	8	20210325160336	G05, G16, G18, G20, G23, G26, G27, G29
	9	GPS0025	90.474	8	20210325160351	G05, G16, G18, G20, G26, G27, G29, G31
	10	GPS0026	92.336	9	20210325160417	G05, G16, G18, G20, G23, G26, G27, G29, G31
	11	GPS0027	97.239	9	20210325160430	G05, G16, G18, G20, G23, G26, G27, G29, G31
	12	GPS0028	92.43	8	20210325160444	G05, G16, G18, G20, G26, G27, G29, G31
	13	GPS0029	91.673	9	20210325160502	G05, G16, G18, G20, G23, G26, G27, G29, G31
	14	GPS0030	94.012	9	20210325160523	G05, G16, G18, G20, G23, G26, G27, G29, G31
	15	GPS0031	91.663	9	20210325160548	G05, G16, G18, G20, G23, G26, G27, G29, G31
	16	GPS0032	92.862	6	20210325160608	G16, G18, G26, G27, G29, G31
	17	GPS0033	92.273	8	20210325160626	G05, G16, G18, G20, G26, G27, G29, G31
	18	GPS0034	80.776	7	20210325160650	G05, G16, G18, G20, G23, G26, G31
	19	GPS0035	116.126	7	20210325160712	G05, G16, G18, G23, G26, G27, G29
	20	GPS0036	121.695	5	20210325160728	G05, G16, G20, G23, G26
	21	GPS0037	117.302	5	20210325160740	G16, G20, G26, G27, G29
	22	GPS0038	94.458	6	20210325160800	G16, G20, G26, G27, G29, G31
	23	GPS0039	120.88	6	20210325160836	G05, G16, G18, G26, G27, G29
	24	GPS0040	98.726	8	20210325160855	G05, G16, G18, G20, G26, G27, G29, G31
	25	GPS0041	108.093	7	20210325160911	G05, G16, G18, G20, G26, G27, G29
	26	GPS0042	112.657	7	20210325160934	G16, G18, G20, G23, G26, G27, G29
	27	GPS0043	72.853	8	20210325161247	G05, G16, G18, G20, G23, G26, G29, G31
	28	GPS0044	84.218	7	20210325161302	G05, G16, G18, G20, G23, G26, G29
	29	GPS0045	83.6	7	20210325161315	G05, G16, G18, G20, G23, G26, G29
	30	GPS0046	82.27	6	20210325161330	G16, G18, G20, G23, G26, G29
	31	GPS0047	86.104	7	20210325161344	G05, G16, G18, G20, G23, G26, G29
	32	GPS0048	86.361	7	20210325161356	G05, G16, G18, G20, G23, G26, G29
	33	GPS0049	87.178	7	20210325161416	G16, G18, G20, G23, G26, G27, G29

(b) Attributes for points in the trajectory.

Figure 3.8: The shapefile product for trajectories.

Chapter 4

Multipath Mitigation

In this chapter, we are going to describe the technique we used to mitigate the multipath effect, especially NLOS errors. We will first introduce the procedure of the mitigation in the first section and look into its detailed steps in the following sections.

4.1 Pipeline for Multipath Mitigation

In a traditional HMM map matching algorithm (see Figure 1.5), a road segment candidate set is constructed for each observation in the trajectory and a probability is assigned to each road segment candidate indicating its likelihood that it is indeed part of the real path. Commonly, this likelihood is inversely related to the shortest distance between the segment and the sample. That is, the nearest road segment gets the highest probability value. However, the drift in the observation caused by multipath errors may result in a poor probability distribution among all segment candidates. Therefore, we aim at mitigating possible multipath effects for each trajectory sample and reveal a more reliable probability distribution among these candidates.

Before we describe the pipeline, here are some notations to clarify first. The distance function $dist(p_i, p_j)$ measures the Euclidean distance between positions p_i and p_j. And the ranging positioning function $\mathcal{RP}(\gamma_1, \gamma_2, \gamma_3, \gamma_4, ...)$ takes at least four pseudo ranges as input and outputs a position. We use three adjectives to describe a pseudo range:

- direct pseudo ranges, which refer to the shortest distance between the satellite and the receiver position.
- measured pseudo ranges, which are the distances that the receiver actually measures. These distances might be affected by NLOS errors, interference errors and other measurement errors.
- hypothesis pseudo ranges, which are measured by geometric models and include NLOS errors.

Also, we define ε_i to be the multipath error for road segment candidate e_i. That is, ε_i is a non-negative number computed by the difference between hypothesis and direct pseudo ranges. Note that $\varepsilon_i = 0$ if there is no multipath exists. Besides, recall that an observation in the trajectory is defined to be $p = (x, y, z, t, L)$ where (x, y, z) is the coordinate of the observation, t is the timestamp and L is a list of visible satellites.

The idea of the multipath mitigation method in our project is first to assume that a position p^e is the actual user position, and then simulate the process of position fixing influenced by NLOS errors to see if the computed hypothesis position p_{hypo} is consistent with the observation. The position p^e has a sufficiently large probability to be the actual user position if p_{hypo} is sufficiently close to the observation. Generally, the method has four steps:

1. construct 3D building model;

2. test intersections between signals and buildings to find obstructed signals (Chapter 4.2 Intersection Test);

3. find reflected path to compute NLOS errors if obstructed signals exist (Chapter 4.3 Multipath Error Simulation);

4. perform ranging positioning and compare the computed position to the observation (Chapter 4.4 Ranging Positioning).

Algorithm 1: Pipeline for Multipath Mitigation for an observation.

input: An observation $p = (x, y, z, t, L)$ where $L = \{s_1, s_2, ..., s_m\}$;
A road segment candidate set $E = \{e_1, e_2, ..., e_n\}$;
A search radius r.
output: A set of positioning drift caused by multipath errors for each road segment candidate $E_r = \{d_1, d_2, ..., d_n\}$.

1 **for** *satellite* $s_i \in L$ **do**
2 | Query the position p_i^s of the satellite s_i by timestamp t using Lagrangian Interpolation;
3 **end**
4 **for** $e_i \in E$ **do**
5 | Extract a point candidate p_i^e from the road segment candidate e_i;
6 | Query the building map by p_i^e and a search radius r. $\mathcal{B} \leftarrow \{B_1, B_2, ...\}$;
7 | **for** $B_j \in \mathcal{B}$ **do**
8 | | Query the elevation h_j of B_j from DSM and DTM;
9 | **end**
10 | **for** $s_j \in L$ **do**
11 | | $f \leftarrow -1$;
12 | | **for** $B_z \in \mathcal{B}$ **do**
13 | | | Test intersection for the signal from p_j^s to p_i^e and B_z ; `// see Algorithm 2`
14 | | | **if** *intersection exists* **then**
15 | | | | $f \leftarrow z$
16 | | | **end**
17 | | **end**
18 | | **if** f = *-1* **then**
19 | | | No multipath effects;
20 | | | $\varepsilon_j \leftarrow 0$;
21 | | | $\gamma_j \leftarrow dist(p_j^e, p_i^s)$;
22 | | **else**
23 | | | **for** $B_z \in \mathcal{B} \setminus B_f$ **do**
24 | | | | Simulate the signal path with multipath errors for B_Z;
25 | | | | Compute the multipath error ε_{jz} ; `// see Algorithm 3`
26 | | | **end**
27 | | | $\varepsilon_j \leftarrow \min_{z=0}^{m}(\varepsilon_{jz})$;
28 | | | Compute the hypothesis pseudo ranges $\gamma_j \leftarrow dist(p_j^e, p_i^s) + \varepsilon_j$
29 | | **end**
30 | **end**
31 | Compute the hypothesis GPS position $p_{hypo} \leftarrow \mathcal{RP}(\gamma_1, \gamma_2, ..., \gamma_m)$ using hypothesis pseudo ranges ; `// see Algorithm 4`
32 | Compute the positioning drift caused by the multipath error $d_i \leftarrow dist(p_{hypo}, p)$
33 **end**

Algorithm 1 describes the detailed method of multipath mitigation for a candidate set containing n road segments $E = \{e_1, e_2, ..., e_n\}$ of an observation $p = (x, y, z, t, L)$ where $L = \{s_1, s_2, ..., s_m\}$. In the first loop (line 1 to line 3), we compute the positions of visible satellites by Lagrangian Interpolation (methodology in Definition 4).

Most of the heavy work is done in the second loop. Theoretically, for a road segment candidate, we should extract some possible positions along the road segment to approximate the actual user position and measure multipath errors. But in practice (as line 5 shows), we choose only one position on the road segment that is closest to the observation in order to cut down the computation cost. This is feasible since the road segments are not very long (with an average length of 38.9 meters). After that, simulation for multipath is performed on this position candidate instead of the road segment candidate. In other words, we assume that we are receiving GPS signals from this point candidate. From line 5 to 9 of Algorithm 1, data of the surrounding environment is collected from the building map (details in Chapter 3.1), DTM and DSM (details in Chapter 3.2).

Later we perform an intersection test for each direct signal of visible satellites and the surrounded environment model from line 11 to 17 (Chapter 4.2 illustrates the intersection test step by step). If the direct signal is not obstructed by any building, we consider that NLOS errors do not exist and the hypothesis pseudo range is equal to the direct pseudo range as shown in line 18 to 21. Conversely, we simulate multipath and compute NLOS errors in hypothesis pseudo ranges as shown in line 22 to 29 (detailed solution for multipath simulation is shown in Chapter 4.3). Sometimes, the situation happens that there are multiple paths for a NLOS signal to travel and reach the point candidate. We take the shortest path to minimize the NLOS error.

After the correction of the direct signals by adding up NLOS errors, we then use ranging positioning (method described in Chapter 4.4) to compute a hypothesis position p_{hypo}. We realize that the calculated hypothesis position p_{hypo} must appear around the observation p if those pseudo ranges are affected by the multipath in the same way as the measured pseudo ranges are. So we store the distance between p_{hypo} and the observation p as an indicator of the likelihood value for this point candidate and also the road segment candidate.

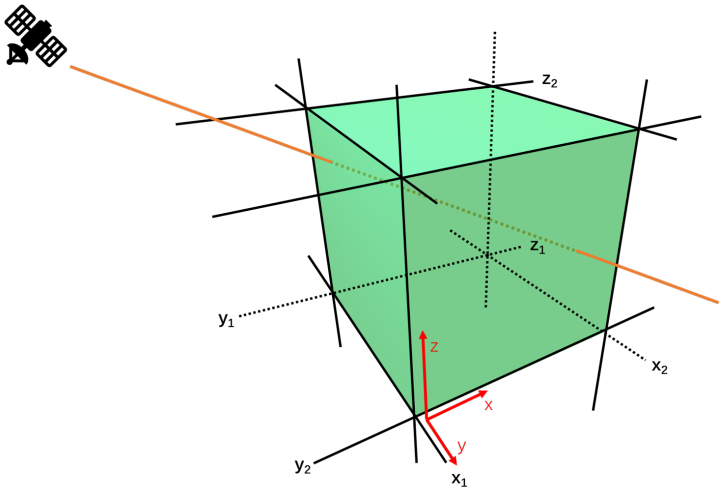
4.2 Intersection Test

The intersection test, or say ray-object intersection test is a technique in ray tracing and it is typically where most of the running time is spent [1]. Trivially, to test if a ray intersects a cube, we need to solve if the ray intersects at least one plane of the cube, which requires at most six times of ray-plane intersection tests. As for an environment containing buildings in different shapes, simply performing an exhaustive search consumes enormous time and effort. The principle for solutions to this is to cull objects which are clearly not in the path of the ray. According to [35], there are two strategies for ray-object intersection culling:

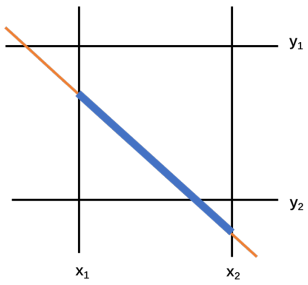
- Hierarchical bounding volumes. These bounding volumes with simple shapes, such as cuboid and sphere, envelop complicated objects that contain multiple planes and take a long time to test intersection. Before testing intersection between a ray and an object, these simpler bounding volumes are tested first to exclude some objects that are far away from the path of the ray. For a complex environment with many complicated objects, a tree for bounding volume is constructed and a volume placed around the entire scene is the root of the tree.

- Space partitioning. This strategy partitions the whole space into regions or voxels. We only need to check those objects in voxels that the ray goes through.

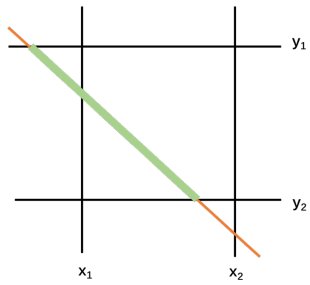
In our case to test intersection between satellite signals and surrounded buildings, neither hierarchical bounding volumes nor space partitioning works, because for less complicated scenes like this, it is not worth it to construct a hierarchical tree or to build up a spacial index for voxels. Therefore, we only adapt the simple bounding volumes to cull unnecessary intersection tests. It is worth mentioning that we only care about if the ray intersects with the bounding volume instead of where the intersection is [35]. Figure 4.1 shows the principle of the intersection test for ray and bounding volumes. Simply speaking, we project the signal to x, y, z directions and compute the interval in each direction.



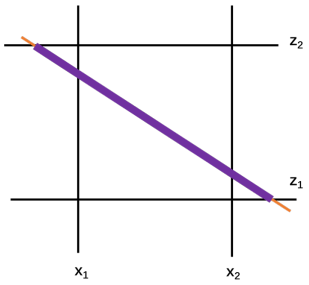
(a) The ray and the bounding volume.



(b) The interval between x_1 and x_2.



(c) The interval between y_1 and y_2.



(d) The interval between z_1 and z_2.

(e) Three intervals overlap with each other so the ray intersects with the bounding volume. The overlapping part is the part on the ray that goes inside the bounding volume.

Figure 4.1: The principle of the intersection test for the ray and the bounding volume. Figure 4.2b and 4.1c are captured along the $-z$ direction while Figure 4.1d is captured along $-y$ direction.

If these three intervals overlap with each other as Figure 4.1e shows, the ray intersects with the bounding volume and further computation is required. Otherwise, the ray does not have intersections with the bounding volume, not to mention the complicated object inside. In this way, we can easily cull down the computation cost. Algorithm 2 shows the details of the intersection test for a building.

Algorithm 2: Intersection test for a ray and an object.

input: A bounding volume $V = (x_1, x_2, y_1, y_2, z_1, z_2)$;
A ray l goes from satellite p^s to a point candidate p^e.
A building B and its height.
output: True if l intersects B; False otherwise.

1 Initialize D to be the direction of the ray, $D \leftarrow p^e - p^s$;
2 **if** *D is parallel to x, y or z axis* **then**
3 check if l intersects with B directly;
4 return True if it intersects and False otherwise;
5 **else**
6 $\{tx_1, tx_2\} \leftarrow \{\frac{x1-p^e.x}{D.x}, \frac{x2-p^e.x}{D.x}\}$, where $tx_1 < tx_2$; `// Compute the interval between` x_1, x_2
7 $\{ty_1, ty_2\} \leftarrow \{\frac{y1-p^e.y}{D.y}, \frac{y2-p^e.y}{D.y}\}$, where $ty_1 < ty_2$; `// Compute the interval between` y_1, y_2
8 $\{tz_1, tz_2\} \leftarrow \{\frac{z1-p^e.z}{D.z}, \frac{z2-p^e.z}{D.z}\}$, where $tz_1 < tz_2$; `// Compute the interval between` z_1, z_2
9 $t_{start} \leftarrow \min(tx_1, ty_1, tz_1)$, $t_{end} \leftarrow \max(tx_2, ty_2, tz_2)$;
10 **if** $t_{start} < t_{end}$ **then**
11 Compute the intersection between the ray and each plane of the building;
12 Return False if no such plane exists and True otherwise;
13 **else**
14 Return False;
15 **end**
16 **end**

4.3 Multipath Error Simulation

If the direct pseudo range is proved to be obstructed by buildings in the intersection test, a multipath error simulation is performed subsequently. Briefly, the simulation examines a hypothesis pseudo range and outputs NLOS error if it really exists. Algorithm 3 describes the basic principle in the multipath error simulation. Algorithm 3 finds the NLOS error for a point candidate p^e and one building. Note that we only consider a single reflection for each possible NLOS signal. Furthermore, the multipath error simulation phase for the point candidate p^e, or say the road segment candidate e, is only finished when Algorithm 3 computes NLOS errors for all the buildings $B \in \mathcal{B}$.

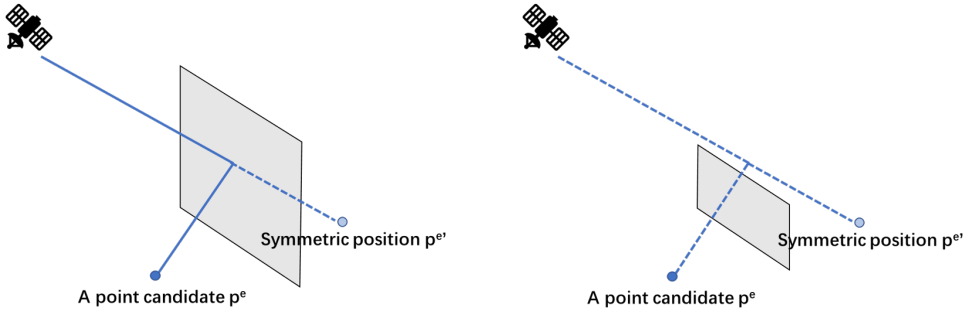
In the first loop of Algorithm 3 from line 2 to 17, we try to construct a multipath for each plane of the building (except for the top and the bottom) and prove its existence. The very first step is to examine the visibility of the plane and remove hidden surfaces (line 2). A multipath will not exist for a plane if the plane can not be seen from the receiver position. A simple and quick way to remove hidden surfaces is to compute the angle between the norm of the plane (pointing outside of the building) and our line-of-sight. If the angle is smaller than 180°, our line-of-sight cannot reach it. However, this hidden-surface test can only give out false negative results, which cannot filter out all invisible planes. Thus, further steps are needed to compute the actual intersection between our line-of-sight and the plane to prove the visibility of the plane. The basic principle of line 4 to 15 can be seen in Figure 4.2 [24]. We assume that all the buildings are constructed using perfectly reflective materials and the generation of a multipath obeys the laws of reflection. Therefore, the reflected path will only exist when the point of reflection is inside the surface (line 6) and the reflected path is not blocked by some other structures (line 7 to 10).

In the second loop of Algorithm 3 from line 18 to 22, we selected the smallest positive value as the NLOS error for the building. If all the values are negative, then there is no multipath exist for this building.

Algorithm 3: Basic principle of the multipath error simulation.

input: A point candidate p^e and a satellite position p^s
A building B with n vertices and its height.
output: NLOS error ϵ (in kilometers) if a hypothesis seudo range exists; 0 otherwise.

Build a 3D model $\mathcal{M}$ of the building using 2D information B and its height;
for *each vertical plane* $F_i \in \mathcal{M}$ **do**
  if F_i *is visible from* p^e **then**
    Compute a symmetric position $p^{e\prime}$ of the point candidate p^e to the plane F_i;
    Connect the symmetric position $p^{e\prime}$ and the satellite position p^s, named ray l';
    if l' *intersects with* F_i **then**
      Perform intersection test between l' and other surroundings;
      if l' *isn't blocked by other buildings* **then**
        $\epsilon^i \leftarrow dist(p^{e\prime}, p^s) - dist(p^e, p^s)$;
      else
        No multipath exists for this plane, $\epsilon^i \leftarrow -1$;
      end
    else
      No multipath exists for this plane, $\epsilon^i \leftarrow -1$;
    end
  end
end
if $max_{i=0}^{n-1} \epsilon_i == -1$ **then**
  No multipath exists for this building, $\epsilon \leftarrow 0$;
else
  $\epsilon \leftarrow min(\{\epsilon^i | \epsilon^i \neq -1, 0 \leq i \leq n-1\})$;
end
Return ϵ;



(a) A multipath exists for this surface. (b) A multipath does not exist for this surface.

Figure 4.2: The principle of constructing a multipath.

4.4 Ranging Positioning

In line 31 of Algorithm 1, a hypothesis GPS position is computed by ranging positioning which takes at least four pseduo ranges as parameters. We adapt the weighted least square solver to achieve this goal. In the rest of this section, we are going to introduce weighted least square solver in detail based on [17, 24].

Given a satellite position $p_i^s = (x_i^s, y_i^s, z_i^s)$ and the actual receiver position $p^u = (x_u, y_u, z_u)$, a measured pseudo range R_i is theoretically written as

$$R_i = \sqrt{(x_i^s - x_u)^2 + (y_i^s + y_u)^2 + (z_i^s - z_u)^2} + c \times dt_i + I_i + T_i + \epsilon \tag{4.1}$$

where c is the speed of light and dt is the clock offset between signal emission and reception; I and T denotes the ionospheric delay distance and the tropospheric delay distance respectively; ϵ refers to the errors produced by multipath, noise and so on. However in our case, a hypothesis pseudo range γ_i is measured directly by geometric models and includes NLOS errors such that it can be written as

$$\gamma_i = \sqrt{(x_i^s - x_u)^2 + (y_i^s - y_u)^2 + (z_i^s - z_u)^2} + \epsilon_{mul} + \epsilon_{other} \tag{4.2}$$

where we don't need to consider errors caused by clock offset and atmosphere. Suppose the hypothesis position we want to compute is defined as $p_{hypo} = (\hat{x}, \hat{y}, \hat{z})$, then Equation 4.2 can be linearized by expanding Taylor's series around p_{hypo} and neglecting the higher terms [17]. Defining $\hat{\gamma}_i$ as γ_i at p_{hypo} gives:

$$\Delta\gamma_i = \gamma_i - \hat{\gamma}_i = b_i^0 \Delta x_u + b_i^1 \Delta y_u + b_i^2 \Delta z_u + \epsilon_{other} \tag{4.3}$$

where

$$\begin{aligned} \rho_i &= \sqrt{(x_i^s - \hat{x})^2 + (y_i^s - \hat{y})^2 + (z_i^s - \hat{z})^2} \\ b_i^0 &= \frac{\hat{x} - x_i^s}{\rho_i},\ b_i^1 = \frac{\hat{y} - y_i^s}{\rho_i},\ b_i^2 = \frac{\hat{z} - z_i^s}{\rho_i} \end{aligned} \tag{4.4}$$

For positioning with n available satellites, Equation 4.7 can be rewritten into matrix format:

$$L = AX + V \tag{4.5}$$

where

$$A = \begin{bmatrix} b_0^0 & b_0^1 & b_0^2 \\ b_1^0 & b_1^1 & b_1^2 \\ \vdots & \vdots & \vdots \\ b_{n-1}^0 & b_{n-1}^1 & b_{n-1}^2 \end{bmatrix}, \quad X = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}, \quad L = \begin{bmatrix} \Delta\gamma_0 \\ \Delta\gamma_1 \\ \vdots \\ \Delta\gamma_{n-1} \end{bmatrix} \tag{4.6}$$

and V is the residual matrix. The weighted least square solver minimizes the residual by minimizing $V^T V$, which equals to

$$\frac{\delta V^T V}{\delta X} = 0 \tag{4.7}$$

By solving Equation 4.7, we can get

$$V^T A = 0 \ \Rightarrow \ A^T V = 0 \ \Rightarrow \ A^T(L - AX) = 0 \ \Rightarrow \ X = (A^T A)^{-1}(A^T L) \tag{4.8}$$

Therefore, following the principle stated above, we start with an approximate position and keep computing an correction X to the position until X is small enough. The procedure of ranging positioning using weighted least square solver is desribed in Algorithm 4.

Algorithm 4: The procedure of ranging positioning using weighted least square solver.

Initialize the approximate position p_{hypo};
Initialize the correction matrix X;
while $||X|| > 1$ **do**
    Construct Matrix A and L;
    Compute X according to Equation 4.8;
    $p_{hypo} \leftarrow p_{hypo} + X^T$;
end
Return p_{hypo};

Chapter 5

HMM Map Matching

In this chapter, we introduce the principle of the Hidden Markov Model (HMM) map matching and how we integrate it with multipath mitigation.

5.1 The Hidden Markov Model

The HMM is based on the Markov chain, which is a model describing the probabilities of sequences of variables, like road segments. It has a strong assumption that if we want to predict the future variables in the sequence, only the current state matters. That is, considering a sequence of state variables $q_1, q_2, ..., q_i$, a Markov chain describes

$$P(q_i = a|q_1 q_2 ... q_{i-1}) = P(q_i = a|q_{i-1}) \tag{5.1}$$

However, HMM is interested in hidden events. For example in the map matching case, we don't observe road segments the user passed directly. Instead, the data we obtain is the GPS trajectory. The previous events are called hidden events and the latter are called observed events. According to Chapter A in [18], the HMM can be defined as Definition 5.

Definition 5 *(Hidden Markov Model) The Hidden Markov Model HMM(Q, A, O, B) is a state machine with the following components:*

- *a set of N states/hidden events $Q = \{q_1, q_2, ..., q_N\}$;*
- *a transition probability matrix $A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1N} \\ a_{21} & a_{22} & ... & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & ... & a_{NN} \end{bmatrix}$ where a_{ij} describes the probability of moving from q_i to q_j;*
- *a sequence of n observed events $O = o_1 o_2 ... o_n$;*
- *a emission probability matrix $B = \begin{bmatrix} b_{11} & b_{12} & ... & b_{1n} \\ b_{21} & b_{22} & ... & b_{2n} \\ \vdots & \vdots & & \vdots \\ b_{N1} & b_{N2} & ... & b_{Nn} \end{bmatrix}$ where b_{ij} representing the probability of the observed event o_j being generated from state q_i.*

We can re-organize the map matching problem based on Definition 5: the road network is equivalent to Q which contains all possible states/hidden events; a GPS trajectory is equivalent to O where a trajectory sample is an observed event. Therefore, the fundamental problem we try

to tackle is: given an observation sequence O, discover the best possible hidden event sequence that is most likely to have generated such an observation sequence.

The most common algorithm for solving such a problem in HMM is the Viterbi Algorithm which is a kind of dynamic programming [18]. Let $v(i,j)$ denotes the probability that the HMM stays in state q_i after experiencing j observed events $o_1, o_2, ..., o_j$. Together with the hidden Markov model $HMM(Q, A, O, B)$, the Viterbi Algorithm computes

$$\begin{cases} v(i,j) = b_{ij} & \text{when } j = 1 \\ v(i,j) = b_{ij} \times max_{t=i}^{N} v(t, j-1) a_{ti} & \text{otherwise} \end{cases} \tag{5.2}$$

and the best score is $max_{i=1}^{N} v(i,n)$ and the best possible hidden state sequence can be computed by tracing back from the state $argmax_{i=1}^{N} v(i,n)$.

5.2 Implementation

In the implementation of adapting HMM to map matching, the first problem is how to define the transition probability matrix A and the emission probability matrix B. According to the idea of Newson et al. [26], we make use of topological and geometric information of the road network to define the transition probability matrix. For example from state q_i to q_j (note that q_i and q_j are road segments), a_{ij} is 0 if q_j is not reachable from q_i. Conversely, if q_j is reachable, we first compute the the shortest path from q_i to q_j. We denote the Euclidean distance along the shortest path by $dist_{sp}(q_i, q_j)$, and take $dist_{sp}(q_i, q_j)$ as an indicator of the transition probability a_{ij}:

$$a_{ij} = \frac{\frac{1}{dist_{sp}(q_i, q_j)}}{\sum_{t=1}^{N} \frac{1}{dist_{sp}(q_i, q_t)}} \tag{5.3}$$

The emission probability b_{ij} measures the possibility of a trajectory sample o_j being generated on the road segment q_i. Commonly, b_{ij} considers the shortest distance between the trajectory sample and the road segment. The larger distance they have, the smaller b_{ij} will be. However, multipath errors might influence the performance of positioning and thus influence the distribution of b_{ij}. This is where we integrate multipath mitigation and make use of the output of Algorithm 1. Equation 5.4 shows how we define the emission probability b_{ij} with the distance function $dist_{pl}(o_j, q_i)$ which computes the shortest distance from a point o_j to a segment q_i.

$$\begin{cases} b_{ij} = \frac{1}{e^{dist_{pl}(o_j, q_i)/C_1}} & \text{when there is no multipath effect;} \\ b_{ij} = \frac{w_1}{e^{dist_{pl}(o_j, q_i)/C_1}} + \frac{w_2}{e^{d_{ij}/C_2}} & \text{when } d_{ij} > 0 \end{cases} \tag{5.4}$$

Note that d_{ij}, the distance between the hypothesis GPS position and the observation o_j, is computed by Algorithm 1. Also, C_1, C_2 are the decay constant and they describe the decrease rate of a quality. Smaller decay constants in Equation 4.4 make the quality vanish more quickly. And w_1, w_2 are weights where $w_1 + w_2 = 1$.

It is important to mention that, in the actual implementation, $v(i,j)$ from Equation 5.2 becomes extremely small when there is a long chain of observations. In case of running out of digits in Python, we add an extra step

$$v'(i,j) = v(i,j) / max_{t=i}^{N} v(t,j) \tag{5.5}$$

which means to scale up the possibilities. In this case, the maximum possibility in this round $max_{t=i}^{N} v'(t,j)$ (after experiencing j observations) is 1.

We adapted an existing implementation of HMM map matching [1] and integrate the idea of

[1] HMM map matcher: https://github.com/simonscheider/mapmatching

multipath mitigation into it. Algorithm 5 briefly describes the whole process.

Algorithm 5: The process of HMM map matching with multipath mitigation.

```
Let t_pr stores the subscript of previously selected road segment, initialise it to -1;
for every sample o_j in the trajectory do
    Select a road segment candidate set E;
    Run Algorithm 1;
    for each road segment candidate q_i ∈ E do
        Compute a_{t_pr i} (Equation 5.3) if t_pr is not -1;
        Compute b_ij (Equation 5.4) using the output of Algorithm 1;
        Compute v'(i, j) by Equation 5.5;
    end
    Select the road segment with maximum probability and store its subscript,
      t_pr ← argmax_{q_i ∈ E} v(i, n);
end
Select the path with the highest probability and trace back to output the path;
```

Chapter 6

Results

We tested the program on 7 trajectories. Table 6.1 shows the parameters we used in experimental tests. We tested different values of parameters for one trajectory and the values in Table 6.1 can output relatively good results. However, these values may not be optimal since we didn't check all possible value combinations for these parameters. In this chapter, we present the results of the

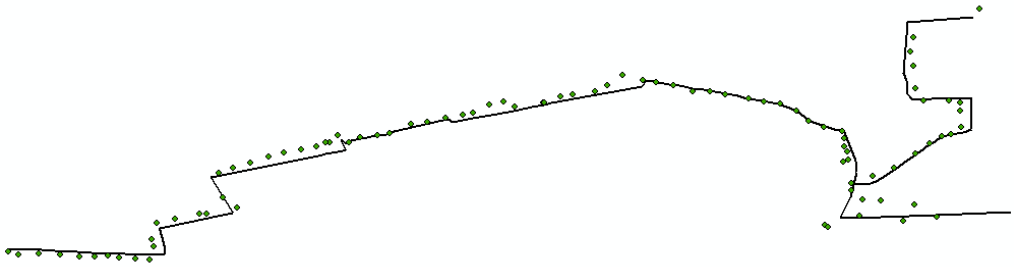
Trajectory id	r	C_1	C_2	w_1	w_2
0325-1	30m	50	100	0.5	0.5
0325-2	30m	50	100	0.5	0.5
0325-3	30m	50	100	0.5	0.5
0401-1	30m	100	200	0.5	0.5
0401-2	30m	50	100	0.5	0.5
0405	50m	50	100	0.5	0.5
0407	30m	50	100	0.5	0.5

Table 6.1: Parameter configuration in experimental tests.

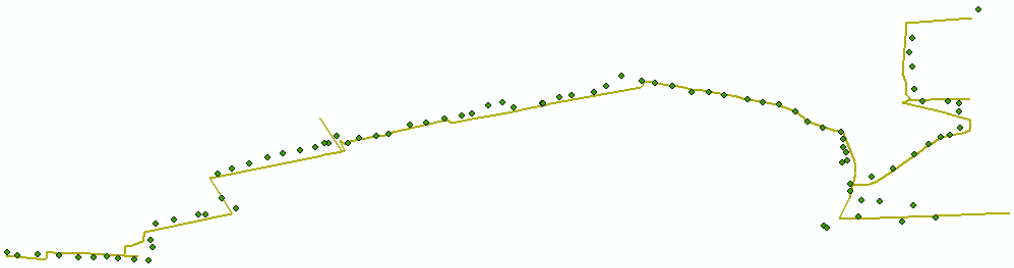
experimental tests and compared them with the results of the conventional HMM map matching in order to investigate the general ability for multipath mitigation to help map matching.

6.1 Analysis

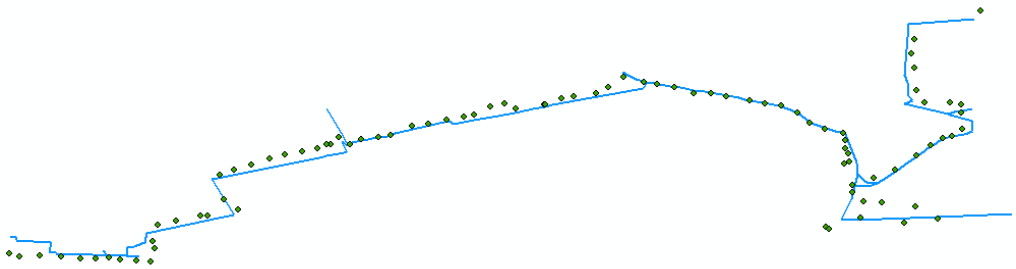
Figure 6.1 shows the actual path and the matching results of the conventional and the proposed methods for Trajectory 0325-1. Figure 6.2 zooms in and highlights the differences between the matching results for Trajectory 0325-1. In the case of Trajectory 0325-1, the proposed method with multipath mitigation cannot correct the matching errors in the conventional HMM map matching (the first, second and fourth columns in Figure 6.2). And it even degraded the matching as the third column in Figure 6.2 shows. The performance of the proposed method on Trajectory 0325-1 didn't meet our expectations.



(a) The actual path for Trajectory 0325-1.



(b) The path computed by the conventional HMM map matching for Trajectory 0325-1.



(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0325-1.

Figure 6.1: Map matching results for Trajectory 0325-1.

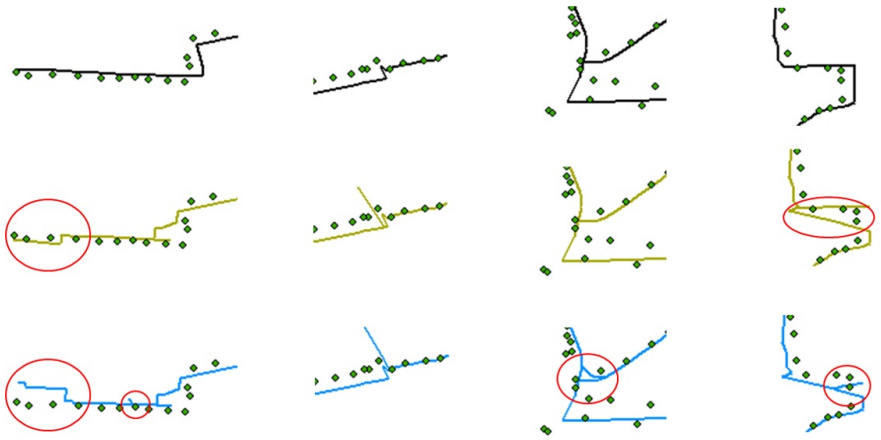
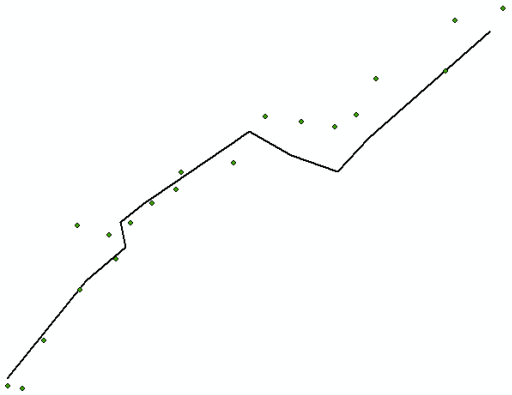
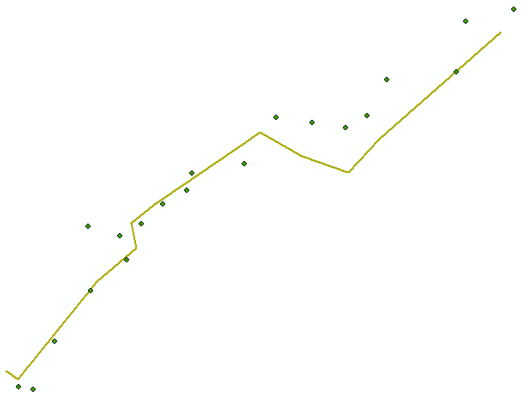


Figure 6.2: Differences between the matching results of the conventional and proposed methods for Trajectory 0325-1.

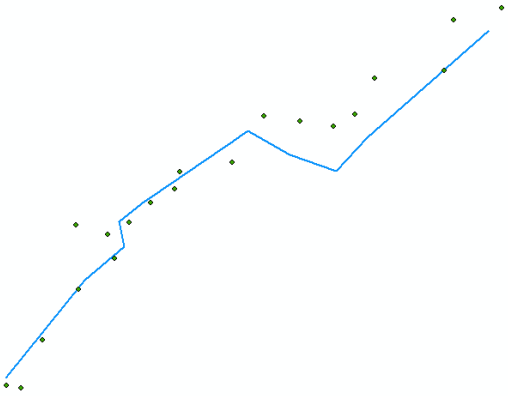
The results of Trajectory 0325-2 are shown in Figure 6.3. With fewer trajectory samples and more spare road network around them, both the conventional and proposed methods work well. But the proposed methods slightly improved the match by correcting the left bottom corner of the result of the conventional method.



(a) The actual path for Trajectory 0325-2.



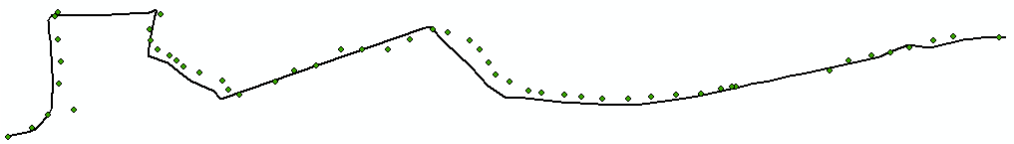
(b) The path computed by the conventional HMM map matching for Trajectory 0325-2.



(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0325-2.

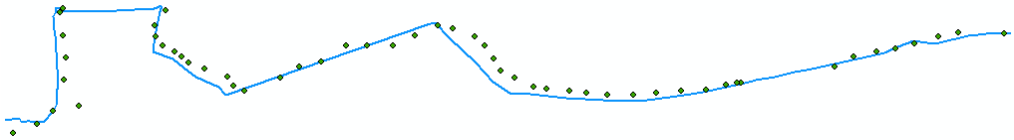
Figure 6.3: Map matching results for Trajectory 0325-2.

Figure 6.4 and Figure 6.5 compare the results of conventional and proposed methods to the actual path for Trajectory 0325-3. From the second and third columns of Figure 6.5 we can see that the proposed map matching method managed to eliminate redundant wrong tracks and select the correct road segments for Trajectory 0325-3.



(a) The actual path for Trajectory 0325-3.

(b) The path computed by the conventional HMM map matching for Trajectory 0325-3.



(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0325-3.

Figure 6.4: Map matching results for Trajectory 0325-3.

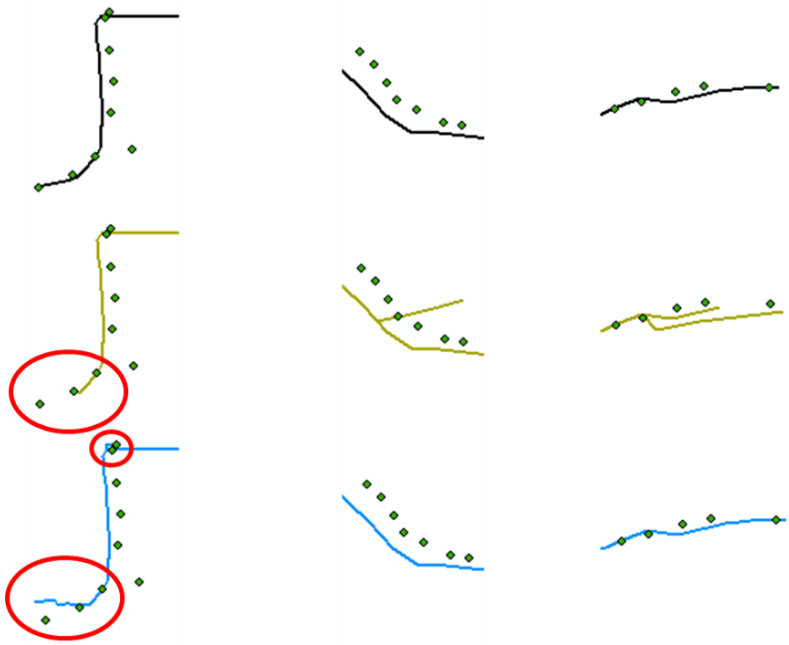


Figure 6.5: Differences between the matching results of the conventional and proposed methods for Trajectory 0325-3.

Samples in Trajectory 0401-1 are not of a good quality and thus, neither of these two map matching methods can give out a fair result as shown in Figure 6.7. Figure 6.6 highlights differences between the results of the conventional and proposed map matching methods. Compared to the result of the conventional method, the output of the proposed method is more accurate (like the first and second columns in Figure 6.6). Even though it failed to correct the match, it slightly improved the performance of map matching by filling the missing road segments (the third column in Figure 6.6) and eliminating redundant segments (the fourth column in Figure 6.6).

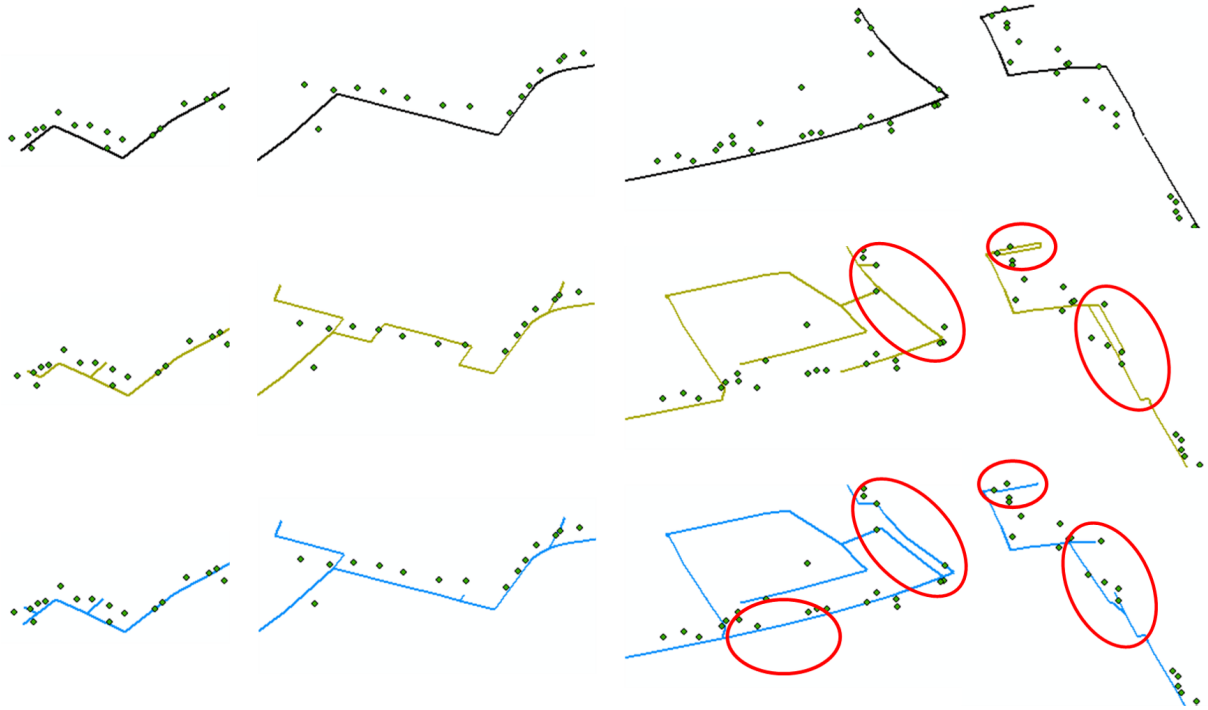
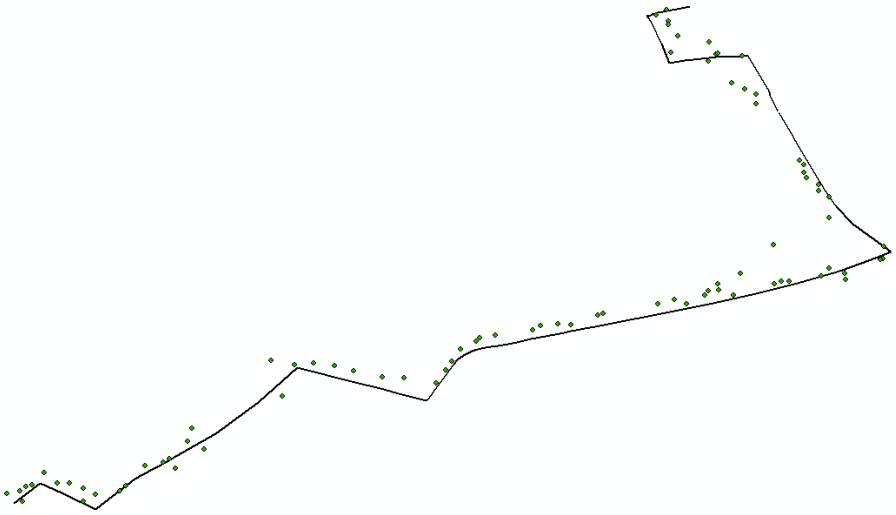
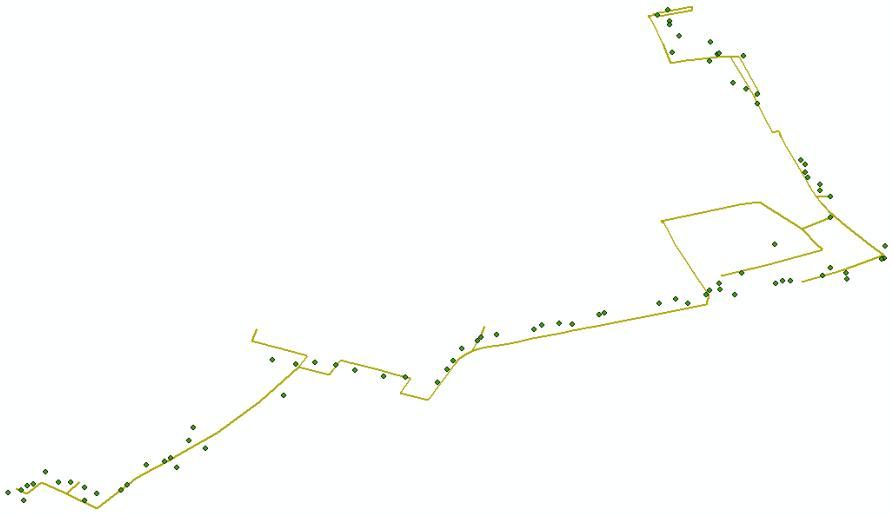


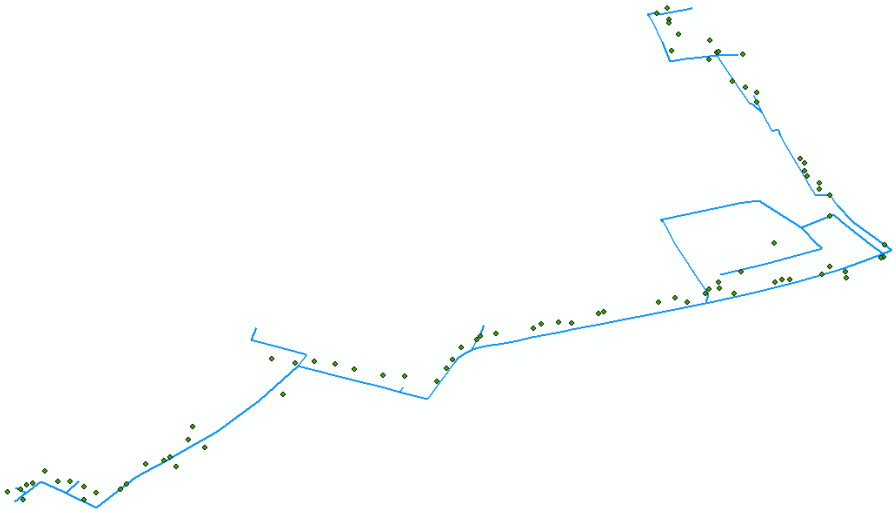
Figure 6.6: Differences between the matching results of the conventional and proposed methods for Trajectory 0401-1.



(a) The actual path for Trajectory 0401-1.



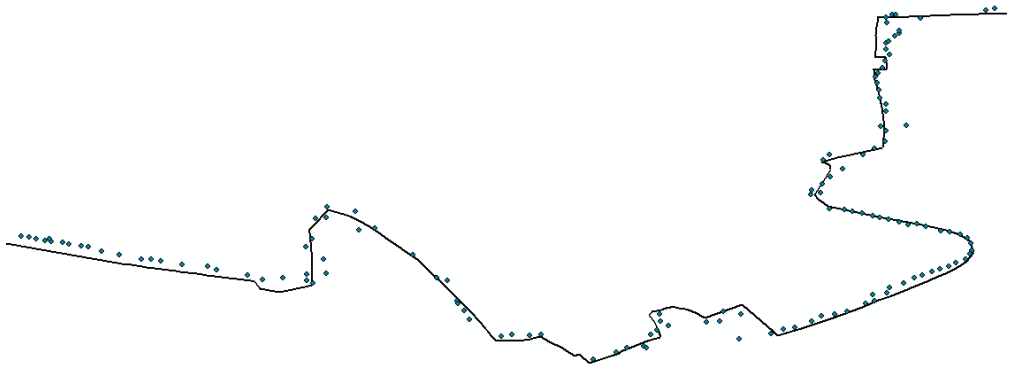
(b) The path computed by the conventional HMM map matching for Trajectory 0401-1.



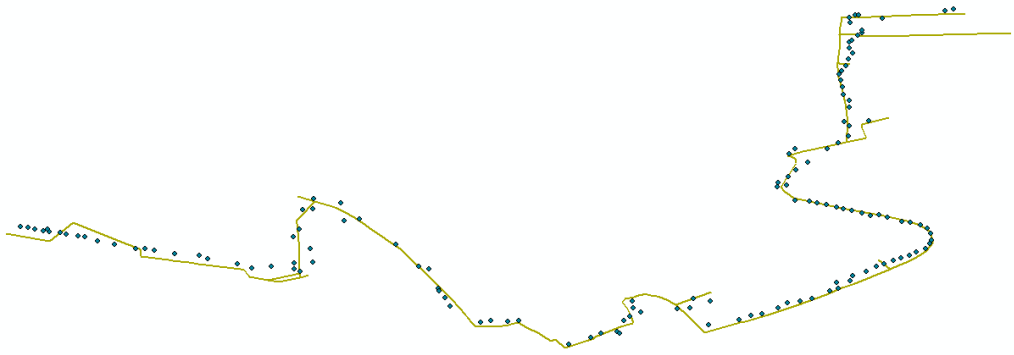
(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0401-1.

Figure 6.7: Map matching results for Trajectory 0401-1.

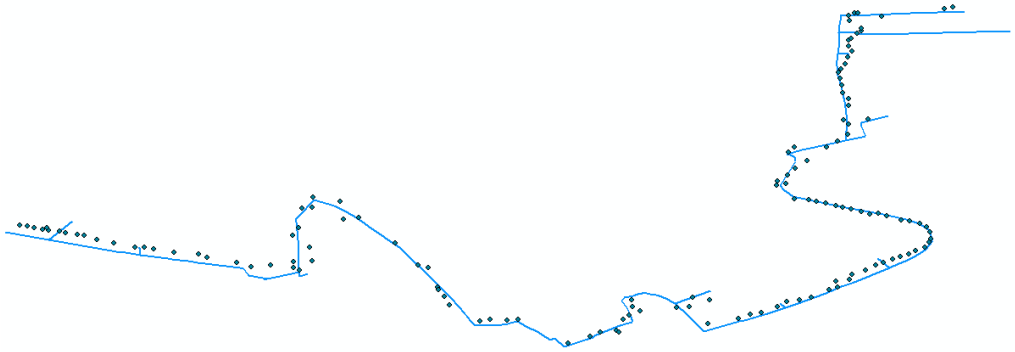
In the case of Trajectory 0401-2, the proposed method had a better performance than the conventional method. Figure 6.8 displays the outputs of these two methods and Figure 6.9 points out their differences. Similar to Trajectory 0401-1, the proposed method can partially correct the match errors in the conventional HMM map matching.



(a) The actual path for Trajectory 0401-2.



(b) The path computed by the conventional HMM map matching for Trajectory 0401-2.



(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0401-2.

Figure 6.8: Map matching results for Trajectory 0401-2.

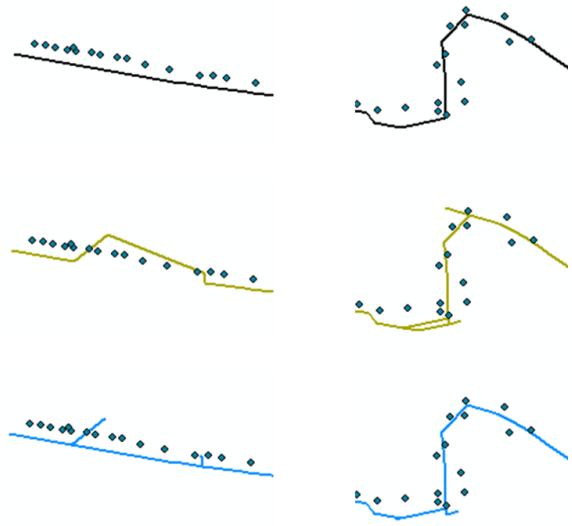


Figure 6.9: Differences between the matching results of the conventional and proposed methods for Trajectory 0401-2.

Figure 6.11 shows the matching results for Trajectory 0405 and Figure 6.10 points out their differences. The proposed method eliminated redundant segments but it also made a wrong correction as the third column shows in Figure 6.10. The matching results of the conventional and proposed method for Trajectory 0407 and the comparison between them are shown in Figure 6.12 and Figure 6.13.

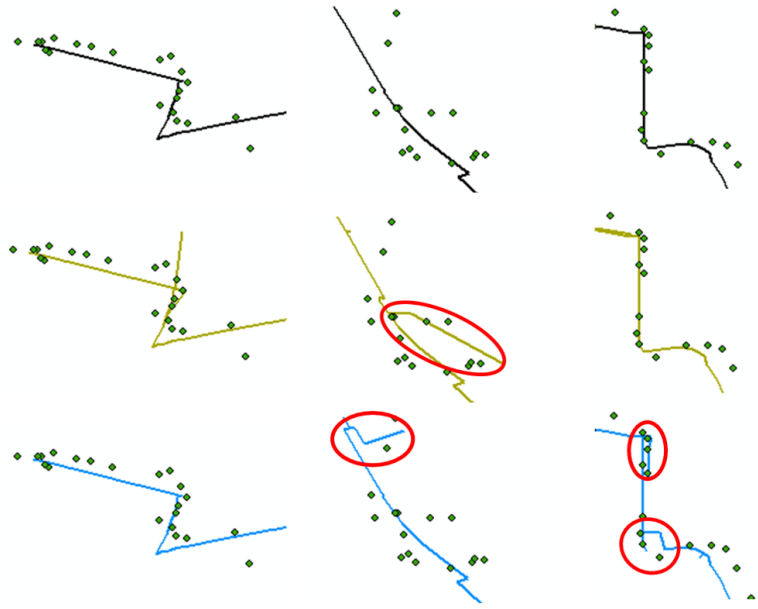
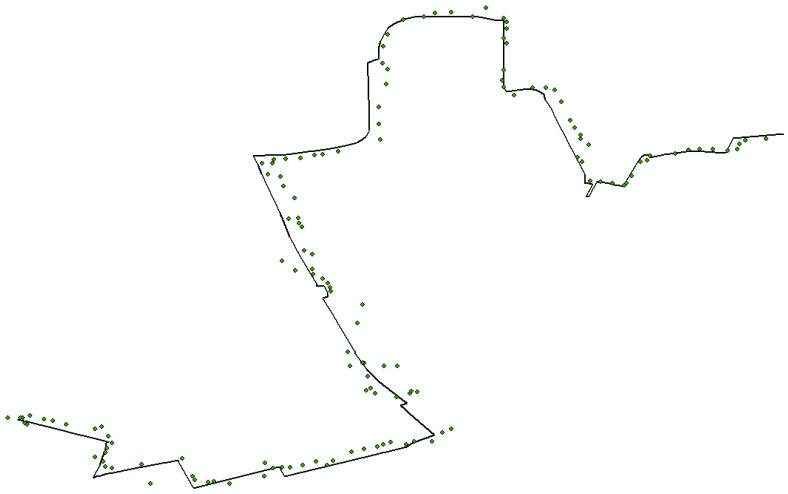
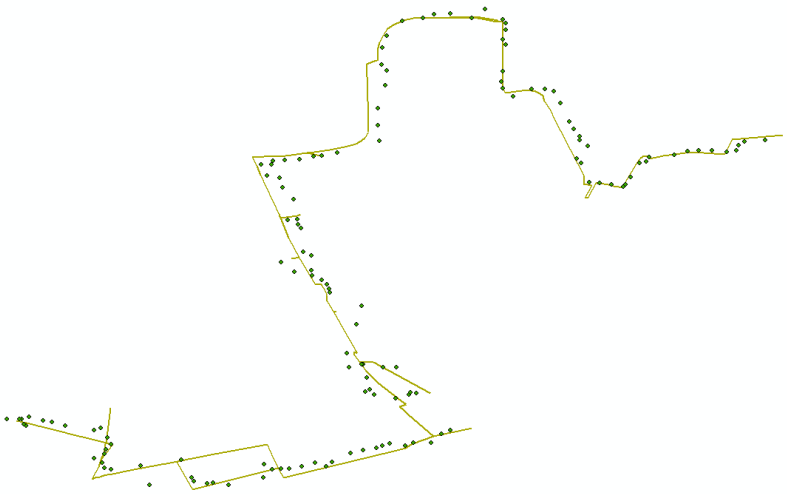


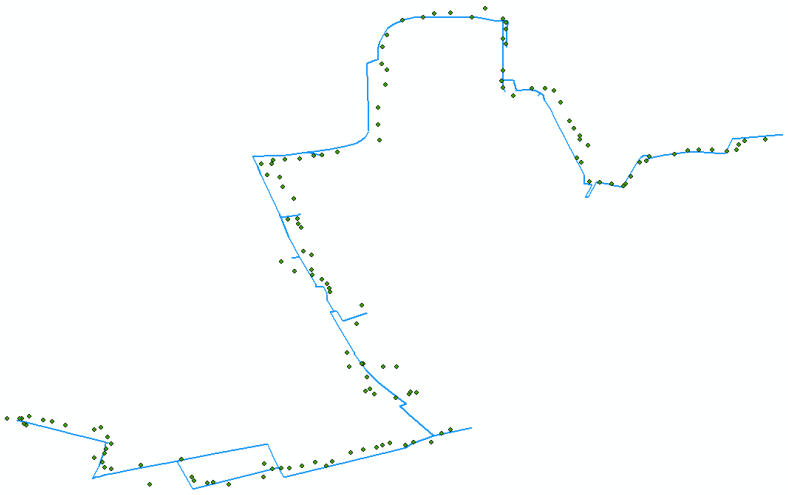
Figure 6.10: Differences between the matching results of the conventional and proposed methods for Trajectory 0405.



(a) The actual path for Trajectory 0405.

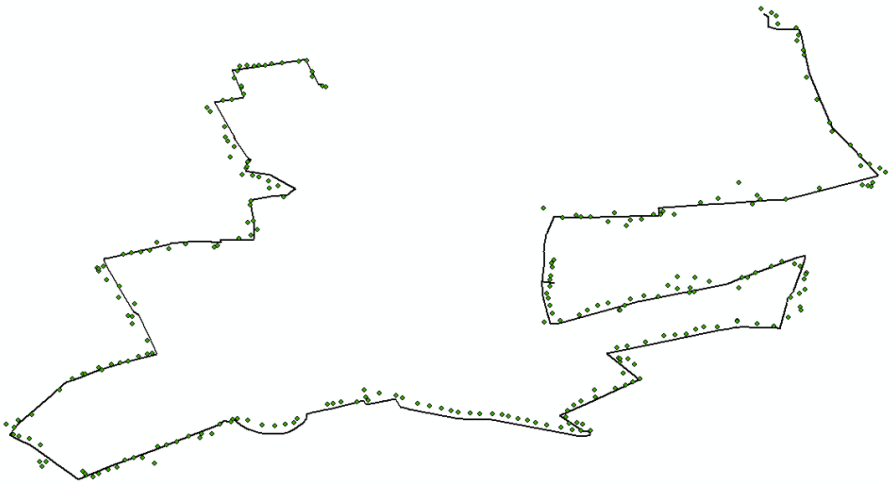


(b) The path computed by the conventional HMM map matching for Trajectory 0405.

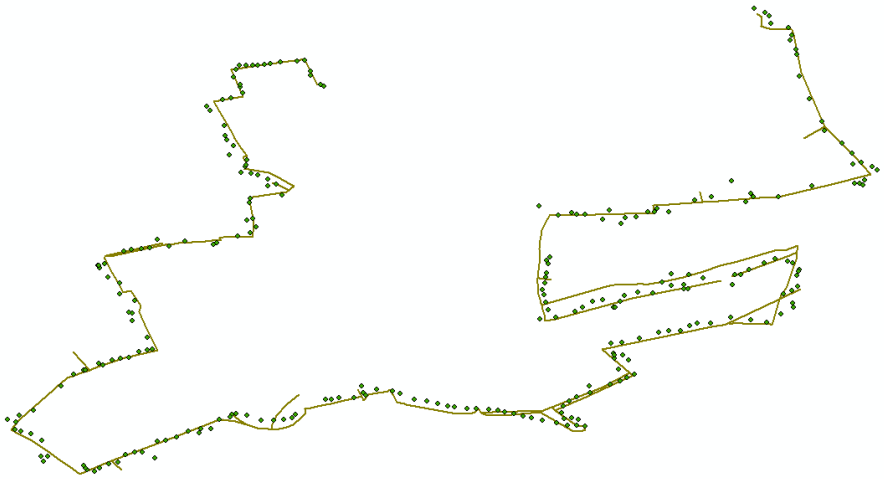


(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0405.

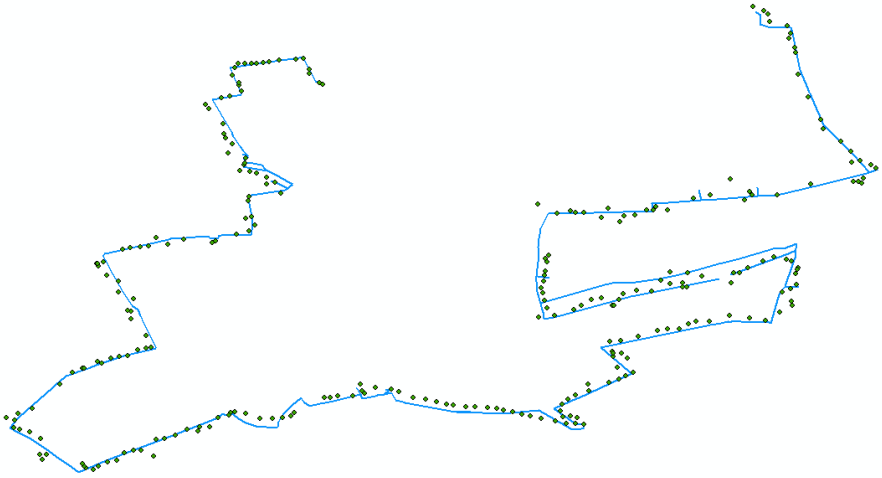
Figure 6.11: Map matching results for Trajectory 0405.



(a) The actual path for Trajectory 0407.



(b) The path computed by the conventional HMM map matching for Trajectory 0407.



(c) The path computed by the HMM map matching with multipath mitigation for Trajectory 0407.

Figure 6.12: Map matching results for Trajectory 0407.

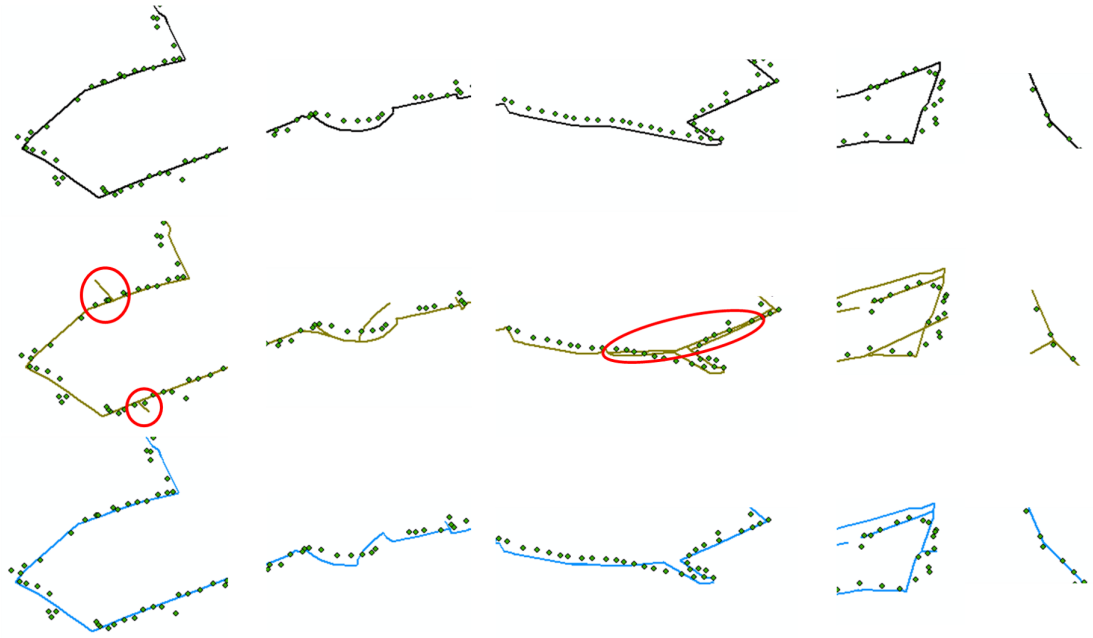


Figure 6.13: Differences between the matching results of the conventional and proposed methods for Trajectory 0407.

Trajectory id	Improvements	Deterioration	Draws
0325-1	0	2	3
0325-2	1	0	0
0325-3	2	0	1
0401-1	3	0	5
0401-2	3	0	5
0405	2	3	5
0407	7	3	7

Table 6.2: Statistics on improvements, deterioration and draws between the results of the proposed method and the conventional method.

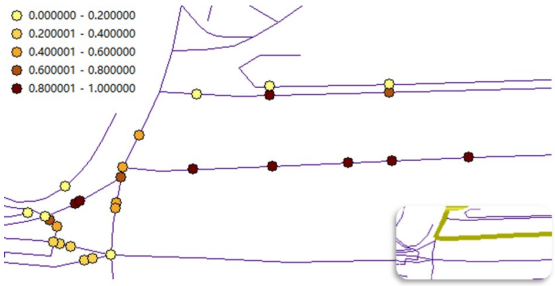
By comparing the results of the proposed and conventional methods, Table 6.2 counts the numbers of the improvements, deterioration and draws, the places that neither the proposed method nor the conventional method provides a good match. It only provides a rough overview of how the proposed HMM map matching method performs on our dataset. It is hard to make a conclusion on the general performance of the proposed HMM map matching method with multipath mitigation, because the trajectory data we collected is not big enough. Based on the examples we have, the proposed method had a positive influence on the results sometimes. In most cases, the proposed method could not correct the mismatch that happened in the conventional HMM map matching. And in a few cases, it selected the wrong road segments, degrading the accuracy of map matching. On the dataset we collected, the proposed method slightly improved the map matching results in general without largely degrading the accuracy in any specific places.

Then we analyse the experiments from a deeper perspective which refers to the candidates and their probabilities for each observed location. However, it is hard to analyse the whole map matching process step by step, because (1) HMM map matching selects the global optima instead of local optima; (2) we didn't implement a canvas to draw intermediate results; (3) the output shapefile (polyline) cannot store intermediate results including points and polyline. Therefore, we make it output the candidates generated and their possibilities ($v'(i,j)$ in Equation 5.5) for

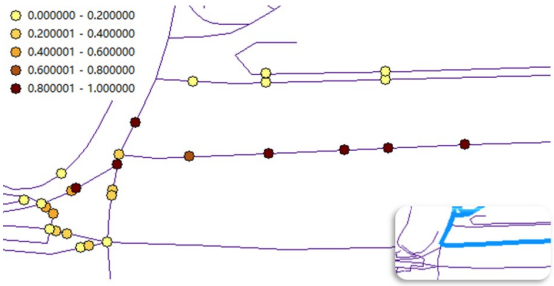
some specific trajectory samples. In this way, we hope to check if the proposed method provided candidates with a better probability distribution, especially in an urban canyon.

Figure 6.14 shows the candidate sets for a part of Trajectory 0325-1. The real world environment for this part of trajectory is complicated with an urban canyon (see Figure 6.14a) and multipath errors, especially NLOS, would likely happen. Figure 6.14b and 6.14c compare the candidate sets generated by the conventional and proposed methods. Subfigures in the bottom right corner of Figure 6.14b and 6.14c recall the map matching results of the conventional and proposed methods. We can see that both conventional and proposed methods made the correct match while the probability distributions of candidates are different. Compared to the conventional method, the proposed method assigned higher probabilities to candidates who are on the correct road segments.

(a) The real world environment for this part of Trajectory 0325-1.



(b) The candidate sets for GPS0036 to GPS0042 computed by the conventional method. The color of the candidates indicates the probability they have. The bottom right corner shows the map matching result of the conventional method.



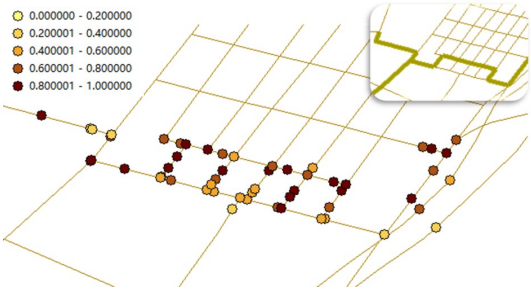
(c) The candidate sets for GPS0036 to GPS0042 computed by the proposed method. The color of the candidates indicates the probability they have. The bottom right corner shows the map matching result of the proposed method.

Figure 6.14: Candidate sets for GPS0036 to GPS0042 in Trajectory 0325-1.

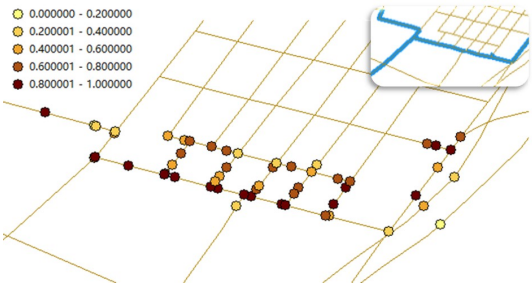
Another part we focus on is shown in Figure 6.15. This time the proposed method successfully corrected the mismatch in the conventional method. From 6.15a, we can see that NLOS errors might happen because buildings stand on both sides of the street. The proposed method did provide candidate sets with better probability distributions compared to the conventional method as shown in Figure 6.15b and 6.15c.

The third example is Figure 6.16. In this example, neither the conventional and proposed method could provide a correct match. Even though the proposed method made a small influence on the probabilities of candidate sets, incorrect road segments were still selected. This is probably because the inaccuracy in positioning here is not mainly related to NLOS errors, but interference errors. Clearly, in Figure 6.16a, the first half of the route didn't pass an area with a city canyon. With tall buildings only standing on one side of the street, multipath interference is more likely to happen rather than NLOS problems (see Figure 1.1).

(a) The real world environment for this part of Trajectory 0401-1.



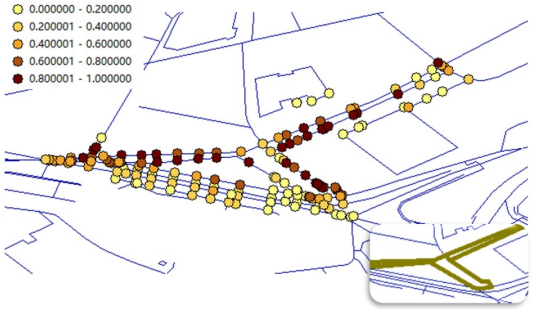
(b) The candidate sets for GPS0236 to GPS0244 computed by the conventional method. The color of the candidates indicates the probability they have. The top right corner shows the map matching result of the conventional method.



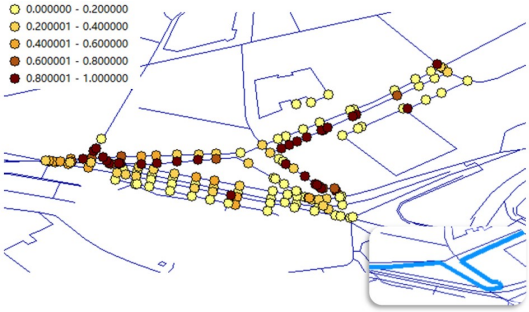
(c) The candidate sets for GPS0236 to GPS0244 computed by the proposed method. The color of the candidates indicates the probability they have. The top right corner shows the map matching result of the proposed method.

Figure 6.15: Candidate sets for GPS0236 to GPS0244 in Trajectory 0401-1.

(a) The real world environment for this part of Trajectory 0407.



(b) The candidate sets computed by the conventional method.

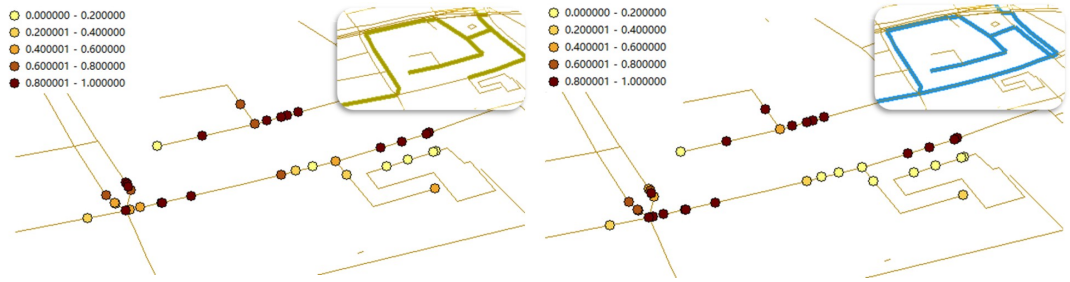


(c) The candidate sets computed by the proposed method.

Figure 6.16: Candidate sets for GPS0654 to GPS0678 in Trajectory 0407.

Similar to the previous example, neither of the conventional and proposed methods outputs the correct match as shown in Figure 6.17. But the real world environment for this example shows that the NLOS errors happened. We went through the candidates and their probabilities and we found that the mismatch is because the search radius is not big enough. As Figure 6.18 shows, the sample marked in the red circle can only have one road segment candidate due to the small search radius. Once we increase the searching radius from 30 meters to 50 meters, both two methods can give the correct match.

(a) The real world environment for this part of Trajectory 0401-1.



(b) The candidate sets computed by the conventional method.

(c) The candidate sets computed by the proposed method.

Figure 6.17: Candidate sets for GPS0208 to GPS0222 in Trajectory 0401-1.

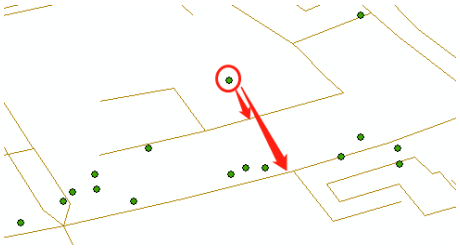


Figure 6.18: Green dots are trajectory samples. The sample marked can only have one candidate close to it due to the small search radius.

Based on the analysis above, the proposed method can slightly improve the accuracy of map matching when NLOS errors are the dominant reason to degrade positioning. But it degrades the accuracy of the conventional one in a few specific places. Possible reasons are:

1. Inappropriate parameters like a small searching radius.
2. The inaccuracy in the trajectory samples is not caused by NLOS errors. In this case, the proposed method will not do any contributions.
3. General errors like the observation errors in our dataset exclude trajectory data, for example, the errors in satellite positions and buildings heights.

6.2 Computation Time

We tested the program on 7 trajectories on two machines with different configurations. Machine 1 is shipped with Intel Core i7-6560U and 8GB memory while Machine 2 uses Intel Core i5-1140 with 16GB memory. Table 6.3 lists the information about these trajectories and the running time they cost.

Trajectory id	Sample size	Time (Machine 1)	Time (Machine 2)
0325-1	92	1050.6 s	33 s
0325-2	20	518.2 s	7.2 s
0325-3	54	585.7 s	11.1 s
0401-1	84	793.9 s	30.4 s
0401-2	135	692.9 s	27.9 s
0405	140	1881.3 s	85.7 s
0407	268	1075.9 s	64.4 s

Table 6.3: Time cost for trajectories on different machine.

Chapter 7

Conclusions

In this final project, we focused on the problem of how we can improve the accuracy of map matching in urban areas. We realized that the bad quality of trajectories in urban areas is mainly caused by multipath errors. Hence, we developed an idea of integrating multipath mitigation (mainly for NLOS errors) into the map matching techniques.

We have proposed an algorithm that uses 3D building models and geometric methods to simulate multipath, and then mitigates them to improve the accuracy of the conventional HMM map matching. By using our geometric methods, obstructed satellite signals can be effectively recognized and the potential NLOS errors can be estimated. By constructing the hypothesis pseudo ranges from direct pseudo ranges and NLOS errors, we can compute the hypothesis position for a GPS trajectory sample using the weighted least square solver. If the calculated position is close to the sample we measured, the road segment that the position stays on is more likely to be selected as a match to the sample. Conversely, the road segment is less likely to be selected. By doing so, we adjust the probability of those road segment candidates for each trajectory sample. Finally, we run HMM map matching on these road segment candidate sets to select the global optima path.

The algorithm was evaluated through experiments on a real world dataset. Trajectory data was manually collected in Newcastle upon Tyne, the UK and the rest of data, including road network, 3D building model and satellite position data were accessed online. It was found that running on this dataset, the proposed algorithm can improve the accuracy of map matching by pruning redundant road segments while filling correct segments to the path. However, the improvement in performance is slight and unstable as the algorithm conducts incorrect modifications and degrades the accuracy in a few specific places. The algorithm slightly degrades the accuracy in a few cases. But from a global perspective of the data set, the improvement overweights the degradation.

The algorithm is time-consuming with a speed of about 100 trajectory samples per 30 seconds on a machine shipping with Intel Core i5-11400 and 16GB memory, or approximately 100 trajectory samples per 1000 seconds on a machine with Intel Core i7-6560U and 8GB memory.

However, the conclusion above is based on the dataset we have with only 7 trajectories. The algorithm should be tested on a larger dataset with more trajectories in future work. With a larger dataset, we can measure the similarity between the result of the proposed method and the actual path by some functions like Fréchet distance. By comparing to the similarity between the result of the conventional method and the actual path, we can then evaluate the performance of the proposed algorithm in general. Also, some details in the proposed algorithm need further discussion. For example, for one road segment candidate, we pick one point from it to do the simulation. It may be better to pick multiple points along a road segment candidate. Furthermore, we only considered NLOS errors in the trajectories for the proposed method and the remaining errors like interference errors need to be investigated in the future.

Bibliography

[1] James Arvo and David Kirk. Fast ray tracing by ray classification. *ACM Siggraph Computer Graphics*, 21(4):55–64, 1987. 23

[2] Alan Bensky. *Wireless positioning technologies and applications*. Artech House, 2016. 9

[3] A Bourdeau, M Sahmoudi, and JY Tourneret. Tight integration of gnss and a 3d city model for robust positioning in urban canyons. In *Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012)*, pages 1263–1269, 2012. 2

[4] Julia Breßler, Pierre Reisdorf, Marcus Obst, and Gerd Wanielik. Gnss positioning in non-line-of-sight context—a survey. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, pages 1147–1154. IEEE, 2016. 1, 2

[5] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. A survey on map-matching algorithms. In *Australasian Database Conference*, pages 121–133. Springer, 2020. 1, 3

[6] Vincent Drevelle and Philippe Bonnifait. igps: Global positioning in urban canyons with road surface maps. *IEEE Intelligent Transportation Systems Magazine*, 4(3):6–18, 2012. 3

[7] Xiao Fu, Jiaxu Zhang, and Yue Zhang. An online map matching algorithm based on second-order hidden markov model. *Journal of Advanced Transportation*, 2021, 2021. 3

[8] N. Giordano. *College Physics: Reasoning and Relationships*. Cengage Learning, 2009. 10

[9] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 776–781. IEEE, 2012. 4

[10] Paul D Groves. Principles of gnss, inertial, and multisensor integrated navigation systems, [book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):26–27, 2015. 9, 10, 11

[11] Paul D Groves, Mounir Adjrad, Han Gao, and CD Ellul. Intelligent gnss positioning using 3d mapping and context detection for better accuracy in dense urban environments. Royal Institute of Navigation, 2016. 1

[12] Paul D Groves, Ziyi Jiang, Morten Rudi, and Philip Strode. A portfolio approach to nlos and multipath mitigation in dense urban areas. The Institute of Navigation, 2013. 3

[13] PD Groves. Gnss solutions: Multipath vs. nlos signals, how does non-line-of-sight reception differ from multipath interference? *Insid. GNSS*, pages 40–42. 1, 2

[14] Christian Hirt. *Digital Terrain Models*, pages 1–6. Springer International Publishing, Cham, 2014. 15

[15] Li-Ta Hsu, Yanlei Gu, and Shunsuke Kamijo. 3d building model-based pedestrian positioning method using gps/glonass/qzss and its reliability calculation. *GPS solutions*, 20(3):413–428, 2016. 6

[16] National Imagery and Mapping Agency. Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems. *Technical Report. 3RD Edition, TR8350. 2"*, 2000. 7

[17] Dah-Jing Jwo, MH Hsieh, and YC Lee. Gps navigation solution using the iterative least absolute deviation approach. *Sci. Iran. B*, 22:2103–2111, 2015. 27

[18] Vlado Keselj. Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6, $115.00, 2009. 29, 30

[19] Rakesh Kumar and Mark G Petovello. A novel gnss positioning technique for improved accuracy in urban canyon scenarios using 3d city model. In *Proceedings of the 27th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, pages 2139–2148, 2014. 3

[20] Lawrence Lau. Chapter 4 - gnss multipath errors and mitigation techniques. In George p. Petropoulos and Prashant K. Srivastava, editors, *GPS and GNSS Technology in Geosciences*, pages 77–98. Elsevier, 2021. 1

[21] Zhilin Li, Christopher Zhu, and Chris Gold. *Digital terrain modeling: principles and methodology*. CRC press, 2004. 15

[22] Yang Liu, Lianjie Fu, Jinling Wang, and Chunxi Zhang. Study of gnss loss of lock characteristics under ionosphere scintillation with gnss data at weipa (australia) during solar maximum phase. *Sensors*, 17(10):2205, 2017. 1

[23] Vida Maliene, Vytautas Grigonis, Vytautas Palevičius, and Sam Griffiths. Geographic information system: Old principles with new capabilities. *Urban Design International*, 16(1):1–6, 2011. 9

[24] Shunsuke Miura, Shoma Hisaka, and Shunsuke Kamijo. Gps multipath detection and rectification using 3d maps. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1528–1534. IEEE, 2013. 3, 6, 25, 27

[25] Shunsuke Miura and Shunsuke Kamijo. Gps error correction by multipath adaptation. In *International Journal of Intelligent Transportation Systems Research*, pages 1–8, 2015. 3, 6

[26] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343, 2009. 4, 30

[27] Marcus Obst, Sven Bauer, and Gerd Wanielik. Urban multipath detection and mitigation with dynamic 3d maps for reliable land vehicle localization. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 685–691. IEEE, 2012. 1, 6, 15

[28] Simon Ollander, Friedrich-Wilhelm Bode, and Marcus Baum. Multi-frequency gnss signal fusion for minimization of multipath and non-line-of-sight errors: A survey. In *2018 15th Workshop on Positioning, Navigation and Communications (WPNC)*, pages 1–6. IEEE, 2018. 1, 2

[29] Takayuki Osogami and Rudy Raymond. Map matching with inverse reinforcement learning. In *IJCAI*, pages 2547–2553, 2013. 4

[30] Oliver Pink and Britta Hummel. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *2008 11th International IEEE conference on intelligent transportation systems*, pages 862–867. IEEE, 2008. 3

[31] Tomaz Podobnikar. Methods for visual quality assessment of a digital terrain model. *SAPI EN. S. Surveys and Perspectives Integrating Environment and Society*, (2.2), 2009. 15

[32] Mohammed A Quddus, Washington Y Ochieng, and Robert B Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation research part c: Emerging technologies*, 15(5):312–328, 2007. 1, 3

[33] Ordnance Survey. A guide to coordinate systems in great britain. v3. 3. 2018. 7, 8

[34] Roger F Tomlinson. *Thinking about GIS: geographic information system planning for managers*, volume 1. ESRI, Inc., 2007. 9

[35] Hank Weghorst, Gary Hooper, and Donald P Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics (TOG)*, 3(1):52–69, 1984. 23

[36] Shuang Zhou, Liang Mi, Hao Chen, and Yishuang Geng. Building detection in digital surface model. In *2013 IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 194–199, 2013. 15