Segmenting trajactory data using the Brownian bridge movement model

Wouter Jongeling

August 21, 2019

Master thesis

ICA-3997626

Utrecht University

Abstract

We have implemented a model-based segmentation framework and used it with the Brownian bridge movement model to segment a trajectory based on transport mode. We find that this setup is not very effective at discovering transport mode transitions, but in the process, we discover that this method is much more effective at segmenting based on other movement patterns such as curves and acceleration than on transport mode.

Contents

1 Introduction

Increasingly many devices, like cars, mobile phones and other smart devices are capturing GPS trajectory data. The wide availability of this data has offered opportunities to study the behaviour of humans, objects and animals. One of the tasks that is often performed on a dataset is *segmentation*. Segmentation is the process of dividing a trajectory into a set of subtrajectories, such that every subtrajectory holds a certain property that is not held by the entire trajectory. This can help reveal patterns that were not visible for the trajectory as a whole, or let us reason more precisely about the behaviour portrayed in the trajectory. Segmentation often goes hand in hand with *classification*, which is the process of assigning a class label to each of these segments.

One of the properties that it is useful to segment a trajectory by is the transportation mode. A hypothetical person may cycle to a train station, then walk to a platform, take a train to another station, walk to the bus station, take a bus to another bus stop, and finally walk to their destination. This hypothetical trajectory can be divided into six segments, characterised by their transportation mode (bike, walk, train, walk, bus, walk). But the GPS device used to record this trajectory cannot make this distinction, so the entire trajectory is recorded as a single unit. A segmentation algorithm can be used to split this trajectory into 6 subtrajectories, each corresponding to their transportation mode. Specifically, the algorithm has to detect the points where the trajectory switches between transport modes. Then, a classification algorithm can assign the correct transport mode to each segment.

In this thesis, we aim implement a segmentation algorithm to find such transitions between different transport modes. We will not implement a classification algorithm, so we are only interested in the segmentation points: the points at which the user transitions between transport modes.

Alewijnse et al. introduced a framework for criterion-based segmentation [1, 2]. This thesis seeks to implement this framework with the Brownian bridge movement model (Section 4), and test it on the Geolife dataset [12, 13, 14] (Section 6) to test how well the model performs on the task of segmenting trajectories by the transport mode.

We find that the Brownian bridge movement model does not handle this task very well, because it tends to detect much smaller features in the trajectory, such as a single curve, instead of detecting a longer segment, that can be assigned to a single transport mode.

2 Related work

2.1 Trajectory segmentation

Buchin et al. [5] segment a trajectory into sufficiently homogeneous segments based on spatiotemporal features such as speed, heading and curvature, while minimising the total number of segments. They define *attribute functions* for these features, and *criteria* that ensure that these attribute values are sufficiently similar within each segment. To achieve such a segmentation, they present a greedy algorithm that runs in $O(n)$ time for monotone criteria, and $O(n \log n)$ time for many other criteria. A monotone criterion is a criterion that,

when it applies to a subtrajectory $\tau' \subseteq \tau$, it also applies to any subtrajectory $\tau'' \subseteq \tau'$.

Aronov et al. [3] present an algorithm to segment a trajectory based on nonmonotone criteria. They split the problem into two subproblems. First, they compute a start-stop diagram. Then, they use this start-stop diagram to compute the optimal segmentation. They show that in general, this second step is NP-hard, but if the start-stop diagram has certain properties, there is a polynomial-time solution. They show two criteria that have such properties, and corresponding polynomial-time algorithms.

Yoon and Shahabi [11] segment trajectories such that in each segment, the object's speed is approximately constant, while at the same time removing any outliers using a pre-specified maximum speed between probes. They present 3 algorithms to perform this task. First, a top-down algorithm that recursively splits the trajectory at the point that most deviates from its estimated position if the vehicle travelled at constant speed, until this maximum deviation is less than some threshold. This runs in $O(kn \log n)$ time, where k is the number of splits, if an efficient data structure is used. Next, a bottom-up algorithm, that starts with n segments of size 1, and greedily merges two adjacent segments with the lowest merge cost until this minimum cost exceeds some threshold. This runs in $O(k \log n)$ time where k is the number of merges. And finally, they show a sliding window algorithm that starts a segment with the first probe, and keeps extending this segment until it breaks the homogeneity requirement. This runs in $O(kn)$ time.

Guo et al. [6] use probabilistic logic to segment a trajectory based on business points for delivery vehicles. These are places where the vehicle is stopping to make a delivery. They first detect all stopping points, and then aim to distinguish stopping points from traffic events like traffic lights from the business points using a probabilistic model based on both trajectory data and attributes from the vehicle, like electronic lock status and fueling tank cap status.

Patterson et al. [10] segment based on three different transport modes (bus, walk, car) using a custom Bayesian model. They also attempted to predict the user's future path based on their trajectory.

Zheng et al. also developed an algorithm to segment a trajectory based on transport mode [12]. In their algorithm, they focus on finding walking segments first, as people often walk a bit when transitioning between transport modes. Then, they use a decision tree on three features (heading change rate, stop rate and velocity change rate) to classifiy the remaining segments according to their transport mode. They also build a spatial index and a graph of common change points from their training data, to further help classify segments by identifying potential locations of interest such as bus stops, train stations, parking garages, etc.

Moosavi et al. [9] developed a method to compute segments in a trajectory based on driving patterns. First, they used a Markov Model to transform the trajectory to *Probabilistic Movement Dissimilarity (PMD)* space. This is a measure of how unlikely the measured behaviour of the driver at that point is, if it were part of the same segment. Very unlikely behaviour, like a sudden acceleration or deceleration will have a high PMD score, and likely behaviour will have a low score. Finally, they use a Dynamic Programming approach to divide this PMD-signal into segments, such that each segment has uniform behaviour.

2.2 Brownian bridge movement model

Horne et al. developed a movement model based on Brownian motion for modelling an animal's motion [7]. Since continuous tracking of the animal is impossible, the only representation of their trajectory available is a discrete set of n probes. The Brownian bridge movement model aims to model the movement of the animal in between these probes. Section 4 describes how the Browian bridge movement model works in detail.

Kranstauber et al. [8] later expanded upon this model by allowing different parts of the trajectory to be modelled by different diffusion coefficients in what they call the *dynamic Brownian bridge movement model* (dBBMM). This way, different features in a single trajectory can be modelled more accurately by using different diffusion coefficients. They showed that this approach always yields equal or better results than the BBMM with a single diffusion coefficient. This is also the model that we will use, as described in Section 5.1.

Buchin et al. used the Brownian bridge movement model to detect movement patterns in animals [4]. They considered the following patterns:

Encounters, where two distance between two creatures at any time is less than some threshold; *avoidance* and *attraction*, where two creatures visit the same locations, rarely at the same time (attraction), or often at the same time (attraction); *regular visit*, where an area is visited with a regular period; and two different definitions for following behaviour.

3 Definitions and notation

This section describes the defintions and notation used in the rest of this thesis.

Trajectory A trajectory is a sequence of n *probes*, where a probe i is a tuple (lat_i, lon_i, t_i). Here, lat_i and lon_i are the latitude and longitude as recorded by the GPS device at time t_i for each probe i ($0 \leq i < n$). In Section 5.1, we describe how we renumber the probes in order to construct the Brownian Bridges.

Brownian Bridge The i-th Brownian bridge is notated as $\tau[i, i+1]$ and consists of probes $[2i, 2i+1, 2i+2]$. Section 5.1 describes how these bridges are constructed, and explains the numberings.

Segment A segment is a consecutive subsequence of a trajectory that holds a certain property.

Segmentation A segmentation is a set of segments that together cover the entire trajectory, but do not overlap, except at the endpoints. The optimal segmentation is the segmentation that matches the segmentation of the ground truth. In our case, this is the segmentation based on transport mode as labelled by the Geolife user (see Section 6).

4 Brownian bridge movement model

Brownian motion is a term from physics, describing the random motion of particles in a liquid or a gas. It is also referred to as a Wiener Process. The Brownian bridge model describes Brownian motion where both endpoints are anchored to predetermined locations. How far the motion strays away from these locations is defined by the model parameter σ_m^2, called *diffusion coefficient*.

If we set the Brownian motion of a moving object to be at point A at time 0, and at point B at some time T, we can construct a Brownian bridge to interpolate the location of this object between these two points. The probability distribution of the location at time t is normally distributed around the mean and variance at that time. These can be described by the following formulae:

$$\mu(t) = A + \tfrac{t}{T}(B - A) \qquad\qquad \sigma^2(t) = t(1 - \tfrac{t}{T}\sigma_m^2)$$

$$(X_t | X_0 = A, X_T = B) \sim \mathcal{N}(\mu(t), \sigma^2(t)) \tag{1}$$

To illustrate this, figure 1 shows the location distributions for $t = \frac{1}{4}T$ and $t = \frac{1}{2}T$ for a hypothetical Brownian bridge. The mean $\mu(t)$ is along the straight line between A and B, proportional to $\frac{t}{T}$. $\sigma^2(t)$ is also dependent on $\frac{t}{T}$, with the highest variance when $\frac{t}{T} = \frac{1}{2}$, and the lowest variances near $t = 0$ and $t = T$.

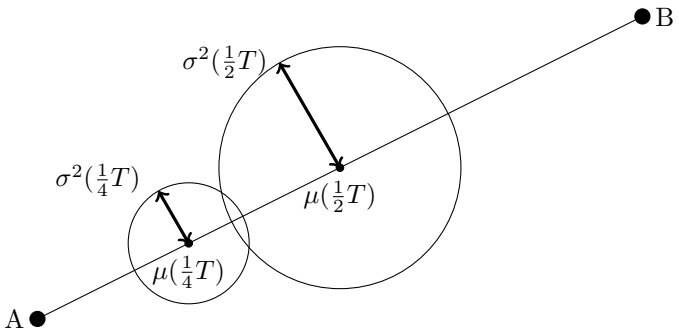

Figure 1: Two location distributions for a Brownian bridge (adapted from [1])

The probability density function over the entire bridge will look something like in Figure 2 (depending on the specific parameters). The two peaks are the endpoints A and B of the bridge. This is the sum over the probability distribution of all possible times $0 \leq t \leq T$, or formally:

$$P(X|A, B) = \int_{t=0}^{T} P(X, t|A, B)dt \tag{2}$$

This section described how a single Brownian bridge works. Section 5.1 will describe how an entire trajectory can be modelled using a sequence of Brownian bridges.

Figure 2: Probability density function for a Brownian bridge (taken from [7])

5 Description of the algorithm

We can model a trajectory using a series of Brownian bridges. This process is described in section 5.1. Each of these bridges must be assigned a diffusion coefficient. This is the model parameter that describes the shape of the Brownian bridge. We describe how we can compute the likelihood of a candidate diffusion coefficient being used to describe a given Browian bridge. The optimal diffusion coefficient for a bridge is the one that has the highest likelihood.

Then, we describe the information criterion we use to prevent oversegmentation in Section 5.2.

Next, we describe how we select a representative set of candidate diffusion coefficients for the Brownian bridges (Section 5.3), and finally, we describe the dynamic programming algorithm that assigns a diffusion coefficient to each Brownian bridge (Section 5.4). Each consecutive set of Brownian bridges with the same diffusion coefficient is considered a separate segment.

The optimal segmentation for the entire trajectory is the one that minimises the information criterion. This function aims to maximise the likelihood of the diffusion coefficient for each individual Brownian bridge, while at the same time not oversegmenting by giving each bridge its own segment. For the convenience of the reader, this section will use the same notation as [1, 2].

5.1 Modelling a trajectory using Brownian bridges

The trajectory probes are used to create a sequence of Brownian bridges. Each bridge consists of 3 probes, where the first and third probe form the endpoints of the bridge, and the second probe is used as an attribute of the bridge. The endpoint of one bridge is also the first point of the next bridge. The probes that are used as endpoints - these are the even numbered probes in the original trajectory - are renamed and renumbered as $\tau(i)$. The odd numbered probes are renumbered as $\tau^b(i)$. The diffusion coefficient of each bridge is also identified by its corresponding bridge id. With this notation, bridge i consists of probes $\tau(i)$, $\tau^b(i)$ and $\tau(i+1)$, and has diffusion coefficient $\sigma_m^2(i)$. See Figure 3 for a

graphical example.

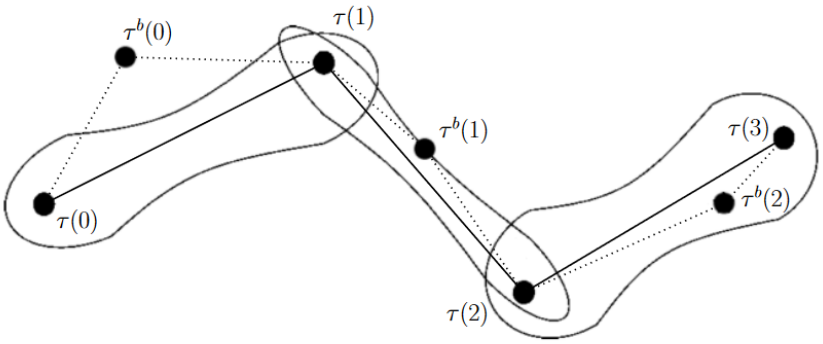

Figure 3: A sequence of Brownian bridges is constructed. Note the overlap of the endpoints of the bridges, and the renumbering of the probes (from [1])

.

If there is one probe left over after constructing the last bridge, this probe is simply discarded in the bridge creation process. The total number of bridges is therefore $\lfloor \frac{1}{2} \cdot (|T| - 1) \rfloor$.

The likelihood of diffusion coefficient s_m^2 for Brownian bridge $\tau[i, i+1]$ is:

$$L(\sigma_m^2 | \tau[i, i+1]) = \frac{1}{2\pi\sigma^2(i)} \cdot \exp\left(\frac{-||\tau^b(i) - \mu(i)||}{2\sigma^2(i)}\right) \tag{3}$$

where, substituting the parameters of the bridge into the formulas in in section 4, with $\alpha = \frac{t_{i+1} - t_i^b}{t_{i+1} - t_i}$:

$$\mu(i) = \tau[i] + \alpha(\tau[i+1] - \tau[i]) \qquad \sigma^2(i) = (t_{i+1} - t_i)\alpha(1-\alpha)\sigma_m^2$$

Since the bridges are assumed to be independent, likelihood of a diffusion coefficient given *multiple* Brownian bridges $\tau[i', i]$ equals the product of the likelihood for the individual bridges:

$$L(\sigma_m^2 | \tau[i', i]) = \prod_{j=i'}^{i-1} L(\sigma_m^2 | \tau[j, j+1]) \tag{4}$$

$$\log(L(\sigma_m^2 | \tau[i', i])) = \sum_{j=i'}^{i-1} \log(L(\sigma_m^2 | \tau[j, j+1])) \tag{5}$$

The optimal diffusion coefficient for the whole trajectory is the one that maximises the likelihood over $\tau[0, n]$, and therefore also maximises the log-likelihood.

This is the standard Brownian bridge movement model as described by Horne et al. [7]. That model models an entire trajectory with the same diffusion coefficient. The *dynamic Brownian bridge movement model* (dBBMM) introduced by Kranstauber et al. [8] allows for multiple diffusion coefficients in the same trajectory. We will use this model to describe the trajectory, because not only

can it allow us to describe a trajectory more accurately [8], it also lets us use the resulting diffusion coefficients in the segmentation process. Every consecutive stretch of bridges with the same diffusion coefficient will be considered a new segment.

5.2 Information criterion

If we simply compute the optimal diffusion coefficient for each bridge, we will end up with n different segments of length 1. To combat this, we need to artificially limit the number of segments in our result. This can be done by introducing a penalty factor for each new segment. Candidate segmentations with a larger number of segments might have a higher total likelihood, but they will be penalised by a higher penalty factor as well. Therefore, we will not compare the candidate segmentations by just their likelihood, but by an *Information criterion*, which includes this penalty factor. For this purpose, we use the Bayesian information criterion (BIC). The general form of the BIC is:

$$BIC = k \cdot \ln(n) - 2 \cdot \ln(L) \tag{6}$$

where n is the size of the input, in our case the number of probes, L is the likelihood of the candidate model (thus $\ln(L)$ is the loglikelihood), and k is the number of parameters. In our case, k is the number of segments in the segmentation. After experimenting with this information criterion (Section 7), we found it to yield inadequate results, so we altered the function to allow us to tweak the performance of the algorithm by introducing a new factor p, which we can adjust in our experiments (Section 7.3):

$$IC = k \cdot p \cdot \ln(n) - 2 \cdot \ln(L) \tag{7}$$

5.3 Selection of the candidate diffusion coefficients

We will try to select the optimal diffusion coefficient for each bridge from a discrete set of *candidate diffusion coefficients*. In order to get the most representative set of candidate diffusion coefficients, we devised the following process:

First, we compute the optimal diffusion coefficient for each individual bridge. Remember, the likelihood of a diffusion coefficient σ_m^2 for bridge $\tau[i, i+i]$ is:

$$L(\sigma_m^2 | \tau[i, i+i]) = \frac{1}{2\pi\sigma^2(i)} \cdot \exp\left(\frac{-||\tau^b(i) - \mu(i)||}{2\sigma^2(i)}\right) \tag{8}$$

We can compute the optimal value of σ_m^2 by taking the derivative. For the purpose of readability of this section, we will substitute $p = ||\tau^b(i) - \mu(i)||$ and $q = (t_{i+1} - t_i)\alpha(1-\alpha)$.

$$
\begin{aligned}
L(\sigma_m^2|\tau[i,i+i]) &= \frac{1}{2\pi q\sigma_m^2} \cdot \exp\left(\frac{-p}{2q\sigma_m^2}\right) \\
\frac{dL(\sigma_m^2|\tau[i,i+i])}{d\sigma_m^2} &= \frac{1}{(2\pi q\sigma_m^2)^2} \cdot 2\pi q\sigma_m^2 \cdot \frac{p}{2q(\sigma_m^2)^2} \cdot \exp(\frac{-p}{2\sigma_m^2}) \\
&\quad - \frac{1}{(2\pi q\sigma_m^2)^2} \cdot \exp(\frac{-p}{2q\sigma_m^2}) \cdot 2\pi q \\
&= \frac{p\exp(\frac{-p}{2q\sigma_m^2})}{4\pi q^2(\sigma_m^2)^3} - \frac{\exp(\frac{-p}{2q\sigma_m^2})}{2\pi q(\sigma_m^2)^2}
\end{aligned} \tag{9}
$$

Equating this to 0 gives us the value of σ_m^2 where the likelihood is maximised.

$$
\begin{aligned}
\frac{dL(\sigma_m^2|\tau[i,i+i])}{d\sigma_m^2} &= 0 \\
\frac{p\exp(\frac{-p}{2q\sigma_m^2})}{4\pi q^2(\sigma_m^2)^3} - \frac{\exp(\frac{-p}{2q\sigma_m^2})}{2\pi q(\sigma_m^2)^2} &= 0 \\
\frac{p\exp(\frac{-p}{2q\sigma_m^2})}{4\pi q^2(\sigma_m^2)^3} &= \frac{\exp(\frac{-p}{2q\sigma_m^2})}{2\pi q(\sigma_m^2)^2} \\
\frac{p}{2q\sigma_m^2} &= 1 \\
\sigma_m^2 &= \frac{p}{2q}
\end{aligned} \tag{10}
$$

Replacing p and q again gives us:

$$
\sigma_m^2 = \frac{||\tau^b(i) - \mu(i)||}{2(t_{i+1} - t_i)\alpha(1-\alpha)} \tag{11}
$$

This means that we can compute the optimal diffusion coefficient for a bridge in $O(1)$ time, or for all bridges in $O(n)$ time. Then, we sort these n values and take the $\frac{1}{k} \cdot (i + 0.5)$th percentiles from that set (where $0 \leq i < k$) to get k candidate diffusion coefficients that are most representative of the entire set. We will call this set of candidate diffusion coefficients V, and index it with v, where $0 \leq v < k$ and $k = |V|$.

5.4 Dynamic programming algorithm

With these techniques, we can now compute the optimal segmentation using a dynamic programming algorithm. We will compute Opt_i, the optimal segmentation of $\tau[0, i]$, in increasing order of i. In order to do this, we need to maintain a dynamic programming table $OptLastFixed$, where each $OptLastFixed_{i,v}$ stores the optimal segmentation of $\tau[0, i]$, where the last segment uses diffusion coefficient v.

When we compute $OptLastFixed_{0,v}$, the resulting segmentation will always contain one segment of length one. The IC of this segmentation is simply the likelihood of diffusion coefficient v plus the penalty factor:

$$
IC_{0,v} = -2 \cdot log(L(v|\tau[0,1])) + p \tag{12}
$$

After computing this for all v, it is trivial to determine Opt_0: this is simply the v that gave the lowest IC for $OptLastFixed_{0,v}$.

For all $OptLastFixed_{i,v}$ with $i > 0$, there are two possibilities to consider.

The first option is that Opt_{i-1} (this is the optimal segmentation from the previous step) is appended with a new segment consisting of only bridge i. This will henceforth be called the *Append* option. In this case, $IC_{i,v} = IC(Opt_{i-1}) + p - 2\ln(L_i(v))$.

The second option is $OptLastFixed_{i-1,v}$ (the previous segmentation that ended with the same diffusion coefficient), but with the last segment extended with bridge i. This will be referred to as the *Extend* option. In this case, $IC_{i,v} = IC_{i-1,v} - 2\ln(L_i(v))$.

Alewijnse et al. have proved [1, 2] that these are the only two possibilities for $OptLastFixed_{i,v}$, and that therefore this greedy choice is correct.

$OptLastFixed_{i-1,v}$ is chosen to be the option that has the lowest IC. Note that if $v = Opt_{i-1}.v$, the *Append* option will never be chosen, because the only difference between the two options is then the addition of the extra penalty factor by adding a new segment.

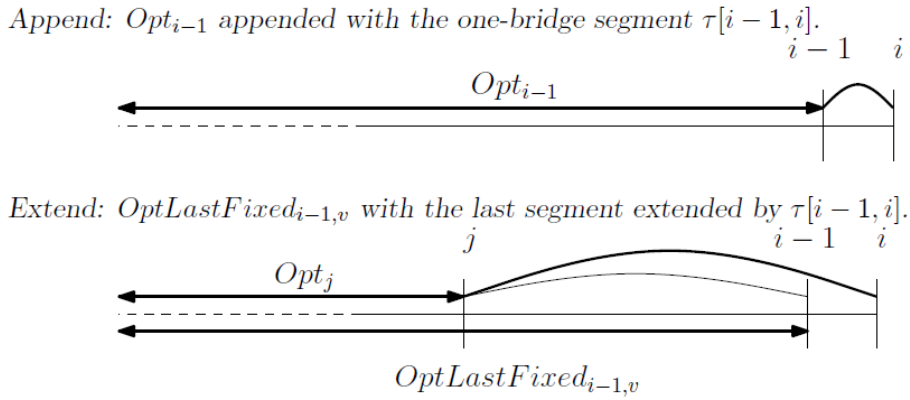

Figure 4: $OptLastFixed_{i,v}$ is always one of exactly two possibilities (taken from [1])

In each cell of the table, we store the total IC of the segmentation up to that point, along with a reference to which previous segmentation it expanded upon. When the entire table has been computed, we can take the entry with the lowest IC in the final row (Opt_{n-1}), and backtrack to find the complete segmentation. The algorithm will return the indices and lengths of the segments in number of bridges. In order to convert this to the number of probes, we multiply the values by two.

5.5 Runtime

The selection of the candidate diffusion coefficients takes $O(n)$ time. The dynamic programming algorithm takes $O(n \cdot |V|)$ time. Backtracking at the end takes $O(n)$ time. Therefore, the entire process can be done in $O(n \cdot |V|)$ time.

6 Geolife data set

To test the segmentation algorithm, we used version 1.3 of the Geolife dataset [12, 13, 14]. This dataset was recorded as part of Microsoft's Geolife project, which was started in 2007 by Zheng et al. It contains trajectories as recorded by 182 unique users during a five year period (april 2007 - august 2012). The entire dataset contains 17,621 trajectories.

73 users have labelled their trajectories with a transportation mode. Using this labelling, we could determine the ground truth segmentation of each of these trajectories. Unfortunately, most trajectories only contained one transportation mode, so they were unsuitable to test our segmentation algorithm. In order to get a good view of the behaviour of our algorithm, we decided to only consider the trajectories that had three or more segments in their ground truth. Figure 5 shows the distribution of the number of segments in the ground truth of the remaining trajectories. Most of the remaining trajectories have 3 segments. The highest number of segments in a single trajectory was 15.

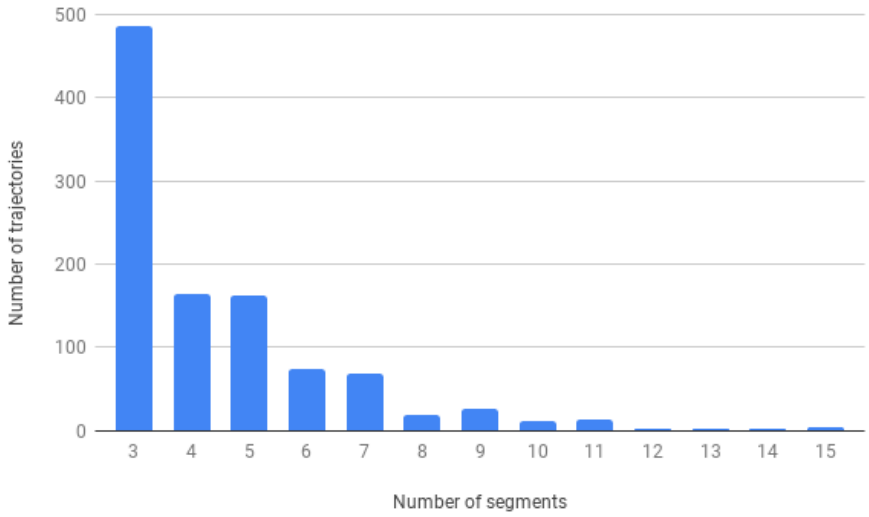

Figure 5: Number of segments in the ground truth

399 trajectories had between 0 and 1,000 probes (Figure 6a). There were 36 trajectories with more than 10,000 probes, and the highest number of probes in a trajectory was 56,781.

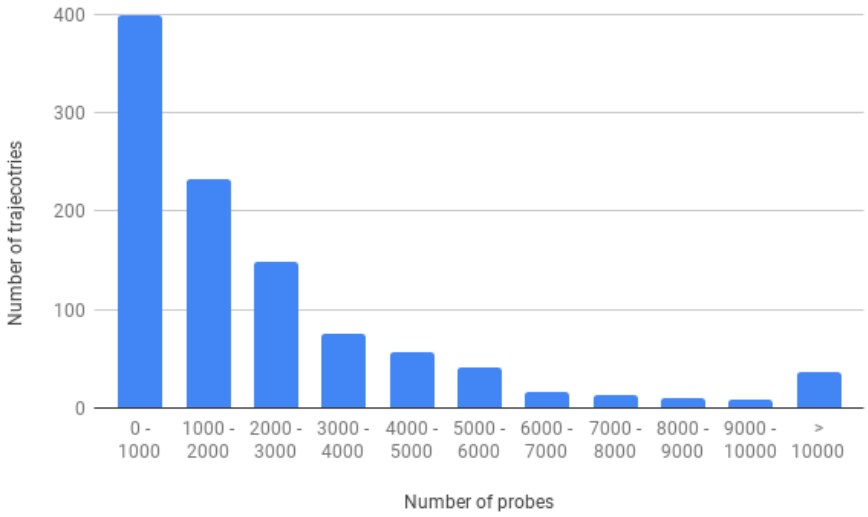

Figure 6: Number of probes per trajectory

Because of the lower number of trajectories of larger sizes, and for ease of visualisation in the Experiments section, we only show the results on trajectories with fewer than 5000 probes. After making this selection, there are 911 trajectories left with an average of 1,531 probes.

7 Experiments

Our algorithms behaviour can now be defined by two parameters: the penalty factor p and the number of candidate diffusion coefficients k.

7.1 Measure of segmentation quality

In order to evaluate the performance of our algorithm, we need a way to compare segmentations to the ground truth. For this purpose, we introduce three measures. A segmentation can be represented by the starting probe index of each segment. We denote the segmentation as $S_{0\ldots n}$ and the ground truth as $G_{0\ldots m}$.

The first metric Q_s measures how far away our segmentation points are from the real segmentation points. For each S_i, we find the nearest index j in G, and sum the differences. Obviously, this measure will have a bias towards segmentations with a low number of segments. In fact, every segmentation with only one segment (starting index 0) will have a score of 0.

The second metric Q_g does the exact opposite: it measures how far away the ground truth's segmentation points are from our segmentation points. For each G_j, we find the nearest index i in S, and sum the differences. This measure has a bias towards segmentations with a large number of segments. In fact, if a segmentation has n different segments (of length 1), the G_g score will always be 0.

Both of these measures have their obvious biases, but they can both give an insight into the behaviour of the algorithm. The best segmentation is the one that minimises both. Therefore, the third metric Q_t is a weighted average of Q_s and Q_g ($0 \leq \alpha \leq 1$). We will start the experiments with $\alpha = 0.5$.

$$Q_s = \sum_{i=0}^{n} \min_{j=0,\dots,m} |S_i - G_j| \tag{13}$$

$$Q_g = \sum_{j=0}^{m} \min_{i=0,\dots,n} |S_i - G_j| \tag{14}$$

$$Q_t = \alpha \cdot Q_g + (1 - \alpha) \cdot Q_s \tag{15}$$

7.2 Number of diffusion coefficients

In our first set of experiments, we will look at the effect of the number of diffusion coefficients k on the quality of our segmentation. We expect a higher k to produce more segments per segmentation, and a lower (better) quality score. It is easy to see that if $k = 1$, every bridge will be assigned the same diffusion coefficient, and the result will be a segmentation consisting of one segment of length n, where n is the number of bridges. This means that $Q_s = 0$, and $Q_t = Q_g$.

We run the algorithm with $k \in \{1, ..., 10\}$ on all trajectories, and look at the average number of segments and quality of the resulting segmentations. We find that the results greatly depend on the size of the trajectory, so we show the results grouped by number of probes in the trajectory.

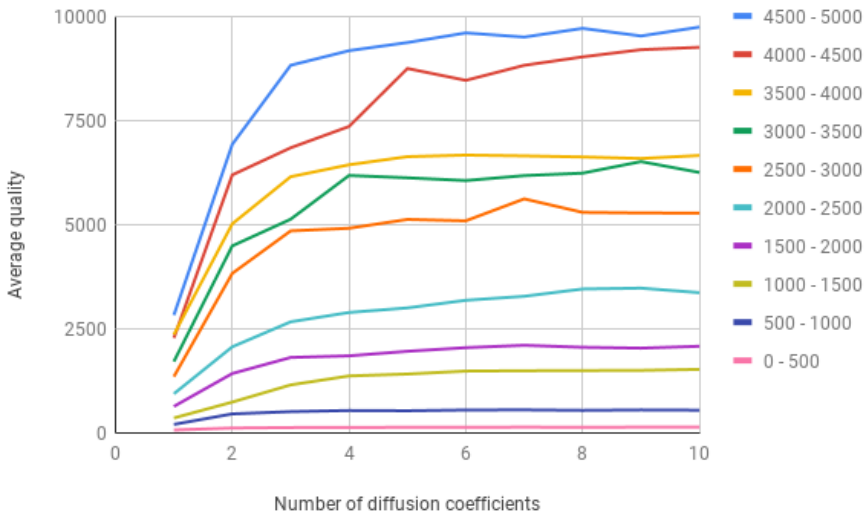

Figure 7: Average quality per number of diffusion coefficients, grouped by number of probes

Shockingly, we find that increasing the number of diffusion coefficients increases the quality score dramatically for these low numbers of diffusion coef-

ficients. Since this quality score needs to be minimised, this is not a desirable outcome. Closer examination of Q_s and Q_g reveals that Q_s shows the same rapid growth as Q_t, while Q_g stays relatively constant if $k > 1$. This leads us to believe that this increasing error rate is caused by a rapidly increasing number of segments. Figure 8 shows the *segment overshoot* per bucket of trajectories: this is the difference between the number of segments in our segmentation and the true number of segments. This confirms our suspicion, and suggests that we need to increase the penalty factor of the algorithm in order to decrease the number of segments generated. Additionally, since the overshoot seems to depend on the number of probes in the trajectory, we will likely need to choose a variable penalty factor depending on the length of the trajectory. Both figure 7 and 8 show that the effect of increasing k diminishes after $k = 4$. Therefore, we will use $k = 5$ in the next set of experiments.

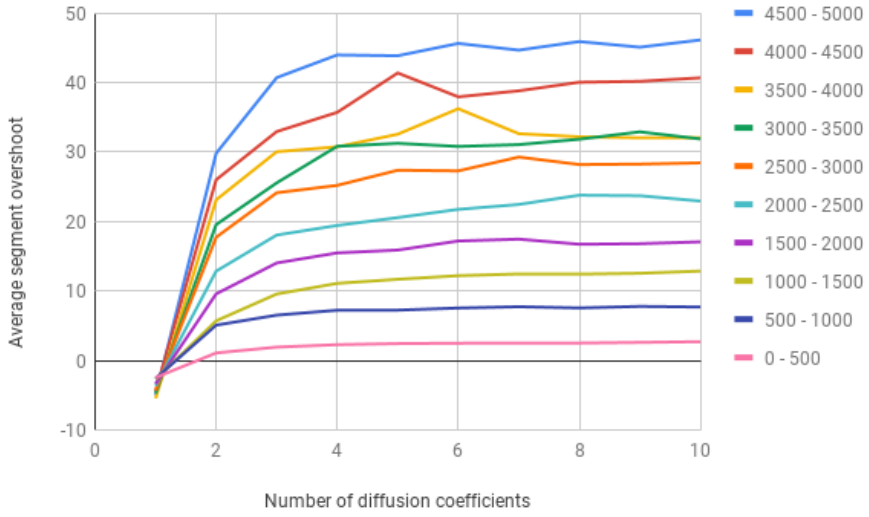

Figure 8: Average segment overshoot per number of diffusion coefficients, grouped by number of probes

7.3 Penalty factor

Next, we look at the effect of the penalty factor p on the quality of the segmentation. We choose $p \in \{1, ..., 20\}$, and again, we split the trajectories into buckets. We expect that a higher penalty factor will decrease the number of segments, and the average quality score will decrease, because the segmentation boundaries can better reflect the ground truth. Figure 9 shows the results. To demonstrate the difference between the different quality measures, Figure 10 shows the average quality according to all three quality measures for trajectories between 2500 and 3000 probes.

We can see that a higher penalty factor does indeed yield a lower quality score. Looking at the average segment overshoot (Figure 11), we also see that for each bucket, there is a point where the average segment overshoot is zero.

Again, the penalty factor corresponding to this point seems to be higher for larger trajectories.

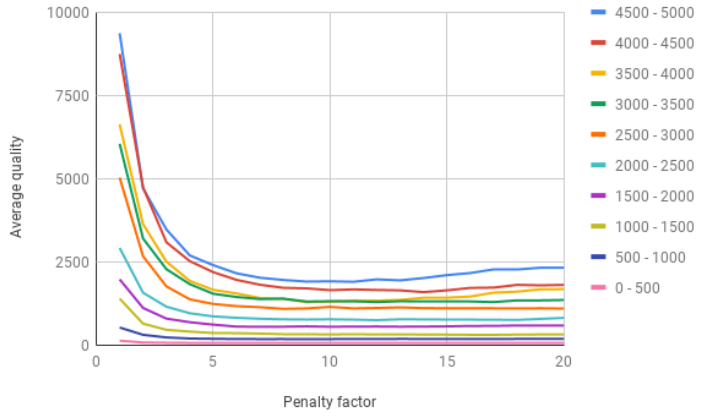

Figure 9: Average quality per penalty factor, grouped by number of probes

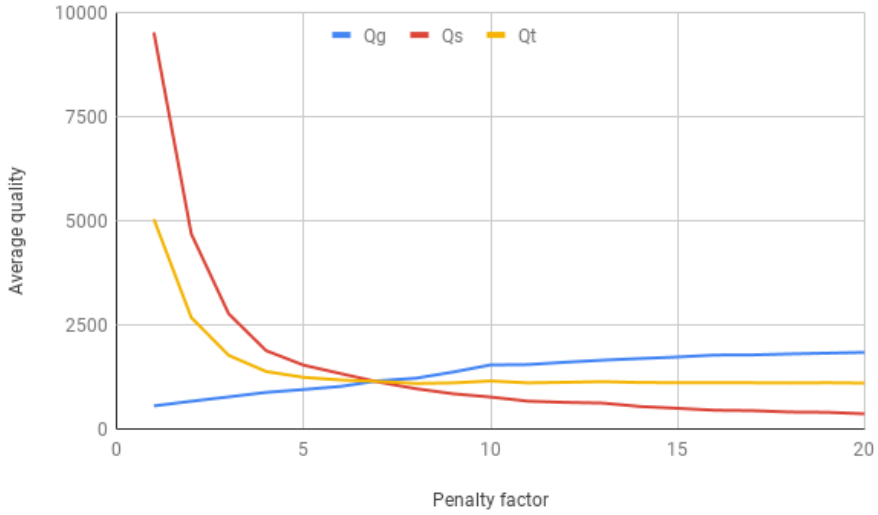

Figure 10: Average quality per penalty factor, for three quality measures, for trajectories between 2500 and 3000 probes

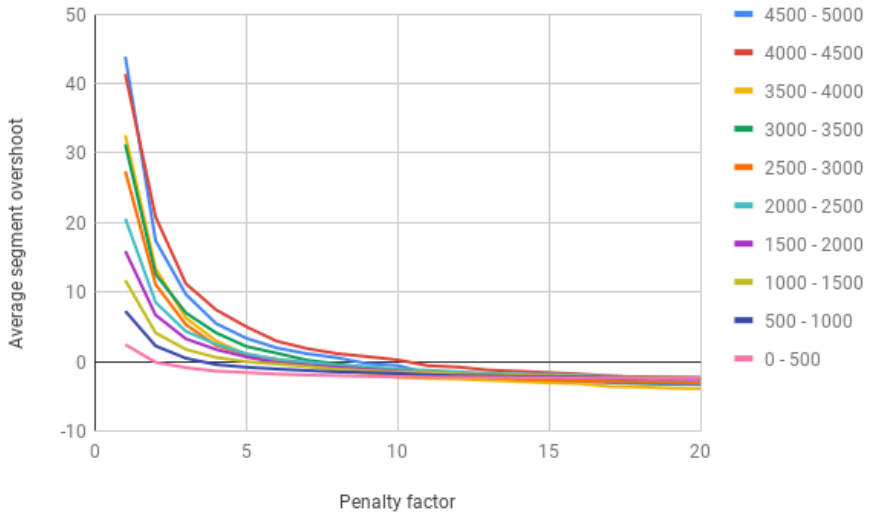

Figure 11: Average segment overshoot per penalty factor, grouped by number of probes

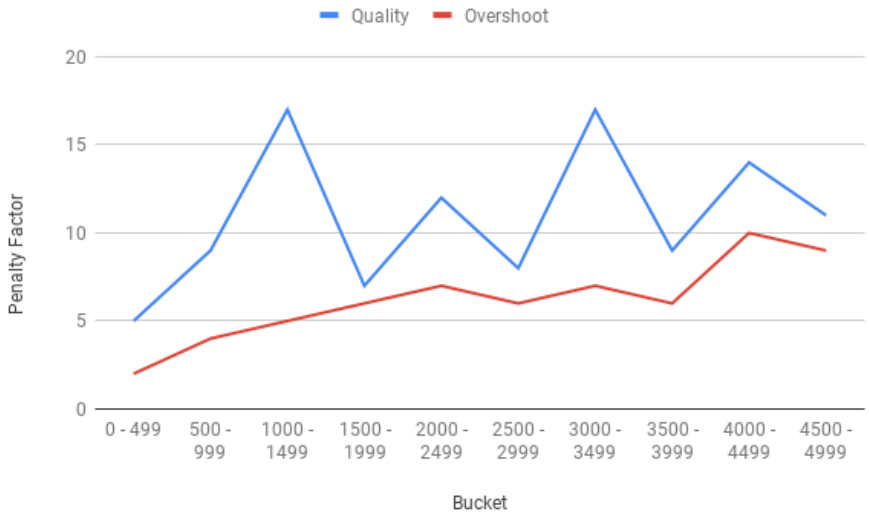

Figure 12: Optimal penalty factor per bucket

Figure 12 shows the optimal penalty factor for each bucket. Optimal here means lowest quality score, or a segment overshoot closest to zero. This figure lets us make two very important observations. First, it seems that the optimal penalty factor increases as the trajectory size increases. This means that we cannot use a constant penalty factor, but that the penalty factor must be a function of the size of the trajectory. The second observation is that the optimal penalty factor for minimising the quality score is consistently higher than the

optimal penalty factor for bringing the segment overshoot to zero. This suggests that the segmentations with the lowest quality score have fewer segments than the ground truth. Our data confirms this: the average segment overshoot when using the penalty factor that yielded the lowest quality score is -1.6. And even the resulting lowest quality scores are still in the thousands, which is not as low as we were hoping for.

If we now look at the resulting segmentations on a map, we can study how the algorithm performs, and where it might go wrong. We find a few examples where segmentation points appear at or near train stations and traffic lights. We suspect these are caused by the acceleration and deceleration of the vehicles at these locations. But one can imagine that the frequency of such situations is far higher than the number of transport mode switches.

7.4 Alpha factor

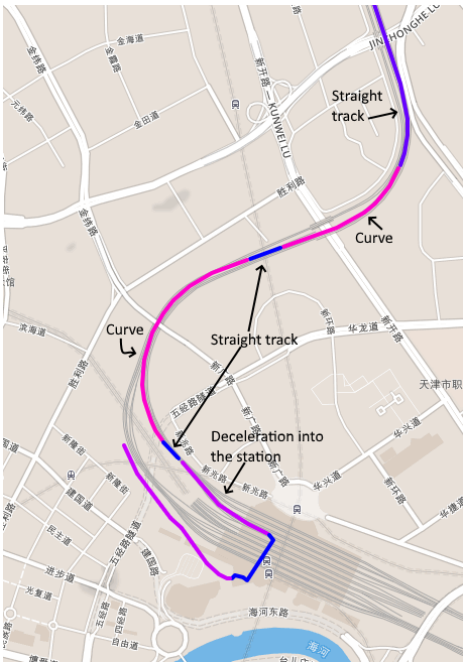

Figure 13: Example of a part of a segmentation with penalty factor 1

Up to now, we have assumed that our segmentation should contain exactly one segment per transport mode. If we relax this constraint to allow for multiple consecutive segments to represent a single transport mode, we can use the observation above to our advantage. Recall that when we used penalty factor 1 in earlier experiments, the resulting segmentation had a very high segmentation overshoot.

Figure 13 shows the last part of a segmentation with penalty factor 1. In this visualisation, the value of the diffusion coefficient is indicated using a gradient scale from blue to red. Blue represents the lowest possible diffusion coefficient from the candidate set, and red represents the highest possible diffusion coefficient. Obviously, the number of segments here is too large, but we can clearly see now that the segmentation has identified deceleration and curves (both sections with a higher diffusion coefficient) and sections with a constant speed and straight track (blue: lowest diffusion coefficient).

Other parts of the trajectory show the same behaviour: segments around stations that have a higher diffusion coefficient, and segments between stations with the lowest diffusion coefficient. If we were to use this approach, the next step would be to combine consective segments that correspond the the same transport mode into segment. This would likely be the task of a classification algorithm.

This means that we need to make sure that at least the true segmentation points are in our resulting segmentation, but we will allow additional segmen-

tation points as well. The quality measure Q_g that we described in section 7.1 does exactly this. So we increase α to give more weight to the Q_g component of the quality score. We have used $\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Now, we see that if $k = 1$, Q_t increases as we increase α. But for all other k, Q_t becomes lower as α increases. An interesting discovery is that for certain values of α (0.8 and 0.9), there is a clear minimum quality when $k = 2$. Figure 14 shows the average quality if $\alpha = 0.8$. This suggests that a distinction between a 'high' diffusion coefficient (curves, deceleration) and a 'low' diffusion coefficient (straight, constant speed) is sufficient to segment based on this criterion.

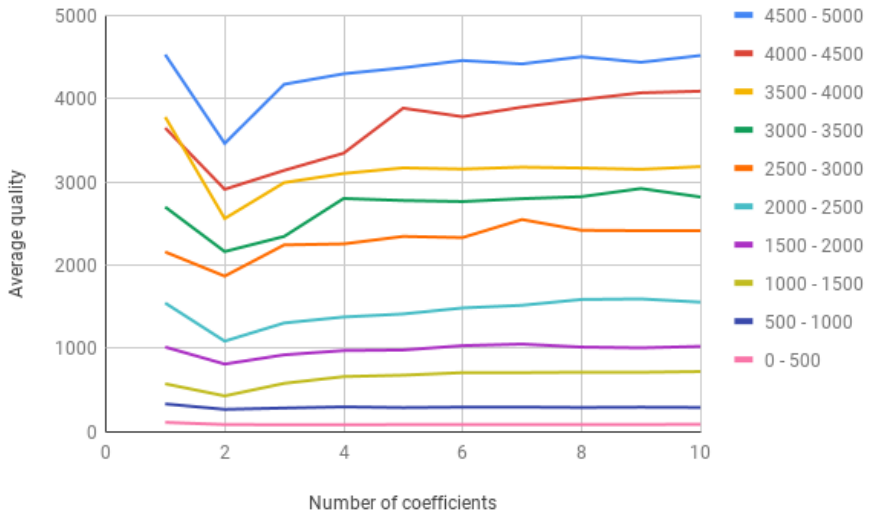

Figure 14: Average quality per number of diffusion coefficients if $\alpha = 0.8$

7.5 Construction of the Brownian bridges

As discussed in section 4, the bridges are constructed by taking the even probes as bridge endpoints, and the odd probes as attribute of the bridge. But this is a rather arbitrary choice. If we were to use the odd probes as bridge endpoints and the even probes as attributes, the resulting likelihood functions might be entirely different. Consider the situation in figure 15.

In the left situation, there are two bridges, both with most likely diffusion coefficient 1. Just by looking at the bridges, one would not be able to tell that a turn has taken place here. In the right situation, the middle bridge has a much higher most likely diffusion coefficient, because the middle point is far away from the straight line between the two endpoints. To test whether this difference has a large influence on the segmentation quality, we run the experiment again with $k = 5$, $p = 10$ and $\alpha = 0.5$, but this time, we discard the first probe of every trajectory. This is comparable to if the GPS device had been switched on several seconds later. This has the effect that every probe that used to be an even point is now an odd point, and vice versa. We will indicate the Q_t from these experiments with Q_t^s (s for skip). Then, we computed the quality ratio QR between the original experiments and these skip-experiments: $QR = \frac{q_t^s}{q_t}$. If

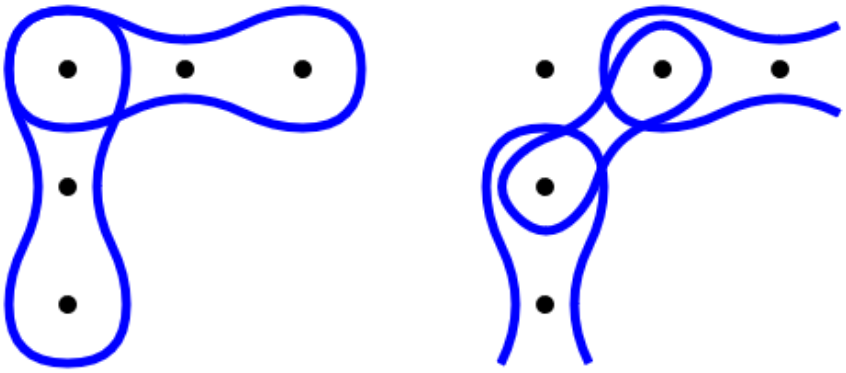

Figure 15: Two ways of building the Brownian bridges

this $QR = 1$, the quality is the same; if $QR > 1$, q_t^s is higher; and if $QR < 1$, q_t is higher.

Min	10%	Q1	Med	Q3	90%	Max
0.088	0.896	0.999	1.000	1.007	1.205	4.158

Table 1: Several percentiles of QR

Table 1 shows that most trajectories had a QR of roughly 1, but there were also some significant outliers. Over 10% of the trajectories had a 20% higher score, and another 10% had a 10% lower score. There were 33 trajectories that had a $QR > 2$ and 18 with a $QR < 0.5$. This demonstrates the way the bridges are constructed can have a signficant influence on the quality of the segmentation.

Further inspection of the outliers showed that this discrepancy is often caused if one of the two segmentations has a very low score, and the other introduces one or more new segmentation points far away from the ground truth, causing a very large increase in score. Table 2 shows an example of the trajectory with the lowest QR, and the two segmentations. This trajectory has 3343 probes. The vertical alignment shows the closest segmentation point in the ground truth, the distance to which is used to calculate the score. The segmentation points in the skip row have been incremented by 1 to make them comparable to the ground truth.

GT	0	354		2110		2214	3343	Q_g	Q_s	Q_t
Regular	0	352	1246	1476	2148		3343	106	1538	822
Skip	0	352		2146			3343	106	38	72

Table 2: Segmentation points for a trajectory with a very low QR

We can see that by skipping the first probe, we lost two segmentation points, which caused Q_s, and therefore Q_t, to drop considerably.

8 Conclusion, discussion and future work

After performing these experiments we can conclude that our setup by itself was not suitable for accurately segmenting based on transport mode. Although this method didn't perform our desired segmentation task very well, we did find that it was a lot more effective at segmenting based on other features such as turns and accelerations. We suggest several improvements that could be made to achieve better results.

Perhaps the Brownian bridge movement model is not very suitable for segmenting trajectories of vehicles because of their restrictions to a road network. This model has been used before to model the movement of particles in a gas or liquid, or the behaviour of animals like birds. Such movement is not usually restricted by such human constructions as train tracks and traffic lights. If one could find another model that better described movement by a vehicle on a road/track network, that model could be used with this framework instead of the Brownian bridge model to improve segmentation accuracy. This model also seems to be somewhat sensitive to the way the probes are split up (as discussed in Section 7.5).

Using the approach described in Section 7.4, a classification algorithm would need to be implemented to assign a transport mode to each segment, and then combine segments with the same transport mode. One could use training data to help identify features for each transport mode, and use these to classify the segments in the testing data. Training data can also be used to learn the probability of transitioning between any two transport modes, as Zheng et al. did in [12].

If the underlying road network is known, one can use a map-matching algorithm and use the associated map data to aid in the classification process.

References

[1] Sander P. A. Alewijnse. A framework for trajectory segmentation by stable criteria and brownian bridge movement model. Master's thesis, Eindhoven University of Technology, 2013.

[2] Sander P. A. Alewijnse, Kevin Buchin, Maike Buchin, Stef Sijben, and Michel A. Westenberg. Model-based segmentation and classification of trajectories. *Algorithmica*, 80(8):2422–2452, 2018.

[3] Boris Aronov, Anne Driemel, Marc J. van Kreveld, Maarten Löffler, and Frank Staals. Segmentation of trajectories on nonmonotone criteria. *ACM Trans. Algorithms*, 12(2):26:1–26:28, 2016.

[4] Kevin Buchin, Stef Sijben, T. Jean Marie Arseneau, and Erik P. Willems. Detecting movement patterns using brownian bridges. In *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*, pages 119–128, 2012.

[5] Maike Buchin, Anne Driemel, Marc J. van Kreveld, and Vera Sacristán. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spatial Information Science*, 3(1):33–63, 2011.

[6] Sini Guo, Xiang Li, Wai-Ki Ching, Dan A. Ralescu, Wai-Keung Li, and Zhiwen Zhang. GPS trajectory data segmentation based on probabilistic logic. *Int. J. Approx. Reasoning*, 103:227–247, 2018.

[7] Jon S. Horne, Edward O. Garton, Stephen M. Krone, and Jesse S. Lewis. Analyzing animal movements using brownian bridges. *Ecology*, 88(9):2354–2363, 2007.

[8] Bart Kranstauber, Roland Kays, Scott D LaPoint, Martin Wikelski, and Kamran Safi. A dynamic brownian bridge movement model to estimate utilization distributions for heterogeneous animal movement. *Journal of Animal Ecology*, 81(4):738–746, 2012.

[9] Sobhan Moosavi, Arnab Nandi, and Rajiv Ramnath. Discovery of driving patterns by trajectory segmentation. *CoRR*, abs/1804.08748, 2018.

[10] Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring high-level behavior from low-level sensors. In Anind K. Dey, Albrecht Schmidt, and Joseph F. McCarthy, editors, *UbiComp 2003: Ubiquitous Computing*, pages 73–89, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[11] Hyunjin Yoon and Cyrus Shahabi. Robust time-referenced segmentation of moving object trajectories. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 1121–1126, 2008.

[12] Yu Zheng, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th ACM conference on Ubiquitous Computing (Ubicomp 2008)*, September 2008.

[13] Yu Zheng, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of International conference on World Wide Web 2009*, April 2009. WWW 2009.

[14] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data(base) Engineering Bulletin*, June 2010.

A Implementation in tulib

The framework described in this thesis was implemented in *tulib*, a trajectory processing library that is being developed by a collaboration between HERE technologies, Utrecht University and Eindhoven University of Technology as part of the Commit2Data project. Commit2data is a cross-sector collaboration project between both public and private partners with the aim to further develop the usage of Big Data.

tulib is a data structure agnostic trajectory processing library, that aims to implement state-of-the-art algorithms and to be easily scalable to very large datasets. Its design is inspired by the designs of CGAL and GUDHI, two existing geometry algorithms libraries. The library is designed to be very extensible to make implementation easy for a wide range of applications. This is achieved by using design principles from generic programming, specifically template metaprogramming, for separating algorithms, data structures and geometry from one another, and delegating the responsibility of connecting them to the client code. This way, each component can be individually implemented, tested, and adapted to satisfy a large variety of use cases.

At the time of writing, the monotone segmentation algorithm by Buchin et al. [5] that was described in Section 2 was already implemented. During this thesis project, the following components were added to tulib:

Geolife import tools The file `geolife.py` has been added to preprocess the geolife data. This process filters the unlabelled probes, adds the labels to remaining probes, and converts the datetime string to a unix timestamp. The `GeolifeProbeTraits.h` and `GeolifeTrajectoryTraits.h` files contain traits classes to define the input probe type and the trajectory type.

Brownian bridge movement model The file `BrownianBridge.h` was created to model the Brownian Bridge movement model. It contains a `Model` class, that builds a collection of Brownian bridges from a collection of probes, as described in Section 5.1. It also contains a Maximum Likelihood Estimator class `MLE`, that computes the optimal diffusion coefficients for a collection of Brownian bridges, and a `ParameterSelector` class that selects k candidate diffusion coefficients, as described in Section 5.3.

Model-based segmentation framework The file `Segmentation.h` now contains a class `ModelBasedSegmentation` that implements the dynamic programming framework as described in Section 5.4. The table `dp_table` is the dynamic programming table $OptLastFixed_{i,v}$. After computing each row of the table, only the last Opt_i is maintained, in the variable `min_ic`. This is used to compute the next row of the table.

Segmentation quality measure The segmentation quality measure as described in Section 7.1 was implemented in `SegmentationQuality.h`. This function computes the sum of the distances between each point in the first parameter to the nearest point in the second parameter. Therefore, the order of the parameters determines whether Q_s or Q_g is computed.