Fakultät für Bauingenieur- und Vermessungswesen
Lehrstuhl für Kartographie
Prof. Dr.-Ing. Liqiu Meng

Algorithm Design and Implementation of Map Matching of City-wide Floating Car Data

Technische Universität München

Master's Thesis

Author: Luyi Han
Supervisor: M.Eng. Jian Yang
Submission Date: January 12, 2015

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

January 11, 2015 Luyi Han

Acknowledgments

I would like to express my gratitude to all those who helped me during the writing of this master thesis.

My deepest gratitude goes to my supervisor M.Eng. Jian Yang, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

Second, I would like to express my heartfelt gratitude to Technische Universität München and the Cartography faculty for giving me the opportunity to study this research topic, and offering me comfort researching environment.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my friends and classmates who gave me their help and time in helping me solve my problems during the difficult course of the thesis.

Abstract

With the rapid development of the urbanization, the city transportation system exposes more and more unreasonable planning and congestion problems. To solve these problems, the Floating Car Data (FCD), including the city-wide driving data of vehicles, plays an important role in studying and analyzing Intelligent Transportation System (ITS). Map matching, which serves the purpose of recovering the original route on a road network from a sequence of noisy GPS observations, is one of the most significant prerequisite procedure of using FCD. The FCD usually contains huge amount of noisy data and is updated frequently, but most of the map matching algorithms published in the last few decades were only tested on prepared small and smooth test data that is not as noisy and sparseness as FCD. In this thesis, new map matching algorithms called Fusion Map Matching (FMM) algorithms are presented. The FMM algorithms comprehensively use the two most important map matching technology branches, global and incremental map matching algorithms, and make use of the strong points and avoid the shortcomings of them. Through the tests on FCD in Shanghai, China with different scenarios, the FMM algorithms show better performances on both the accuracy and efficiency perspective compared with other existing global algorithms. In addition, a work flow is proposed for map matching FCD, which can be directly deployed in the map matching project.

Keywords: map matching, floating car data, intelligent transportation system

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, the amounts of vehicles rapidly increase with the acceleration of the urbanization process in cities. While providing convenience to people, the city transportation system also exposes a lot of unreasonable planning and congestion problems. With the rapid development of the positioning technologies, more and more movement data are being collected nowadays. In particular, Global Positioning System (GPS)-based positioning technology has become a standard in transportation, which help to capture huge amount of movement data of vehicles in the road networks. And there is lots of on-going research to make better use of this data.

Floating Car Data (FCD), which includes the city-wide driving data of vehicles, is one of the most advanced sources to analyze the transportation system in the Intelligent Transportation System (ITS). The basic idea of FCD is to connect it with the city road network in the aspects of time and space based on the location, direction and speed information. This data can be recorded by the GPS devices installed in the vehicles, and then get the traffic information of the road network. Taxis, which has the best spatial and temporal coverage of the urban road traffic, are the ideal agents for collecting floating car data for city-wide applications.

The GPS data, which contains latitude, longitude and timestamps informations, is not precise due to including various measurement errors. Therefore, the map matching procedure should be done before the GPS data is used in the further analysis. The map matching techniques, which integrate positioning data with road network data, are developed to provide accurate and reliable positioning information. To be more specific, map matching is to assign each GPS trajectory point to the correct road segment with given information and algorithm. Since the concept of map matching was put forward, various researches ([12][13][15][17][18] [21][23][24]) have been done to design map matching algorithms for different scenarios.

As one of the preprocessing procedures, the result of map matching has significant influence on the performance of FCD analysis. It is always a challenge to map matching

FCD due to its two characteristics: huge date volume and quickly updating. The FCD is gathered from the actual complex scenarios, and is usually noisy, sparseness and not well organized. To the best of our knowledge, previous works have focused on map matching either experimental data with a data collection under control or small real world dataset with arbitrary spatial distribution[21]. In addition, none of the previous work has proposed a complete and clear work flow of map matching. Therefore, a map matching algorithm that can efficiently and accurately get the matching results, and can also be achieved on FCD, is required.

In this thesis, we present two new map matching algorithms that meet the requirements of map matching FCD, and evaluate the performances of our algorithms through comparing them with other existing map matching algorithm on the city-wide taxi FCD in Shanghai, China. In addition, we propose a complete and clear work flow of dealing with map matching FCD from the GPS data handling to the result representation, which can be directly deployed in the map matching project.

2 Overview of Map Matching

Map matching is the process that serves the purpose of recovering the original route on a road network from a sequence of noisy GPS observations[23]. From 1970s, various map matching algorithms based on different techniques have been developed. In this chapter, we will introduce the map matching problem, the principles and characteristics of some typical map matching algorithms.

2.1 Definitions

In this thesis, the definitions relate to map matching are described as below:

GPS Log: A GPS Log is a set of information of GPS points. The basic fields in GPS log are the geographical information (longitude and latitude) and the corresponding timestamps (T).

GPS Trajectory: A GPS trajectory is defined as a set of successive GPS points (P) with specific condition(s), such as same vehicle, or (and) continues timestamps. Trajectory J_i with n GPS points (P_1 to P_n) is defined as $J_i\{\{P_1, T_1\}, \{P_2, T_2\}, \ldots, \{P_n, T_n\}\}$.

Road Segment: A road segment (R) is composed with two vertexes and the edge between them, and the basic fields in each road segment is *id* and *geometry* (Figure 2.1).

Road Network: A road network is a set of road segments$\{R_1, R_2, \ldots, R_n\}$ within specific conditions (area, city range, etc).

Map Matching Results: Following a specific map matching algorithm, the points on the GPS trajectories are assigned on road segments as the map matching results (M).

Ground Truth Data: In map matching category, ground truth data (G) is the set of trajectory points and the corresponding road segments, and it can be recorded during the GPS data collection or labeled by human after the collection.

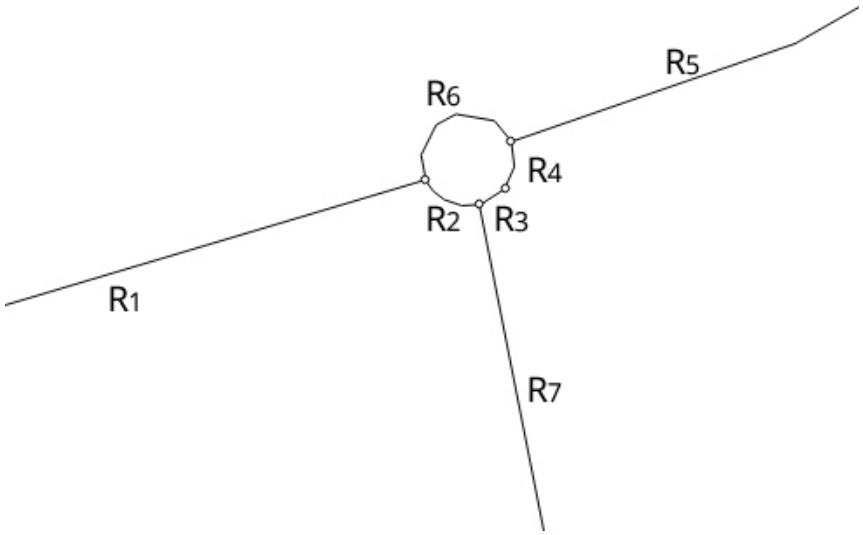


Figure 2.1: Road Segments(R) and Vertexes

Figure 2.2 shows the difference between the map matching results and ground truth data.

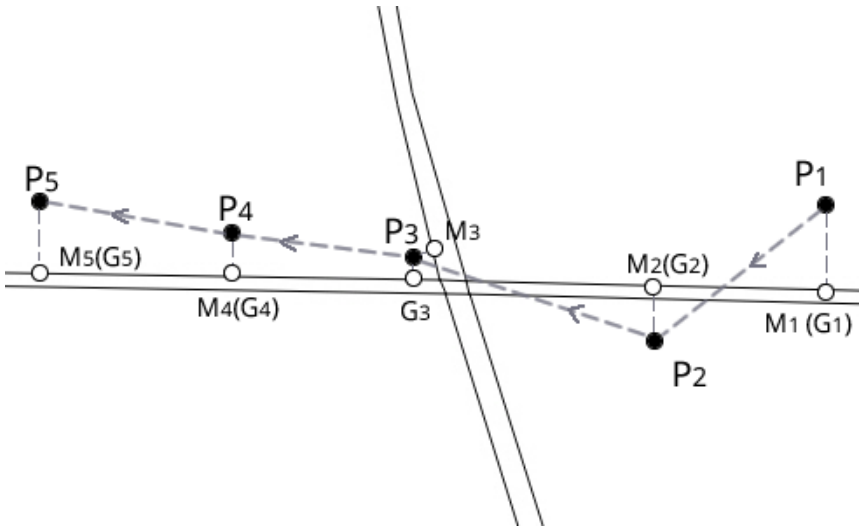


Figure 2.2: GPS Trajectory(P), Map matching results(M) and Ground truth data(G)

2.2 Problem Statement

As shown in Figure 2.3, every trajectory point has several potential candidates, and it is easy for human to distinguish the actual route in this example (sometimes it is also hard

for human to decide the actual routes). To deal with large amount of GPS data, a suitable map matching algorithm is necessary. In this case, if directly choose the closest candidate as the matching result, points P_1, P_2, P_3 will all be matched to the incorrect road segments. Therefore, map matching algorithms need to consider more factors to get more reliable results.

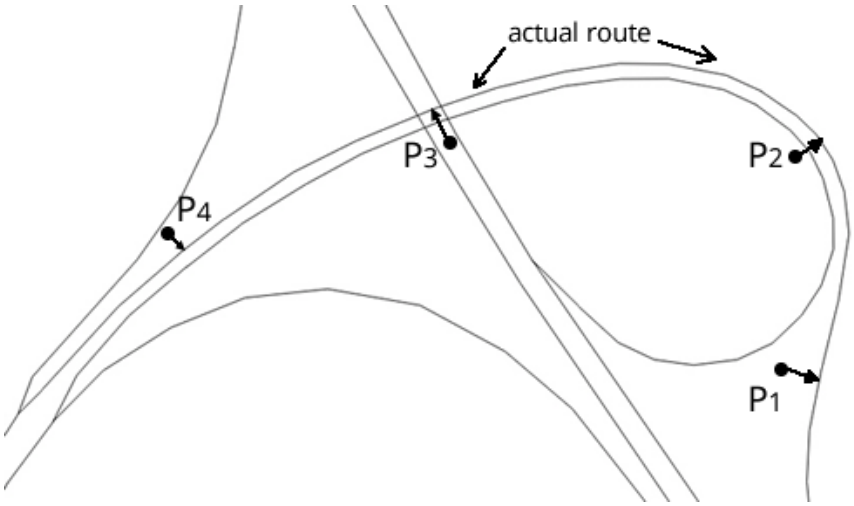


Figure 2.3: Map matching problem

2.3 Classification of Map Matching Methods

There are several perspectives to classify map matching algorithms. From the executing time, map matching algorithms can be divided into real-time and post-processing algorithms; the former one can be used for car navigation, and the latter one can be served in the transportation system analysis with large amount of data. According to the calculation factors, the algorithms can be classified into topological and probabilistic based algorithms. Another perspective is the range of GPS trajectory points that are considered in each calculation, and depend on this, the algorithms can be divided into incremental and global algorithms.

2.3.1 Topological and probabilistic methods

The topological map matching algorithm makes use of the geometry of the links as well as the connectivity and contiguity of the links[19]. In the concept of Geographical in-

formation system (GIS), topology is used to describe the relationships between entities (points, lines and polygons), and the relationships include touch, cross, intersect and disjoint. In the concept of map matching, the entities and their relationships specifically refer to the connectivity of the road segments. Besides these basic topological information, to improve the performance of the algorithms, some additional informations, such as road level classification and turn restriction, are also be used in some of topological based map matching algorithms. The advantage of topological algorithms are "fast, simple and very efficient"[20], but it needs to get very detailed of road topological information, and some of them are not available in every case.

The probabilistic algorithm requires the definition of an elliptical or rectangular confidence region around a position fix obtained from a navigation sensor[19]. The most simple probabilistic map matching algorithm is to choose the nearest road segment of each trajectory point, which only considers the GPS measure error distribution. Other possibilities, such as the transition possibility based on the relationship between the shortest path and great circle distances of successive trajectory points, are also used in some probabilistic map matching algorithms.

2.3.2 Incremental and global methods

The incremental algorithm iteratively calculates the next max-weight candidate based on the current best candidate or a summary of all previous samples[21]. Several factors can be used to calculate the weights of the candidates.

Different with incremental algorithms, the global algorithm tries to calculate the max-weight values of the whole trajectory, and iteratively go back to get the candidates that can get the max-weight values[21], and then get the map matching results for the whole trajectory. Global map matching algorithm can also be used in the navigation scenario when limiting the number of trajectory points participated in each calculation.

Compare the performances of the incremental and global algorithms, generally speaking, the incremental algorithm has better efficiency due to less computation needed, and the global algorithm has better accuracy.

2.3.3 The state of art

Practically, the existing map matching algorithms integrally have the characters mentioned in the last section. In this section, we will introduce some typical examples that can represent current development of map matching algorithms.

Nagendra[20] publishes an incremental weighted-based topological map matching algorithm, which considers four factors: heading, proximity, link connectivity and turn restriction, and a weight value is assigned to each factor. Based on Nagendra's research, Yang [22] publishes an enhanced weighted-based topological map matching algorithm. In this research, three scenarios of the city, dense area, common area and sparse area, are defined according to the road densities. For each scenario, different weight values of the four factors are designed to improve the map matching accuracy. In these kind of topological algorithms, the determination of the first trajectory point is very important, and only comparing the distances of the first trajectory point to the road segments and choosing the smallest one is not robust enough in some scenarios. In addition, a road network with comprehensive informations is an essential precondition.

Brakatsoulas[6] introduces a global topological map matching algorithm that compares the similarity between the curves of road segments and each trajectory through calculating the Fréchet distance. The Fréchet distance is described as following: imagine a dog walking along one curve and the dog's owner walking along the other curve, connected by a leash. Both walk continuously along their respective curve from the prescribed start point to the prescribed end point of the curve. Both may vary their speed, and even stop, at arbitrary positions and for arbitrarily long. However, neither can backtrack[5]. The Fréchet distance between the two curves is the length of the shortest leash. The smaller the Fréchet distance, the bigger the similarity of the two curves.

Liu[14] presents an incremental topological map matching algorithm, which considers the intersection relationship between the trajectory and the road segment to simplify the calculation and improve the accuracy. The basic consideration of this algorithm is that if the GPS trajectory has intersection with a road segment, the vehicle's position can be matched to this road segment. A Passby algorithm is designed to carry out the map matching process with three cases. The first case is that the trajectory passes by both intersections of a road, and this road segment can be chosen as the map matching result without individual check; the second case is that the trajectory only passes by the starting intersection of a road, and the candidate edges that next point may move on can be generated; the last case

covers the remaining points, and a traditional incremental algorithm will be used.

Griffin[10] proposes a global topological map matching algorithm, which firstly identifies key waypoint in GPS trajectory using a modified Peuker curve reduction algorithm, and then sends the key waypoints to a black-box driving directions service, which returns a route utilizing each of the key waypoints. After validating checking, the route will be chosen as the map matching result for the whole trajectory.

Mazhelis[17] proposes a probabilistic map matching algorithm, which estimates the possibility of each road segments with recursive Bayesian estimation, and the road link is identified using maximum a posteriori probability principle. In this method, three factors are took into account: the distance between the trajectory point and road segment, the discrepancy between the measurements-driving heading and the road segment heading, and the circle distance between the candidates of two successive trajectory points. The advantage of this algorithms is that it doesn't need the topological information of the road segments, thus it can be used with the imperfect road network data.

Hidden Markov Model (HMM) is a typical theory used in global probabilistic based map matching algorithm. In HMM based algorithms, the basic elements are the measurement (observation) and transition possibility.

Lamb and Thiébaux[13] are the earliest authors who start to use HMM in map matching application. In their research, they use several Kalman filters to track the vehicle along different hypothesized paths, and then use HMM to choose between all the paths.

In Hummel's[11] HMM based map matching algorithm, the measurement possibility relates to the circle distance between the trajectory point and the road segment, and the transition possibility relates to the road topology (candidates that adjacent with the current road segment) and road restrains (one way road, U-turn).

Krumm's [12] algorithm has similar measurement possibility definition with Hummel[11], and the transition possibility is the relationship between the estimated and actual traversal times between each two successive trajectory points: the smaller the difference, the bigger the transition possibility.

Newson[18] publishes a HMM based map matching algorithm that is similar to Krumm's research[12], but modify the transition possibility to the value differences between the shortest path and the circle distances of each two trajectory points, which are not that sensitive to the traffic conditions compared with time differences.

Lou[15] proposes a ST (Spatial and Temporal) algorithm that focus on the study for low-sampling-rate GPS trajectory. It adds a temporal analysis on the basic HMM based algorithm. The basic idea of the temporal analysis is to take the speed limitations of the road segments into account.

Wei's algorithm[21] combined the advantages of the algorithms from Newson[18] and Lou[15] in the performances of different sampling rates, and the performance is robust against varying sampling rates.

Yuan[24] proposed an interactive-voting based map matching algorithm, which considers the mutual influence between GPS points by given influence weight values according to the distances between GPS trajectory points.

The HMM based map matching algorithm will be detailed subsequently since it is also used in the algorithms proposed in this thesis.

Table 2.1 shows the map matching data and road network used in each algorithm.

	Category	Dataset size	Data collection	Original sampling interval	Maximum sampling interval	Data source and coverage	Scenario	Device
Nagendra[20]	incremental topological	35km totally	planned			London, UK Washington, USA	Urban, Suburban, Rural	GPS
Yang[22]	incremental topological	721 points	random	30s		Harbin, China	Urban, Suburban, Rural	GPS in 500 taxis
Brakatsoulas[6]	global topological	45 trajectories / 4177 edges (less than 4177 points)		30s		Athens, Greece		GPS / Emphasis Telematics
Liu[14]	incremental topological	10 trajectories / 14436 GPS points		1s	30s	Washington, USA / Open Street Map		GPS
Wei[21]	global probabilistic				64s			
Griffin[10]	global topological	341 trajectories		1-2s		Wichita Falls,Texas		GPS
Mazhelis[17]	probabilistic	15658 points				Finland / Open Street Map	Urban, Suburban, Rural	GPS in Mobile phone
Newson[18]	global probabilistic	7531 points	planned	1s	600s	Washington,USA		GPS
Lou[15]	global probabilistic	28 trajectories	random	30s	300s	Beijing, China		GPS/GeoLife system

**Blank indicates the algorithm does not provide the corresponding information.*

Table 2.1: Map matching data and road network used in different algorithms

3 Hidden Markov Model Based Map Matching Algorithm

Hidden Markov Model (HMM) is a typical theory which can be used in probabilistic map matching algorithm, and several related algorithms have been published. In this chapter, the principle of HMM and the detailed HMM based map matching algorithm is discussed.

3.1 Introduction of Hidden Markov Model

A HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states. Figure 3.1 shows the general architecture of an instantiated HMM. Each oval shape represents a random variable that can adopt any of a number of values. The random variable $x(t)$ is the hidden state at time t (with the model from Figure 3.1, $x(t) \in \{x1, x2, x3\}$). The random variable $y(t)$ is the observation at time t (with $y(t) \in \{y1, y2, y3, y4\}$). The arrows in the diagram (often called a trellis diagram) denote conditional dependencies[4].

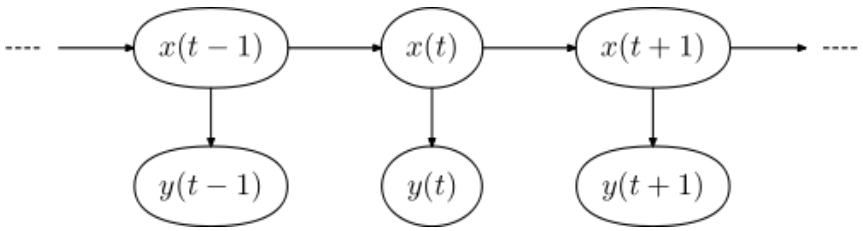


Figure 3.1: General architecture of an instantiated HMM[4]

From Figure 3.1, it is clear that the conditional probability distribution of the hidden variable $x(t)$ at time t, given the values of the hidden variable x at all times, depends only on the value of the hidden variable $x(t-1)$: the values at time $(t-2)$ and before have no influence. This is called the Markov property. Similarly, the value of the observed variable $y(t)$ only depends on the value of the hidden variable $x(t)$ (both at time t)[4].

3.2 Hidden Markov Model Based Map Matching

In probabilistic map matching, the basic work flow is firstly choosing candidates with specific conditions, then determining the result from the candidates. The HMM based map matching is to determine the map matching result in the frame of Hidden Markov Model definitions. In map matching category, the states of the HMM are the individual road segments, and the state measurements are the geographical locations measured by the GPS sensor, and the vehicle's movement along the road segments from point to point describes the states transition. There are several ways to evaluate the measurement possibilities and transition possibilities.

3.2.1 Candidates preparation

The candidate is defined as the projection point of the trajectory point on the road segment; if the projection point is not on the extend of the road segment, then the candidate will be set on the closest vertex of the road segment to the trajectory point. As one of the typical probabilistic map matching algorithms, preparing candidates is the prerequisite in HMM based algorithm. Theoretically speaking, for each trajectory point, every road segment has the possibility to be the true road segment that the vehicle was running on; thus the candidate number can be equal to the number of the road segments. But in practice, we know that the road segments whose distances to the trajectory point are larger than a value will have a very little possibilities to be the true matching result, and can be ignored. This value is called candidate searching radius, and one of the tasks in candidate preparation progress is to define the suitable value for the searching radius.

As shown in Figure 3.2, trajectory point P_t has 7 candidates that are located on 7 road segments if no candidate searching radius is set (assume that there are only these 7 road segments in the whole road network). It can be seen that for road segments R_3, R_4, R_7, the candidates C_t^3, C_t^4, C_t^7 are located on the vertexes. With the searching radius r_1, only can-

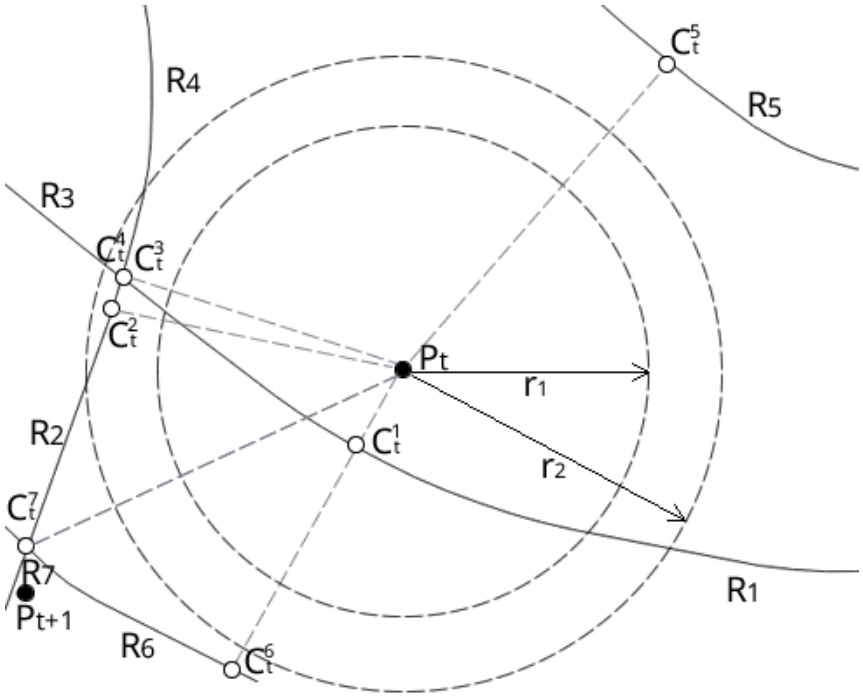


Figure 3.2: Searching radius and candidates

didate C_t^1 is chosen as the candidate that will participate in calculation, but with radius r_2, three more candidate R_2, R_3, R_4 will be added. The chosen of search radius has sufficient effect on the accurate and efficient performances of the map matching algorithm. We will discuss the selection procedure in the later chapter.

3.2.2 Measurement possibility

As we know, the location information obtained from the GPS devices contains errors, especially from the GPS devices that are built inside the vehicles, which are continuously moving and only have use a very short time to calculate the location information. The errors come from the GPS devices itself, the ephemeris and clock errors caused by the satellites, the atmospheric effects, and the multi-path effects caused by the environment at the point. Thus, the GPS errors are the composition of errors come from various sources, and vary with the changing of time and locations.

In probabilistic map matching, for each trajectory point P_t, every road segment R_i has a measurement possibility $p_t(C_t^i)$. This measurement possibility, which represents the GPS errors, is related to the circle distance from the trajectory point to the road segment (Figure

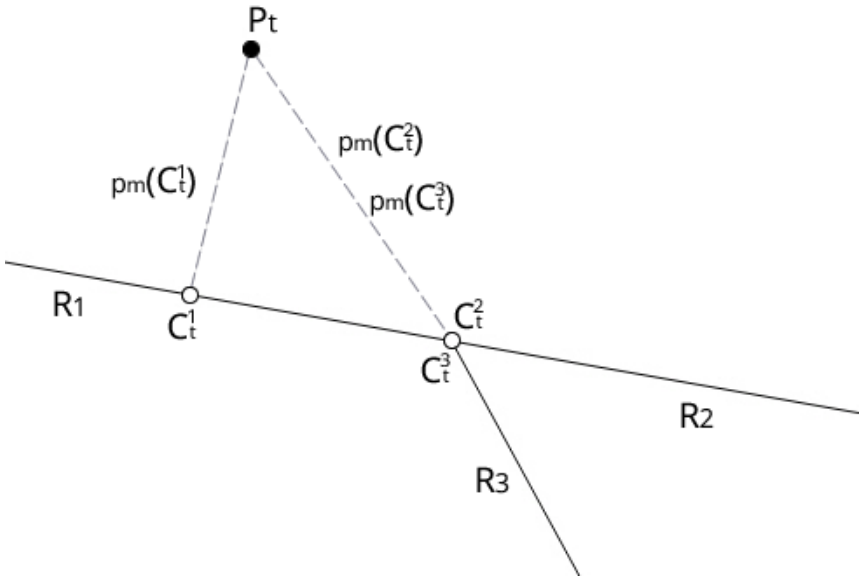


Figure 3.3: Measurement Possibility

3.3). Generally speaking, the closer the road segment to the trajectory point, the bigger the possibility for the road segment to be chosen as the map matching result (Figure 3.4). Base on VanDiggelen's research[7], the GPS errors can be modeled as zero-mean Gaussian, and the measurement possibility can be defined as:

$$p_m(C_t^i) = \frac{1}{\sqrt{2\pi}\sigma_p} e^{-0.5(\frac{|P_t - C_t^i|}{\sigma_p})^2} \tag{1}$$

where C_t^i is the projection of trajectory point P_t on road segment R_i, and $|P_t - C_t^i|$ is the circle distance between them. σ_p is the standard deviation of the GPS measurements. Because the model of the GPS device and the surrounding environment are unknown, thus the σ_p is an unknown value, but it can be evaluated by the previous measure results. We will discuss the evaluation of σ_p in the experimental chapter later.

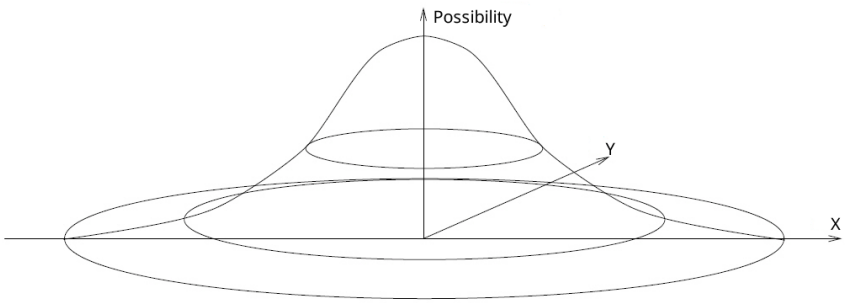


Figure 3.4: GPS measurement possibility distribution[6]

3.2.3 Transition possibility

In HMM, the transition possibility is the possibility of changing from one state to another one. In map matching, transition possibility means the possibility of a vehicle to choose to move to next road segment (or same segment) from current road segment between GPS sampling time. Various methods of defining transition possibility have been proposed by different researchers. Hummel[11] uses road topology, and Krumm[12] uses the route time differences, while other researches try to use the relationship between the route distance and circle distance to define the transition possibility. Here route means the trajectory of a vehicle moves along the road segments with restrains, such as one-way road and turning prohibition, etc. Through comparing the routes distance and circle distances in the ground truth data, Newson[18] pointed out that these two distances are about the same with the correct matching, because the relatively short distance traveled on the road(s) between a pair of correct matches will be about the same as the distance between the measured GPS points. This conclusion also fits the fact that drivers tend to choose the shortest route between two points in reality (there are also exceptions, such as the traffics, the road is temporary disabled, etc). Based on the ground truth data, Newson[18] created a histogram which indicated the difference between the circle and route distances , and they found that histogram fit well to an exponential probability distribution:

$$p_m(C_t^i \to C_{t+1}^j) = \frac{1}{\beta} e^{\frac{-d_t}{\beta}} \tag{2}$$

Where d_t is the difference between circle and route distances:

$$d_t = \left| |P_{t+1} - P_t|_{great\ circle} - |C_{t+1}^i - C_t^j|_{route} \right| \tag{3}$$

Lou[15] uses the ratio of the circle distance to the route distances to calculate the transition possibilities. The formula is defined as:

$$p(d_t) = \frac{|P_{t+1} - P_t|_{great\ circle}}{|C_{t+1}^i - C_t^j|_{route}} \tag{4}$$

Wei[21] uses the same definition of transition possibility as Newson[18], but simplified it to not consider the great circle distances, because they will all be counteracted in the comparison.

Through analyzing, we can see that in Newson and Wei's formulas, the smaller the difference between the circle and route distances, the larger the transition possibility; in Lou' formula, the smaller the route distance, the larger the transition possibility. In another

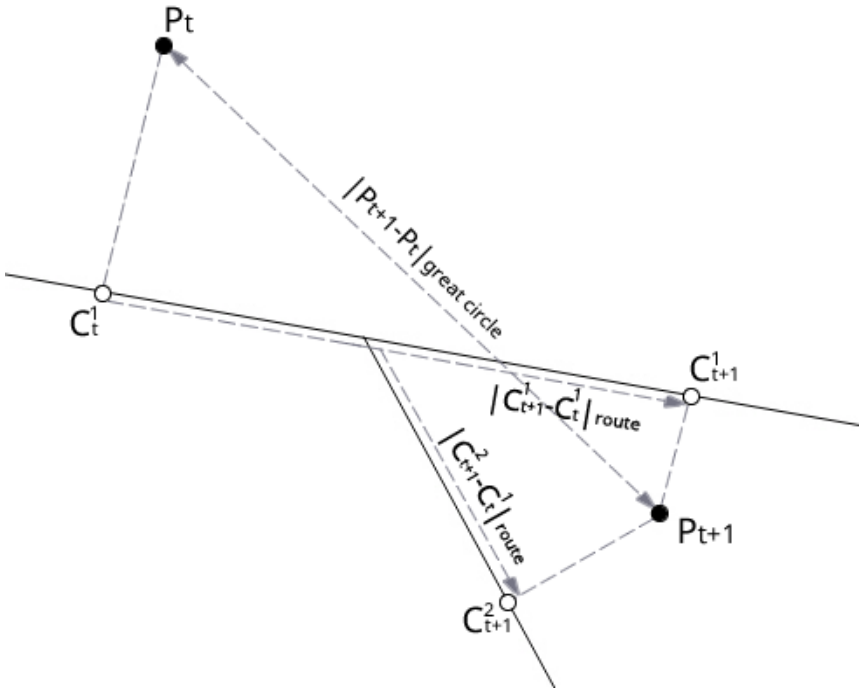


Figure 3.5: Great circle and route distance

word, Newson[18] tries to find the route most close to the circle distance, and Lou[15] tries to find the shortest route.

Besides, Lou[15] takes the speed limitation into account, and add another possibility called temporal analysis, which partly helps decide the true road segments, especially among the parallel ones with different speed limitations. In this thesis, due to the lack of speed limitation information, we do not consider this temporal factor when using Lou's algorithm.

3.2.4 Weights of measurement and transition possibilities

Measurement and transition possibility need to be combined together to get the ultimate possibility to evaluate the final output candidate sequence. It is a very important task to decide the weight of each element, which means how many role of each possibility plays in the result road segment decision.

Although HMM based map matching algorithms are not that sensitive to the influence of the uncertain choice of one point as incremental algorithm, which is the advantage of global algorithms, but the weight choosing of each possibility is still very important to the matching results. As shown in Figure 3.6, C_2^1 and C_2^3 are two candidates of P_2; C_2^1

has bigger measurement possibility, but smaller transition possibility than C_2^3, therefore, in different algorithms, or different weight parameters setting in the same algorithm, the matching result could be different.

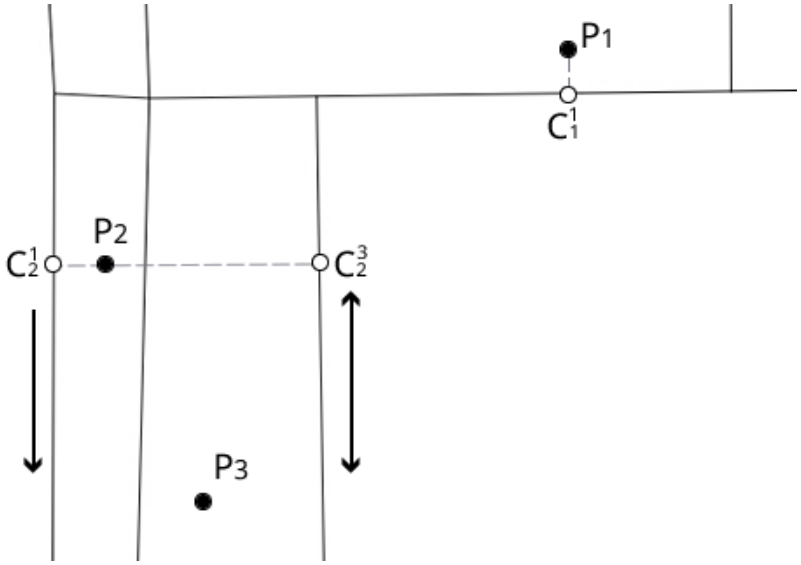


Figure 3.6: The wight of measurement and transition possibility

From the definitions of the measurement and transition possibility in the HMM algorithms from Newson[18], Lou[15] and Wei[21], we can see that they have used different strategies in the consideration of weights. All the algorithms think that the ultimate possibility for each trajectory point is the product of the measurement and transition possibility, and as global map matching algorithms, Newson[18] and Wei[21] use product to connect the possibilities of sequence, and Lou[15] use addition. Besides, Wei[21] considers the sampling intervals in the measurement possibilities, which means the bigger the sampling intervals, the smaller the measurement possibility.

To be more specific, Newson's map matching algorithm[18] is defined as:

$$\arg\max \prod_{t=0}^{n} \left(\frac{1}{\sqrt{2\pi}\sigma_p} e^{-0.5(\frac{|P_t - C_t^i|}{\sigma_p})^2} \frac{1}{\beta} e^{\frac{-d_t}{\beta}} \right) \tag{5}$$

and in the practical terms, the measurement and transition possibility are very likely to be very small values. To avoid the underflow in the result, the logarithm of Newson's formula is used:

$$\arg\max \sum_{t=0}^{n} \left(-0.5\big(\frac{|P_t - C_t^i|}{\sigma_p}\big)^2 - \frac{d_t}{\beta} \right) \tag{6}$$

Lou's algorithm[15] without considering temporal analysis is defined as:

$$\arg\max \sum_{t=0}^{n} \left(\frac{1}{\sqrt{2\pi}\sigma_p} e^{-0.5(\frac{|P_t - C_t^i|}{\sigma_p})^2} \frac{|P_{t+1} - P_t|_{great\ circle}}{|C_{t+1}^i - C_t^j|_{route}} \right) \tag{7}$$

The logarithm of Wei's algorithm[21] is defined as:

$$\arg\min \sum_{t=0}^{n} \left(t_i^2 |P_t - C_t^i|^2 + \frac{2\sigma_p^2}{\beta} |C_{t+1}^i - C_t^j|_{route}^2 \right) \tag{8}$$

where t_i is the time difference between points P_t and P_{t+1}.

3.2.5 Solving HMM via Viterbi algorithm

The Viterbi algorithm is a recursive optimal solution to the problem of estimating the state of a discrete-time finite-state Markov process observed in memoryless noise[8].

In HMM based map matching, the goal is to find the most likely candidates sequence through the measurement and transition possibility, or with other possibility if used. The initial possibilities for the candidates of first point are only the measurement possibilities. From the second points of each trajectory, the ultimate possibility will be the product of the measurement and transition possibility. During the procedure, the candidates which lead to the largest possibility values of the candidates of next successive trajectory points are recorded as parents. In the end of the calculation, the candidate with largest possibility of the last trajectory point will be chosen as the map matching result, and then trace back to find the candidate of previous trajectory point that lead to this largest possibility, and this candidate will be chosen as the map matching result of the second to last trajectory point. The rest will be done in the same manner, until reaching to the first trajectory point. The algorithm of using Viterbi algorithm to find the most likely candidate sequence is shown in Algorithm 1.

Algorithm 1: Finding most likely road sequence via Viterbi

Input: N_T {number of trajectory points}
$N(P_t)$ {candidate number of trajectory point P_t}
$p_m(C_t^i)$ {measurement possibility of candidate C_t^i}
$p_t(C_{t-1}^i \to C_t^j)$ {transition possibility between candidates C_{t-1}^i, C_t^j}

Output: X {Most likely candidate sequence list};

{Initialization}

1: **for** $i = 1$ **to** $N(P_1)$ **do**

$p(C_1^i) \leftarrow p_m(C_1^i)$;
$X.add(0)$;
end for
{Recursion}
for $i = 1$ **to** N_T **do**
  for $j = 1$ **to** $N(P_t)$ **do**
    $temp \leftarrow 0$;
    $s\_id \leftarrow 0$;
    for $k = 1$ **to** $N(P_{t-1})$ **do**
      $p_{temp} \leftarrow p(C_{t-1}^k) \times p_t(C_{t-1}^k \rightarrow C_t^j)$;
      if $p_{temp} > temp$ **then**
        $temp \leftarrow p_{temp}$;
        $s\_id \leftarrow k$;
      end if
    end for
    $p(C_m^j) \leftarrow temp \times p_m(C_m^j)$;
    $X.add(s\_id)$;
  end for
end for
return X;

Here we use Newson's HMM based algorithm to explain the principle of Viterbi algorithm. As shown in Figure 3.7, we define a trajectory with only 3 GPS points, and each of the three successive GPS points has several candidates within the searching radius; each candidate has a measurement possibility based on the circle distance to the GPS point; Each pair of candidates of two GPS points has a transition possibility based on the shortest path between each other. The measurement and transition possibilities are listed in Table 3.1 (the sum of the possibilities are not necessarily equal to 1). It can be seen that C_3^1 has the largest possibility value, then look back to check the parents that lead to this possibility value. The parent of C_3^1 is C_2^1, and the parent of C_2^1 is C_1^1, and finally the whole candidate sequence is determined: $C_1^1 \rightarrow C_2^1 \rightarrow C_3^1$.

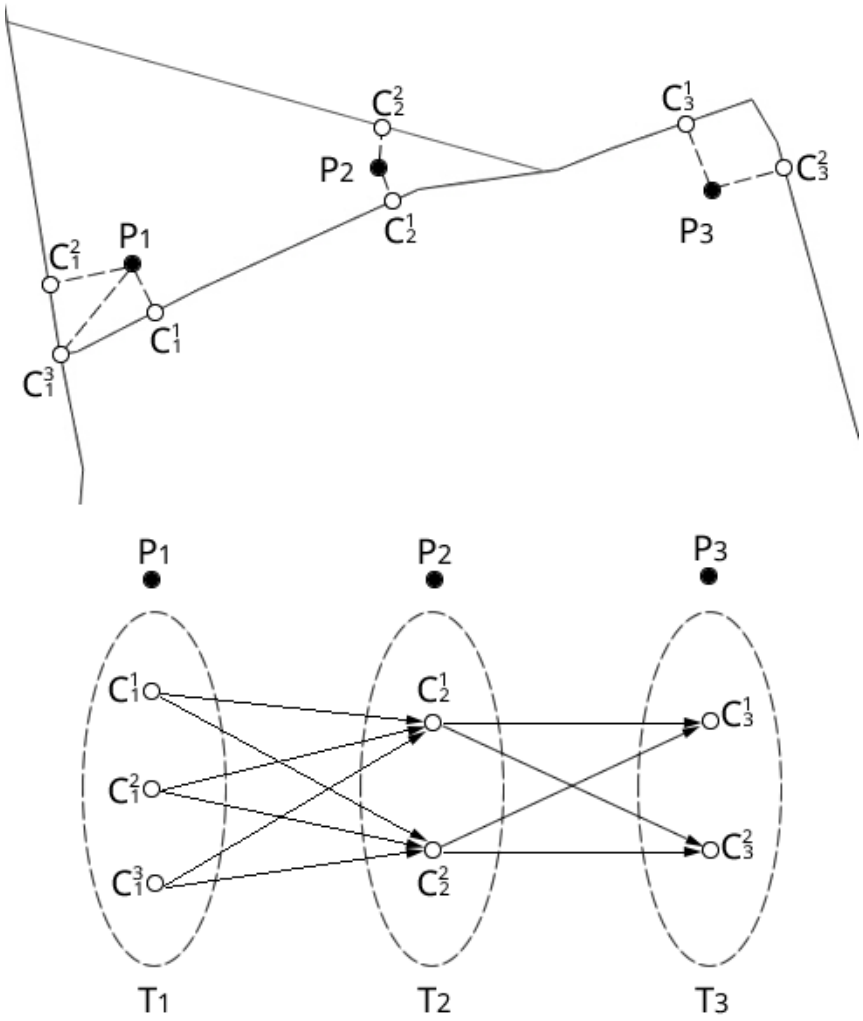


Figure 3.7: Example of finding most likely candidate sequence

(a) Measurement possibility

C_1^1	0.80	C_2^1	0.92	C_3^1	0.75
C_1^2	0.60	C_2^2	0.90	C_3^2	0.74
C_1^3	0.30				

(b) Transition possibility

	C_2^1	C_2^2
C_1^1	0.90	0.70
C_1^2	0.60	0.40
C_1^3	0.70	0.50

	C_3^1	C_3^2
C_2^1	0.80	0.60
C_2^2	0.90	0.58

Table 3.1: Measurement and transition possibility

$\boldsymbol{P_1}$ **:**

$$p(C_1^1) = p_m(C_1^1) = 0.80$$

$$p(C_1^2) = p_m(C_1^2) = 0.60$$

$$p(C_1^3) = p_m(C_1^3) = 0.30$$

$\boldsymbol{P_2}$ **:**

$$p(C_2^1) = max(\underbrace{0.80 \times 0.92 \times 0.90}_{C_1^1 \to C_2^1 (parent)}, \underbrace{0.60 \times 0.92 \times 0.60}_{C_1^2 \to C_2^1}, \underbrace{0.30 \times 0.92 \times 0.70}_{C_1^3 \to C_2^1}) = 0.6624$$

$$p(C_2^2) = max(\underbrace{0.80 \times 0.90 \times 0.70}_{C_1^1 \to C_2^2 (parent)}, \underbrace{0.60 \times 0.90 \times 0.40}_{C_1^2 \to C_2^2}, \underbrace{0.30 \times 0.90 \times 0.50}_{C_1^3 \to C_2^2}) = 0.504$$

$\boldsymbol{P_3}$ **:**

$$p(C_3^1) = max(\underbrace{0.6624 \times 0.75 \times 0.80}_{C_2^1 \to C_3^1}, \underbrace{0.504 \times 0.75 \times 0.90}_{C_2^2 \to C_3^1}) = 0.39744$$

$$p(C_3^2) = max(\underbrace{0.6624 \times 0.74 \times 0.60}_{C_2^1 \to C_3^2}, \underbrace{0.504 \times 0.74 \times 0.58}_{C_2^2 \to C_3^2}) = 0.2941$$

4 Fusion Map Matching Algorithm

In general, because the global algorithms take all the trajectory points into account, they can achieve better accuracy than the incremental ones. Accordingly, global algorithms have higher computation complexities, thus the efficiency are worse than incremental algorithms. The FCD, which is collected to analyze the traffic model and the passenger's behaviors in ITS, normally includes tens of millions data. To map matching such large amount of data, the efficiency is also very important. Therefore, we propose two new map matching algorithms that have better performance than the current algorithms, which suit for large scale map matching project.

4.1 Principle of Fusion Map Matching Algorithm

From the principle of incremental algorithm, it can be seen that the map matching result of the current trajectory point has significant influence on the matching result of the next trajectory point, because part of the weight values are based on the current result. Therefore, the beginning of each trajectory is very important for the matching accuracy of the whole trajectory. But unfortunately in practice, the first point is the most fragile part of the trajectory, because for that point, no previous point can be used as reference, and the only criterion is the circle distance to the candidate points (measurement possibility). This causes a much bigger chance to get incorrect matching result for the first point of each trajectory, and then lead the following points also get wrong map matching results. Besides, during the precess of incremental algorithm calculation, once the trajectory is already been matched to the wrong road segment, it will be very difficult for the map matching result to go back to the correct road segment, especially when the transition possibility is the major factor of the whole possibility, and this will also lead a large range of map matching mistakes. These are the reasons that the current incremental map matching algorithms show not ideal accuracy performance.

To partly overcome these disadvantages of incremental algorithm and improve the per-

formance of the incremental algorithms , we propose a Fusion Map Matching algorithm (FMM) in this thesis, which synthetically combines the global and incremental map matching algorithms together.

The characteristic of "fusion", which means the alterations between the global and incremental algorithms, are occurred in two scenarios. The first alterations are occurred at the beginning of each trajectory, where the global map matching algorithm is firstly used to get more confident matching results for the first few points, and then start the incremental algorithm, which uses the result of the last previous point to calculate the weight values of candidates for next point. The second alternations are happened during the running of incremental algorithm. In this scenario, the shortest route between the last matching point and all the current candidates will be calculated. If the lengths of all the shortest routes are lager than a reasonable value, which is related to longest distance a vehicle can run in the sampling time with the fastest speed, the algorithm will enlarge the candidates search radius to get more candidates. If the new candidates still can not fit the condition, then the algorithm will judge that the correctness of the last matching result can not be fully determined, and will make a break here. Then the algorithm will start to map matching the next few trajectory points with global map matching algorithm, and then goes to incremental algorithm. The whole algorithm runs like this until it reaches the end of the trajectory.

The work flow of FMM algorithm is shown in Figure 4.1 and Figure 4.2.

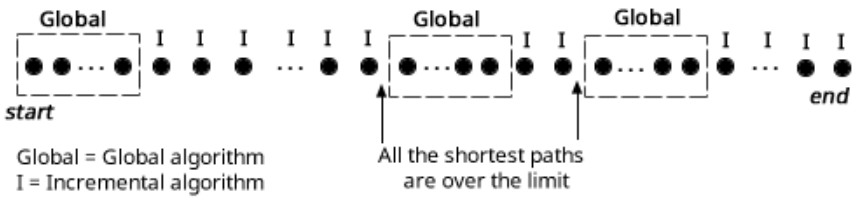


Figure 4.1: The alternations of global and incremental algorithms in FMM algorithm

4.1.1 Candidate search radius

As mentioned in section 3.2.1, the selection of the search radius has significant influence to the performance of the map matching algorithm. In most of exist algorithms, the search radius is defined either by a fixed maximum value (normally from 50m to 200m), or by the

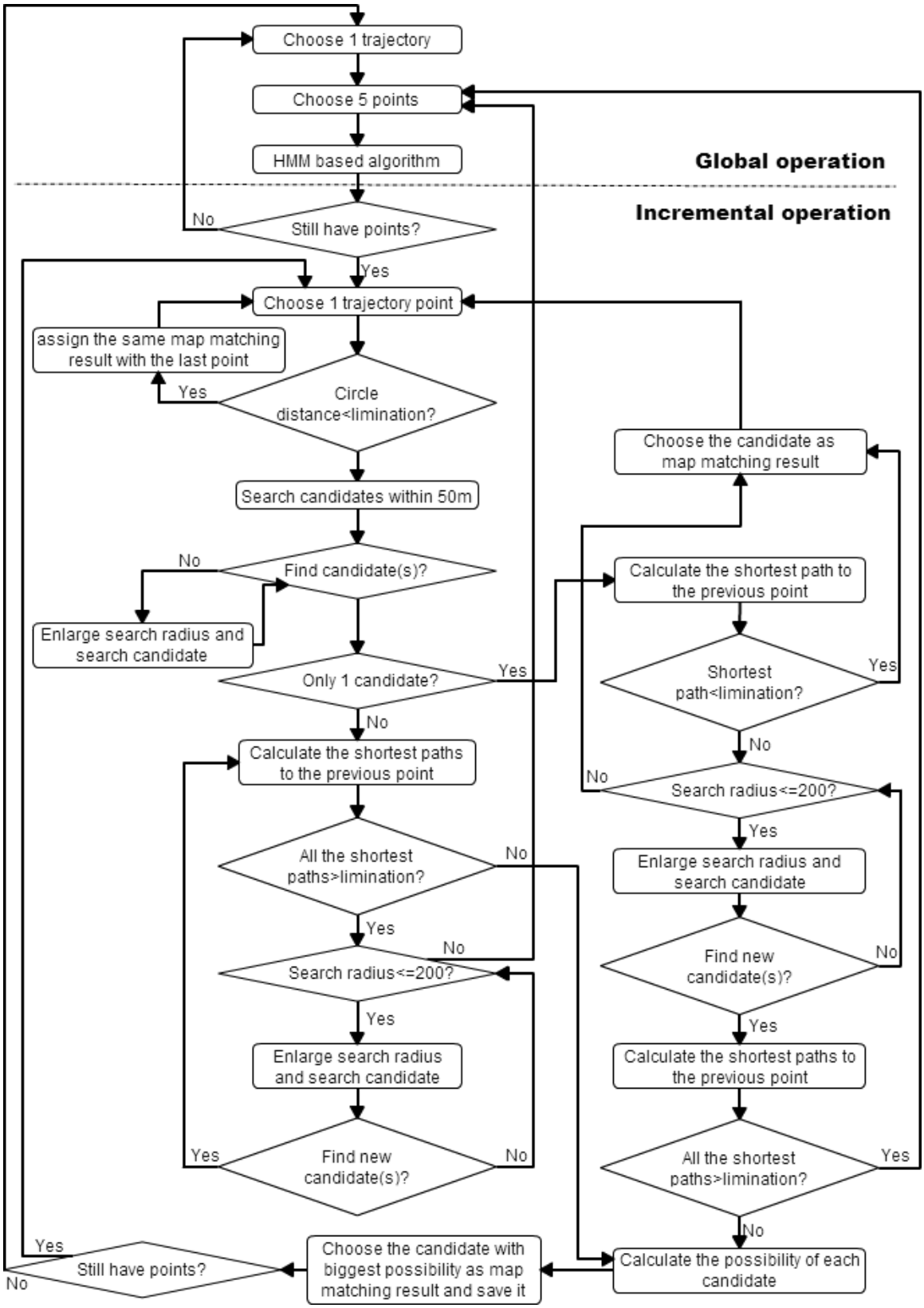


Figure 4.2: The work flow of FMM algorithm

maximum number of candidates, or both. This kind of definition is direct and simple, but has some potential problems: if the search radius or the maximum number of candidates are defined too small, it is possible that no candidate is found or the true road segment is not selected as candidate, and the later situation could occurs more often especially in the urban area, where the road density is big; oppositely, if the two values are defined too big, irrelevant road segments may be chosen as candidates, and this will effect the efficiency of the algorithm.

In the FMM algorithm, a flexible search radius is proposed, which means the search radius will change according to different situations. To be more specific, the initial search radius is defined as 50m, and if there is no candidate found, then enlarge the value by 50m, until at least one candidate is found. Besides, the maximum distance for a vehicle is able to travel in a sampling interval can be calculated by estimating a max speed of a vehicle. Then the algorithm checks the shortest routes between the previous matching result and the candidates, if all the shortest routes are bigger than the maximum distance and the search radius is smaller than 200m, then enlarge the search radius by 50m, until at least one candidate that can fit this requirement is found or altering to the global algorithm.

4.1.2 Global calculation

As discussed in Chapter 3, HMM is a typical theory used in global probabilistic based map matching algorithm, and several HMM based map matching algorithms have been proposed. Therefore, in our FMM algorithm, we also choose an HMM based algorithm in the global calculation part. More specifically, we propose a new HMM based algorithm based on Lou's[15] algorithm, and make several modifications on it.

Firstly, we modify the relationship between the possibilities of successive trajectory points from addition to multiply, because we think multiply fits the HMM model better, and has better accuracy performance.

Secondly, we modify the definition of shortest route. In our definition, the shortest route is not only consider the route distance shown in Figure 3.5, but also includes the circle distance between the trajectory points to the candidates road segments. This means we also take the distance to the road segment as a factor in the transition possibility (Figure 4.3), and the larger of this distance, the smaller the transition possibility.

In addition, we introduce the azimuth possibility to improve the performance of the algo-

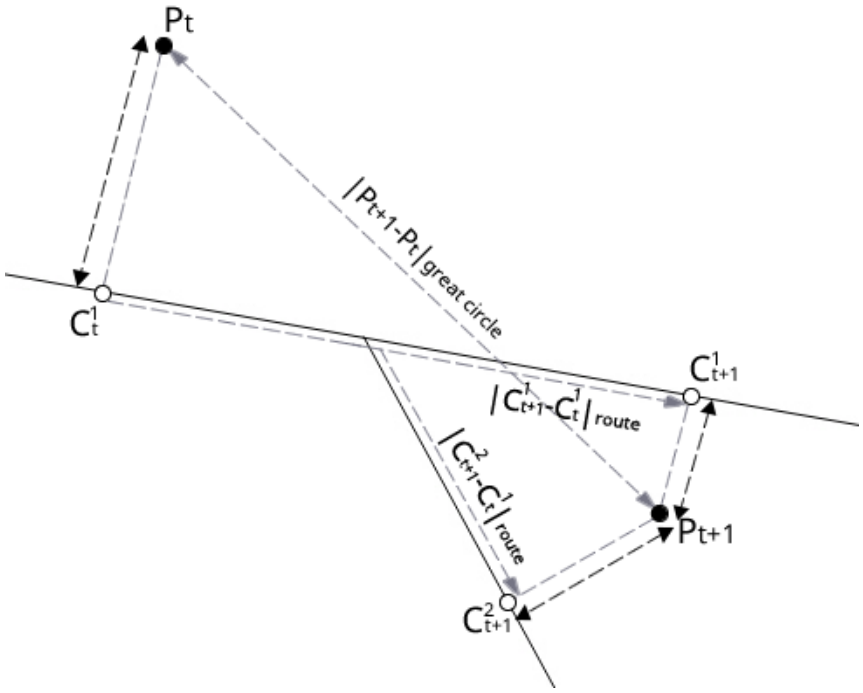


Figure 4.3: A shortest route with circle distance between the trajectory points and road segments

rithm. An azimuth is an angular measurement in a spherical coordinate system[1]. The azimuth possibility is used to describe the azimuthal difference between the trajectory point (driving direction) and road segment. The smaller the difference, the bigger the azimuth possibility. The formula of azimuth possibility is defined as:

$$p_a(C_t^i) = \cos^5 \theta \tag{9}$$

Where θ is the azimuth difference between the driving direction and road segment, and an uneven number of the power value of the $\cos\theta$ means that this azimuth possibility can get negative value when the difference of the azimuth is in the region of $(\frac{\pi}{2}, \frac{3\pi}{2})$.

The direction of the vehicle can be directly read from the GPS log, and the direction of the road segment at the candidate point is the direction of the tangent line in the point, but this tangent line can not be got directly. In this thesis, we design a method to approximately calculate the direction of the tangent line (road segment). As shown in the Figure 4.4, we create a point on the same road segment with the candidate point, and the distance between the two point is very short, and in the thesis, this distance is set to $0.001 * length(R_i)$. Then we calculate the azimuth value of the line that connect these two points and follow the direction of the road segment, and this azimuth value can approximately be seen as

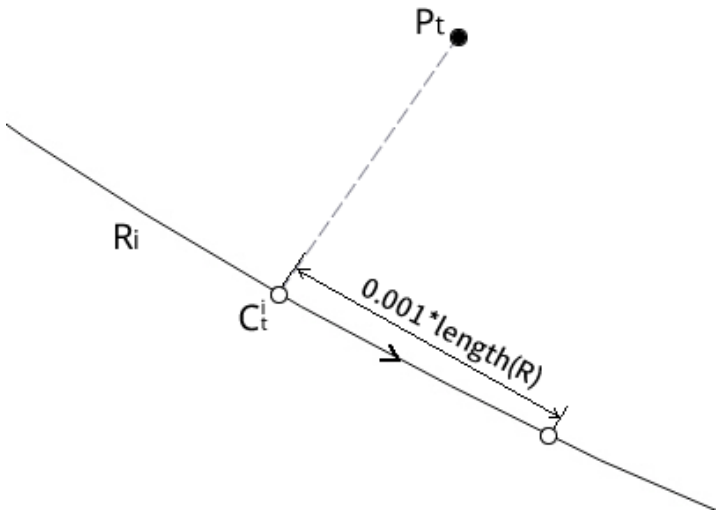


Figure 4.4: Calculation of the azimuth of the road segment

the azimuth of the road segment at the location of the candidate. It should be noted that the two-way road segments have two azimuth which are 180 difference with each other, and in this situation, the direction that can get bigger azimuth possibility (more similar to the driving direction) will be chosen.

Therefore, the original formula of our HMM based algorithm is defined as:

$$\arg\max \prod_{t=0}^{n} \left(\frac{1}{\sqrt{2\pi}\sigma_p} e^{-0.5(\frac{|P_t - C_t^i|}{\sigma_p})^2} \frac{|P_{t+1} - P_t|_{great\ circle}}{|C_{t+1}^i - C_t^j|_{route}} \cos^5\theta \right) \tag{10}$$

To avoid the underflow problem, we also need to transfer the original algorithm to a logarithm type. In addition, because the azimuth possibility have negative value, the product of the azimuth possibilities values may get opposite result, and damage the comparison result of the whole ultimate possibilities. Therefore, the azimuth possibility can not be simply conversed without prejudice. In this thesis, we define the logarithm formula of azimuth possibility as:

$$p_a(C_t^i) = -10 \times \ln(\cos^5\theta + 1) \tag{11}$$

and the logarithm of our HMM based algorithm is then defined as:

$$\arg\min \sum_{t=0}^{n} \left(0.5\big(\frac{|P_t - C_t^i|}{\sigma_p}\big)^2 - log\big(\frac{|P_{t+1} - P_t|_{great\ circle}}{|C_{t+1}^i - C_t^j|_{route}}\big) - 10 \times \ln(\cos^5\theta + 1) \right) \tag{12}$$

4.1.3 Incremental calculation

In FMM, the incremental calculation uses the same definitions as our HMM based algorithm for measurement, transition and azimuth possibility. The incremental algorithm is designed to find the candidate that has the maximum ultimate possibility among all the candidates of the same trajectory point:

$$max_{(i=0\ to\ n)}\left(\frac{1}{\sqrt{2\pi}\sigma_p}e^{-0.5(\frac{|P_t-C_t^i|}{\sigma_p})^2}\frac{|P_{t+1}-P_t|_{great\ circle}}{|C_{t+1}^i-C_t^j|_{route}}\cos^5\theta\right) \tag{13}$$

4.2 Forward-looking Incremental Algorithm

Compared with global algorithms, incremental algorithms only consider the matching result of the previous points, thus it is easy to have matching mistakes in the Y-split area. As shown in Figure 4.5, here assume that road segments R_1, R_2, R_3 are all one-way road. For human it is very easy to label the trajectory to road segments R_1 and R_2, but in incremental algorithm, point P_6 is much closer to R_3 than R_2, then the measurement possibility is larger for the candidate on R_3, and the transition possibilities for candidates on R_2 and R_3 are similar, thus the incremental algorithm will choose the candidate R_3 as the matching results. Because R_3 is one-way road, there will have a big possibility for the following points all be matched on R_3 based on the matching result of P_6.

To partly solve this kind of problem, we propose a forward-looking incremental algorithm. As shown in Figure 4.6, P_1, P_2, P_3 are three successive trajectory points, and C_1^1 already be chosen as the matching result of P_1, and now need to find the best candidate for P_2. Compared with normal incremental algorithm, the forward-looking incremental algorithm takes the subsequent point(s) into consideration. P_2 has two candidates C_2^1, C_2^2 located on R_3, R_2 respectively. If C_2^1 is the matching result, then based on this point to calculate the weight values of the three candidates of P_3, and C_3^2 has the biggest possibility. Similarly, if C_2^2 is the matching result, based on this point it is also able to find one candidate of P_3 that get the biggest possibility, C_3^1 for example. Then the final possibilities of C_2^1 and C_2^2 are $p(C_2^1) \times p(C_3^2)$ and $p(C_2^2) \times p(C_3^1)$, and the one with bigger possibility will be chosen as the map matching result of P_2. This algorithm can also consider more than one subsequent points, but the computation time will also increase accordingly. The formula of forward-looking incremental algorithm is defined as:

$$p(C_t^i) = \arg\max \prod_{d=0}^{n} p(C_{t+d}^j) \tag{14}$$

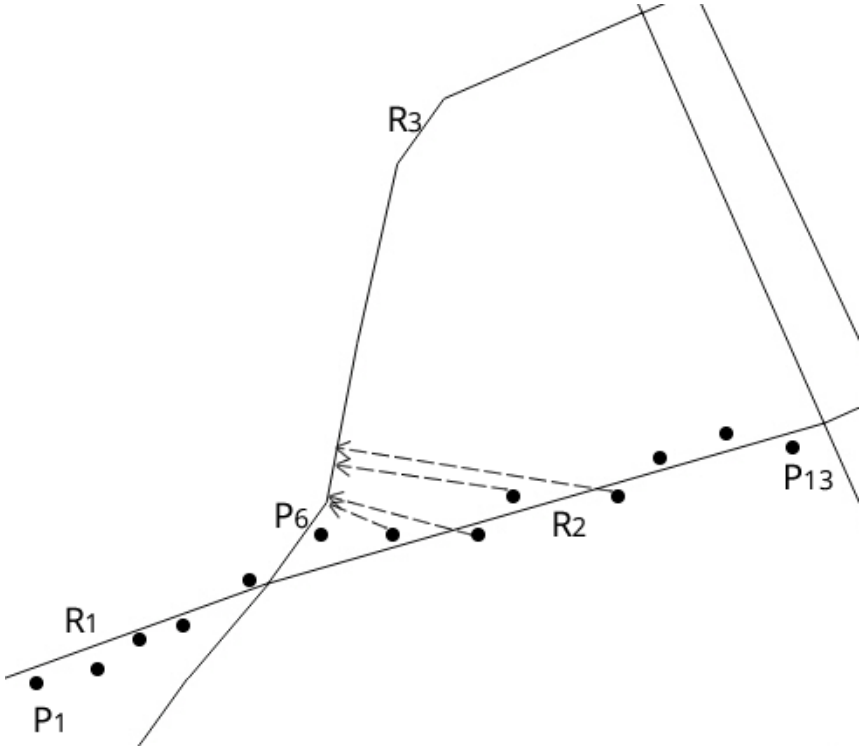


Figure 4.5: Map matching errors on Y-split road segments

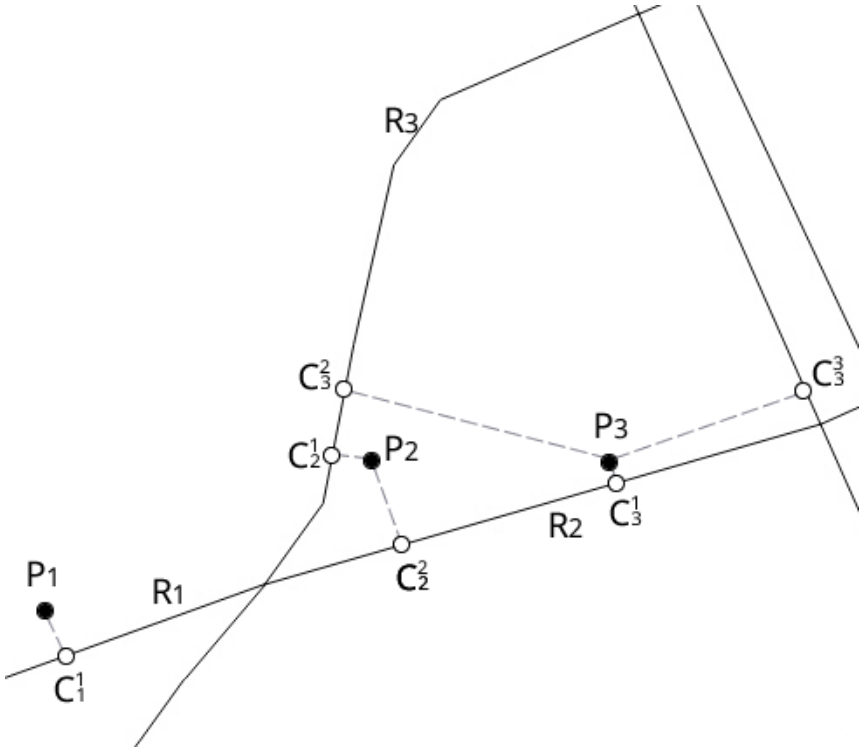


Figure 4.6: Forward-Looking FMM algorithm

Where n is the maximum number of forward-looking points.

Although forward-looking incremental algorithm dose not have such serious underflow problem like HMM based algorithm due to its smaller computation complexity, but to drastically avoid the influence of the underflow problem to the map matching result, just like what has been done in the transformation of the HMM based algorithms, the formula of forward-looking incremental algorithm is also be transformed to a logarithm type. The logarithm form forward-looking incremental algorithm is defined as:

$$p(C_t^i) = \arg\min \sum_{d=0}^{n} \left(0.5\big(\frac{|P_{t+d} - C_{t+d}^i|}{\sigma_p}\big)^2 - log\big(\frac{|P_{t+d+1} - P_{t+d}|_{great\ circle}}{|C_{t+d+1}^i - C_{t+d}^j|_{route}}\big) - 10 \times \ln(\cos^5\theta + 1) \right) \tag{15}$$

4.3 Handling Special Cases

To improve the efficiency and the accuracy of the map matching, several special situations are considered in the FMM algorithms.

4.3.1 Only one candidate found

If there is only one candidate be found within the searching radius, and the shortest route to the previous point is less than the maximum allowed distance, this candidate point will be directly chosen as the matching result.

4.3.2 Successive points within a distance

Due to the errors of GPS, when the vehicle stops for a while or moves in a very slow speed, the locations recorded by the GPS are normally not same, but very close to each other. The distances between the points depend on the precision of the GPS device, the surrounding environment of the vehicle, and the distribution of the GPS satellites at that time. These points are recorded at the same or almost same location, thus the map matching results for these points should be same, which means after using map matching algorithm to calculate the first point of this kind of points group, the matching results for other points can be directly assigned to the same road segment with the first one, and this procedure can save calculation time.

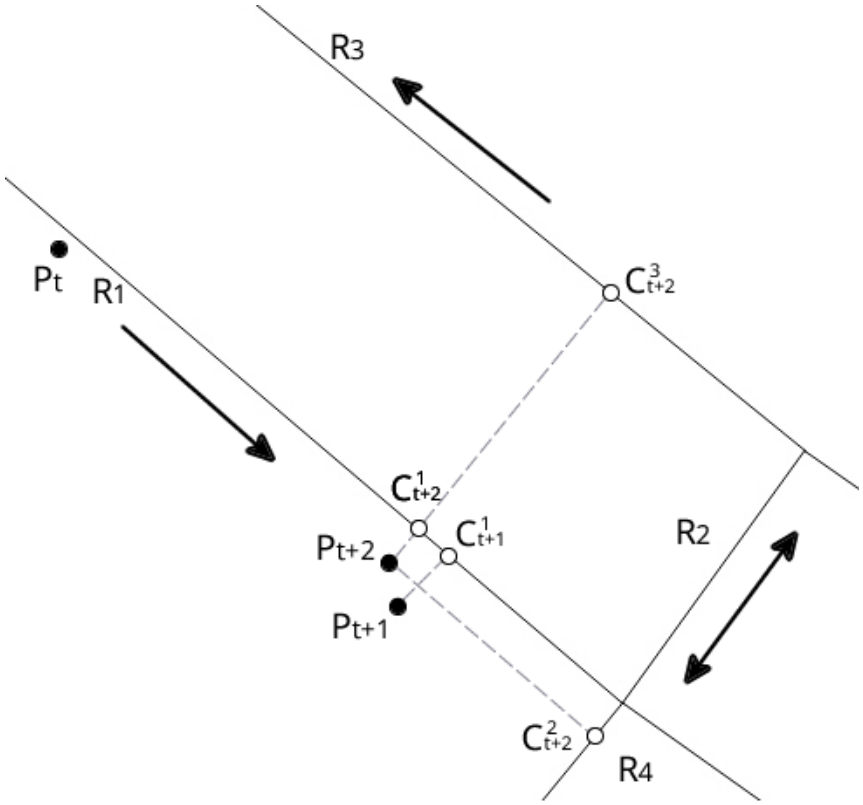


Figure 4.7: Close points lead to matching errors

Besides improving the efficiency, this procedure can also avoid some matching errors. As shown in Figure 4.7, P_t, P_{t+1}, P_{t+2} are three successive trajectory points, and it can be easily labeled that the vehicle was driving along road segment R_1 by human, and P_t, P_{t+1}, P_{t+2} should be matched to R_1. Because of the GPS error, the location of P_{t+2} was recorded behind of P_{t+1}. $C^1_{t+2}, C^2_{t+2}, C^3_{t+2}$ are three candidates of P_{t+2}, which locate on road segment R_1, R_3, R_4 respectively. C^1_{t+1} and C^1_{t+2} should be the correct matching results for P_{t+1} and P_{t+2}, which are both on R_1. R_1 is a single direction road, thus after C^1_{t+1} is assigned to the matching result for P_{t+1}, when try to use the FMM algorithm to get the matching result for P_{t+2}, the shortest route between C^1_{t+1} and C^1_{t+2} has to pass through R_1, R_2, R_2, and clearly C^1_{t+2} will be out of the selection due to small transition possibility, and the matching result for P_{t+2} will become C^3_{t+2} on R_3, which is not correct. This mistake will not only affect point P_{t+2}, but also possibly affect the matching result of next point that is based on this wrong matching result of P_{t+2}. In FMM algorithm, two successive trajectory points within ω meters will be assigned to same road segment, and we will discuss the value of ω in the later chapter.

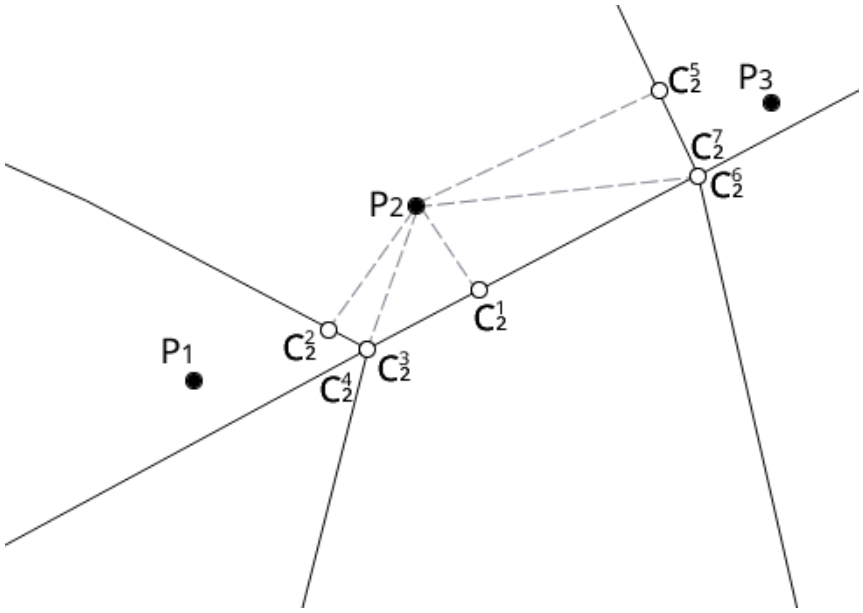


Figure 4.8: Candidates on the vertexes of the road segment

4.3.3 Candidates on the vertexes of the road segment

As shown in Figure 4.8, point P_2 has 7 candidates within the searching radius, and $C_t^3, C_t^4, C_t^6, C_t^7$ locate on the vertexes of the road segments. It is clearly that these four candidates are impossible to be the correct results. Generally speaking, there are only two scenarios that the true matching results could locate on the vertexes of the road segments:

1. The trajectory point was recorded just on the vertex of a road segments;
2. The trajectory likes Figure 4.9, and in this scenario the candidate C_t^3 on the vertex should be the correct map matching result.

In practice, the possibility is very small to meet the above situations. In our test data which includes 24482 trajectory points, only 86 points (0.35 %) meet the second scenario, and the ground truth result are located on the vertexes; none of trajectory point locates on the vertex. After checking the candidates distribution of training data, which is limited to at most 5 candidates of each trajectory point, we found that 47257 of 108426(43.58%) candidates are located on the vertexes of the road segments, which means over 40% of candidates are located on the vertexes, but only less than 0.2% of them are correct results. But these candidates need a lot of calculation time, especially in HMM algorithm, which is more sensitive to the number of candidates.

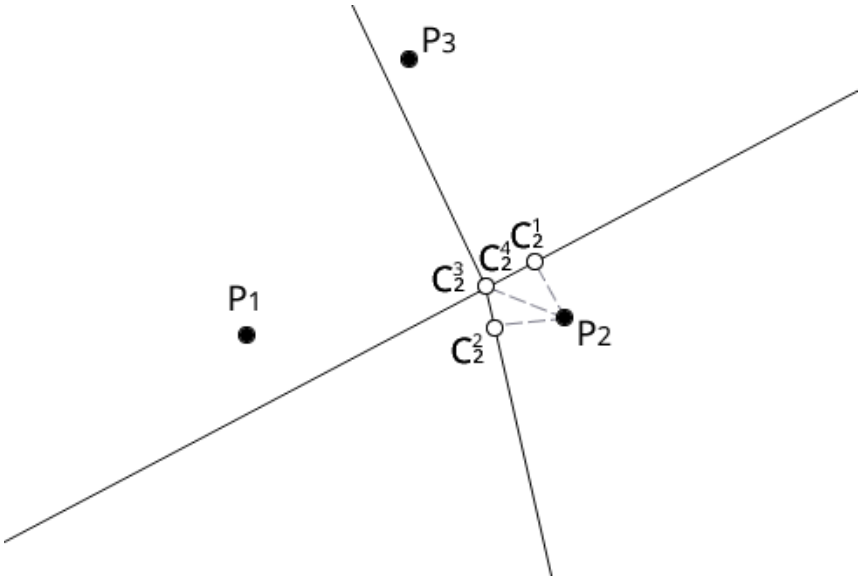


Figure 4.9: Matching result on the vertex of the road segment

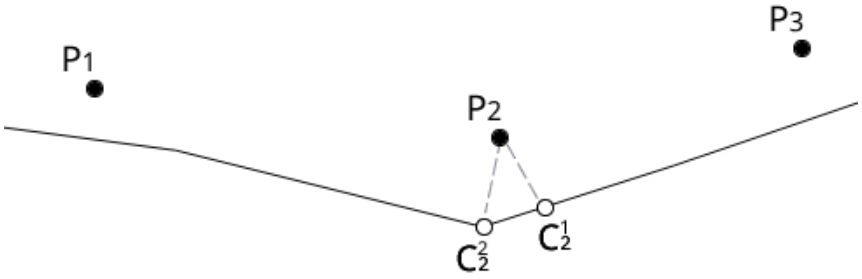


Figure 4.10: Incorrect choosing candidates on the vertex

Besides, delete the candidates on the vertexes will not only improve the efficiency, but also the accuracy. With the current FMM/HMM algorithm, sometimes the candidates on vertexes get bigger possibility values than other candidates, and as we mentioned above, most of them are not the correct matching results. As shown in Figure 4.10, trajectory point P_2 has two candidates, and C_2^2 is located on the vertex; after checking the driving direction, we know that C_2^1 should be the correct matching result, but C_2^2 has bigger transition possibility because the shortest routes is almost equal to the circle distance between P_1 and P_2, and the map matching algorithm may choose C_2^2 as the matching result with some settings of weights.

Take the FMM algorithm as example, the accuracy of considering and without considering candidates on vertexes are shown on Figure 4.11. It can be seen that without considering

the candidates on the vertexes, there are over 1% accuracy improvement. Therefore, in our FMM algorithm, we consider the candidates on the vertexes only if the trajectory point itself is located on the vertex.

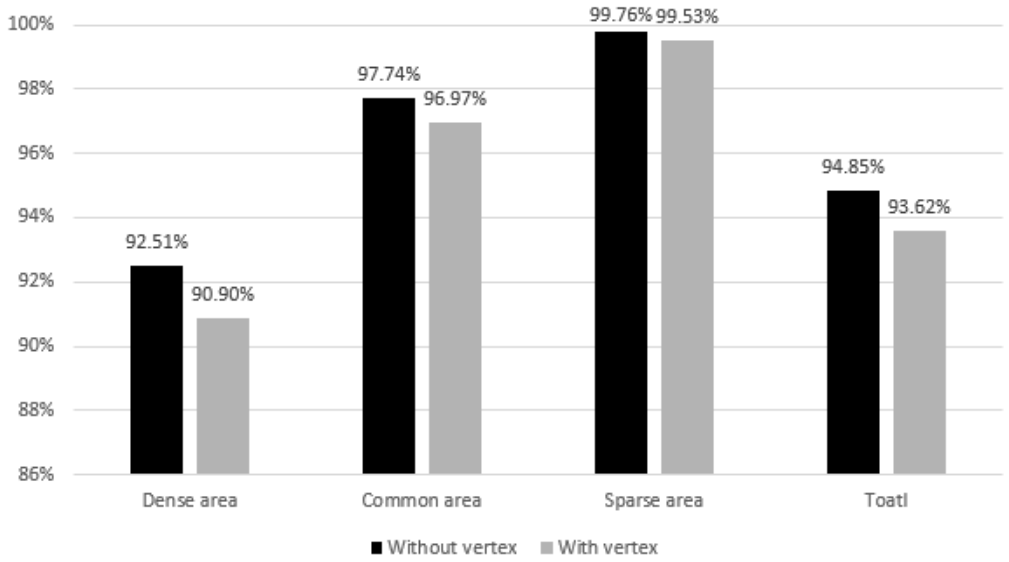


Figure 4.11: Accuracy performances of map matching algorithm of with and without considering the candidates located on vertexes

5 Calculating Shortest Route

5.1 Principles of Shortest Route Calculation

The shortest route calculation is a significant operation of computing the transition possibility. In the FMM algorithm, the shortest route calculation is carried out via an extension called pgRouting in PostgreSQL. pgRouting can be used to analysis the road network and calculate the shortest route between two vertexes of roads, and it contains a series of routing related functions. In this thesis, the core function of shortest route calculation is pgr_kdijkstraPath. To use this routing function, several preparation works need to be done firstly.

pgr_nodeNetwork: The function reads edges from a not ”noded” network table and writes the ”noded” edges into a new table. Two attributes called ”source” and ”target”, which indicate the vertexes and the direction of one-way road, are added into the road table. As shown in the Figure 5.1, in one way-road, the road direction always follows the order from ”source” to ”target”, and two-way road does not follow this order. Besides, an additional table called ”vertives_pgr” that saves all the information of vertexes will be used later.

pgr_createTopology : The function builds a network topology based on the geometry information.

pgr_kdijkstraPath: The function returns the paths for K shortest paths using Dijkstra algorithm.

With pgRouting, only the shortest routes between the vertexes of the roads can be calculated; but in most of the cases, the candidates do not fall on the vertexes. Thus in this thesis, a new function is created to calculated the lengths of the shortest routes between any two candidates located on the arbitrary positions of the road segments. The functions of pgRouting used in our function is listed below:

ST_LineLocatePoint: Returns a float between 0 and 1 representing the location of the clos-

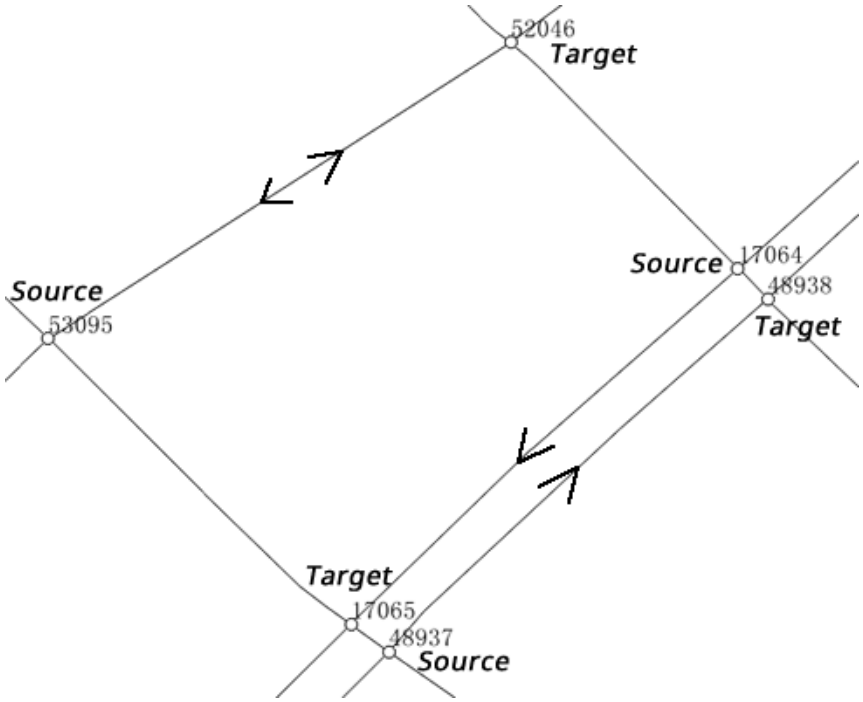


Figure 5.1: Source and target vertexes of road segments

est point on LineString to the given Point.

ST_Line_SubString: Return a linestring being a substring of the input one starting and ending at the given fractions of total 2d length.

ST_Length: Returns the 2d length of the geometry if it is a linestring or multilinestring.

ST_Linemerge: Returns a (set of) LineString(s) formed by sewing together a multilinestring.

ST_MakeLine: Creates a Linestring from point or line geometries.

ST_Buffer: Returns a geometry that represents all points whose distance from this Geometry is less than or equal to distance. Calculations are in the Spatial Reference System of this Geometry.

8 parameters used in this shortest route calculation function are listed in Table 5.1.

There are several scenarios when dealing with shortest route calculation, and we use different strategies to deal with these scenarios in our shortest route calculation function.

1. C_t^i and C_{t+1}^j locate on the same road segment;
2. C_t^i and C_{t+1}^j locate on two connected road segments;

3. C_t^i and C_{t+1}^j locate on two non-connected road segments.

Parameter	Description
start line	the geometry of the line where C_t^i located
end line	the geometry of the line where C_{t+1}^j located
start point	the geometry of C_t^i
end point	the geometry of C_{t+1}^j
start target	the "target" value of the road segment where C_t^i located
end source	the "source" value of the road segment where C_{t+1}^j located
cost file	the name of the attribution of cost
reverse cost file	the name of the attribution of reverse cost

C_t^i and C_{t+1}^j are the candidates of two successive points respectively.

Table 5.1: Parameters in shortest route calculation

As shown in Figure 5.2, $C_t^1, C_{t+1}^1, C_{t+2}^1, C_{t+3}^1$ are the candidates of trajectory points P_t, P_{t+1}, P_{t+2} and P_{t+3}, and locate on road segments R_1, R_2, R_2, R_4 respectively. C_t^1 and C_{t+1}^1 locate on two connected road segments R_1 and R_2; C_{t+1}^1 and C_{t+2}^1 locate on the same road segment R_2; C_{t+2}^1 and C_{t+3}^1 locate on two non-connected road segments R_2 and R_4. The function of our shortest route calculation is described in Algorithm 2.

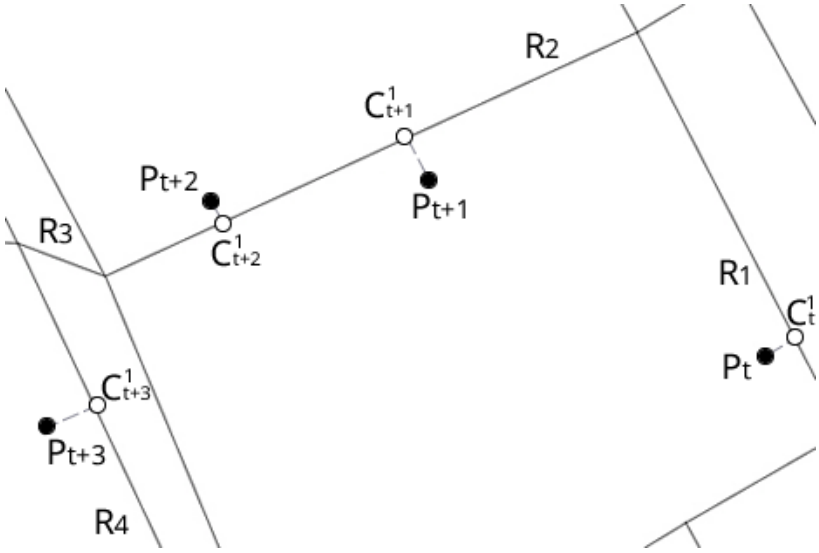


Figure 5.2: Candidates locate on same,connected,non connected road segments

Algorithm 2: Calculate the length of shortest route between C_t^i and C_{t+1}^j

Input: 8 parameters in Table 5.1

Output: the length of the shortest route between C_t^i and C_{t+1}^j;
{C_t^i and C_{t+1}^j locate on the same road segment;};

1: **if** start line=end line **then**
2: perStart ← ST_Line_Locate_Point(start line, start point);
3: perEnd ← ST_Line_Locate_Point(start line, end point);
4: **if** perStart<perEnd **then**
5: shPath ← ST_Line_SubString(start line,perStart,perEnd);
6: Length ← ST_Length(shPath);
7: **else**
8: **if** start line is a two-way road **then**
9: shPath ← ST_Line_SubString(start line,perEnd,perStart);
10: Length ← ST_Length(shPath);
11: **end if**
12: **else**
13: res ← pgr_kdijkstraPath;
14: res ← ST_Linemerge(start line,res);
15: Length_ring ← ST_Length(res);
16: Length = ST_Length(shPath);
17: Length = Length_ring-Length;
18: **end if**
{C_t^i and C_{t+1}^j locate on two connected road segments;};
19: **else**
20: **if** start target=end source **then**
21: res ← ST_Linemerge(start line,end line);
22: perStart ← ST_Line_Locate_Point(start line, start point);
23: perEnd ← ST_Line_Locate_Point(start line, end point);
24: shPath ← ST_Line_SubString(start line,perStart,perEnd);
25: Length = ST_Length(shPath);
26: **end if**
{C_t^i and C_{t+1}^j locate on two non-connected road segments.};
27: **else**
28: res ← pgr_kdijkstraPath;

```
    res ← ST_Linemerge(start line,end line);
    perStart ← ST_Line_Locate_Point(start line, start point);
    perEnd ← ST_Line_Locate_Point(start line, end point);
    if perStart<perEnd then
        shPath ← ST_Line_SubString(start line,perStart,perEnd);
    else
        shPath ← ST_Line_SubString(start line,perEnd,perStart);
    end if
    Length = ST_Length(shPath);
end if
return Length;
```

5.2 Optimizing Shortest Route Calculation

The shortest route calculation is the most time consuming procedure in the HMM and FMM algorithms. When using pgRouting to do the shortest route calculation, the pgRouting will use all the possible permutations and combinations of the road segments in the whole road net work to find the shortest route between two given vertexes. To accelerate this procedure, we can limit only the road segments that are close to the trajectory will be considered as the possible set of the shortest route result. In this thesis, a buffer of the trajectory is used. To be more specific, the first step is to use ST_Makeline to connect all the

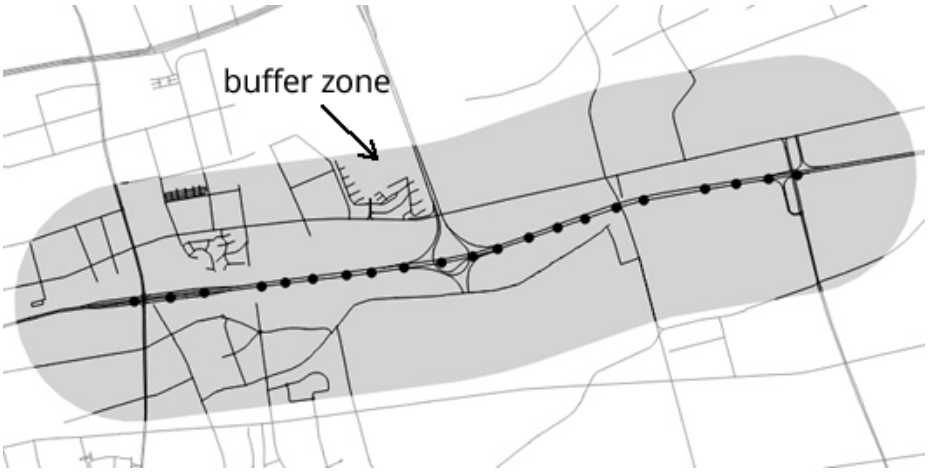


Figure 5.3: Buffer zone of trajectory

GPS points in the same trajectory by time series, then according to this trajectory line, use ST_buffer to get a buffer zone; the last step is to only keep the road segments inside the buffer zone, and use this new set of road segments to do the shortest route calculation. The size of the buffer zone is significant to the accuracy and efficiency performance, and we will discuss the size of the buffer zone in the experimental chapter.

6 A Work flow of map matching FCD

Not like map matching the data prepared for algorithm designation, map matching FCD is a much more difficult task, due to the big amount and the complexity of the data. Therefore, we propose a work flow to show the full procedure of map matching FCD, from the data preparation to the presentation of the map matching result. The work flow is shown in Figure 6.1.

6.1 Trajectory Data Preparation

6.1.1 Overview of Shanghai FCD

In this thesis, we use the floating car data in Shanghai, China to explain the work flow, and the original data is saved in a .txt file. The File records 20703863 GPS trajectory points, which contains the GPS logs of 6973 taxis in Shanghai for 24 hours, from 20:30:00 31.03.2010 to 20:00:00 01.04.2010. The original intervals of the GPS points is 10 seconds. The attributes of the points are shown in Table 6.1.

6.1.2 Data validation

In this thesis, we choose PostgreSQL to save the FCD, which is convenient for calculating the shortest routes between trajectory points later. Because the original data has some errors, data validation is necessary before importing the FCD into database. We define three levels of data validation in this thesis.

Errors like missing or invalid formats of attributions, which cannot be saved into database, can be dealt with the first level of data validation. A Visual Basic program is designed to scan the completeness of the attributions and transfer the original csv format (.txt) into SQL statements. After this step, 20233886(97.73%) pieces of data remains, and can be imported into database.

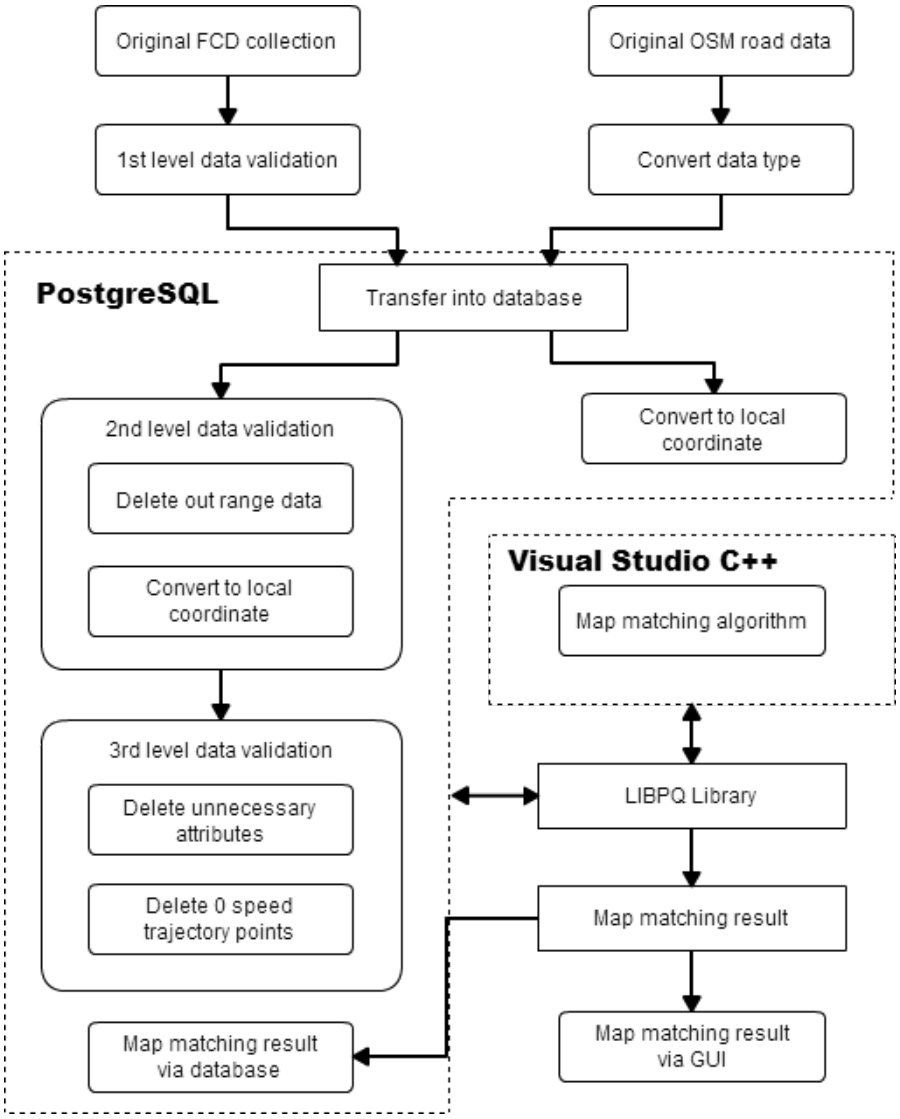


Figure 6.1: The work flow of map matching FCD

The second level of data validation is designed to deal with the implicit errors of GPS information. As shown in Figure 6.2, some GPS points are located on the sea area, which are obviously out of the range of Shanghai city area. This kind of error is caused during the procedures of recording or uploading GPS information. After checking the city area of Shanghai on map, we delete the GPS points outside the coordinates {(120.29, 30.0),(121.98, 31.93)}.

Fields	Description
date	Date of GPS measurement.In format of '2010-03-31'
time	Time of GPS measurement.In format of '20:37:16'
car owner	string code of taxi company. ie.'QS'
car id	unique ID of the car, ie. 18384
longitude	longitude in degress, with precision length of six decimal
latitude	latitude in degree,with precision length of six decimal
velocity	instant velocity of the taxi in km/h. ie.39.5
driving direction	deiving direction of the car, ranges from 0 to 355, with 0 pointing north, and increase clockwisely
car status	binary code, 1 means driving with passenger, otherwise 0
signal status	binary code indicate the validation of gps measurement. 1 for validate, 0 otherwise
record time	time when data is collected to datacenter

Table 6.1: Floating Car Data in Shanghai

Figure 6.2: Floating car data out of city area

To simplify the calculations, especially the ones to get circle distances and shortest routes, it is better to convert the sphere geographic coordinate system, which is WGS84 in the original GPS data, to a local plane coordinate system. In addition, the data type "Geometry" in PostgreSQL, which include the location information (longitude and latitude) and the coordinate system information, is more convenient to be used. Consider Shanghai is located in China, and the meridian crosses close to the middle of the city is about 120 east, thus we choose the EPSG:2385 Xi'an 1980 /3-degree Gauss-Kruger CM 120E[2] (Table 6.2) as the coordinate system in this thesis, and transfer all the GPS points in the geometry type with the above local plane coordinate system, and then save them into the PostgreSQL database.

EPSG:2385	Xian 1980 / 3-degree Gauss-Kruger CM 120E
WGS84 Bounds:	118.5000, 21.9300, 121.5000, 53.3300
Projected Bounds:	345017.9483, 2426808.3104, 654982.0517, 5912397.9623
Scope:	Large scale topographic mapping, cadastral and engineering survey.
Last Revised:	June 22, 2002
Area:	China - 118.5°E to 121.5°E

Table 6.2: EPSG: 2385 Xian 1980 / 3-degree Gauss-Kruger CM 120E

This thesis focuses on map matching based on HMM algorithm, thus we only need few fields in the original data. After the second level of data validation, the fields saved in PostgreSQL are listed in Table 6.3.

Fields	**Type**	**Description**
id	integer	unique ID of the trajectory point,ie. 1
car id	integer	unqiue ID ededof the car,ie. 18384
time	timestamp without time zone	Date and time of gps measurement.In format of '2010-03-31 22:28:00'
geometry	gemotry(point,2385)	geometry value of the GPS points
driving direction	integer	driving direction of the car, ranges from 0 to 355, with 0 pointing north, and increase clock wisely

Table 6.3: FCD after validation and simplification

The third level of data validation is designed to accelerate the map matching process. We

notice that there exist a big of amount of successive GPS points with speed of 0, which may be created when the taxi was stopped in one location for a long time without turning off the GPS. These points cannot be served to analyze the states of taxis, but it will bring negative influence to the efficiency performance of map matching. To delete these points, we execute a SQL query with the following definition:

Delete GPS points with same car ids, same geometry locations, and speeds equal to 0, except the ones with smallest row ids.

After the second and third data validations, 13986856(69.12%) points remain, and can be used to do the further analysis.

6.2 Road Network Data Preparation

The road network is gathered from Open Street Map, which is an openly licensed map of the world being created by volunteers using local knowledge, GPS tracks and donated sources[3]. The original road network file is in the term of .shp, and imported into PostgreSQL with being converted to the geometry type and the EPSG:2385 local plane coordinate system. The fields in the road network is shown in Table 6.4.

Fields	Type	Description
id	integer	Unique local id of each road polyline. ie.1
length	double precision	The length of the road polyline. ie.100.25
reverse cost	double precision	The reverse cost indicate whether the road is two-way road or not.if the road is two-way road,then reverse cost=length; otherwise 100000000000
geometry	geometry (linestring,2385)	The geometry of the road polyline, includes coordinate system information

Table 6.4: Fields of Open Street Map

In this thesis, instead of using osm id, which is a worldwide unique id given by Open Street Map for each road segment, we choose to use the unique local id, and it can help calculate the accuracy of the algorithm more precisely. As shown in Figure 6.3, osm id is corresponding with the name of the road, and several road segments may have same osm id, but local id is corresponding to the segmentation of the road.

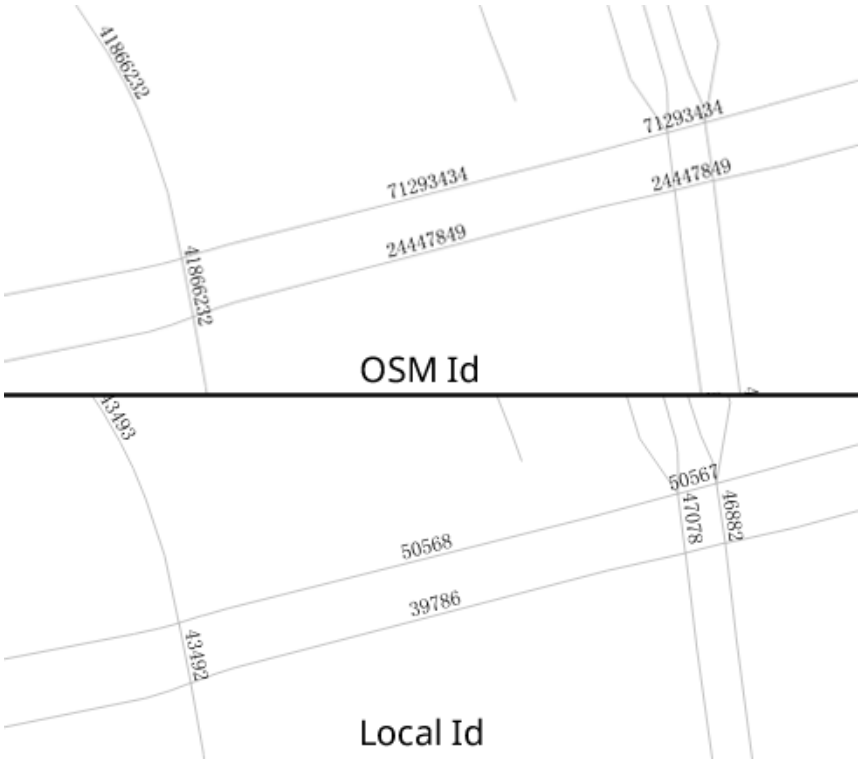


Figure 6.3: Comparison of OSM id and local id

6.3 Algorithm Implementation Environment

The FMM algorithm is coded in C++ in Visual Studio, and the C++ program is connected to the PostgreSQL via LIBPQ library, which is a set of library functions that allow client programs to pass queries to the PostgreSQL backend server and to receive the results of these queries. The detail test and runtime environment is shown in Table 6.5.

Operaction System	Windows 7 64bit
CPU	Intel Core i7 2.67GHz
RAM	8 GB DDR3
Storage	128 GB SSD
Coding Platform	C++ via Visual Studio 2010
Database	PostgreSQL 9.3

Table 6.5: Test and runtime environment

7 Case Study: Map Matching of Shanghai FCD

Due to the large amount of FCD, we choose part of the data from Shanghai FCD as test data, to evaluate the feasibility of the map matching work flow and the performances of the algorithms.

7.1 Test Data

Test data is defined to evaluate the performances of the map matching algorithms in the perspectives of accuracy and efficiency, and the following requirements should be reserved:

1. Represent the whole area, from the perspective of distribution and density;
2. Suitable quantity, which means it should be large enough to give credible result, but runs within an acceptable time.

To choose the appropriate test data for evaluating the performance of the map matching algorithms, we define three scenarios based on the road density: dense area, common area, and sparse area. Then for each scenario, we extract an amount of GPS points with the same proportion of the whole area. The detail choosing procedure works as below:

1. We illustrate grids with equal size on the whole range covered by the GPS points, and in each grid, we calculate the road density(km/km^2)=total road length(km)/total area(km^2);
2. We define the range of each scenarios as: the dense areas road density is more than 8 km/km^2, the sparse areas is less than 3 km/km^2 and the rest parts are common areas. The distribution of the grids and the corresponding road densities are shown in Figure 7.1;

	Grid Number	Training Area (km^2)	Original Points	Test Points
Dense Areas	7	30.76	1675638	13884
Common Areas	52	230.01	1121043	9749
Sparse Areas	119	524.18	118831	849

Table 7.1: Distribution of the test points in three scenarios

3. We choose the test areas with the same proportion of the grids, and to not damage the completeness of the trajectories, we try to make the expressway or arterial road as the boundaries of each area;
4. We randomly choose the final test trajectories in each scenario with the same proportion of the total points in each scenario, and try to make all the roads in the chosen areas be covered. The borders of each area and the distribution of the test points are shown in Table 7.1 and Figure 7.2.
5. We manually label the test data, and for the blurry trajectories, we delete them or mark both candidate road segments correct, which is much common when the trajectory is in the middle of two parallel road segments and both of them are looked correct.

The original sampling intervals of FCD is 10s. To evaluate the algorithm performance in low sampling rates, we create the bigger sampling intervals sets of points based on the original FCD data. We choose 20s, 30s, 60s and 120s as our new sampling intervals. We do not do the new sampling intervals on sparse area, because the lower sampling rate sets will have only few points, and the result will not be reliable. The lower sampling rates points for dense and common area is shown in table 7.2.

	20s	30s	60s	120s
Dense Area	7995	5802	3301	1807
Common Area	5625	4108	2315	1261

Table 7.2: Trajectory points in low sampling rates

Labeling test data by human is a very time consuming and tedious procedure, and also very easy to make errors due to losing attention of human. In addition, some complex trajectories are very difficult for human to identify the correct road segments, especially the ones travel on the parallel road segments in the city center. Based on our experience,

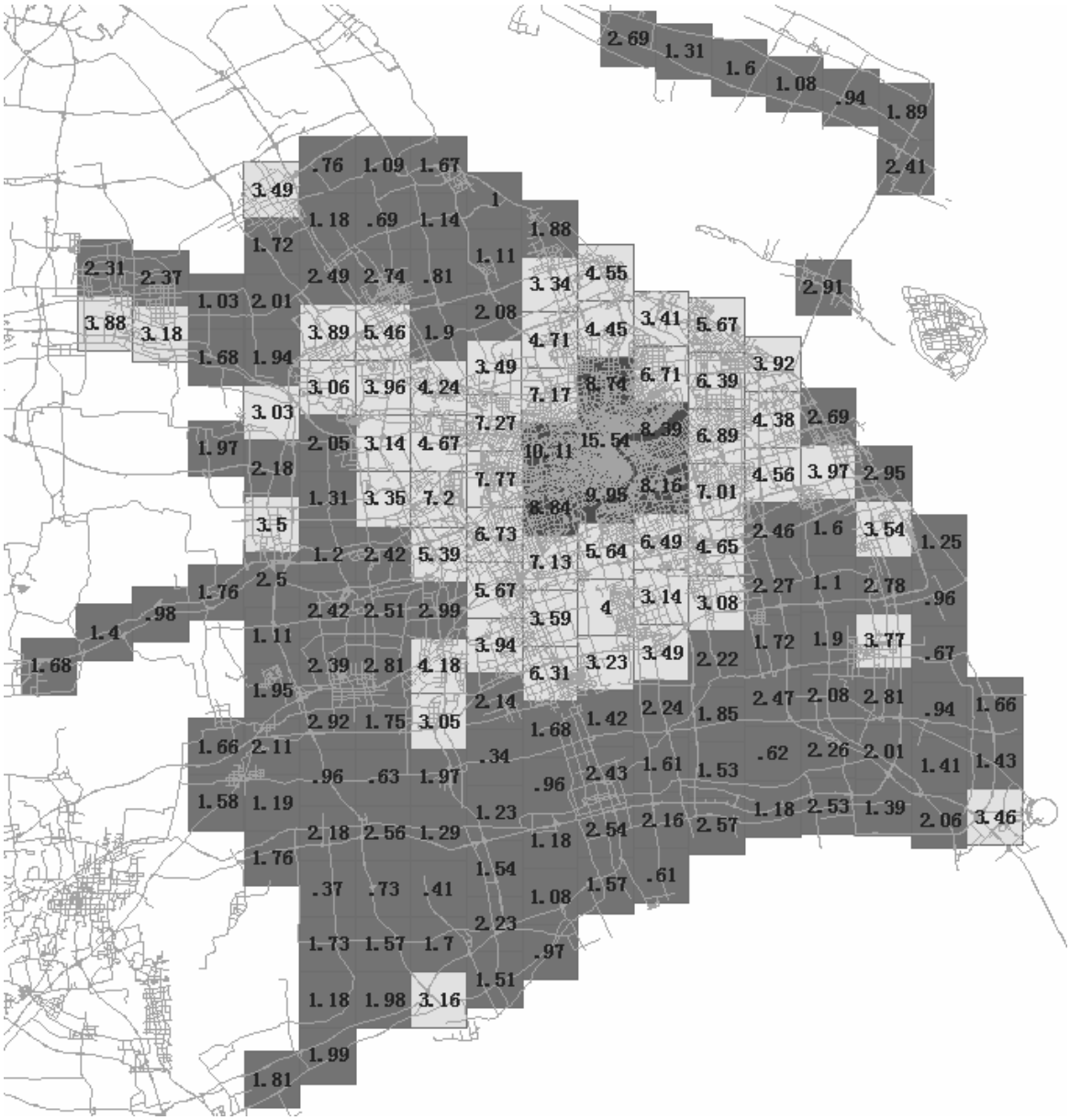


Figure 7.1: Road network densities in girds

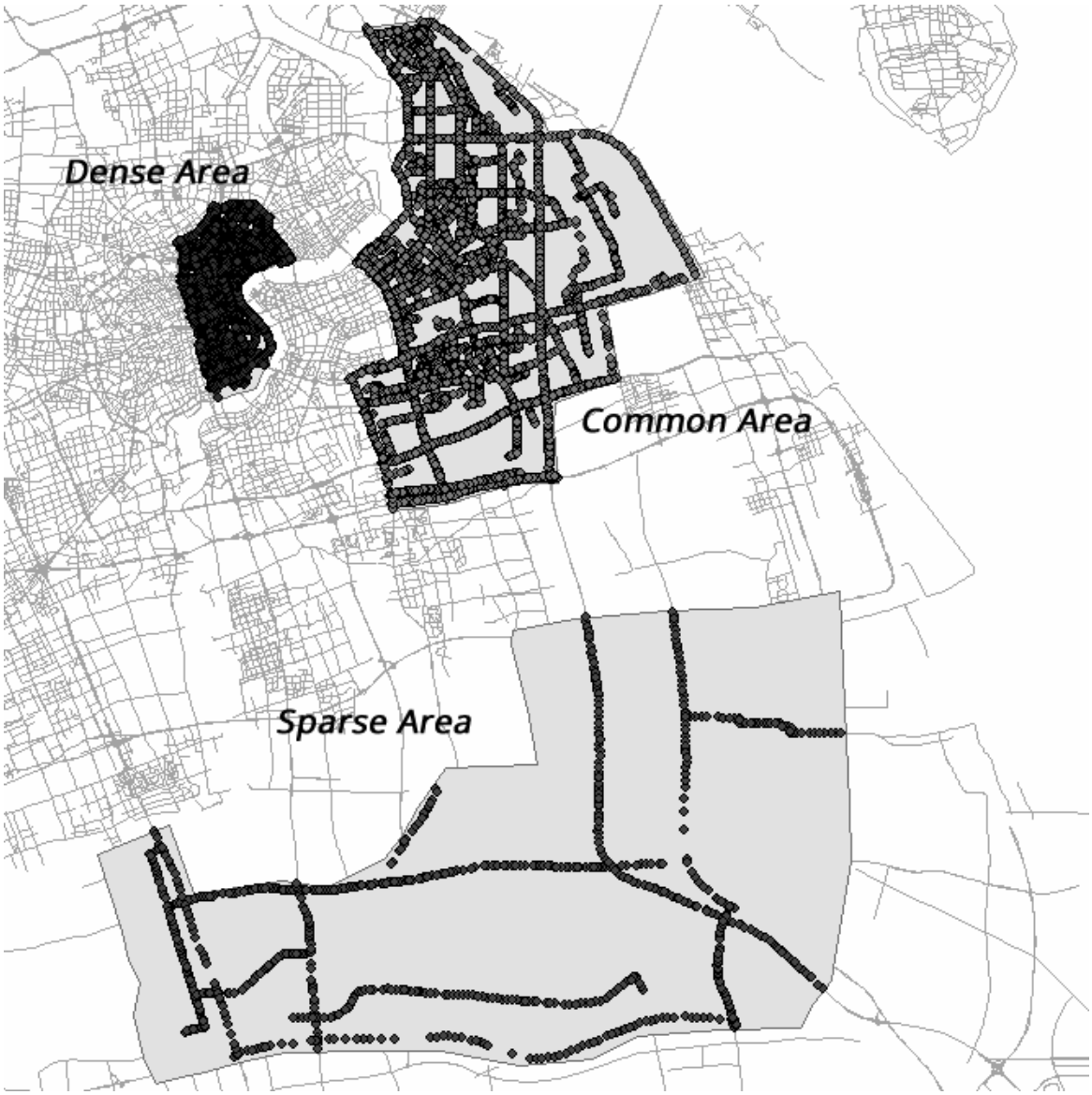


Figure 7.2: Borders of test areas and distribution of points

human can label about 1000 points each day, which does not include the time to check the errors.

In this thesis, we propose a improved efficiency test data labeling method , which focuses on labeling the entire trajectory at one time. As we know, every trajectory follows a serious successive road segments and most time it follows the shortest route between the origin and destination, especially in this FCD Shanghai data recorded by taxis. Therefore we use an open source project called Open Source Routing Machine (OSRM), which uses the road network data of Open Street Map, and can quickly get the shortest route between every two points. The OSRM has the option to save the path into a GPX file, which includes the line geometry information of the paths, and exchanges it between programs. In addition, the OSRM supports setting route markers on the path, and dragging them arbitrarily, and this can help make the path become totally same with the trajectory that need to be matched. The OSRM can be run on personal computer after a series of set up, and in this thesis, we use an on-line website[16] created by the author of OSRM.

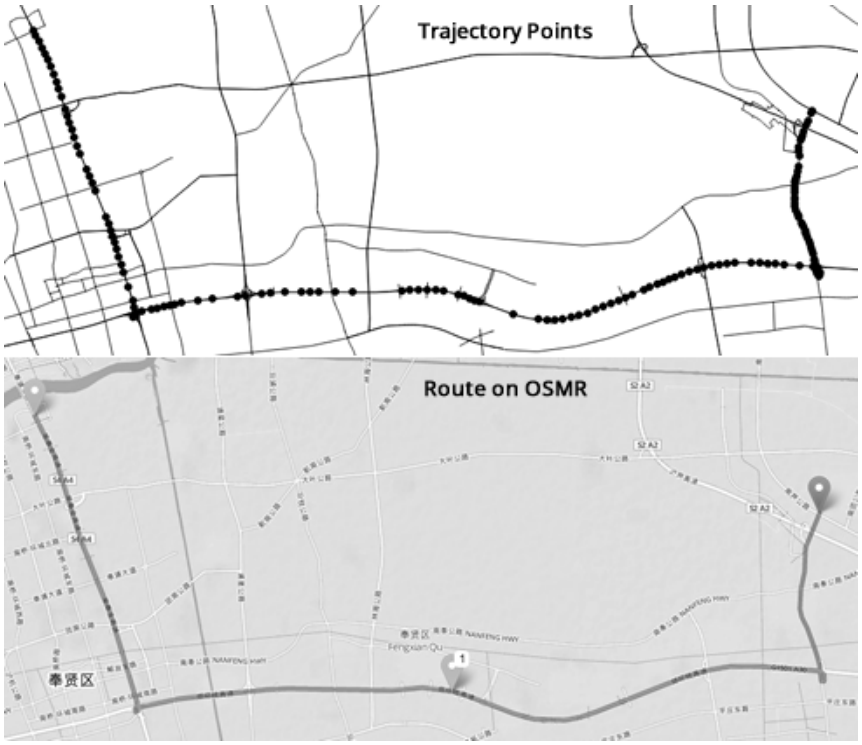


Figure 7.3: Route on OSMR

The procedure of the method goes like this: firstly a route is created in OSRM with the same shape of the trajectory, as shown in Figure 7.3. Then the GPX can be used to find

the corresponding road segments with local ids in the road network. Next, each trajectory point can be assigned to a local id of the closest road segment by a program automatically. At last, we check the labeling result and fix the errors manually, which is much easier than doing it from the beginning. With this method and our experiences, more than 80 trajectories can be finished in a day, and about another one day is needed to check and the fix the errors.

After checking by human, there still a big possibility that existing errors in the labeled ground truth data. To partly solve this, we check the shortest routes between each two successive ground truth points. As we know, the speed of vehicle in a city area is limited, and if the shortest route between two successive ground truth points is much bigger than the longest distance that a vehicle can run within the sampling interval, then there is big chance that this area exists labeling errors. After this procedure, the ground truth data is more reliable and can be used to evaluate the performances of map matching algorithms.

7.2 Parameters Estimation

7.2.1 Trajectory segmentation

Trajectory is defined as the collection of continues GPS points from the same vehicle. For the FCD in Shanghai, the car id attribute can be used to distinguish different taxis, but the driving path of each taxi is not a continuous trajectory in most of the time, and it has break points. Therefore, we need to separate the GPS points from the same taxi into several trajectories based on the break of the time sequence, or the distance between successive GPS points, or both. What need to be pointed out is that because of the huge amount of data and the complex situations of different trajectories, none of these two fields can divide the trajectory 100% correctly. But overall, this procedure can make the trajectories more close to the real situations.

In this thesis, we choose both the time and distance as the judgments of breaking trajectories. To be more specific, for two successive GPS points from same taxi, if the time difference is bigger than 1 hour, or the distance is larger than α, the two points will be seen as the last and the first point of two trajectory.

In the Shanghai city, the biggest allowed driving speed is 120km/h, which is about 35m/s. Therefore, we define α =35m/s $\times t \times 2$, where t is the sampling interval.

Accordingly, we use 35m/s $\times t$ as the threshold parameter of alternating global and incremental algorithms in our FMM algorithms.

7.2.2 Buffer zone size for shortest route calculation

As mentioned in section 5.2, the determination of the buffer zone for shortest route calculation is a key procedure that effects the performance of the algorithm in the aspects of accuracy and efficiency. The priority consideration of the buffer size is the influence on the accuracy, which means the buffer zone must cover the correct candidate road segments of the trajectory. In this thesis, to decide the most suitable buffer zones for the HMM and FMM algorithms, we make a series tests on the test data, which has been labeled with ground truth data, to find the most suitable buffer size for shortest route calculation. In these tests, we calculate the shortest route between each two successive trajectory points in two situations: with and without buffer zone setting. In addition, the maximum circle distance between the trajectory point and the corresponding road segment is around 180m, which means the buffer zone must bigger than 180m to make sure the correct road segments is included, and the shortest route should be the same in these two situations. Since the smaller the buffer zone, the faster the calculation, the goal of the tests are to find the minimum size of the buffer zone which have the same accuracy with the no buffer zone situation. After testing several buffer zone values in the test data with different sampling intervals. The most suitable buffer zone sizes for different sampling intervals are shown in Table 7.3.

Sampling interval	10s	20s	30s	60s	120s
Buffer zone size	200m	200m	300m	400m	800m

Table 7.3: Buffer zone sizes for different sampling intervals

The efficiency performance of the buffer zone setting is evaluated through counting the running time of calculating shortest routes. When using buffer zone, the needed time is composed of two parts: creating buffer zone for each trajectory and calculating the shortest route within the buffer zone. The performance is shown in Figure 7.4, and it can be seen that using buffer zone can saving over 40% of calculation time.

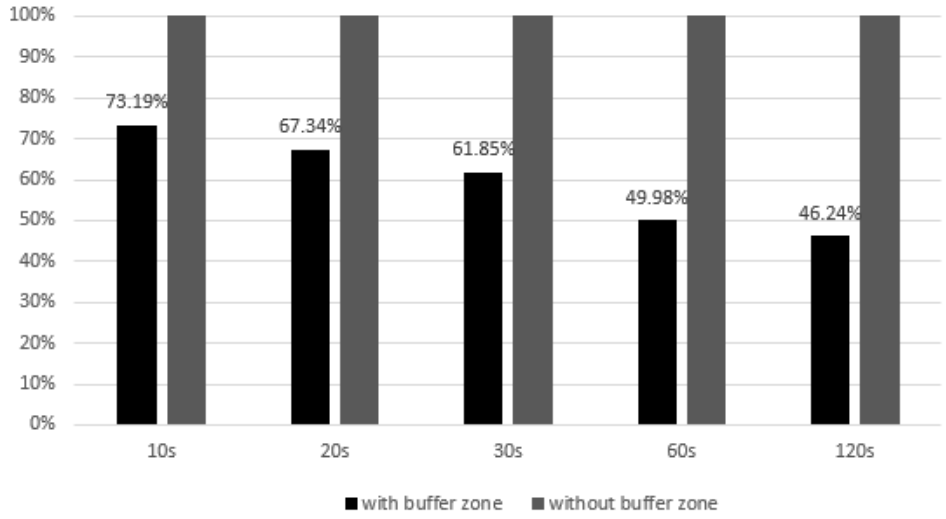


Figure 7.4: The efficiency performance of using buffer zone

7.2.3 Standard deviation of the GPS measurement

In HMM and FMM algorithms, the standard deviation of the GPS measurement σ_p need to be estimated. From the labeled test data, which can be seen as the ground truth data, the circle distances from each trajectory point to its location on the road segment can be calculated, and these values indicate the error distribution of the GPS measurement. According to Newson's research[18], σ_p could be estimated by using the median absolute deviation (MAD), which is a robust estimator of standard deviation:

$$\sigma_p = 1.4826(|P_t - C_t^i|_{great\ circle}) \tag{16}$$

	Average (m)	Median (m)	Maximum (m)
Dense area	7.09	4.54	127.06
Common area	6.00	4.19	181.17
Sparse area	5.23	3.80	39.32
Total	6.60	4.37	181.17

Table 7.4: Average, median and maximum disperses of three scenarios

Table 7.4 shows the trajectory points disperses situation in the dense, common and sparse area. The results indicate that the disperses in dense area is more serious due to the complex surrounding environment. Figure 7.5 shows the distribution of the disperses. It can

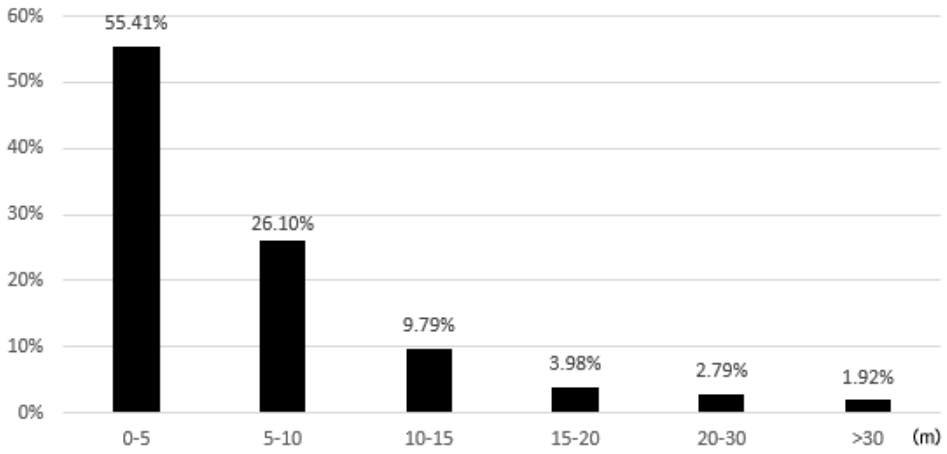


Figure 7.5: Disperses of trajectory points of ground truth data

be seen that most of the disperses are within 10 meters. The maximum circle distance between the trajectory points and the road segments in ground truth data is 181.17m, which fits the configuration of the search radius (50m to 200m) in our FMM algorithms.

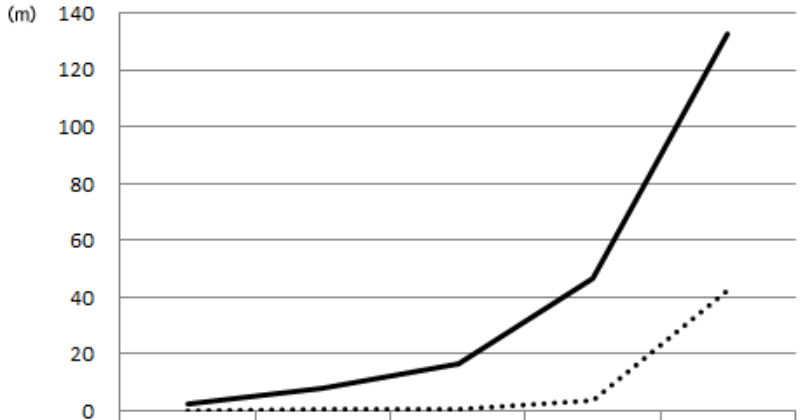
In this thesis, σ_p is estimated with value 6.48m, and the ω for determining the distance between trajectory points that are recorded in the location is 6.60m.

7.2.4 Parameter in transition possibility

In the HMM based algorithms from Newson[18] and Wei[21], the parameter β from the exponential distribution in equation (2), which describes the differences between the circle distances and shortest routes, need to be estimated. Newson's research[18] proposed that β could be estimated by using a robust estimator suggested by Gather[9]:

$$\beta = \frac{1}{\log(2)} median(|P_{t+1} - P_t|_{great\ circle} - |C^i_{t+1} - C^j_t|_{route}) \tag{17}$$

Figure 7.6 shows the differences between the circle distances and shortest routes in different sampling time intervals. It can be seen that with the increasing of the sampling time interval, the disperses get larger, which means the performance of transition possibility get worse with bigger sampling time intervals in map matching. A larger value of β, represents more tolerance of non-direct routes. Therefore, in this thesis, the β value is various with different time intervals, and we choose the values in Figure 7.6 as the β in the algorithms of Newson[18] and Wei[21].



	10 s	20 s	30 s	60 s	120 s
Average	2.45	8.19	16.49	46.80	132.52
Median	0.14	0.37	0.72	3.74	42.35
β	0.47	1.21	2.41	12.41	140.69

Figure 7.6: Differences between circle distances and shortest routes in various sampling intervals

7.2.5 Parameters in forward looking algorithm

In our forward-looking FMM algorithm, through experiments, we notice that when setting the number of forward looking point bigger than 1, the accuracy of the algorithm can only get a very tiny improvement, but will significantly increase the calculation time. Therefore, in our forward-looking algorithm, we set the forward looking point number to 1. In addition, to accelerate the calculation, we limit the largest number of candidates for this forward looking point to 3.

7.3 HMM Based Algorithms Comparison

To choose the suitable HMM based algorithm for the global operation in our FMM algorithms, we compare the accuracy performances of three existing algorithms and our HMM based algorithm. In this thesis, we define the accuracy performance of a map matching algorithm as:

$$\frac{number\ of\ trajectory\ points\ assigned\ with\ correct\ local\ road\ segment\ id}{number\ of\ total\ trajectory\ points} \tag{18}$$

The accuracy performances of the algorithms are checked in two scenarios.

The first one is the post-processing scenario, which use all the continues points in one

trajectory to get the map matching results, and the results are shown in Figure 7.7. It can be seen that our algorithm has the best and stable performances in all the sampling intervals; Newson's algorithm [18] also has a stable accuracy performance; Wei's algorithm [21] has better performance in high-sampling-rate, but seriously decrease with the increasing of the sampling intervals; Without considering the temporal possibility, Lou's algorithm [15] shows an unaccountable results, but it does has better performance in low-sampling-rate data.

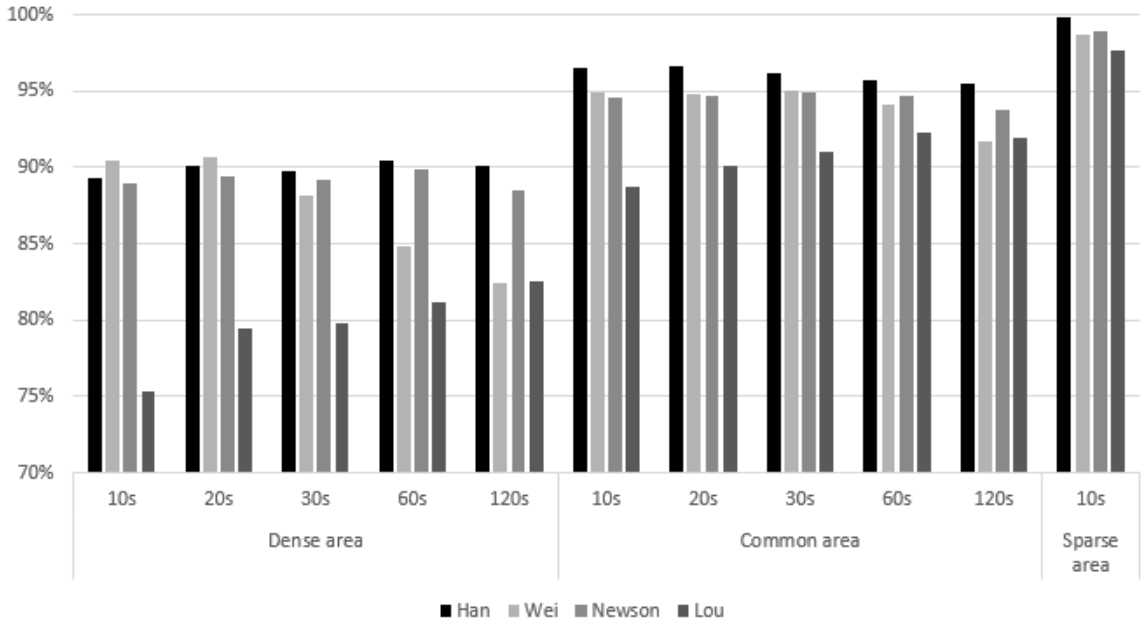


Figure 7.7: Accuracy performances of HMM based algorithms with full trajectory

Because the FMM algorithm only takes 5 trajectory points to do the global calculation at the beginning and in the middle of the trajectory when needed, thus we design the second scenario to use every 5 successive trajectory points to get the map matching results. In addition, take 5 points is a reasonable amount when HMM based algorithms are used in real-time navigation scenario. From Figure 7.8, we can see that all the algorithms show reduced accuracy performance compared with the results with full trajectory, but our algorithm has the lowest amount of decreasing, and still have the best and most stable performance among the four HMM based algorithms. This indicates that our HMM based algorithm also has the best performance in the real time navigation scenario. The algorithms from Newson [18] and Wei [21] show deteriorative performance in short trajectory with small sampling intervals, and we speculate this relates to the values of β. Following Newson's assumption of β, which are very small in high-sampling-rate, it requires a very

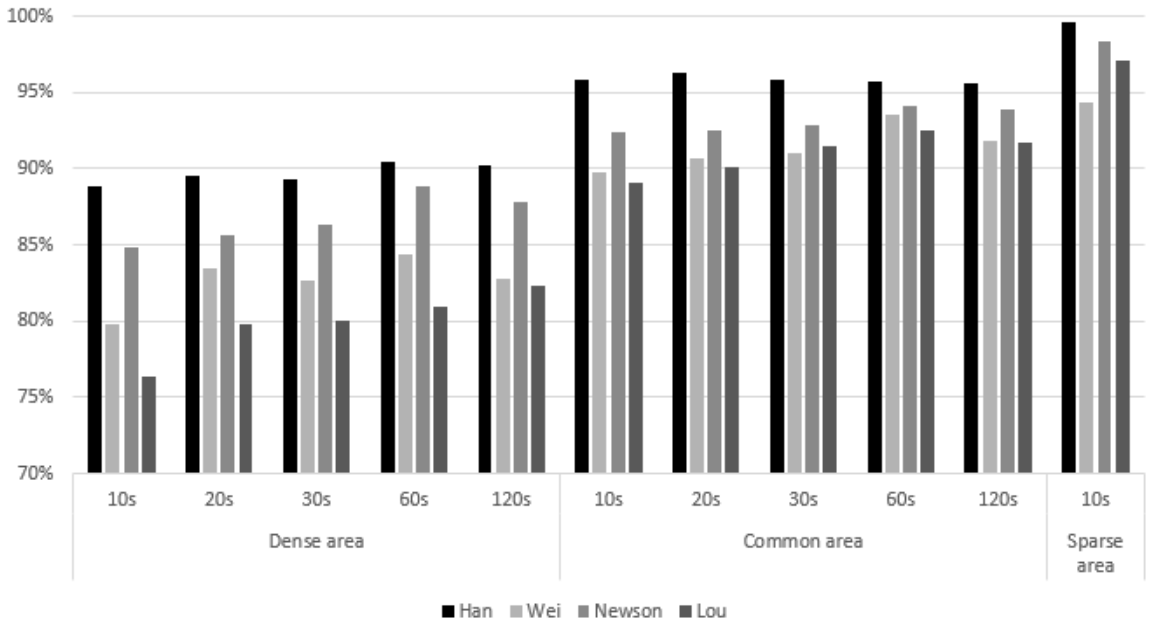


Figure 7.8: Accuracy performances of HMM based algorithms with 5 points limitation

strict equality between the shortest routes and circle distances, and makes the transition possibility influences too much to the final matching result. With bigger β selection, the performances of these two algorithm can be improved.

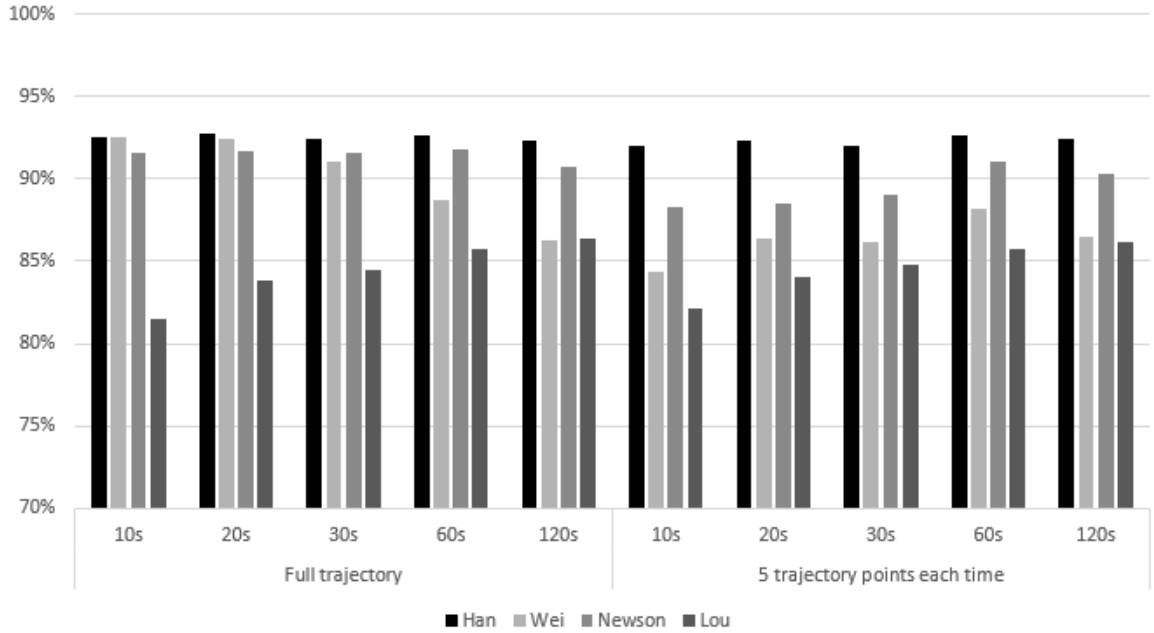


Figure 7.9: Accuracy performances of HMM based algorithms in the whole test area

The accuracy performances of HMM based algorithms in the whole test area are shown in Figure 7.9. The detail data of accuracy performances of HMM based algorithms is shown in Table 7.5.

7.4 Experiment Results

7.4.1 Accuracy and efficiency performances of the FMM algorithms

The performance of an map matching algorithm can be shown in two perspectives, accuracy and efficiency. To evaluate these two performances of the FMM algorithms, we use the HMM based algorithms with best performances as baselines. For different sampling intervals, we choose the HMM based algorithms with the best accuracy performances, since the accuracy is the major focus in this thesis. Therefore, our baselines are made by the following HMM based algorithms:

Dense area					Common area					Sparse area
10s	20s	30s	60s	120s	10s	20s	30s	60s	120s	10s
W	W	N	N	N	W	W	W	N	N	W

*W = Wei[21], N = Newson[18]

Table 7.6: Baselines made by HMM based algorithms

Figure 7.10 shows the accuracy performances of two kinds of FMM algorithms in three scenarios. It can be seen the forward-heading FMM algorithm has the best performances in most of the sampling intervals, and normal FMM algorithm has 1% lower performances than forward-heading FMM algorithm. Compared with the baselines made by HMM based algorithms, both FMM algorithms have a much better accuracy performances.

As we described, the shortest route calculation is the most time consuming procedure in the map matching algorithms. Therefore, we use the amount of shortest routes calculation as the main factor of evaluating the calculation complexities. To simply describe the complexity, here we take a trajectory with m $(m > 5)$ trajectory points as example, and each trajectory has n $(n >= 3)$ candidates. Our FMM algorithms are more complex to calculate the complexities because it use the global and incremental algorithm dynamically, and the proportions of each algorithm used in FMM algorithms are depended on the specific circumstances of trajectories. In our experiments on our test data, 9.63% (2354) of trajectory

		Dense area					Common area					Sparse area
		10s	20s	30s	60s	120s	10s	20s	30s	60s	120s	10s
Full trajectory	Han	89.27%	90.11%	**89.80%**	**90.49%**	**90.15%**	**96.51%**	**96.59%**	**96.23%**	**95.72%**	**95.48%**	**99.88%**
	Wei[21]	**90.46%**	**90.72%**	88.16%	84.88%	82.40%	94.88%	94.76%	95.03%	94.13%	91.75%	98.70%
	Newson[18]	88.91%	89.47%	89.23%	89.91%	88.54%	94.59%	94.74%	94.89%	94.47%	93.81%	98.94%
	Lou[15]	75.36%	79.41%	79.82%	81.22%	82.51%	88.76%	90.08%	91.02%	92.31%	91.99%	97.64%
5 points each time	Han	88.90%	89.53%	89.28%	90.49%	90.26%	95.83%	96.32%	95.86%	95.68%	95.64%	99.65%
	Wei[21]	79.85%	83.43%	82.70%	84.40%	82.79%	89.82%	90.67%	91.02%	93.56%	91.83%	94.35%
	Newson[18]	84.80%	85.60%	86.32%	88.85%	87.77%	92.35%	92.57%	92.87%	94.13%	93.89%	98.35%
	Lou[15]	76.34%	79.77%	80.08%	80.91%	82.35%	89.11%	90.12%	91.43%	92.53%	91.75%	97.06%
		Total area										
Full trajectory	Han	**92.53%**	**92.78%**	**92.46%**	**92.65%**	**92.34%**						
	Wei[21]	92.51%	92.39%	91.01%	88.69%	86.25%						
	Newson[18]	91.53%	91.64%	91.57%	91.79%	90.71%						
	Lou[15]	81.48%	83.82%	84.46%	85.79%	86.41%						
5 points each time	Han	**92.04%**	**92.33%**	**92.01%**	**92.63%**	**92.47%**						
	Wei[21]	84.33%	86.42%	86.15%	88.18%	86.51%						
	Newson[18]	88.28%	88.48%	89.03%	91.03%	90.29%						
	Lou[15]	82.15%	84.05%	84.78%	85.70%	86.21%						

∗*Bold value indicates the best performance in corresponding category.*

Table 7.5: Accuracy performances of HMM based algorithms

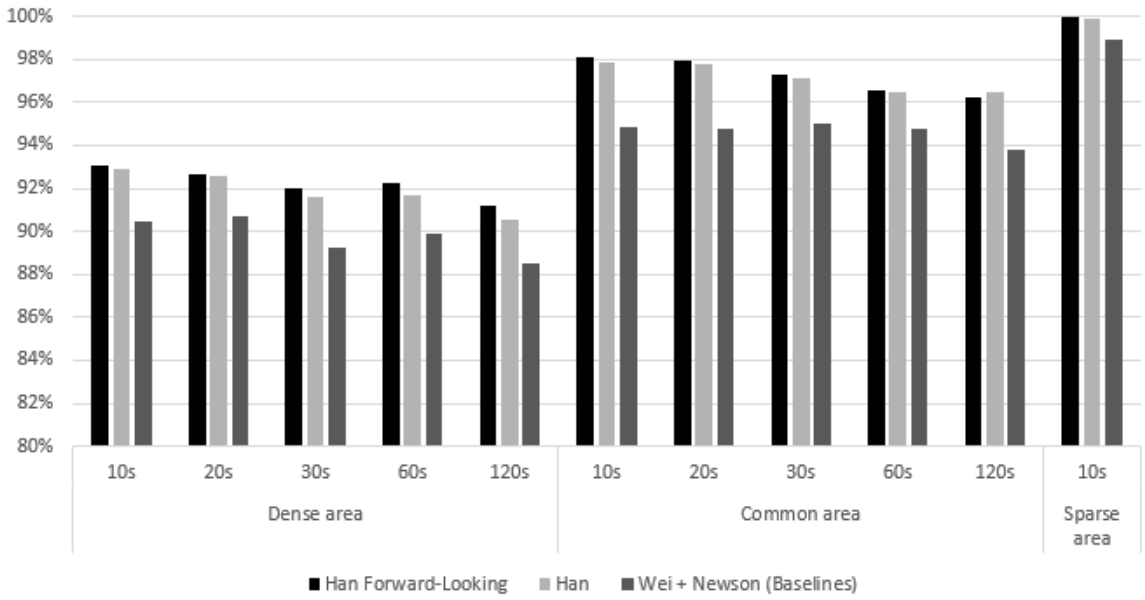


Figure 7.10: Accuracy performance of FMM algorithms in three scenarios

points are map matched by HMM algorithm, thus we use 10% to approximately evaluated the proportions of the usages of HMM algorithm in FMM algorithms. The computation complexities of the map matching algorithms are:

HMM	Normal incremental	F-L incremental	Normal FMM	F-L FMM
mn^2	mn	$4mn$	$0.1mn^2 + 0.9mn$	$0.1mn^2 + 3.6mn$

Table 7.7: Calculation complexities of map matching algorithms

Besides using calculation complexities, we also do experiments to compare the efficiency performances of all the map matching algorithms. We try to use the same environment (same CPU and RAM occupancy rates before running the algorithms) to test different algorithms and record the running time. To indicated the efficiency performances, instead of using the absolute time results, we choose to use the relative results, because different configuration of the running environment could have different absolute time results. In the presentation of results, we compare the algorithm running times with the one has longest running time in the same sampling rate and scenario, and set this longest running time as 100%. The relative efficiency performances are shown in Figure 7.11. It can be seen that the HMM based algorithms have the worst efficiency performances, and the normal FMM algorithm has the best efficiency performances among the three algorithms, and it only use 20% of time of HMM based algorithms. The forward-looking FMM algorithms can save

about 60% of running time.

Figure 7.12 and Figure 7.13 show the accuracies and efficieies of FMM algorithms compared with baselines in the whole test data area. The detail data of the accuracy and efficiency performances of FMM and HMM algorithms is shown in Table 7.8.

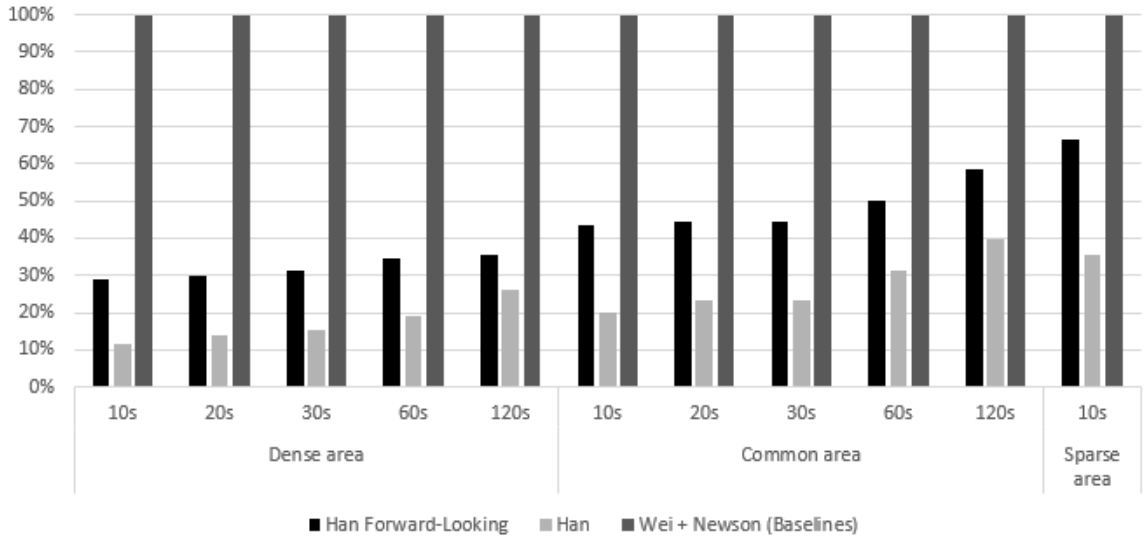


Figure 7.11: Efficiency performance of FMM algorithms in three scenarios

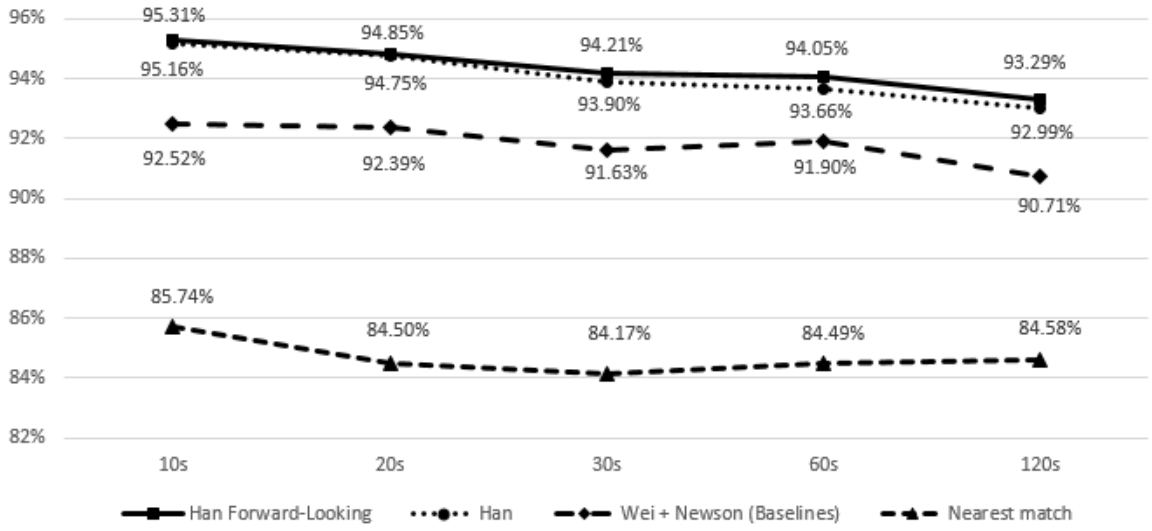


Figure 7.12: The accuracies of FMM algorithms

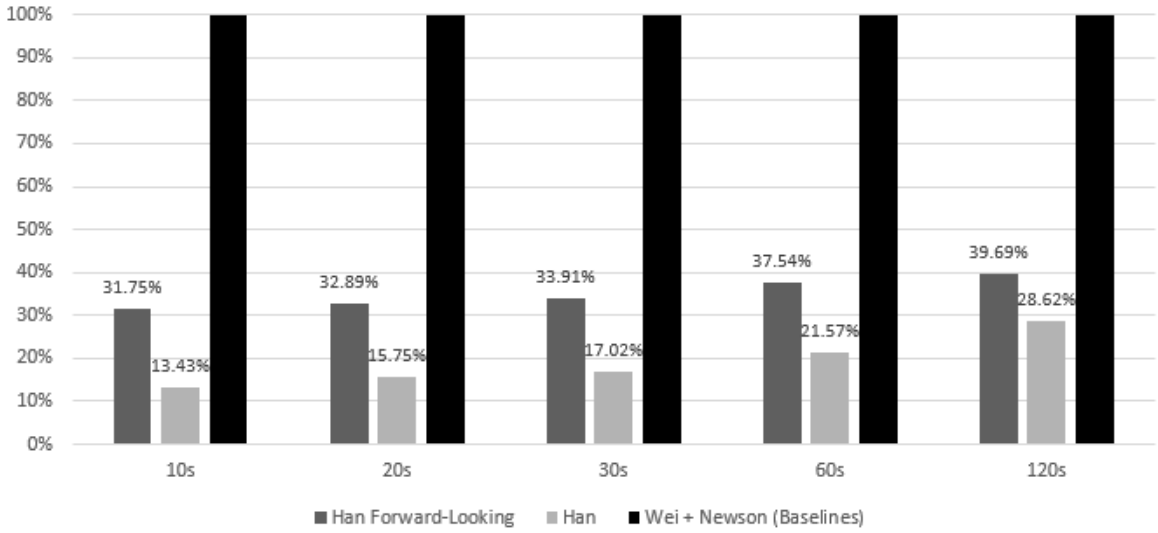


Figure 7.13: The efficiencies of FMM algorithms

7.4.2 FMM algorithms map matching improvement examples

In this section, we use some examples from the test data to describe the accuracy improvement in the FMM algorithm.

Figure 7.14 shows that with considering the trajectory points within a distance, the accuracy of the algorithm is improved. The trajectory has a group of intensive GPS points, and there are "abnormal" points from id 144 to 148, which against the driving direction. As a result, the HMM algorithms will match this part of points to the closest vertex, but FMM algorithm will consider these points are created in the same location, thus match the points to the same road segment with point 143, which are correct.

Figure 7.15 shows that the map matching accuracy improves with considering the driving direction. Trajectory point 5725 is more close to horizontal road segment, therefore the measurement possibility is bigger than the candidate on the vertical road segment. The transition possibilities of the candidates on the two road segments are similar. But the driving direction in the GPS log is 341°, which indicates the vehicle was driving on the vertical road segment. If not consider the driving direction in the map matching algorithm, it will give wrong map matching result to map it on the vertical road segment.

Figure 7.16 shows that FMM algorithm have a better accuracy performance in the parallel road segments. As what we can see, the HMM based algorithm matched these points to

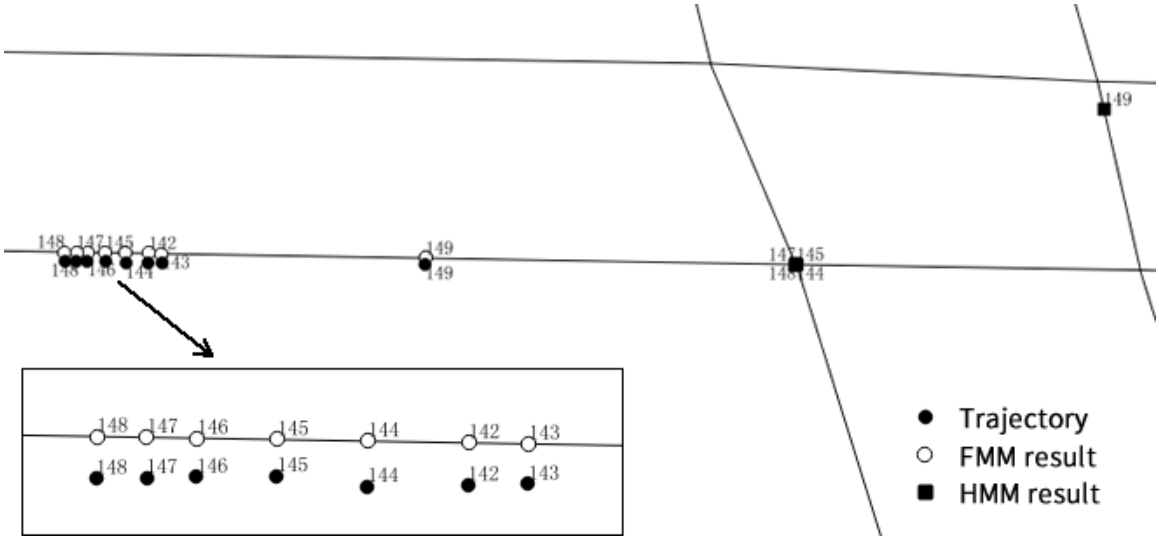


Figure 7.14: Map matching results comparison (1)

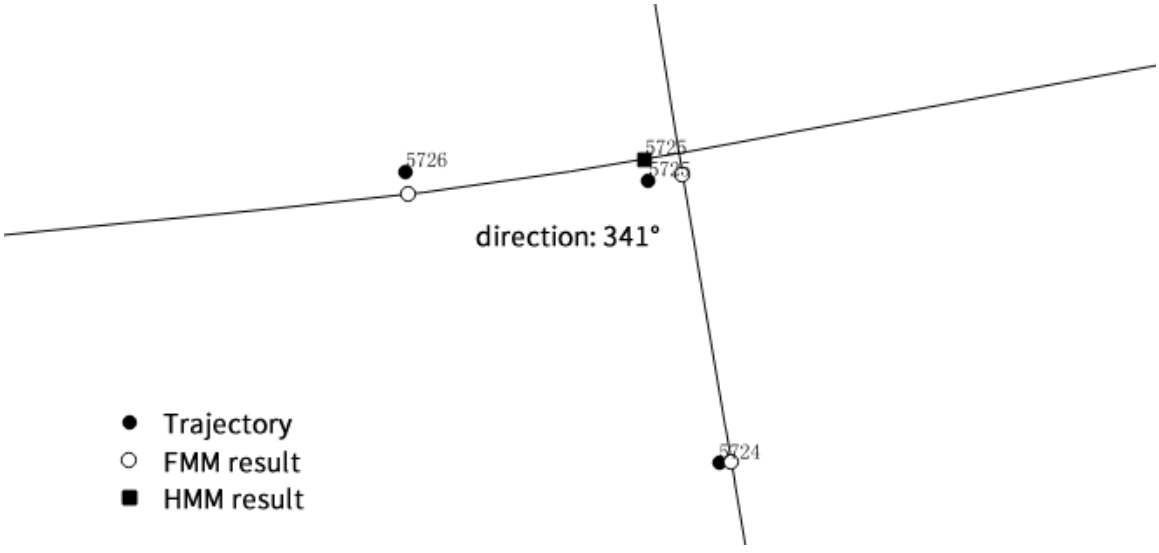


Figure 7.15: Map matching results comparison (2)

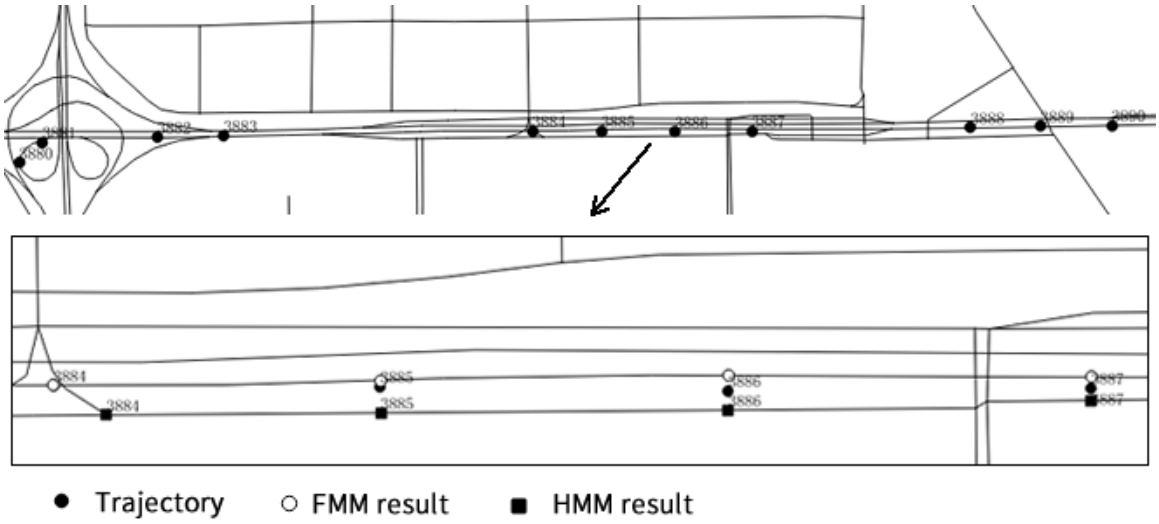


Figure 7.16: Map matching results comparison (3)

the side road, which does not fit the driving habit so much, because before and after these part of trajectory points, the vehicle was traveling on the viaduct; besides, most of the trajectory points are more close to the viaduct.

		Dense area					Common area					Sparse area
		10s	20s	30s	60s	120s	10s	20s	30s	60s	120s	10s
Accuracy	Han F-L	**93.07%**	**92.67%**	**92.02%**	**92.28%**	**91.20%**	**98.09%**	**97.94%**	**97.30%**	**96.59%**	96.27%	**100.00%**
	Han	92.94%	92.60%	91.59%	91.67%	90.54%	97.90%	97.81%	97.15%	96.50%	**96.51%**	99.88%
	Wei + Newson	90.46%	90.72%	89.23%	89.91%	88.54%	94.88%	94.76%	95.03%	94.73%	93.81%	98.94%
Efficiency	Han F-L	28.86%	29.80%	31.32%	34.60%	35.62%	43.27%	44.25%	44.25%	49.85%	58.29%	66.67%
	Han	**11.80%**	**14.08%**	**15.40%**	**19.27%**	**26.17%**	**19.82%**	**23.50%**	**23.50%**	**31.23%**	**39.81%**	**35.56%**
	Wei + Newson	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
		Total area										
Accuracy	Han F-L	**95.31%**	**94.85%**	**94.21%**	**94.05%**	**93.29%**						
	Han	95.16%	94.75%	93.90%	93.66%	92.99%						
	Wei + Newson	92.52%	92.39%	91.63%	91.90%	90.71%						
	Nearest match	85.74%	84.50%	84.17%	84.49%	84.58%						
Efficiency	Han F-L	31.75%	32.89%	33.91%	37.54%	39.69%						
	Han	**13.43%**	**15.75%**	**17.02%**	**21.57%**	**28.62%**						
	Wei + Newson	100.00%	100.00%	100.00%	100.00%	100.00%						

∗F-L = Forward-Looking

∗*Bold value indicates the best performance in corresponding category.*

Table 7.8: Accuracy and efficiency performances of FMM algorithms

8 Conclusion and Future Work

8.1 Conclusion

As floating car data becomes an important source of intelligent transportation system for studying the behaviors of the vehicles and solving the increasingly serious problems in the city transportation system, it is important to have an accurate and efficient map matching algorithm and a corresponding complete work flow to deal with the city wide FCD. In this thesis, we propose two fusion map matching algorithm derived from one new HMM based global algorithm and two incremental algorithms designed by us. In addition, we introduce a work flow of map matching FCD from data reduction to the result presentation. To check the performances of the algorithms, we use the test data selected from the real city wide FCD in Shanghai. Moreover, to detect the algorithm performances in different situations, we separate our test data into dense, common and sparse scenario. Then for each scenario we choose the data amount by the proportion of the distribution of FCD, in order to simulate the constitution of the real city transportation system.

Our test results show that our fusion map matching algorithms can reach 95.31% (with forward-looking 1 trajectory point) and 95.16% (normal FMM algorithm) accuracies for the original sampling rate in FCD, and have a more constant performance in the low sampling rate data. Having less computation complexities, our FMM algorithms have over 2% accuracy improvement, and only need around 32% (with forward-looking 1 trajectory point) and 14% (normal FMM algorithm) of running time compared with existing HMM based map matching algorithms. Furthermore, With considering the distribution of the FCD in different scenarios, we believe that our test results is robust for the city-wide data.

8.2 Future Work

In this thesis, we have been focusing more on improving the accuracy performances of the map matching algorithms, but as already mentioned, the efficiency is also important for map matching FCD. Although our algorithms has better efficiencies than existing HMM based algorithms from the perspective of computation complexities and running time, they can still be improved through various speedup techniques, such as the efficiency improvement of the shortest paths calculation, which is the most time consuming procedure in the algorithm; and fully using the potential of calculation by using multi-cores of computer and parallel calculation techniques. This part of work will be the next research target of this study.

Bibliography

[1] Azimuth. http://en.wikipedia.org/wiki/Azimuth, 07.10.2014. [Online; accessed 10.01.2015].

[2] EPSG:2385 & Xian 1980 / 3-degree Gauss-Kruger CM 120E. http://spatialreference.org, 2002. [Online; accessed 10.01.2015].

[3] Open Street Map. http://www.openstreetmap.org/about, 2014. [Online; accessed 10.01.2015].

[4] Hidden Markov model. http://en.wikipedia.org/wiki/Hidden_Markov_model, 28.09.2014. [Online; accessed 10.01.2015].

[5] Fréchet distance. http://en.wikipedia.org/wiki/Frechet_distance, 30.05.2014. [Online; accessed 10.01.2015].

[6] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *VLDB '05 Proceedings of the 31st international conference on Very large data bases*, pages 853–864, 2005.

[7] VanDiggelen F. Gnss accuracy: Lies, damn lies, and statistics. *GPS World*, pages 26–32, 2007.

[8] JR. G. DAVID FORNEY. The viterbi algorithm. *Electrical Engineering*, pages 268–278, 1973.

[9] U. Gather and V. Schultze. Robust estimation of scale of an exponential distribution. pages 327–341, 2001.

[10] Terry Griffin, Yan Huang, and Shawn Seals. Routing-based map matching for extracting routes from gps trajectories. In *Geo '11 Proceedings of the 2nd International Conference on Computing for Geospatial Research and Applications*.

[11] Britta Hummel. Map matching for vehicle guidance. *Dynamic and Mobile GIS: Investigating Space and Time*, 2006.

[12] John Krumm, Julie Letchner, and Eric Horvitz. Map matching with travel time constraints. In *SAE World Congress (2007 : Detroit, Mich.)*, pages 1–7.

[13] Peter Lamb and Sylvie Thiébaux. Avoiding explicit map-matching in vehicle location. In *Intelligent Transportation Systems (ITS-99)*.

[14] Kuien Liu, Yaguang Li, Fengcheng He, Jiajie Xu, and Zhiming Ding. Effective map-matching on the most simplified road network. *SIGSPATIAL '12 Proceedings of the 20th International Conference on Advances in Geographic Information Systems)*, pages 609–612, 2012.

[15] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. *ACM SIGSPATIAL GIS*, 2009.

[16] Dennis Luxen. OSRM Website. `http://map.project-osrm.org/`, 2014. [Online; accessed 10.01.2015].

[17] Oleksiy Mazhelis. Using recursive bayesian estimation for matching gps measurements to imperfect road network data. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference*, pages 1492–1497.

[18] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *GIS '09 Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 336–343, 2009.

[19] 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*, pages 312–328, 2007.

[20] Nagendra R. Velaga, Mohammed A. Quddus, and Abigail L. Bristow. Developing an enhanced weight-based topological map- matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, pages 672–683, 2009.

[21] Hong Wei, Yin Wang, George Forman, Yanmin Zhu, and Haibing Guan. Fast viterbi map matching with tunable weight functions. *SIGSPATIAL '12 Proceedings of the 20th International Conference on Advances in Geographic Information Systems)*, pages 613–616, 2012.

[22] Haiqiang Yang, Shaowu Cheng, Huifu Jiang, and Shi An. An enhanced weight-based topological map matching algorithm for intricate urban road network. In *Intelligent*

and Integrated Sustainable Multimodal Transportation Systems Proceedings from the 13th COTA International Conference of Transportation Professionals (CICTP2013), pages 1670–1678, 2013.

[23] Jian Yang and Liqiu Meng. Feature selection in conditional random fields for map matching of gps trajectories. *Progress in Location-Based Services 2014*, pages 121–135, 2014.

[24] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guang-Zhong Sun. An interactive-voting based map matching algorithm. In *MDM '10 Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, pages 43–52, 2010.