g-RIPS Sendai 2022

Mitsubishi-A Group

Final Report

Authors:
Tomoya Akamatsu[1]
Gabriel Gress[2]
Katelynn Huneycutt[3]
Seiya Omura [4]

Mentors:
Dr. Shunsuke Kano[+]
Dr. Masashi Yamazaki[*]

[1] Osaka University
[2] University of New Mexico
[3] Ohio State University
[4] Nagoya University
[+] Academic Mentor, Tohoku University, RACMaS
[*] Industrial Mentor, MITSUBISHI Electric Corp

August 17, 2022

Contents

1 Introduction **2**

2 Problem Summary and Statement **2**

3 Background **3**
3.1 Geometric Model . . . 3
3.2 Hidden Markov Model . . . 4
3.2.1 Application to Map-Matching . . . 5

4 Mathematical formulation for our Approach **5**

5 Wasserstein Distance Method **7**
5.1 A brief introduction of (discrete) optimal transport theory . . . 7
5.2 The case in which input data is only the trajectory coordinates and timestamps . . . 8
5.3 Incorporating speed and direction information . . . 10
5.3.1 Our motivation . . . 10
5.3.2 Our strategy . . . 10
5.3.3 The definition of weights including speed and direction information . . . 11
5.3.4 Briefly explanation of route determination in Figure 5.13 using Wasserstein method . . . 12
5.4 Future problems . . . 14

6 Methods by Physical Analogy **15**
6.1 "Electric" Method . . . 15
6.1.1 Details of method . . . 15
6.1.2 Observations . . . 16
6.2 "Harmonic oscillator" method . . . 17
6.2.1 Details of method . . . 18
6.2.2 Observations . . . 19

7 Implementation **20**
7.1 Preprocessing Routes . . . 21
7.1.1 Restricting to a Sub-graph . . . 21
7.1.2 Obtaining Candidate Routes . . . 22
7.1.3 Candidate Route Interpolation . . . 22
7.2 Wasserstein Distance Implementation . . . 22
7.3 Electrical Method and Harmonic Oscillator Implementation . . . 23
7.4 Evaluation (Error) Method . . . 24
7.5 Preliminary Results . . . 25
7.6 Computational Complexity . . . 25
7.7 Metric-based Algorithm Framework . . . 26
7.8 Data Fusion . . . 26
7.9 Other Utilities . . . 28

8 Future Work (Implementation) **28**
8.1 IMU Inclusion . . . 29
8.2 Incomplete Map Matching . . . 29
8.3 Extension to Higher Dimensions . . . 29

9 Conclusion **29**

Appendix A Geometric background of Wasserstein method **29**
A.1 Ricci curvature of graphs . . . 29
A.2 "Ricci curvature" between the trajectory and route . . . 31

1 Introduction

A fundamental question when dealing with geospatial information is, given GPS trajectory data and a road map, how can one determine which route on a map this trajectory corresponds to? This problem is called the map-matching problem. Data-driven methods, such as hidden Markov models, are currently the most prominent approach to the map-matching problem; however, they are inflexible changes in the data. In contrast, a mathematical model will have the benefit of being more adaptable to changes in the data, such as the extension from two to three-dimensional data or the inclusion of speed and angle data. Our project aims to develop new solutions to the map-matching problem, favoring mathematical formulations, rather than a data-driven models, that are stable under small perturbations in the road map and trajectory data. We propose three novel methods: Wasserstein distance method, electrical force method, and harmonic oscillator method. Each of these methods has an associated loss (objective) function whose minimum is the "matched" map. The loss function of each of the three methods employs the mathematical object for which they are named. We implement the Wasserstein distance method and electrical force method in Python and evaluate their performance using both theoretical and numerical techniques.

2 Problem Summary and Statement

Here we will describe the mathematical formulation of our problem. We will adopt the problem statement from definitions 2.1-2.4 in [CXHZ] with slight modifications. The geospatial data points will be modeled by a trajectory, see Definition 2.1, and map data will be encapsulated by a road network Definition 2.2.

Let us fix $N \in \mathbb{N}$, $N \geq 2$, (but almost everywhere we consider the case $N = 2$).

Definition 2.1 (Trajectory)**.** A *trajectory* Tr is a sequence $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ of points in $\mathbb{R}^N$ equipped with

- a sequence, $t(\mathbf{p}) = (t_1, \ldots, t_n)$, of positive numbers satisfying $t_1 < t_2 < \cdots < t_n$, called the *timestamp* of $\mathbf{p}$,
- a sequence, $\mathsf{spd}(\mathbf{p}) = (\mathsf{spd}_1, \ldots, \mathsf{spd}_n)$, of positive numbers called the *speed* of $\mathbf{p}$ (optional),
- a sequence, $u(\mathbf{p}) = (u_1, \ldots, u_n)$, of unit vectors in $\mathbb{R}^N$, called the *direction* of $\mathbf{p}$ (optional).

Definition 2.2 (Road Network)**.** A *road network* (also known as map) is a directed graph $G = (V, E)$ consists of the set V (resp. E) of vertices (resp. edges) with an embedding $\phi : |G| \to \mathbb{R}^N$ of the geometric realization $|G|$ of G. We will identify G and the image $\phi(|G|)$ by ϕ as long as there is no confusion.

Definition 2.3 (Local Road Network)**.** A *local road network* is a directed connected subgraph of $G = (V, E)$.

Definition 2.4 (Route)**.** A *route* r on a road network $G = (V, E)$ is a sequence of connected edges $(e_1, e_2, \ldots, e_n) \subset E$, i.e. the head of e_i coincides with the tail of e_{i+1} for each $i = 1, 2, \ldots, n-1$. Let R denote the set of all routes.

Definition 2.5 (Candidate Routes)**.** For the local road network graph as H of the road network G, we define

$$\mathcal{CR}_H = \mathcal{CR} := \{\text{routes on a local road network graph } H\},$$

Definition 2.6 (Map-Matching)**.** Given a road network $G = (V, E)$ and a trajectory Tr, the map-matching, $\mathcal{MR}_G(Tr)$, is the route that is the argument of the minimum of some function $L : \mathcal{CR} \to \mathbb{R}^+$, called the *loss function*.

In the map-matching problem, the trajectory and road network are given as input data. Preprocessing can narrow the scope of the problem to compare candidate routes on the local road network. The main goal of this project is to find a suitable loss function such that $\mathcal{MR}_G(Tr)$ is the route that is the minimal distance from the actual route traveled by a vehicle or pedestrian taken concerning a chosen metric (see § 7.4 for details on chosen metrics). A flow chart of our approach to the map-matching problem can be seen in Figure 2.1.

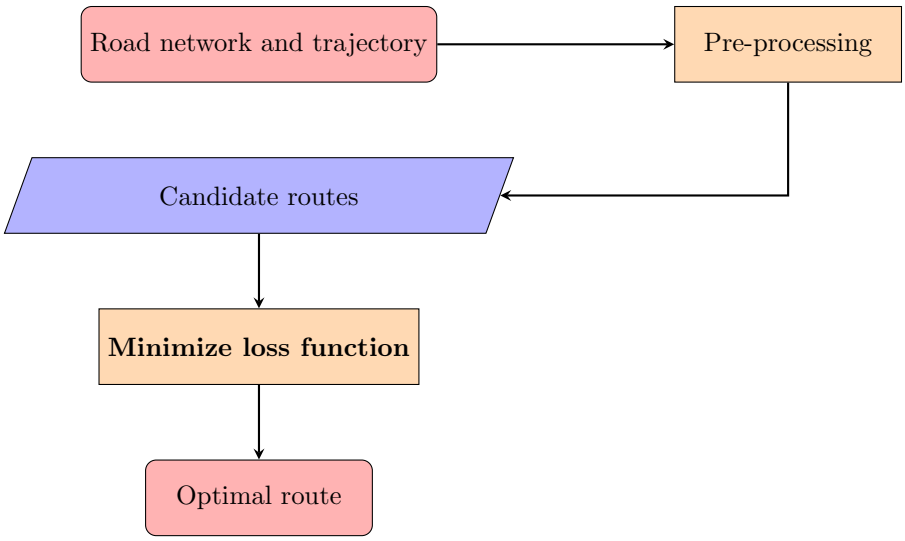


Figure 2.1: Steps of the map-matching process

3 Background

In this section, we will review three existing map-matching methods, the point-to-point method, the point-to-curve method, and the Hidden Markov Model method, and evaluate their strengths and weaknesses. Surveys of these and many other existing map-matching algorithms are available in [CXHZ, QON].

3.1 Geometric Model

Descriptions of several geometric methods including point-to-point, point-to-curve, and curve-to-curve methods can be found in [BK]. The simplest geometric map-matching algorithm is the point-to-point method (for this method only, consider a route to instead be a sequence of vertices in the road network). Suppose there are n trajectory points, p_i with $1 \leq i \leq n$ and m vertices v_j with $1 \leq j \leq m$. The procedure for the point-to-point method is given in [BK] to be

1. For each trajectory point, p_i, compute the distance from p_i to v for each $v \in V$,
2. Find $v^* \in V$ such that the euclidean distance between p_i and v^* is minimal.
3. Let the route found be the sequence of the v^*'s found for each p_i, $1 \leq i \leq n$.

Similarly, the procedure for the point-to-curve method is

1. Project each trajectory point onto each edge in the road network,
2. Compute the distance between each trajectory point and all of its projections,
3. Determine which edge minimizes this distance for each trajectory point
4. Let the route found be the sequence of the edges found in step 3.

Geometric methods are easily implemented and have low time complexity [BK]. Techniques such as creating Voronoi diagrams can be used to further decrease computational time. For example, one can partition a subset of $\mathbb{R}^2$ based on which points are closer to a given vertex on the graph than all others. This partition could be computed once and used to map-match different trajectories on the same road network. See Figure 3.2 for an example of a Voronoi diagram of a road network. The black lines and circles in the figure correspond to the road network while the blue lines partition the space into regions whose points are closest to the given vertex.

However, these geometric methods, can be sensitive to measurement errors and are highly dependent on the network structure [BK]. Bernstein and Kornhauser presented the example in Figure 3.3 to demonstrate this. Suppose the sequence of red points labeled x_1, x_2, x_3 is our trajectory and the lines and circles represent the vertices and edges of the underlying road network. Clearly, the trajectory appears to be closest to the path from vertex v_4 to v_5, but the closest node to trajectory point x_2 is v_2.

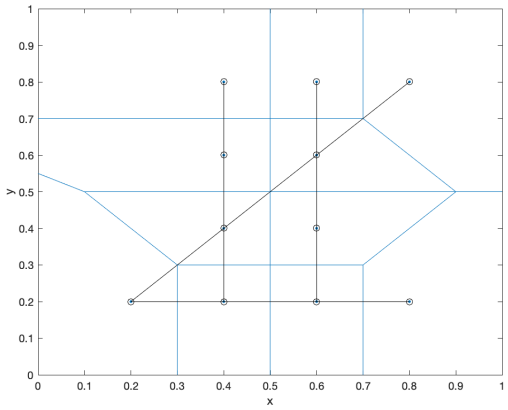


Figure 3.2: Voronoi Example

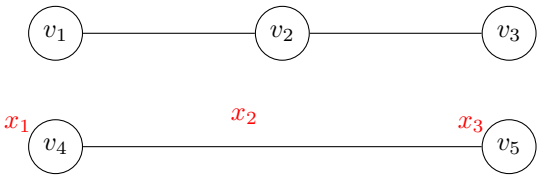


Figure 3.3: Geometric Method Error

3.2 Hidden Markov Model

The use of Hidden Markov Models (HMM) is one of the most popular approaches to the map-matching problem [CXHZ]. An open source example of a map-matching algorithm using HMM is GraphHopper [GH].

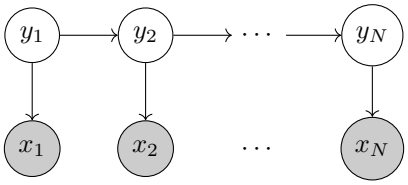


Figure 3.4: Hidden Markov Model (HMM)

A Markov chain is a probabilistic model for sequential events subject to the condition that the probability of a given event depends on the on the previous event alone, i.e. it is a sequence of random variable $z_1, \ldots, z_n$ satisfying

$$p(z_n|z_1, \ldots, z_{n-1}) = p(z_n|z_{n-1}).$$

A Hidden Markov Model assumes that observations, $x_1, x_2, \ldots, x_n$ are generated by a Markov chain of unobserved states $y_1, \ldots, y_n$, seen in Figure 3.4. The joint probability of the observed and unobserved states is

$$p(x_1, x_2, \ldots, x_n, y_1, \ldots, y_n) = p(y_1)p(x_1|y_1)\prod_{i=2}^{n} p(y_i|y_{i-1})p(x_i|y_i).$$

The probability $p(x_i|y_i)$ is called the emission probability, $p(y_i|y_{i-1})$ is the transition probability, and $p(y_1)$ is the initial distribution. If there are a finite number of states each x_i and y_i can take on, then we can form emission, transition, and initial distribution matrices. Each probability is a parameter in our model. Once these parameters are established, one of several existing algorithms can be used to compute the probabilities of some sequences of y_i values occurring given the sequence of observed x_i's. This method can be used to find the sequence of events with the highest probability given a sequence of observations.

Example 3.1. A simple application of an HMM can be used to determine how busy a teacher is, given the observed lecture quality. Take the observed states $x_1, \ldots, x_n \in \{\text{good, bad}\}$ to be the quality of the lecture on days $1, \ldots, n$, and the unobserved states to be $y_1, \ldots, y_n \in \{\text{busy, not busy}\}$. In this example, the emission matrix, A, and transition matrix, B, are given below based on the quantities decided in Figure 3.5:

$$A = \begin{bmatrix} .7 & .3 \\ .6 & .4 \end{bmatrix}, \qquad B = \begin{bmatrix} .2 & .8 \\ .9 & .1 \end{bmatrix}.$$

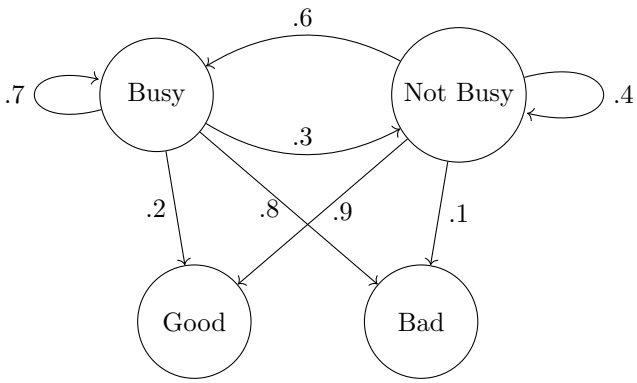


Figure 3.5: Transition and Emission probabilities for Example 3.1.

3.2.1 Application to Map-Matching

From the survey from Chao [CXHZ], HMM models for map-matching have the following setup. The observed quantities in the HMM for the map-matching problem are the sequence of geospatial data points and the hidden variables are the possible edges on the road network. The transition probabilities describe the likelihood of the route containing some edge, given the previous edge, but the emission probabilities describe the probability of observing a trajectory point given some edge in the road network. The choice of these parameters differentiates existing HMM models for map-matching. After the parameters are chosen, one can compute the route in the road network with the highest probability given the observed trajectory. Unfortunately, this approach to map-matching is limited, because a two-dimensional hidden Markov model for map-matching cannot be extended to include three-dimensional model or be changed to include speed information without training on the new data altogether.

4 Mathematical formulation for our Approach

We introduce the two methods each employing one of the following mathematical tool and physical concept:

- Wasserstein Distance,
- Coulomb forces and harmonic oscillator.

We will give our situation settings used in Section 5 and Section 6, with mathematical formulations.

First, we explain how to handle the GPS error. Preprocessing can reduce the effect of GPS error, but it cannot eliminate it. Although the average value of the GPS error can be obtained, the value of the error at each trajectory point is often not known. However, for simplicity, we impose and discuss here a rather strong setting in which the GPS error at each point is obtained. The error Err is used to mean that the true location is **likely** to exist within radius Err of the location where the trajectory was observed. For simplicity, we handle the error Err in the sense that there is **always** a true position within a radius Err. In addition, in areas with low signal reception, such as skyscrapers, GPS data is less reliable, so speed and direction

information is obtained from the IMU[1] on the mover[2]. In this case, the GPS data has a large error rate, but the accuracy of the speed and direction information remains high. We make the following assumptions taking these into account.

Assumption 4.1 (The GPS error)**.**

- Set the *GPS error* as $\mathsf{Err} : \mathbf{p} \to \mathbb{R}_{\geq 0}$. Then, suppose that the *spherically-symmetric probability measure* γ_p is given such that $\operatorname{supp}\gamma_p = \overline{B}\big(p; \mathsf{Err}(p)\big) \coloneqq \big\{v \in \mathbb{R}^N \;\big|\; d_{\mathbb{R}^N}(v,p) \leq \mathsf{Err}(p)\big\}$. We assume that p is truly located at v with (conditional) probability $\gamma_p(v)$ (see Figure 4.6).
- Suppose that **there are NO errors for both the speed and direction information**.

Remark 4.2.

- **On spherically-symmetric probability measure:** For example, consider the Gaussian measure parameterized so that the total measure is close enough to 1 in the bounded domain $B\big(p; \mathsf{Err}(p)\big)$. One can then cut off the measure by $B\big(p; \mathsf{Err}(p)\big)$ and normalize it so that the total measure is 1.
- **Validity of the error:** Notice that for any $p \in \mathbf{p}$, there exists $e \in E$ such that $B\big(p; \mathsf{Err}(p)\big) \cap e \neq \emptyset$.

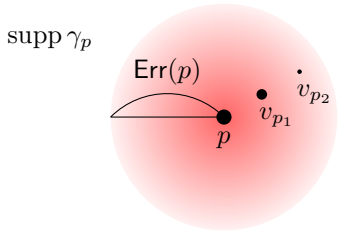


Figure 4.6: $\operatorname{supp}\gamma_p$ which represents the true region of existence of p. The mover exists at v_{p_i} with probability $\gamma_p(p_i)$ for $i = 1, 2$. then $\gamma_p(p) > \gamma_p(v_{p_1}) > \gamma_p(v_{p_2})$ holds.

If we try to handle GPS errors more rigorously, we face the following problem.

Problem 4.3 (Renormalization of $(\gamma_p)_{p\in\mathbf{p}}$)**.** Although in Remark 4.2, we defined $\operatorname{supp}\gamma_p$ as a ball in $\mathbb{R}^N$ for each $p \in \mathbf{p}$, this is unreasonable, given that the true location of the mover cannot exist outside of the road (see Figure 4.7). Hence, $\operatorname{supp}\gamma_p$ should be restricted to $\operatorname{supp}\gamma_p \cap E$ $(\subset \mathbb{R}^N)$ and renormalized so that the total probability of being there is 1. However, since we do not consider the width of the road in our setting, a high-level mathematical theory such as geometric measure theory would be needed to justify such a renormalization. We, therefore, used a simplified setup, as in Remark 4.2, in this project. This renormalization will be an important procedure when considering the width of the road (see Figure 4.8).

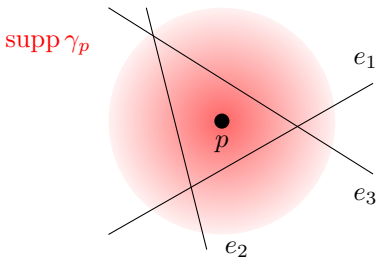


Figure 4.7: The situation where $\operatorname{supp}\gamma_p$ and edges e_1, e_2, e_3 are intersecting. Since the width of the edges is not considered in our setup, the true position of p is the whole $\operatorname{supp}\gamma_p$.

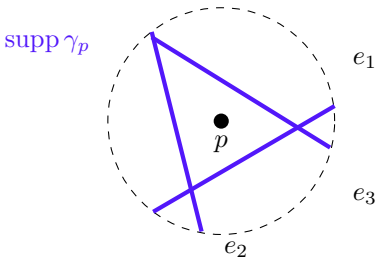


Figure 4.8: The renormalized $\operatorname{supp}\gamma_p$ by restricting it to where the mover can be located in $\operatorname{supp}\gamma_p$. This renormalization will be necessary especially when considering the width of the road.

[1] Inertial Measurement Unit

[2] see https://www.onosokki.co.jp/HP-WK/products/keisoku/vehicle/lc8_principle.htm, https://www.digikey.jp/ja/articles/use-inertial-measurement-units-to-enable-precision-agriculture (both articles are in Japanese).

It is unreasonable to use $\mathbb{R}^N$-distance $d_{\mathbb{R}^N}$ in quantifying how far away a trajectory point is from the edge due to GPS error. To address this, we introduce a distance taking into account the GPS error between a trajectory point and an edge.

Definition 4.4 (The "distance" with error between $p \in \mathbf{p}$ and $e \in E$)**.** We define the *"distance" with error* $\mathsf{d}_{\mathsf{Err}}$ between $p \in \mathbf{p}$ (**with errors**) and $e \in E$ (**without errors**) as

$$\mathsf{d}_{\mathsf{Err}}(p, e) \coloneqq \int_{v_p \in B\left(p;\mathsf{Err}(p)\right)} d_{\mathbb{R}^N}(v_p, e)\, \mathrm{d}\gamma_p(v_p).$$

Remark 4.5. Note that this $\mathsf{d}_{\mathsf{Err}}$ is a quantity that describes "how far is it $p \in \mathbf{p}$ to $e \in E$ can be considered to be" and not used as a "distance function".

Observation 4.6. If $p \in e$, i.e. $d_{\mathbb{R}^N}(p, e) = 0$, then $\mathsf{d}_{\mathsf{Err}}(p, e) = 0$. It is only in this case that $\mathsf{d}_{\mathsf{Err}}(p, e) = 0$.

5 Wasserstein Distance Method

In this section, we explain the method using Wasserstein distance which comes from optimal transport theory. We will briefly describe discrete optimal transport theory (see also [FG, Sa, Vi], etc.).

5.1 A brief introduction of (discrete) optimal transport theory

Consider transporting sand from a sand pile at $x_1, \ldots, x_n$ to a hole at $y_1, \ldots, y_m$. Note that $n, m \in \mathbb{N}$ are independent. Suppose that each sand pile $x_1, \ldots, x_n$ has $\mu(x_1), \ldots, \mu(x_n)$ mass of sand, respectively, and each hole $y_1, \ldots, y_m$ can contain $\nu(y_1), \ldots, \nu(y_m)$ mass of sand, respectively. Moreover, we assume that the cost of transporting from a sand pile of x_i to a hole of y_j is linearly dependent on their distance $d(x_i, y_j)$: it costs $d(x_i, y_j)\pi(x_i, y_j)$ to transport sand of mass $\pi(x_i, y_j)$ from the sand pile at x_i to the hole at y_j. Since the sum of the mass of sand transported from each x_i equals the sum of the mass of sand placed in each hole y_j, the following holds:

$$\sum_{i=1}^{n} \mu(x_i) = \sum_{j=1}^{m} \nu(y_j). \tag{5.1}$$

For simplicity, we normalize both sides of (5.1) to be 1. Then, μ, ν can be regarded as probability measures, respectively. Optimal transport problem is the problem such minimizing total cost for transporting. In other words, we consider the following the minimizing problem.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} d(x_i, y_j)\pi(x_i, y_j) \\
\text{subject to} \quad & \pi(x_i, y_j) \geq 0 && \text{for } i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m, \\
& \mu(x_i) = \sum_{j=1}^{m} \pi(x_i, y_j) && \text{for } i = 1, 2, \ldots, n, \\
& \mu(y_j) = \sum_{i=1}^{n} \pi(x_i, y_j) && \text{for } j = 1, 2, \ldots, m.
\end{aligned} \tag{5.2}$$

We call the map π that satisfies these conditions the *optimal transport plan* or the *coupling* of μ, ν. $\Pi(\mu, \nu)$ denotes a set of all couplings of μ, ν.

Definition 5.1 ($(L^1$-$)$Wasserstein distance)**.** For probability measures μ, ν with $\operatorname{supp}\mu = \{x_1, \ldots, x_n\}$ and $\operatorname{supp}\nu = \{y_1, \ldots, y_m\}$, we define

$$W_1(\mu, \nu) \coloneqq \inf_{\pi \in \Pi(\mu,\nu)} \sum_{i=1}^{n}\sum_{j=1}^{m} d(x_i, y_j)\pi(x_i, y_j). \tag{5.3}$$

Under the appropriate conditions, W_1 is a distance function on the probability measure space. We call this *$(L^1$-$)$Wasserstein distance* between probability measures μ and ν.

A probability measure space often contains geometrical information about its underlying space. Therefore, it is useful to analyze the probability measure space using Wasserstein distance to understand the geometrical structure of its underlying space.

Remark 5.2. Although $\Pi(\mu, \nu)$ is an infinite set since the mass of sand transported can be continuously varied, this is known to be a bounded closed, i.e. compact set. Hence, the right-hand side of (5.3) attains the minimum value.

5.2 The case in which input data is only the trajectory coordinates and timestamps

First, we introduce probability measures that are determined only from the coordinate information of the trajectory points.

Definition 5.3 (Probability measures associated with the trajectory and candidate route)**.** Let $G = (V, E)$ be the local road network graph and $m \in \mathbb{N}$ be large enough. We divide the candidate route $A \in \mathcal{CR}_G$ into $m+1$ equal parts and let $a_1, \ldots, a_m$ denote the m threshold points, then set $V(A, m) \coloneqq \{a_1, a_2, \ldots, a_m\}$. Then, we define *probability measures* $\mu_{\mathbf{p}}$ *and* $\nu_{A,m} = \nu_A$ *associated with the trajectory* $\mathbf{p}$ *and* $A \in \mathcal{CR}_G$, respectively, as follows:

$$\mu_{\mathbf{p}} \coloneqq \frac{1}{n} \sum_{p \in \mathbf{p}} \delta_p, \quad \nu_{A,m} = \nu_A \coloneqq \frac{1}{m} \sum_{a \in V(A,m)} \delta_a.$$

Remark 5.4.

- **Independence of n and m:** The number of trajectory points n and the number of divisions m are independent. On the other hand, for each candidate route $A \in \mathcal{CR}$ to be compared, the number of divisions m is the same: for any $A, B \in \mathcal{CR}$, $\#\operatorname{supp} \nu_A = \#\operatorname{supp} \nu_B$.
- **Size of m:** Although the threshold points $a_1, \ldots, a_m$ is placed in the image of a discrete approximation of the route A, we still do not know exactly what the minimum magnitude of m should be (see also Problem 5.18).
- **On the definition of $\mu_{\mathbf{p}}$:** Considering the GPS error, γ_p should be adopted as the definition of $\mu_{\mathbf{p}}$ instead of δ_p at each $p \in \mathbf{p}$, but for simplicity, we defined $\mu_{\mathbf{p}}$ in this way (see also § 5.4).

Assumption 5.5. In the following, unless otherwise noted, we proceed with the discussion in this section assuming that **the dividing point of the candidate route is** m.

Examine $\mu_{\mathbf{p}}$ and ν, i.e. $\nu_{A,m} = \nu_A$ and $\nu_{B,m} = \nu_B$ in the following example as the simplest case where is it is difficult to determine the true route.

Example 5.6 (Square model)**.** Consider the case where the edges of the local road network graph ($\subset \mathbb{R}^2$) form a square, and suppose that the trajectory points $p_1, \ldots, p_n$ is observed near the diagonal as shown in Figure 5.9. Although the trajectory points are drawn in a zigzag pattern in Figure 5.9, our approach can be used for anything near the diagonal line. We assume that we know from the information of timestamps that the trajectory began near v_1 and ended near v_4. It is difficult to determine from this trajectory along whether $(v_1, v_2) \to (v_2, v_4)$ (we call this *route A*) or $(v_1, v_3) \to (v_3, v_4)$ (we call this *route B*) is the true route traveled. Therefore, we propose the Wasserstein method, which is the method of route determination based on the optimal transport theory described in § 5.1.

We will find $\mu_{\mathbf{p}}$ and $\nu_{A,m}$ and $\nu_{B,m}$ in this situation, respectively. From the definition, $\mu_{\mathbf{p}}$ can be written directly as

$$\mu_{\mathbf{p}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{p_i}.$$

Hence, we put weights of $1/n$ on each point $p \in \mathbf{p}$ as shown in Figure 5.10. To find $\nu_{A,m}$ and $\nu_{B,m}$, we first divide the route A and B into $m+1$ equal parts. These are obtained by putting weight $1/m$ on each of the m threshold points in this way, respectively (see also Figure 5.11 and Figure 5.12):

$$\nu_{A,m} = \nu_A = \frac{1}{m} \sum_{j=1}^{m} \delta_{a_j}, \quad \nu_{B,m} = \nu_B = \frac{1}{m} \sum_{j=1}^{m} \delta_{b_j}.$$

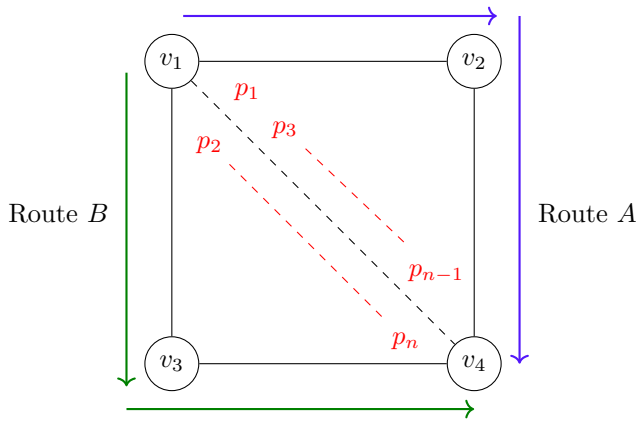


Figure 5.9: The case where the trajectory points $p_1, \ldots, p_n$ is observed near the diagonal of a square road network.

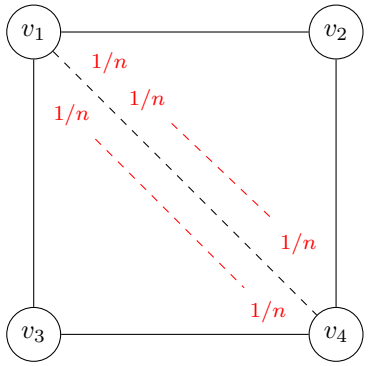


Figure 5.10: A probability measure $\mu_{\mathbf{p}}$ on the trajectory points. The weight $1/n$ is placed on the trajectory points $p_1, \ldots, p_n$ respectively.

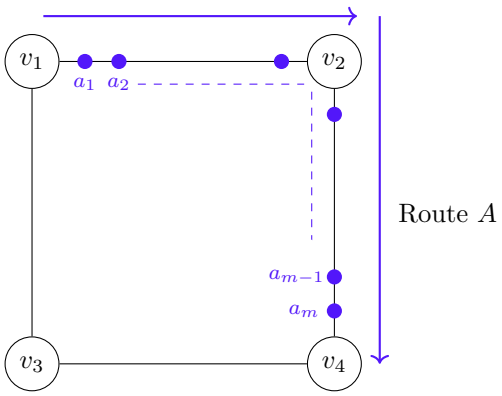


Figure 5.11: A probability measure $\nu_{A,m}$ associated with the route A. The edges (v_1, v_2) and (v_2, v_4) forming the route A are divided into $m+1$ equal parts, each with $1/m$ weight on its threshold points $a_1, \ldots, a_m$.

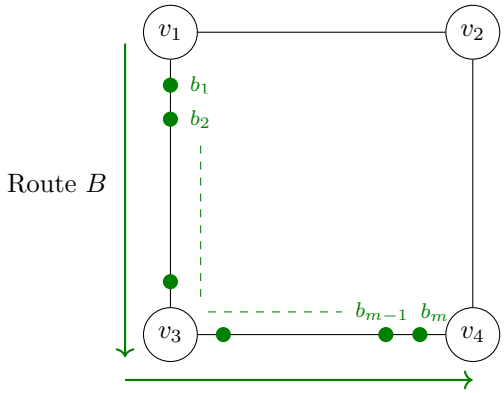


Figure 5.12: A probability measure $\nu_{B,m}$ associated with the route B. The edges (v_1, v_3) and (v_3, v_4) forming the route B are divided into $m+1$ equal parts, each with $1/m$ weight on its threshold points $b_1, \ldots, b_m$.

Remark 5.7. If m is odd and can be expressed as $m = 2\ell + 1$, then the threshold points a_ℓ and b_ℓ coincide with the nodes v_2 and v_3, respectively.

If input data is only the trajectory coordinates and timestamps, then the true route is determined by Wasserstein distance between $\mu_{\mathbf{p}}$ and ν.

Remark 5.8. Although $\varphi_m(A)$ take finite values for any $A \in \mathcal{CR}$ and any $m \in \mathbb{N}$, they might not converge as $m \to \infty$. Although we would conjecture that they converge as m approaches infinity in an even or odd state, respectively, their values may be different, i.e. their values may oscillate as $m \to \infty$.

Remark 5.9. Even if all trajectory points are on the candidate route $A \in \mathcal{CR}$, $W_1(\mu_{\mathbf{p}}, \nu_{A,m}) \neq 0$ holds in general. Therefore, we should apply this method only when it is non-trivial which route was passed such as the trajectory points are clustered near the diagonal.

Remark 5.10. Note that the trajectory $\mathbf{p}$ is a finite set, and a route is a curve, that is, a set of infinite points. The point-to-curve method projects from the trajectory point to the edges that compose the candidate route, so it is a "point-to-(point on curve)" iteration. In contrast, our method transports weights from trajectory $\mathbf{p}$ to a candidate route $A \in \mathcal{CR}$, in other words, it is a "**trajectory-to-route**" process.

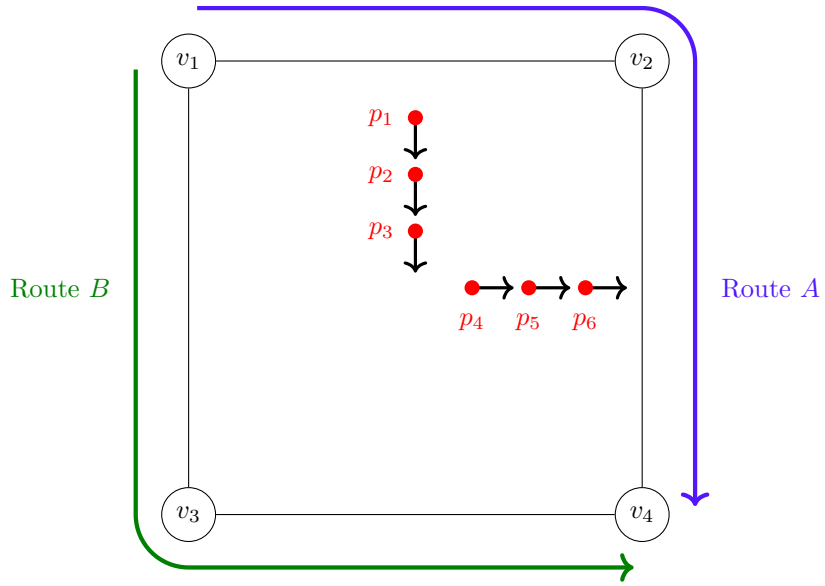


Figure 5.13: A local road network graph G. The trajectory is close to the route A, while the unit vector is similar to the shape of the route B. We should judge the route B in such a case.

5.3 Incorporating speed and direction information

In this subsubsection, we propose a method that takes speed and direction information into account.

5.3.1 Our motivation

Consider an extreme example such as Figure 5.13. Suppose that each p_i is located between edges v_1v_3 and v_2v_4 for $i = 1, 2, 3$, and each p_j is located between edges v_1v_2 and v_3v_4 for $j = 4, 5, 6$. Consider $\mu_{\mathbf{p}}$, ν_A and ν_B in this case, and determine the route from only location information using the method described in § 5.2. From the situation,

$$W_1(\mu_{\mathbf{p}}, \nu_A) > W_1(\mu_{\mathbf{p}}, \nu_B) \tag{5.4}$$

holds and We would have to select the route A. In this case, however, the GPS data is probably noisy due to skyscrapers and other factors, making the location data unreliable. Also, since direction should have a small margin of error even in such cases, we would like to determine the route from the direction information as much as possible. Thus, in this case, the true route would be the route B.

5.3.2 Our strategy

The first idea is to use speed and direction information to modify the transport method, i.e., W_1 distance. In this case, however, we would modify the optimal transport plan, not the loss function for W_1 distance. Although the modification itself may not be difficult, the modified W_1 distance that is constructed as a result is likely to be very hard to handle, in particular, the triangle inequality may no longer hold. Then, we modify the probability measures $\mu_{\mathbf{p}}$ and ν using speed and direction information. The following is a brief description of the flow of the method.

1. Place a weight on each threshold point $a \in V(A, m)$ that takes speed and direction information into account. This weight is separated into two parts: one w_A which is uniformly placed on each route and another $w_{\mathbf{p},A}$ which is affected by speed and direction.

2. According to the weights, the local road network graph is updated to a **weighted graph**. This weighted graph is a graph that is defined for each route A and we denote it as G_A.

3. Perturb $\mu_{\mathbf{p}}$ and ν_A on the weighted graph G_A by ε, where $0 < \varepsilon \ll 1$. We denote the probability measures obtained in this way as $\mu_{\mathbf{p}}^{\varepsilon}$ and ν_A^{ε}, respectively. We similarly define ν_B^{ε} for $B \in \mathcal{CR}$.

4. Then, comparing the values

$$\frac{W_1(\mu_{\mathbf{p}}^{\varepsilon}, \nu_A^{\varepsilon})}{W_1(\mu_{\mathbf{p}}, \nu_A)} \quad \text{and} \quad \frac{W_1(\mu_{\mathbf{p}}^{\varepsilon}, \nu_B^{\varepsilon})}{W_1(\mu_{\mathbf{p}}, \nu_B)},$$

determine the route with the smallest value to be the true route. In other words, we adopt the value

$$\frac{W_1(\mu_{\mathbf{p}}^{\varepsilon}, \nu_A^{\varepsilon})}{W_1(\mu_{\mathbf{p}}, \nu_A)} \tag{5.5}$$

as the quantity for route determination.

Remark 5.11 (The quantity to be used to determine route)**.** Note that since $0 < \varepsilon \ll 1$, $W_1(\mu_{\mathbf{p}}^{\varepsilon}, \nu_A^{\varepsilon})$ and $W_1(\mu_{\mathbf{p}}^{\varepsilon}, \nu_B^{\varepsilon})$ are not yet valid quantities for determining route because of the strong influence of location information. There is also a technique to make this ε larger, but in that case, the question is how to proportion it to the location information. Therefore, we adopted quantities normalized by $W_1(\mu_{\mathbf{p}}, \nu_A)$ for $0 < \varepsilon \ll 1$. This quantity (5.5) is very useful in Riemannian geometry on graphs. See Appendix A for more details.

5.3.3 The definition of weights including speed and direction information

We will put additional weighted edges in this vertex set V. Our method quantifies the distance between $\mathbf{p}$ and each route separately. Therefore, it is necessary to consider two types of weighted edges, since each of them is evaluated from a different point of view.

The first type is a "natural" weight in the sense that it decides which routes to consider.

Definition 5.12 (Natural weights for each route)**.** For each trajectory point $p \in \mathbf{p}$ and each candidate route $A \in \mathcal{CR}$, we define $w_{p,A}$, $w_{p,B} : \mathbb{R}^N \to \{0, 1\}$ as

$$w_{p,A}(v) \coloneqq \begin{cases} 1 & (v \in V(A)), \\ 0 & (\text{otherwise}), \end{cases}$$

$V(A)$ is the threshold points set of $A \in \mathcal{CR}$.

Next, we introduce a weight including speed and direction information. This weight is a quantity that depends on the input data at each trajectory point $p \in \mathbf{p}$, and is commonly defined independently of which route is considered.

We will make some preparations to define this weight.

Definition 5.13 (Weights depending on speed and direction information)**.**

- For each $p \in \mathbf{p}$ and $e \in E$, we define

$$\mathsf{C}_{p,e} \coloneqq \mathsf{spd}(p) \cdot \frac{\langle u_p, \mathsf{e} \rangle}{\exp\left(\mathsf{d}_{\mathsf{Err}}(p, e)\right)},$$

where e is the unit vector parallel to e for each $e \in E$. Then, we define $\mathsf{C}_p \coloneqq \sum_{e \in E} \mathsf{C}_{p,e}$.

- With the above preparations, we define $w_p : \mathbb{R}^N \to \mathbb{R}_{\geq 0}$ for each $p \in \mathbf{p}$ as follows:

$$w_p(v) \coloneqq \sum_{\substack{e \in E; \\ e \ni v}} \frac{\mathsf{C}_{p,e}}{\sum_{e \in E} \mathsf{C}_{p,e}} \cdot \frac{1}{\#\big(e \cap V(A)\big)}.$$

Observation 5.14 (Weighted local road network graph)**.** The weight introduced in Definition 5.12 and Definition 5.13 modifies the local road network $G = (V, E)$ to the **weighted graph** $G = (V, E, w)$. **Notice that the weights of each $e_A \in E(A)$ are all zero, so these edges vanished.** Moreover, note that the

vertex set V_A of G_A determined by the weights $w_{\mathbf{p},A}$ and $w_{\mathbf{p}}$ is $V_A = \mathbf{p} \cup V(A)$. In order to consider the perturbation (of the probability measure $\mu_{\mathbf{p}}$ associated with the trajectory) at $G = (V, E, w)$, let us now calculate the *weighted degree* $d_{p,A,m} = d_{p,A}$ of each $p \in \mathbf{p}$, which can be considered in $\#\mathcal{CR}$ types depending on the different weights $w_{p,A}$ for each $A \in \mathcal{CR}$, but the values are the same:

$$d_{p,A} \coloneqq \sum_{v \in V} \{w_{p,A}(v) + w_p(v)\} = \left(\sum_{v \in V(A)} 1 \right) + \sum_{v \in V(A)} \sum_{\substack{e \in E; \\ e \ni v}} \frac{\mathsf{C}_{p,e}}{\sum_{e \in E} \mathsf{C}_{p,e}} \cdot \frac{1}{\#\big(e \cap V(A)\big)} = m + 1.$$

This right-hand side does not depend on $A \in \mathcal{CR}$. For any $A \in \mathcal{CR}$, we have $d_{p,A} = m + 1$.

Finally, we get to the definition of ε-perturbations $\mu^{\varepsilon}_{\mathbf{p},\cdot}$ and $\nu^{\varepsilon}_{\cdot}$.

Definition 5.15 (The ε-perturbations of the probability measures associated with the trajectory and routes)**.** For each $\varepsilon \in [0, 1]$, $p \in \mathbf{p}$, candidate route $A \in \mathcal{CR}$ and $a \in V(A, m)$, we define as follows:

$$\mu^{\varepsilon}_{p,A}(v) \coloneqq \begin{cases} 1 - \varepsilon & (v = p), \\ \varepsilon \cdot \frac{w_{p,A}(x) + w_p(x)}{d_{p,A}} & (v \in V(A, m)), \end{cases} \qquad \nu^{\varepsilon}_{a}(v) \coloneqq \begin{cases} 1 - \varepsilon & (v = a), \\ \varepsilon \cdot \frac{1}{n} & (v \in \mathbf{p}), \\ 0 & (\text{otherwise}), \end{cases}$$

Next, we define the *ε-perturbations* of the probability measures associated with the trajectory and routes as follows, respectively:

$$\mu^{\varepsilon}_{\mathbf{p},A} \coloneqq \frac{1}{n} \sum_{p \in \mathbf{p}} \mu^{\varepsilon}_{p,A}, \qquad \nu^{\varepsilon}_{A} \coloneqq \frac{1}{m} \sum_{a \in V(A,m)} \nu^{\varepsilon}_{a}.$$

Remark 5.16. ν^{ε}_{A} and ν^{ε}_{A} can be simply expressed by the definition as

$$\nu^{\varepsilon}_{A}(v) = \begin{cases} \frac{1-\varepsilon}{m} & (v \in V(A, m)), \\ \frac{\varepsilon}{n} & (v \in \mathbf{p}). \end{cases}$$

Observation 5.17. It is clear from the definition that $\mu^{0}_{\mathbf{p},A} = \mu_{\mathbf{p}} = \mu^{0}_{\mathbf{p},B}$, $\nu^{0}_{A} = \nu_A$ and $\nu^{0}_{B} = \nu_B$ hold.

5.3.4 Briefly explanation of route determination in Figure 5.13 using Wasserstein method

Consider Figure 5.13 again and use the Wasserstein method to determine the route.

First, we find each C. For simplicity, suppose that $\mathsf{spd}(p)$, $\mathsf{Err}(p)$ are constant at each $p \in \mathbf{p}$ ($\mathsf{spd}(p) =: \mathsf{S}$), $\mathsf{vec}(p(i)) \perp \overrightarrow{v_1 v_3}$ for $i = 1, 2, 3$ and $\mathsf{vec}(p(j)) \perp \overrightarrow{v_1 v_2}$ for $j = 4, 5, 6$. Then, we obtain for $i = 1, 2, 3$

$$\mathsf{C}_{p_i,v_1v_3} = \mathsf{C}_{p_i,v_2v_4} = \mathsf{spd}(p_i) \cdot \frac{\langle u_{p_i}, \mathsf{e} \rangle}{\exp\big(\mathsf{d}_{\mathsf{Err}}(p,e)\big)} = \frac{\mathsf{S}}{\exp\big(\mathsf{d}_{\mathsf{Err}}(p,e)\big)}, \quad \mathsf{C}_{p_i,v_1v_2} = \mathsf{C}_{p_i,v_3v_4} = 0,$$

where $\exp\big(\mathsf{d}_{\mathsf{Err}}(p,e)\big) \coloneqq \exp\big(\mathsf{d}_{\mathsf{Err}}(p,v_1v_3)\big) = \exp\big(\mathsf{d}_{\mathsf{Err}}(p,v_2v_4)\big)$. We also obtain for $j = 4, 5, 6$

$$\mathsf{C}_{p_j,v_1v_3} = \mathsf{C}_{p_j,v_2v_4} = 0, \quad \mathsf{C}_{p_j,v_1v_2} = \mathsf{C}_{p_j,v_3v_4} = \mathsf{spd}(p_j) \cdot \frac{\langle u_{p_j}, \mathsf{e} \rangle}{\exp\big(\mathsf{d}_{\mathsf{Err}}(p,e)\big)} = \frac{\mathsf{S}}{\exp\big(\mathsf{d}_{\mathsf{Err}}(p,e)\big)},$$

where $\exp\big(\mathsf{d}_{\mathsf{Err}}(p,e)\big) \coloneqq \exp\big(\mathsf{d}_{\mathsf{Err}}(p,v_1v_2)\big) = \exp\big(\mathsf{d}_{\mathsf{Err}}(p,v_3v_4)\big)$. Then, we have for $i = 1, 2, 3$ and $j = 4, 5, 6$

$$\frac{\mathsf{C}_{p_i,v_1v_3}}{\mathsf{C}_{p_i}} = \frac{1}{2} = \frac{\mathsf{C}_{p_i,v_2v_4}}{\mathsf{C}_{p_i}}, \quad \frac{\mathsf{C}_{p_i,v_1v_2}}{\mathsf{C}_{p_i}} = 0 = \frac{\mathsf{C}_{p_i,v_3v_4}}{\mathsf{C}_{p_i}},$$
$$\frac{\mathsf{C}_{p_j,v_1v_3}}{\mathsf{C}_{p_j}} = 0 = \frac{\mathsf{C}_{p_j,v_2v_4}}{\mathsf{C}_{p_j}}, \quad \frac{\mathsf{C}_{p_j,v_1v_2}}{\mathsf{C}_{p_j}} = \frac{1}{2} = \frac{\mathsf{C}_{p_j,v_3v_4}}{\mathsf{C}_{p_j}}.$$

Suppose that we divide the routes A and B into $m = 2\ell + 1$ equal parts. Then, the weight $w_p(\cdot)$ is distributed as follows:

$$w_{p_1}(v) = w_{p_2}(v) = w_{p_3}(v) = \begin{cases} \frac{1}{2} \cdot \frac{1}{\ell+1} & (v = a_{\ell+1}, \ldots, a_{2\ell+1}, b_1, \ldots, b_{\ell+1}), \\ 0 & (\text{otherwise}), \end{cases}$$

$$w_{p_4}(v) = w_{p_5}(v) = w_{p_6}(v) = \begin{cases} \frac{1}{2} \cdot \frac{1}{\ell+1} & (v = a_1, \ldots, a_{\ell+1}, b_{\ell+1}, \ldots, b_{2\ell+1}), \\ 0 & (\text{otherwise}). \end{cases}$$

Then, it holds that $\mu^{\varepsilon}_{\mathbf{p},A} = \mu^{\varepsilon}_{\mathbf{p},B}$ for any $\varepsilon \in [0,1]$. The supports of probability measures $\mu_{\mathbf{p}}, \nu_A, \nu_B, \mu^{\varepsilon}_{\mathbf{p},A} (= \mu^{\varepsilon}_{\mathbf{p},B}), \nu^{\varepsilon}_A$ and ν^{ε}_B are illustrated in Figure 5.14, Figure 5.15, Figure 5.16,Figure 5.17, Figure 5.18 and Figure 5.19, respectively. Let α be $\alpha \coloneqq 1 - \varepsilon$.

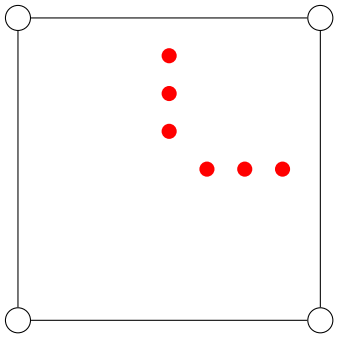


Figure 5.14: $\operatorname{supp} \mu_{\mathbf{p}}$. • = $1/6$.

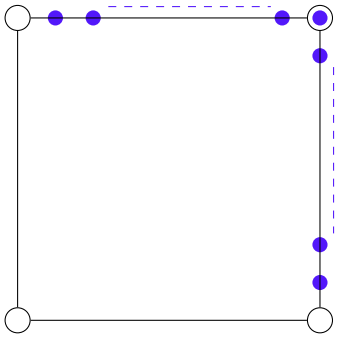


Figure 5.15: $\operatorname{supp} \nu_A$. • = $1/m$.

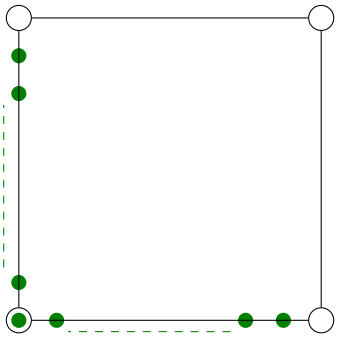


Figure 5.16: $\operatorname{supp} \nu_B$. • = $1/m$.

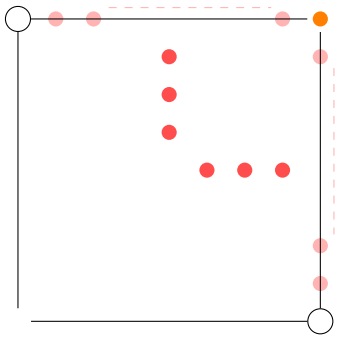


Figure 5.17: $\operatorname{supp} \mu^{\varepsilon}_{\mathbf{p},A}$ • = $\alpha/6$, • = $\varepsilon/4(\ell+1)$ and • = $\varepsilon/2(\ell+1)$.

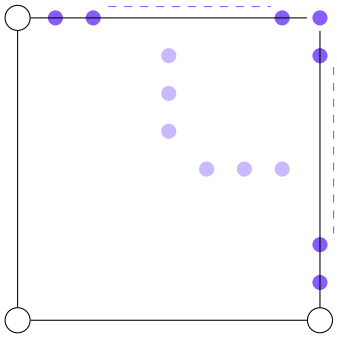


Figure 5.18: $\operatorname{supp} \nu^{\varepsilon}_A$. • = $\varepsilon/6$ and • = α/m.

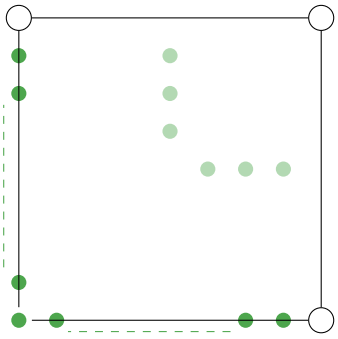


Figure 5.19: $\operatorname{supp} \nu^{\varepsilon}_B$. • = $\varepsilon/6$ and • = α/m.

Now note that the value (5.5) can be decomposed as follows:

$$\frac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A)}{W_1(\mu_{\mathbf{p}}, \nu_A)} = \frac{A(\varepsilon) + \delta(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_A)} = \frac{A(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_A)} + \frac{\delta(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_A)}, \tag{5.6}$$

$$\frac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_B)}{W_1(\mu_{\mathbf{p}}, \nu_B)} = \frac{B(\varepsilon) + \delta(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_B)} = \frac{B(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_B)} + \frac{\delta(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_B)}, \tag{5.7}$$

where $A(\varepsilon)$ and $B(\varepsilon)$ are the transport costs from the trajectory to each route, respectively, and $\delta(\varepsilon)$ is the transport cost from one route to the other. Note that $\delta(\varepsilon)$ is a common quantity in both transports. Moreover, we can see that

$$\frac{A(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_A)} \coloneqq \frac{B(\varepsilon)}{W_1(\mu_{\mathbf{p}}, \nu_B)}$$

is valid when ε is close enough to 0. Then, the magnitude relationship between (5.6) and (5.7) is determined by the magnitude relationship between $W_1(\mu_{\mathbf{p}}, \nu_A)$ and $W_1(\mu_{\mathbf{p}}, \nu_B)$.

Hence, since (5.4) holds, we obtain

$$\frac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A)}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \frac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_B)}{W_1(\mu_{\mathbf{p}}, \nu_B)}. \tag{5.8}$$

Therefore, the Wasserstein method outputs the route B as the true route in a situation like Figure 5.13.

5.4 Future problems

We list the problems that should be addressed to develop the Wasserstein method in the future in the following aspects:

- the definition of a probability measure associated with the trajectory (Definition 5.3),
- how to distribute weights by speed and direction information (Definition 5.13),
- the property to be shown for implementation:
  - the case where determination is based only on location information (§ 5.2),
  - the case where determination is based on speed and direction information as well (§ 5.3).

The definition of a probability measure associated with the trajectory (Definition 5.3)

Inspired by Definition A.6, we defined the probability measure $\mu_{\mathbf{p}}$ associated with the trajectory as Definition 5.3, but it may be useful to make the definition correspond to $(\gamma_p)_{p\in\mathbf{p}}$, which represents the range of GPS error see also § A.1. This is natural considering the original meaning of "coarse Ricci curvature", which was introduced in [Ol], as defined on metric measure spaces. However, the concrete calculations would be very complicated beyond our setup.

How to distribute weights by speed and direction information (Definition 5.13)

After all, the distribution of weights in Definition 5.13 is the most important point in the map-matching problem. The weight in Definition 5.13 was simply assigned equally to all the edges of the local road network, but I think it should be more sophisticated. For example, the weight value to be placed on the threshold point of each edge can be different depending on the location of each $p \in \mathbf{p}$ and the $\mathsf{spd}(p)$.

The property to be shown for implementation

This is a common problem for both § 5.2 and § 5.3 in the sense that the goal is to derive the property corresponding to Theorem A.5 in each setting. However, the properties to be shown in each situation are somewhat different and will be explained separately.

The case where determination is based only on location information (§ 5.2)

For each candidate route $A \in \mathcal{CR}$, as m is increased, $\operatorname{supp}\nu_{A,m}$ approaches to the route A respectively. Our idea is that we can determine which route is closer to the trajectory points $\mathbf{p}$ by computing $W_1(\mu_{\mathbf{p}}, \nu_{A,m})$ for a sufficiently large $m \in \mathbb{N}$. Then, we need to show the following.

Problem 5.18. For any $A, B \in \mathcal{CR}$, there exists a $\widetilde{m} \in \mathbb{N}$ such that $W_1(\mu_{\mathbf{p}}, \nu_{A,m}) - W_1(\mu_{\mathbf{p}}, \nu_{B,m})$ is a monotone function for any $m \in \mathbb{N}$ such that $m \geq \widetilde{m}$.

We can determine which route the trajectory points is closer to by checking the sign of $W_1(\mu_{\mathbf{p}}, \nu_{A,m}) - W_1(\mu_{\mathbf{p}}, \nu_{B,m})$ for a sufficiently large m if we can prove Problem 5.18. Hence, if $W_1(\mu_{\mathbf{p}}, \nu_{A,m}) < W_1(\mu_{\mathbf{p}}, \nu_{B,m})$ holds, then we can conclude that the trajectory points is closer the route A, which is the true route.

Remark 5.19. Although $W_1(\mu_{\mathbf{p}}, \nu_{A,m})$ take finite values for any $A \in \mathcal{CR}$ and any $m \in \mathbb{N}$, they might not converge as $m \to \infty$. Although we would conjecture that they converge as m approaches infinity in an even or odd state, respectively, their values may be different, i.e. their values may oscillate as $m \to \infty$.

The case where determination is based on speed and direction information as well (§ 5.3)

We could not show the piecewise linearity like Theorem A.5 in LLY–Ricci curvature yet within the period of this project (see also Remark A.12):

Problem 5.20. Is the function $\varepsilon \mapsto \kappa(\varepsilon; \mathbf{p}, A)$ piecewise linear on $[0, 1]$?

This is due to the complexity of our setting, and I conjecture $\kappa(\varepsilon; \mathbf{p}, A)$ will almost certainly be piecewise linear on $\varepsilon \in [0, 1]$ for each candidate route $A \in \mathcal{CR}$. However, the number of segments will depend on n and m, not 3. Therefore, we will calculate $\kappa(\varepsilon; \mathbf{p}, A)/\varepsilon$ by assigning a value sufficiently close to 0 to ε. It would have been much more useful in terms of implementation of this first linear part $\varepsilon \in [0, \tilde{\varepsilon}(n, m)]$ could have been derived.

6 Methods by Physical Analogy

We introduce the "electric method" and the "harmonic oscillator method" as a way to define our new loss function.

6.1 "Electric" Method

First, we propose a method to measure the closeness of trajectory points and routes by the interaction of Coulomb forces. In this method, we consider not only trajectory points, but also the entire polyline connecting the trajectory points, and compare it with the entirety of each route. That is, we consider a "trajectory segments-to-route method."

6.1.1 Details of method

We describe the specific procedure of this "electric" method. The abstract of the step is as follows:

1. Connecting trajectory points with segments and parameterizing them and each candidate route.
2. Giving the polyline and routes opposing electrical charge.
3. Calculating electric forces between the polyline and routes.
4. Choosing "the closest route" as the true route.

First, we connect trajectory points with segments and parametrize the candidate route. We assume that this polyline can move but the road map is fixed.

Definition 6.1 (polyline)**.** We define the *polyline* $l_{\mathbf{p}} : [0, 1] \to \mathbb{R}^N$ associated with a trajectory $\mathbf{p} = (p_i)_{0 \le i \le n}$ by

$$l_{\mathbf{p}}(t) := (1 - t)p_i + tp_{i+1} \quad \left(\frac{i}{n} \le t \le \frac{i+1}{n} \right). \tag{6.1}$$

Definition 6.2 (parameterized route)**.** Let $r = (e_1, \ldots, e_m)$ be a route. We define the *parametrised edge* $e_i : [0, 1] \to \mathbb{R}^N$ by parametrising $|e_i|$ by arclength parameter and the *parametrised route* $r(t) : [0, m] \to \mathbb{R}^N$ by union of the parametrised route e_i.

Remark 6.3. We assume that route is piecewise smooth and regular, thus arclength parameter exists on each interval.

Next, we give the candidate routes and our trajectory polyline opposing electrical charges. For simplicity, we assume that electric charge density is uniform.

Definition 6.4 (electric charge)**.** We define *electric charge density* as uniform probability density function on the polyline $\rho_{\text{polyline}} : [0, 1] \to \mathbb{R}$ and on the candidate route $\rho_{\text{route}} : [0, 1] \to \mathbb{R}$.

Then, all of the candidate routes exert an electric force on our polyline. The magnitude of Coulomb forces is represented as follows:

$$\int_{[0,m]} \int_{[0,1]} \frac{(l(t) - r(s)) \cdot \dot{l(t)}}{\mathsf{d_{Err}}(l(t), r(s))^2} \rho_{\text{polyline}}(t) \rho_{\text{route}}(s)\, dtds. \tag{6.2}$$

We choose the route which exerts the most force on the polyline as the true route.

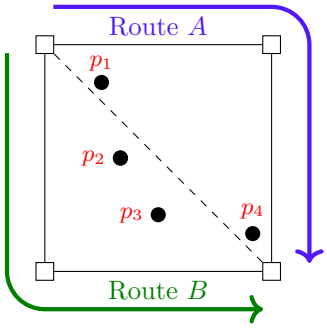


Figure 6.20: Given trajectory points and candidate routes.

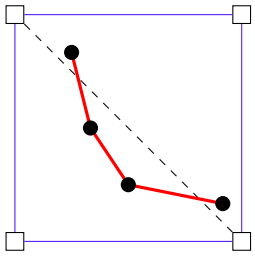


Figure 6.21: Connecting trajectory points and giving opposing electric charges. The red colored area have positive charge and the blue colored area have negative charge.

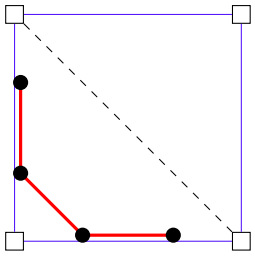
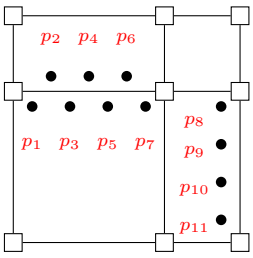
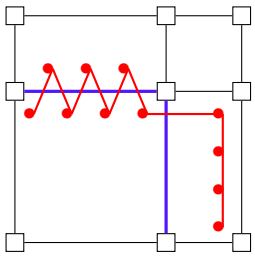


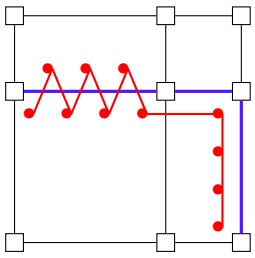
Figure 6.22: The polyline moving to "closer" route by Coulomb forces.

6.1.2 Observations

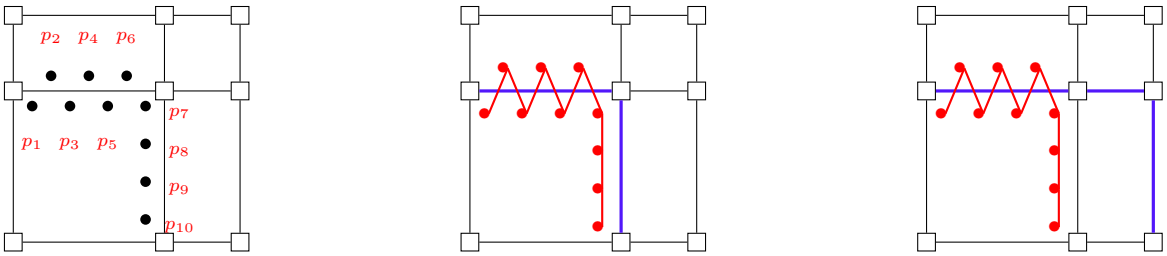
- Strength:
  By using this method, we can consider both the whole polyline and the entirety of each route all at once, because the electric forces are determined by the whole body of each.
- Problems: We have two problems with this "electric" method;
  - The theories about static electrodynamics and rigid body dynamics do not tell us how to treat speed and direction data.
  - Coulomb forces are proportional to square inverse of distance, so when the polyline intersects with the route they diverge. We can avoid divergence by adding a constant to the denominator. But, if a constant is small, the effects of intersection points are still dominant. Thus it is difficult to choose the correct route which has fewer intersection points than other wrong routes like the below figure.







Also, due to the nature of floating-point arithmetic, it is difficult for the computer to distinguish routes that share intersection points like the below figure.



As we see in above figures, consequently, we must take the appropriate constant. But because it is dependent on the number of intersection points, we cannot choose it arbitrarily. This problem was found in the implementation step.

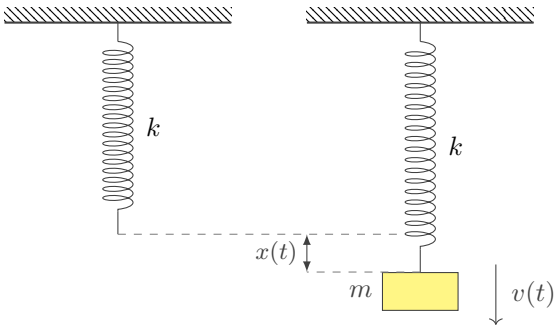
- Strategy to solve this problem:

  Our option to avoid the above issue is to instead invert the problem: that is, we replace maximizing inverse squared distance $\max_{A \in \mathcal{CR}} \frac{1}{r^2}$ to minimizing squared distance $\min_{A \in \mathcal{CR}} r^2$. The latter term is similar to the potential of harmonic oscillators. We consider the analogy of this model.

6.2 "Harmonic oscillator" method

Based on the consideration of the "electric" method, we developed a policy to minimize the square of the distance, and this equation inspired us to consider an analogy with the analytical dynamical description of harmonic oscillators.

Harmonic oscillators are a system in which a material point is subjected to a force proportional to its distance from some fixed point. An example is the motion of a mass connected to a wall by a spring.



If mass m has velocity $v(t)$ and displacement $x(t)$ from the natural length of the spring at a time t, this system has momentum $\frac{1}{2}mv(t)^2$ and potential $\frac{1}{2}kx(t)^2$ where k is a spring constant. Then the Hamiltonian $H(t)$ and the energy E of this system are defined by

$$H(t) := \frac{1}{2}mv(t)^2 + \frac{1}{2}kx(t)^2,$$
$$E := \int L(t)\, dt.$$

We formulate our map-matching problem as the following setup: we take the true root as the location where the potential of the system is minimal. Then, assume that the trajectory point has a minute displacement from the point of minimum potential due to error.

We now assume that our assumptions about error place more emphasis on speed and orientation than on location information. Therefore, we formulate the distance to the edge to be small. This operation is consistent with the picture that the harmonic oscillator approximates the system at small displacements from the point where the potential is smallest.

6.2.1 Details of method

We describe the specific procedure of this "harmonic oscillator" method. The abstract of the step is as follows:

1. Connecting trajectory points by segments
2. Giving speed, direction, and error information to points on polyline connecting trajectory points.
3. Defining a score of edges included in the candidate route and choosing the edge which minimizes the score.
4. Connecting points on the polyline and anchors on the "closest" edge by "spring".
5. Defining "momentum" and "potential" of each point on the polyline.
6. Calculating "energy" of "Hamiltonian" for the polyline and each candidate route.

Definition 6.5. Let $\mathbf{p} = \{p_0, \dots, p_n\}$ be trajectory points

1. We define *polyline* $l_{\mathbf{p}} : [0,1] \to \mathbb{R}^N$ by

$$l_{\mathbf{p}}(t) := (1-t)p_i + tp_{i+1} \quad \left(\frac{i}{n} \le t \le \frac{i+1}{n}, \quad 0 \le i \le n-1\right). \tag{6.3}$$

2. We define the speed, direction, and error of each point on the polyline by piecewise linear interpolation:
   - (speed) $\mathsf{spd}_{l_{\mathbf{p}}}(t) := (1-t)\mathsf{spd}_i + t\mathsf{spd}_{i+1} \quad \left(\frac{i}{n} \le t \le \frac{i+1}{n}, \quad 0 \le i \le n-1\right)$;
   - (direction) $u_{l_{\mathbf{p}}}(t) := (1-t)u_i + tu_{i+1} \quad \left(\frac{i}{n} \le t \le \frac{i+1}{n}, \quad 0 \le i \le n-1\right)$;
   - (error) $\mathsf{Err}_{l_{\mathbf{p}}}(t) := (1-t)\mathsf{Err}_i + t\mathsf{Err}_{i+1} \quad \left(\frac{i}{n} \le t \le \frac{i+1}{n}, \quad 0 \le i \le n-1\right)$.

Definition 6.6 (Score of edges included in candidate route)**.** Let $\mathbf{p}$ be a trajectory and $l_{\mathbf{p}}$ denotes the polyline formed by $\mathbf{p}$ Let $\mathbf{p}$ be a trajectory, $l_{\mathbf{p}} : [0,1] \to \mathbb{R}^N$ be the polyline formed by $\mathbf{p}$ and $A = \{e \in E\} \in \mathcal{CR}$ be a candidate route.

1. We define the *score* $S_{l_{\mathbf{p}}} : [0,1] \times A \to \mathbb{R}$ by

$$S_{l_{\mathbf{p}}}(t,e) := \langle \mathsf{spd}_{l_{\mathbf{p}}(t)} u_{l_{\mathbf{p}}}(t), \vec{e} \rangle_{\mathbb{R}^N} \cdot d_{\mathbb{R}^N}(l_{\mathbf{p}}(t), e), \tag{6.4}$$

   where $\vec{e} \in \mathbb{R}^N$ is the unit vector induced naturally from timestamp information of trajectory.

2. We define the *anchor map* $\mathsf{Anc}_{l_{\mathbf{p}},A} : [0,1] \to A; t \mapsto \mathsf{Anc}_{l_{\mathbf{p}},A}(t)$, so that $S(t, \mathsf{Anc}_{l_{\mathbf{p}},A}(t)) = \max_{e \in A} S_{l_{\mathbf{p}}}(t,e)$.

Definition 6.7. Let $\mathbf{p}$ be a trajectory and $l_{\mathbf{p}}$ denotes the polyline formed by $\mathbf{p}$. We define the "*mass*" and "*spring constant*" of this system then use them to define the "*momentum*" and the "*potential*".

1. We define the *mass* $m_{l_{\mathbf{p}}} : [0,1] \to \mathbb{R}$ by

$$m_{l_{\mathbf{p}}}(t) := \frac{1}{\mathsf{Err}_{l_{\mathbf{p}}}(t) + 1}.$$

2. We define the *momentum* $M_{l_{\mathbf{p}}} : [0,1] \times \mathcal{CR} \to \mathbb{R}$ by

$$M_{l_{\mathbf{p}}}(t,A) := m_{l_{\mathbf{p}}}(t) \left\langle \frac{\mathsf{spd}_{l_{\mathbf{p}}}(t) u_{l_{\mathbf{p}}}(t)}{\log(1 + \mathsf{spd}_{l_{\mathbf{p}}}(t))}, \overrightarrow{\mathsf{Anc}_{l_{\mathbf{p}},A}(t)} \right\rangle_{\mathbb{R}^N}^2 \tag{6.5}$$

   where $\overrightarrow{\mathsf{Anc}_{l_{\mathbf{p}},A}(t)} \in \mathbb{R}^N$ is the unit direction vector of $\mathsf{Anc}_{l_{\mathbf{p}},A}(t)$ induced naturally from timestamp information of trajectory.

3. We define the *spring constant* $k_{l_\mathbf{p}} : [0,1] \to \mathbb{R}$ by

$$k_{l_\mathbf{p}}(t) := \exp\left(-\mathsf{Err}_{l_\mathbf{p}}(t)\right). \tag{6.6}$$

4. We define the *potential* $P_{l_\mathbf{p}} : [0,1] \times \mathcal{CR} \to \mathbb{R}$ by

$$P_{l_\mathbf{p}}(t, A) := k_{l_\mathbf{p}}(t) \cdot \left(d_{\mathsf{Err}_{l_\mathbf{p}}(t)}\left(l_\mathbf{p}(t), \mathsf{Anc}_{l_\mathbf{p},A}(t)\right)\right)^2. \tag{6.7}$$

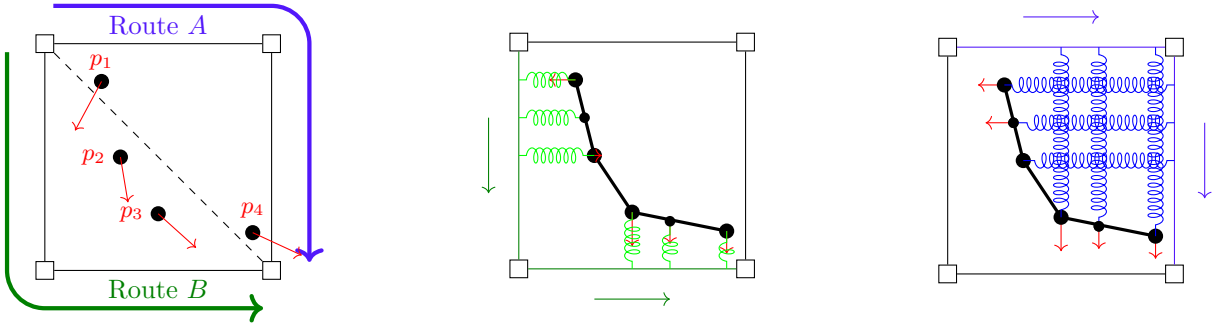
Remark 6.8. 1. We have now decided to focus on speed and direction rather than positional information in our assumptions, and have formulated the distance to be small. Therefore, the spring constants involved in the potential, of which distance is a major component, are allowed to decay exponentially with respect to the error. On the other hand, for mass, it is similarly affected by errors, but the choice of the edges connecting the springs is based on the score determined to direction, and since the direction is assumed to have no error, the correction is assumed to decay about a first order polynomial. 1 is added to eliminate singularity.

2. Velocities vary widely among pedestrians, bicyclists, and motor vehicles, and momentum can be very different for them, which can lead to significant differences in results. This is undesirable in light of the validity of the method. The addition of 1 is to eliminate singularity.

Definition 6.9 (loss function)**.** Let $Tr = \mathbf{p}$ be a trajectory. We define a loss function $E_{Tr} : \mathcal{CR} \to \mathbb{R}$ by

$$E_{Tr}(A) := \int_{t\in[0,1]} \left\{M_{l_\mathbf{p}}(t, A) + P_{l_\mathbf{p}}(t, A)\right\} dt. \tag{6.8}$$

When several candidate routes are given, we calculate the energy of each route on the trajectory points and choose the route that has the least energy as the true route. For example, trajectory points and candidate routes are given as below figure, we calculate $E_{Tr}(A)$ and $E_{Tr}(B)$. Then if $E_{Tr}(B) < E_{Tr}(A)$ holds, we choose route B as the true route.



6.2.2 Observations

- Strength:
  - There is no divergence problem in contrast to the "electric" method.
  - We can take into account information about speed, direction, and error naturally.
- Weakness:
  The edges connecting the springs were formulated so that only one edge with the smallest score is selected and the springs are connected to that edge, but this method is inadequate to reflect the shape of the trajectory points and the shape of the entire route. Consider the case where a route and trajectory points are given such that a similar shape is repeated.

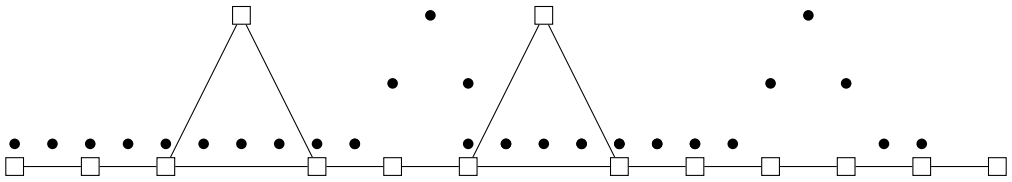


Figure 6.23: Given trajectory points and candidate routes.

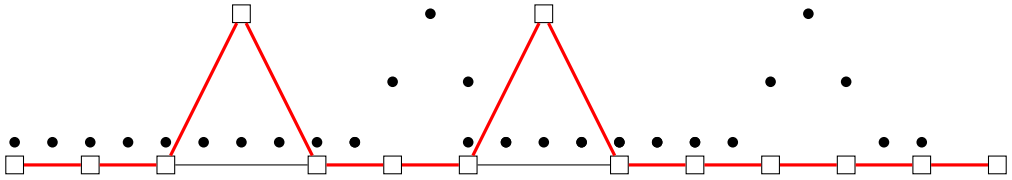


Figure 6.24: The candidate route that seems to be the true route.

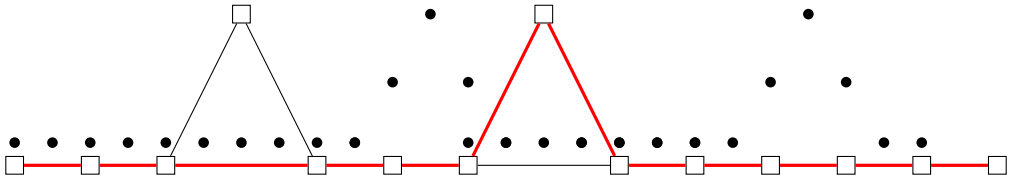


Figure 6.25: The candidate route that is chosen to be the true route by "harmonic oscillator" method in which the only one route is chosen to be connected to trajectory points

In this case, the true route should be the one that has the same number of peaks. However, the method of selecting one edge and connecting it to the spring would select a route with only one peak.

To improve this, we want to connect all edges with springs and give them appropriate weights, so that the shape can be more reflective of the shape.

7 Implementation

After formulating the proposed mathematical methods into robust map-matching algorithms, we implemented them in Python to evaluate their performance numerically. The code written for the project can be found at: https://github.com/gjgress/G-RIPS-2022-Mitsubishi-A. We will show the resulting solution of our algorithms on two cornerstone examples:

1. the **Square** road network with two candidate routes to decide between and a trajectory in a neighborhood of the diagonal Figure 7.26
2. the **Sendai** map example with road network from OpenStreetMap and trajectory collected from walk around Sendai. Figure 7.27

The square example has a total of 6 trajectory points and four edges in its road network. The left and bottom edges (in blue) make up route 1, and the top and right edges (in yellow) make up route 2. The Sendai example has 645 trajectory points (in blue) and 191 edges (in white).

The dataset we used for evaluation of the electrical and harmonic oscillator methods was the *Dataset for testing and training of map-matching algorithms* [KCMMN]. This dataset was a good fit for the project because it included ground truths, despite only possessing GPS data. We were unable to apply the Wasserstein

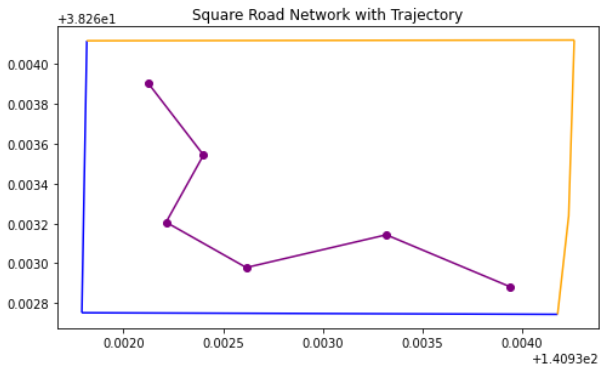


Figure 7.26: Square Example

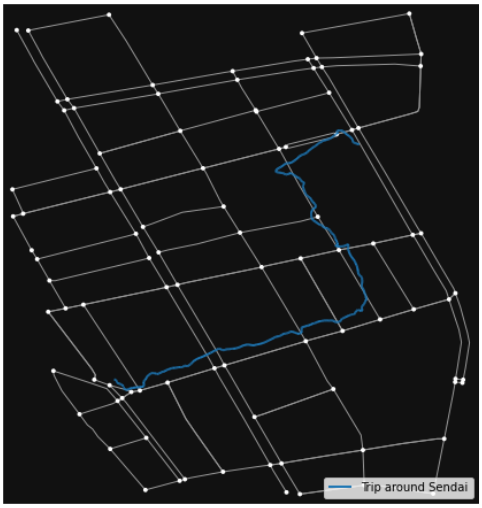


Figure 7.27: Sendai Example

method to this data set due to issues discussed in § 7.2. Although we pre-processed the BDD100K open dataset provided by Berkeley for evaluation, it does not contain ground truths, so we were unable to utilize it for this project. [YCWXCLMD].

7.1 Preprocessing Routes

As the number of edges in a directed graph increases, the number of routes between two nodes grows at a rate similar to the factorial (and can be infinite if loops are allowed); moreover, the determination of all such routes is NP-hard. As a result, it is not sufficient to simply determine a suitable loss function and evaluate all possible routes. Finally, our loss functions are often expressed as integrals, so even once we have our routes, we need to interpolate each route so that the sum sufficiently approximates the integral. In practice, we do all this by restricting to a sub-graph, using an algorithm to obtain a restricted set of candidate routes, and then interpolating the resulting routes. We refer to these steps as preprocessing.

Of course, the preprocessing process need only be done once for a given dataset. That is, once we preprocess our graph with our trajectory, we can utilize the processed information for all of our algorithms. Furthermore, all the preprocessed information can readily be saved to disk, so for larger datasets, this need only be done once.

7.1.1 Restricting to a Sub-graph

The method we use to restrict to a sub-graph is relatively simple. We choose a $k \in \mathbb{Z}_+$, and for each point p_i in the trajectory, we find the set of k-nearest edges with their respective nodes (or all the possible edges, in the rare case that there are fewer than k edges), denoted $E^k_{p_i}$. Our restricted sub-graph $\tilde{G}$ is then given by

$$\tilde{G} = \bigcup_{p_i} E^k_{p_i}. \tag{7.1}$$

Unlike k-nearest nodes, k-nearest neighbors are more computationally difficult. A k-d tree is not possible because we have to consider *all* points around an edge. As a result, we are forced to rely on geometric tools to find this set. We do this by choosing an $r > 0$ and creating a square of size $2r$ centered at each p_i, and finding the geometric intersection with the road network. If at a given point the square does not intersect at least k edges, we double the value of r and repeat the process, until we have found enough edges or we have doubled the value ten times. This method was inspired by the implementation of FMM [YG].

This method is quite inefficient. In the future, we hope to explore using mathematical geometric tools to better find our sub-graph.

7.1.2 Obtaining Candidate Routes

Once we have obtained our sub-graph $\tilde{G}$, we utilize a variant on Dijkstra's algorithm to generate a list of short paths between our source and target. First we identify the source and target node on $\tilde{G}$ by finding the closest neighbor to p_1 and p_n. Then we follow the standard procedure of Dijkstra's algorithm by visiting the nodes connected to the route with lowest cost. Unlike Dijkstra's algorithm, we keep each route obtained, even if we have found a shorter route to get to the same intermediate node. We continue Dijkstra's algorithm until we have either enumerated every route or have obtained the number of routes desired. We do return the cost of each route as well, but we do not use this information in our algorithms– the shortest route is not necessarily the best candidate route!

This method is a reasonably fast way to take a breadth-first search approach, but nevertheless is still quite slow. Furthermore, while Dijkstra's algorithm can be modified to allow multiple source and target nodes, we choose not to apply this as it would require us to generate many more candidate routes to ensure the true route is within our set.

There are some edge cases where this algorithm will fail. For one, if the number of nearest edges is not sufficiently high, the sub-graph will not be able to find any candidate route. In this situation one has to increase the number of candidate edges and start the algorithm from scratch– which can be very computationally expensive. The algorithm is also not designed to handle routes where the same edge is traversed more than once; these cases are exceedingly rare, and so we did not find it necessary to modify the algorithm for this purpose.

This algorithm also is difficult to run in parallel. Conceptually, it should be possible to search multiple routes at once, but this requires a more robust task assignment process. We did not have the time to explore this option, and more importantly, it may not be worth it if better candidate route generation methods are available.

Despite these shortcomings, the candidate routes obtained are usually high quality. Unlike some data-driven implementations, we also know that when we find our optimal route, it will be a valid route.

7.1.3 Candidate Route Interpolation

The interpolation process is rather simple. Because our objects are Shapely geometries, we can interpolate each edge using Shapely's built-in interpolation. We choose constants $n_r, n_t \in \mathbb{Z}_+$ and interpolate each edge of the candidate route into n_r segments and each edge of the trajectory into n_t segments.

One concern is how to determine the constants n_r, n_t a priori. We found that choosing $n_r \approx 100$ and $n_t \approx 10$ produced the best results; however, in the future we wish to implement a more dynamic approach which interpolates based on the relative density of the points.

Typically with simpler loss functions such as the electric method or harmonic oscillator method, increasing the interpolation points does not significantly increase computation time. However, in the case of the Wasserstein method, it had a big impact on computation time. Thus the optimal choice of n_r and n_t depend greatly on the method.

A shortcoming of this method is that it interpolates every edge equitably. This means that very small edges will have a high density of points, while a long edge will be more sparse. This may give an unfair weighting to small edges. In the future we wish to explore a dynamic choice of interpolation points based on the lengths of an edge.

7.2 Wasserstein Distance Implementation

Let $\{p_i\}$ be the set of trajectory points and discretize $A \in \mathcal{CR}_G$ into $m+1$ equal parts to obtain the set of threshold points $V(A, m) \coloneqq \{a_1, a_2, \ldots, a_m\}$. Then, from § 5.1, we know can find the Wasserstein distance

by solving the following linear program

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} d(p_i,a_j)\pi(p_i,a_j) \\ \text{subject to} \quad & \pi(p_i,a_j) \geq 0 \qquad \text{for } i=1,2,\ldots,n,\ j=1,2,\ldots,m, \\ & \mu(p_i) = \sum_{j=1}^{m} \pi(p_i,a_j) \qquad \text{for } i=1,2,\ldots,n, \\ & \nu(a_j) = \sum_{i=1}^{n} \pi(p_i,a_j) \qquad \text{for } j=1,2,\ldots,m. \end{aligned} \tag{7.2}$$

After selecting n, m, we solve this linear program in python using the linprog command from Scipy's optimization package. The constraint matrix constructed will be size $(n+m)\times nm$, but only $2mn$ elements will be nonzero. Thus, the constraint matrix can be saved as a sparse matrix. For the computation, we remove one of the constraints, because the system is overdetermined as written. In contrast, it is likely that no $d(p_i,a_j)$ will be zero. Therefore, for each candidate route, we must compute nm distances.

For the simple example of a trajectory contained in a square road network, see Figure 7.28, the loss computed for each route is the Wasserstein distance. The chosen route in this instance is Route 1. We can see that the loss for Route 1 is smaller, and is the route we would have chosen to be correct through visual analysis.

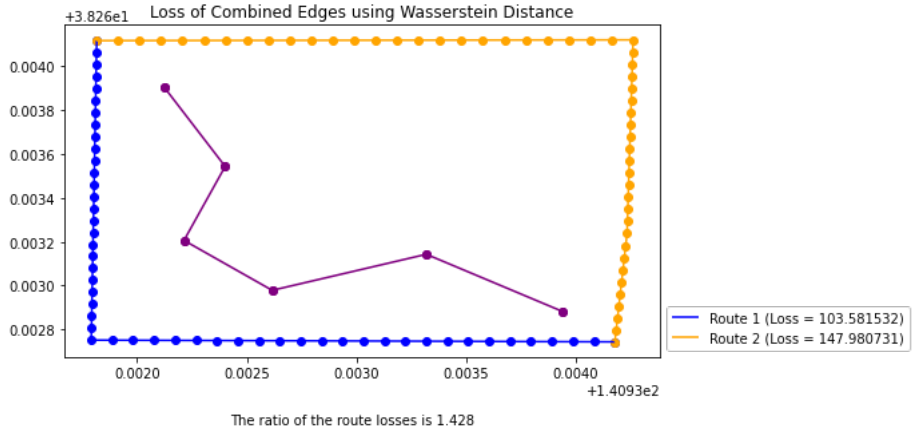


Figure 7.28: Wasserstein solution to the Square Example

In contrast, when we apply Wasserstein distance to the example of the Sendai map, the route that minimizes Wasserstein distance is longer and more complex than the true route. In figure Figure 7.29, we see the route with minimal Wasserstein distance. Not only does it stray from the trajectory substantially, but it also contains a loop. At this time, we are unsure if this is an issue with the method itself or some implementation error. One obstacle to determining this is the computational time. Computing the Wasserstein distances on the set of candidate routes for the Sendai map alone can take several hours.

7.3 Electrical Method and Harmonic Oscillator Implementation

In comparison, the electric method and harmonic oscillator are simpler to compute. Let $\mathrm{Tr} = \{p_i\}$ be the set of trajectory points and $A = \{q_i\}$ be the set of points along a candidate route. For each p_i we determine the k-nearest neighbors from the set $\{q_i\}$; that is, for each p_i we obtain a subset $\{q_j\}_{j\in 1,\ldots,k}^{p_i} \subset \{q_i\}$. Then

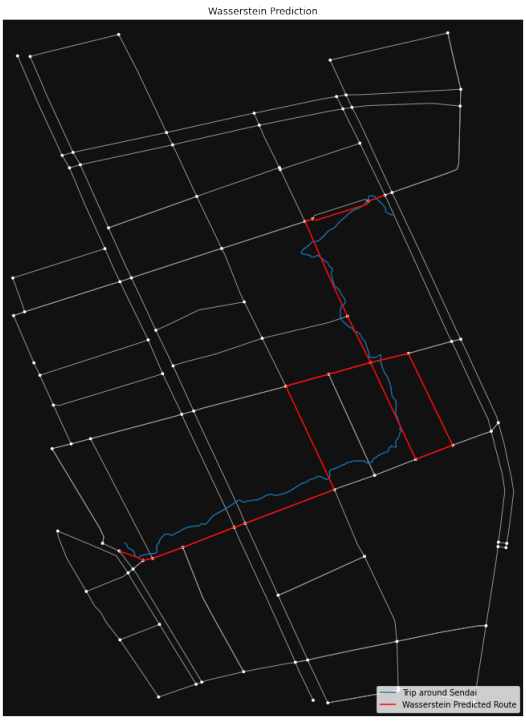


Figure 7.29: Wasserstein solution to the Sendai Example

we compute the sums:

$$E_{\mathrm{Tr}}(A) = \sum_{p_i} \sum_{j=1}^{k} \frac{-1}{d(p_i, q_j)^2 + \varepsilon} \qquad \text{(Electrical Method)} \tag{7.3}$$

$$\mathrm{Act}_{\mathrm{Tr}}(A) = \sum_{p_i} \sum_{j=1}^{k} d(p_i, q_j)^2 \qquad \text{(Harmonic Oscillator Method)} \tag{7.4}$$

where $0 < \varepsilon \ll 1$ is a small constant chosen to prevent divergence.

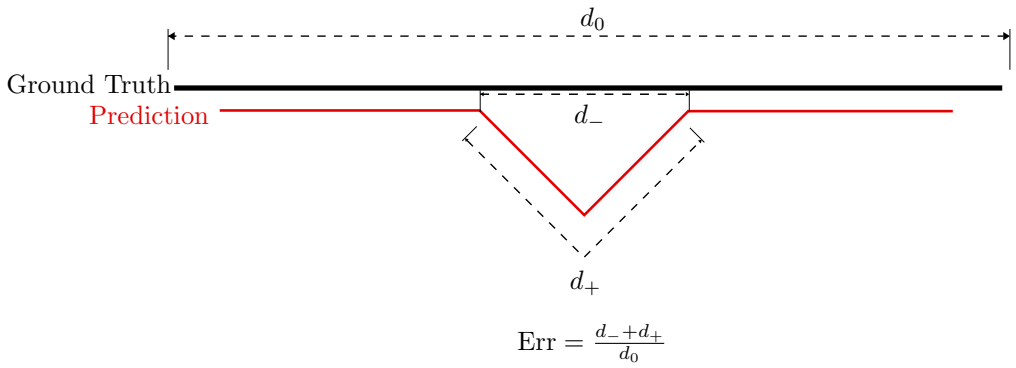
In particular, for the electric method, we choose $k = \#\{q_i\}$ (that is, we compare to all the points along the candidate route), and for the harmonic oscillator method, we choose $k = 1$ (that is, we only compare to the nearest neighbor). On one hand, choosing large k means you take into account more of the polyline; however, points far away from the polyline may have a large influence on the loss. Typically this does not change the relative losses between two candidate routes, but it does make the losses numerically much closer– and in extreme cases, we may lose this relative information due to floating point errors. Conversely, for small k we are only considering the part of the candidate route most relevant to a given point; however, outliers in the trajectory may cause the loss of a candidate route to be lower than expected. Therefore, it is likely that the optimal value for k lies somewhere in between. Of course, it is difficult to determine a priori what k is appropriate for a given case.

7.4 Evaluation (Error) Method

How do we measure the accuracy of our prediction? We use the following formula presented by Newson and Krumm [NK] to calculate the error:

$$\mathrm{Err} = \frac{d_- + d_+}{d_0}$$

where d_0 is the length of the correct route, d_- is the length the prediction erroneously subtracted from the ground truth, and d_+ is the length the prediction erroneously added outside the ground truth. See Figure 7.30.



d_0 = length of ground truth
d_- = length of prediction route erroneously subtracted
d_+ = length of prediction route erroneously added

Figure 7.30: Error Formula by Newson and Krumm

Figure 7.31: FMM evaluated on the annotated dataset

7.5 Preliminary Results

When tested on the dataset, we found that FMM had an error of 14.7% on average 7.31.

Unfortunately, due to processing power and time constraints, we were unable to evaluate our algorithms on the entirety of the dataset. In particular, the preprocessing stage was too slow on the much larger networks to allow for testing. As a result we can only provide preliminary results for two of the methods 7.32 7.33.

We anticipate that the geometric methods will perform more accurately than FMM on the dataset, but at the cost of (significantly) greater computation time.

7.6 Computational Complexity

We tested the computational complexity of the preprocessing stage and our methods. In particular, the preprocessing stage scales linearly with the number of candidate routes to find, but scales exponentially with the number of candidate edges provided 7.34. Both the Electric Method and the Harmonic Oscillator Method seem to scale linearly $\mathcal{O}(n)$ with the number of candidate routes 7.357.36. When the number of distance calculations were increased (i.e. more nodes in the candidate route and trajectory), runtime increased, but not significantly– certainly slower than linear growth. Unfortunately, the Wasserstein method grew too rapidly when distance calculations were increased, and so a similar figure could not be produced for the method. We suspect that it scales as a polynomial of degree > 1 with respect to distance calculations.

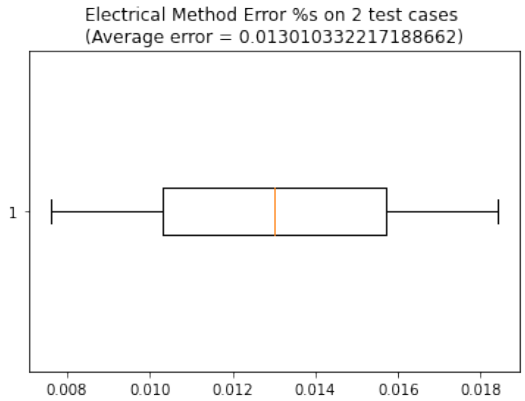


Figure 7.32: Electric Method evaluated on annotated dataset

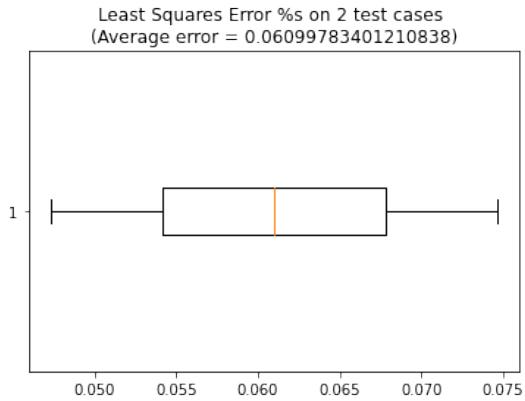


Figure 7.33: Harmonic Oscillator Method evaluated on the annotated dataset

The metric algorithm framework can parallelize over the number of candidate routes, which significantly improves the computation complexity. It is unlikely that improvements can be made within the loss functions to parallelize further, as Electric Method and Harmonic Oscillator Method utilize array operations, and Wasserstein utilizes linear programming.

While candidate route generation has limited room for parallelization, one can easily parallelize the process across the entirety of the dataset. The simulators can also be easily run in parallel on the entirety of the dataset.

7.7 Metric-based Algorithm Framework

Because the basic formulation of our proposed map matching algorithms are all purely metric-based, we realized it was more efficient to write the algorithm in a modular capacity. This led to the creation of the generalized metric_mm algorithm class.

To create a metric_based map matching algorithm simulator, one first creates a loss function in one of two ways:

- Providing simple interior and exterior functions which operate in stages on the distance arrays
- Providing a singular loss function which takes as arguments the candidate route distance array and the trajectory distance array.

Often it is simpler to provide the latter, but the former may be convenient in the case that one wishes to make small modifications to the loss function procedure in a systematic capacity.

If preprocessing has already been performed by another simulator, the interpolated candidate route nodes and trajectory can be provided directly.

The class then provides a preprocessing method if preprocessing has not already been performed. It's only required argument is a GeoDataFrame consisting of LineStrings of the trajectory (this can be generated from a sequence of points using a utility function provided in another module). If candidate routes are not provided, it applies the variant on Dijsktra's algorithm described in 7.1.2.

To run the simulator, one simply has to call the run method. This will return the best candidate route (and its loss) or the loss of all candidate routes depending on parameters passed.

7.8 Data Fusion

In addition to map matching, the Jupyter notebooks providing a framework for implementing and testing algorithms on driving datasets. One issue that pervades this field of research is the high variety in file formatting, and in particular, these formats are not well-suited for containing IMU data. Our notebooks provides a rudimentary fusion method to align asynchronous data and incorporates IMU data into a GeoDataFrame

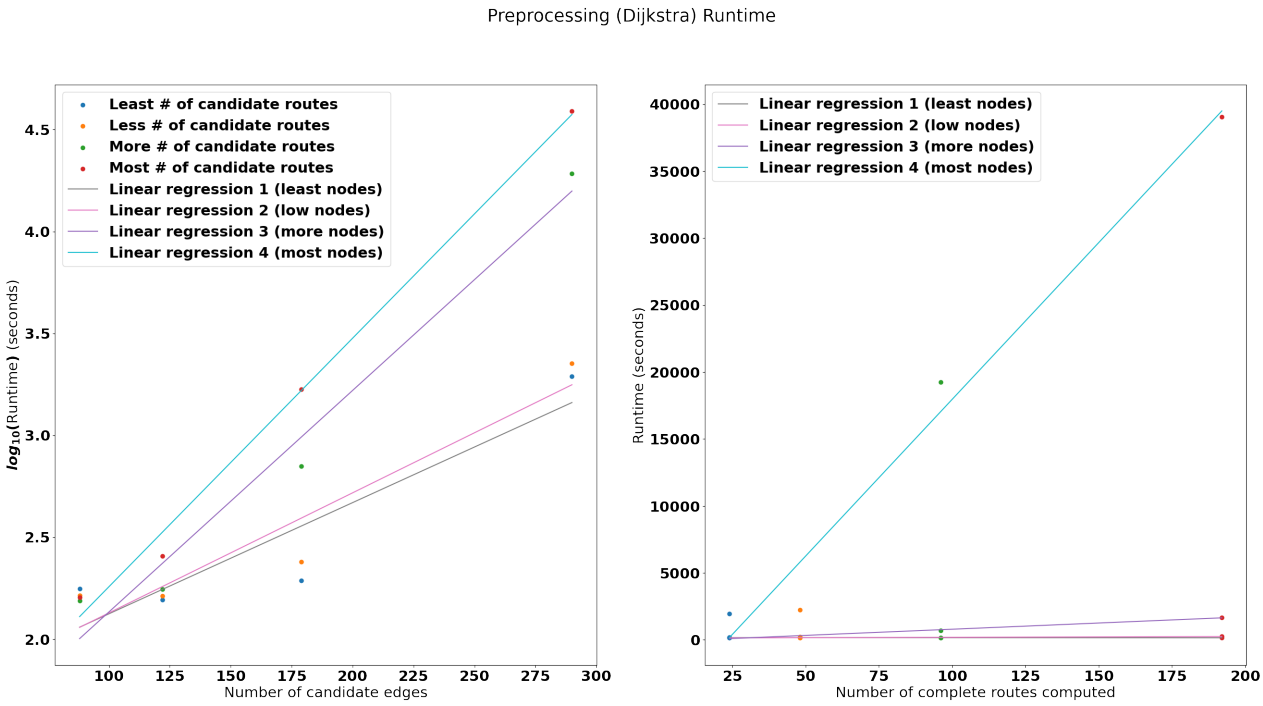


Figure 7.34: Preprocessing Computation Runtimes

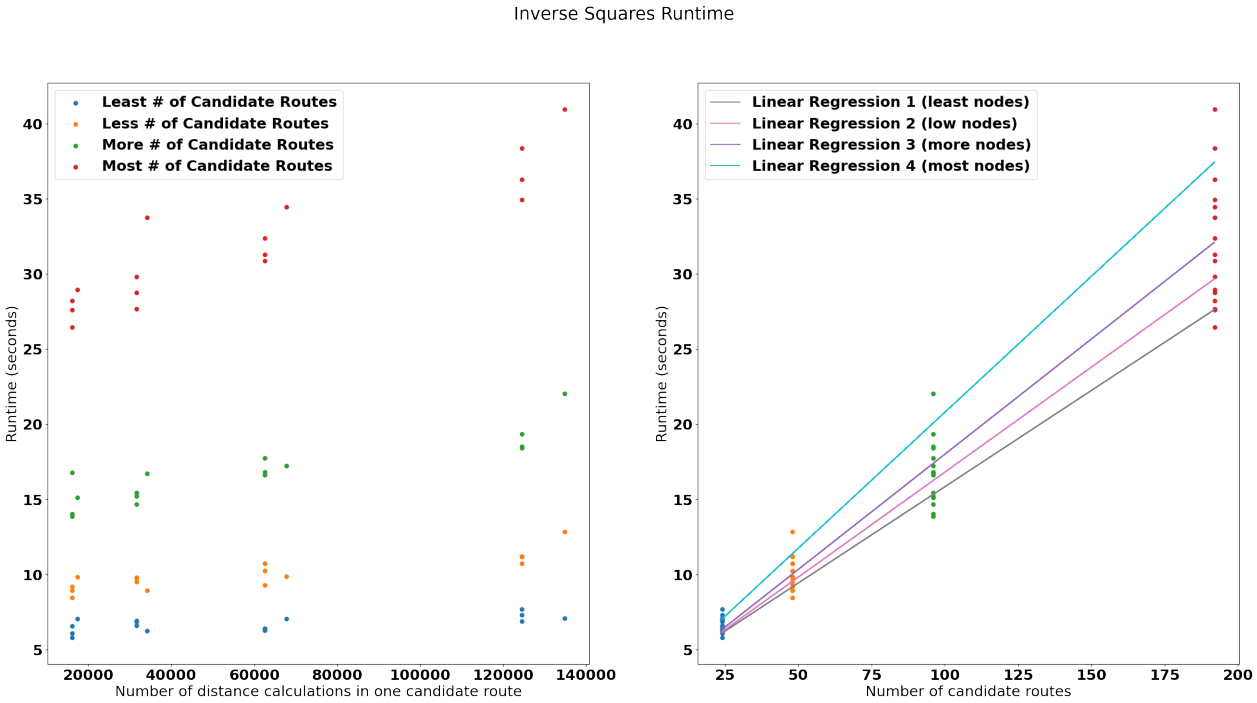


Figure 7.35: Electric Method Computation Runtimes

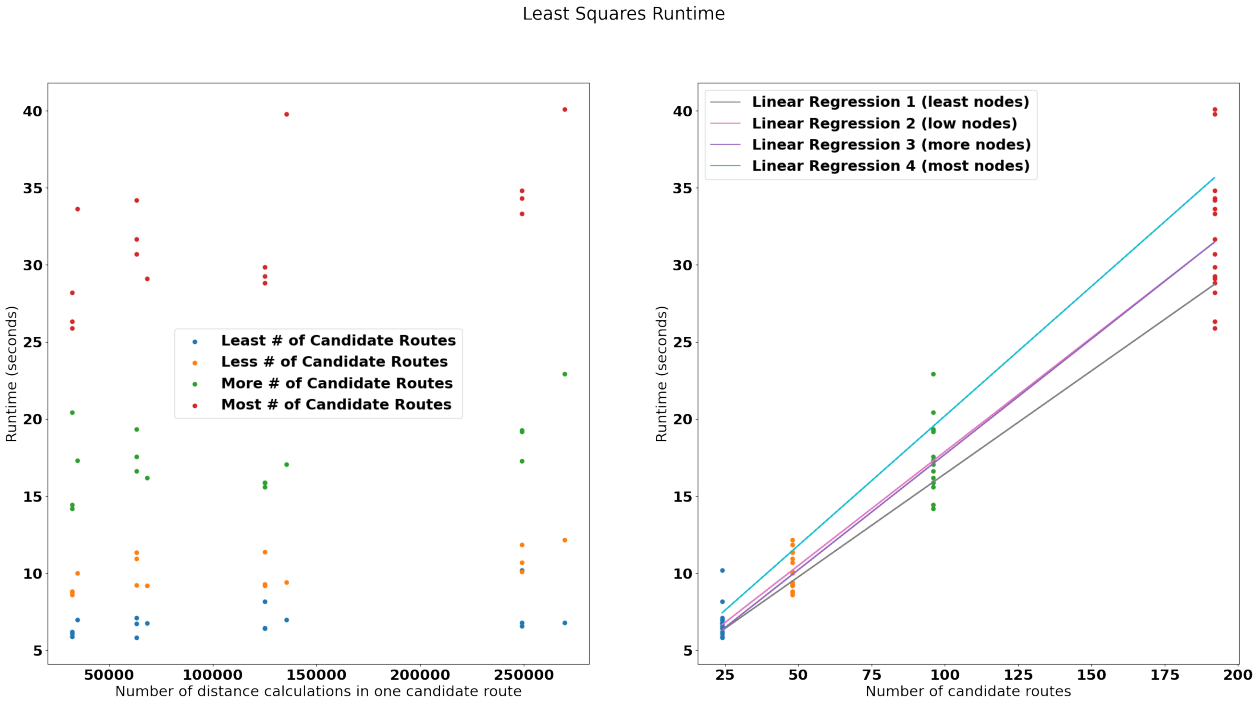


Figure 7.36: Harmonic Oscillator Method Computation Runtimes

(which can be exported to GPX, GeoJSON, KML, etc.) in a sensible manner. In particular, this method can sample any discrete-time data such as speed, accelerometer, and gyroscope, and merge it with the GPS data sequence.

7.9 Other Utilities

In the process of creating this framework, several utility functions and classes have been provided. Within mm_utils.py we provide functions such as:

- Basic plotting methods
- An evaluation method to calculate error
- A method to create trajectory edges (LineStrings) from a sequence of coordinates (Points)
- A method which provides a road network from OSM (as a MultiDiGraph or GeoDataFrame) from the trajectory data only
- A method to find k nearest points using k-d trees
- A method to find k nearest edges using the method described in 7.1.1

8 Future Work (Implementation)

We briefly mention again improvements discussed in the previous section:

- Modify k-nearest edge search with more efficient methods
- Modify or replace Dijsktra's algorithm for candidate route generation
- Investigate and improve upon Wasserstein method

8.1 IMU Inclusion

While IMU approaches were proposed theoretically, we did not have the time to include IMU information into our algorithm implementations. Currently the generalized metric-based map matching framework does not pass IMU data to the loss functions, but it is preserved in all the intermediate stages, so it should be simple to modify it so that the relevant data is passed along.

8.2 Incomplete Map Matching

One area of interest with map matching problems is incomplete maps. Our assumption is that our road network includes every possible path in reality. Of course, some older roads or even new roads may not be incorporated into our road network information. One direction of interest is then to explore if we can determine when roads are missing from our network based on the loss values along the trajectory.

8.3 Extension to Higher Dimensions

Including the z-axis into our implementation faces one major hurdle: OpenStreetMaps does not include elevation. Because this information is not included within our network, we cannot test our algorithms in 3-space. One can circumvent this by merging elevation data from an external source. However, highways and roads are often not incorporated into these data sets, and so it is approximate at best. Instead, it is probably best to source network information from more detailed data sets, such as proprietary sources, but results can vary greatly depending on the region.

Fortunately, the Python framework developed is quite modular, so if one does have elevation data in their network and test data, it is easy to implement within our notebooks. Hopefully, with time, data sets such as OpenStreetMaps will gain detail, and this line of inquiry becomes more accessible to the scientific community.

9 Conclusion

We developed 3 new loss functions for solving the map-matching problem: Wasserstein, Electrical, and Harmonic Oscillator. Within each theoretical formulation, not only can we determine a route from distance information, but also by accounting for speed and direction information. For the Wasserstein distance method, this achieved through a perturbation of the probability measures on both the trajectory data and the threshold points on the candidate routes. For the harmonic oscillator, the speed and direction are incorporated through the momentum term of the Hamiltonian. Each method was also implemented in Python, but using only the distance information. The results for Wasserstein distance suggests either the formulation or the implementation requires more investigation. However, the electric and harmonic oscillators results are promising. Although their computational time is greater than that of FMM, in the experiments that we have run so far, the accuracy seems much improved.

Appendix A Geometric background of Wasserstein method

This section provides the geometric background of the Wasserstein method in Section 5. Wasserstein method is based on the concept of "Ricci curvature of graph", which was introduced by [Ol, LLY]. This concept is a metric to measure the strength of cohesion between two vertices and has attracted attention as a new tool for graph analysis and has already been applied to real problems ([JL, NLGGS, NLLG], etc.). We modified Ricci curvature of graphs to quantify the "strength of cohesion" between the trajectory and each route.

A.1 Ricci curvature of graphs

In this section, we briefly describe the Ricci curvature of graphs. We consider a weighted graph. Denote the weight of edge e as w_e. In this case, the node degree d_x of vertex x is $d_x = \sum_{y \sim x} w_{xy}$. We call a graph

with a weight of 1 on all edges a *combinatorial graph*. First, we introduce a transition probability measure on a node. In this section, we use x, y as symbols that denote nodes.

Definition A.1 (Random walk with idleness parameter $\varepsilon \in [0,1]$)**.** For each node $x \in V$ and $\varepsilon \in [0,1]$, we define a random walk μ_x^ε as follows:

$$\mu_x^\varepsilon(y) \coloneqq \begin{cases} 1-\varepsilon & (y = x), \\ \varepsilon \cdot \frac{w_{xy}}{d_x} & (y \sim x), \\ 0 & (\text{otherwise}). \end{cases}$$

The Ricci curvature with idleness parameter ε is defined as follows.

Definition A.2 (ε-Ollivier–Ricci curvature; [Ol, LLY])**.** Let $\varepsilon \in [0,1]$. The *ε-Ollivier–Ricci curvature* $\kappa(\varepsilon; x, y)$ between two nodes $x, y \in V$ is defined as

$$\kappa(\varepsilon; x, y) \coloneqq 1 - \frac{W_1\left(\mu_x^\varepsilon, \mu_y^\varepsilon\right)}{d(x,y)}.$$

When $\varepsilon = 0$, $\kappa(0; x, y) = 0$ for any nodes x, y. Then, in [LLY], they defined a curvature as the (right) limit $\lim_{\varepsilon \downarrow 0} \kappa(\varepsilon; x, y)/\varepsilon$ instead of simply assigning $\varepsilon = 0$ (Definition A.6). To confirm the existence of this limit, the following two lemmas were shown.

Lemma A.3 ([LLY, Lemma 2.2])**.** For any $\varepsilon \in [0,1]$ and $x, y \in V$, we have

$$|\kappa(\varepsilon; x, y)| \leq \frac{2\varepsilon}{d(x,y)}.$$

Lemma A.4 ([LLY, Lemma 2.1])**.** For two vertices x, y, $\kappa(\varepsilon; x, y)$ is concave in $\varepsilon \in [0,1]$.

The shape of $\kappa(\varepsilon; x, y)$ in $\varepsilon \in [0,1]$ was later examined in more detail.

Theorem A.5 ([BCLMP, Theorem 3.4], [CK, Theorem 3.2])**.** For any connected, locally finite and simple graph $G = (V, E)$ and any of its nodes x, y, the function $\varphi : \varepsilon \mapsto \kappa(\varepsilon; x, y)$ is concave and piecewise linear on $[0,1]$. Moreover, the number of its partitions is at most 3. In particular, $\varphi(\varepsilon) \coloneqq \kappa(\varepsilon; x, y)/\varepsilon$ is constant for ε sufficiently close to 0.

In [BCLMP], the case of $x \sim y$ was shown, then in [CK] established the general case by using the Kantorovich duality of W_1 distance.

From Lemma A.3 and Lemma A.4, we can see the existence of the (right) limit $\lim_{\varepsilon \downarrow 0} \kappa(\varepsilon; x, y)/\varepsilon$.

Definition A.6 (LLY–Ricci curvature; [LLY])**.** We define the *LLY–Ricci curvature* $\kappa(x, y)$ between two nodes $x, y \in V$ as

$$\kappa(x, y) \coloneqq \lim_{\varepsilon \downarrow 0} \frac{\kappa(\varepsilon; x, y)}{\varepsilon}.$$

Although LLY–Ricci curvature $\kappa(x, y)$ is a limit value, it can be obtained by linear programming ([CKLLS]) thanks to Theorem A.5. Moreover, the Graph Curvature Calculator[3] has been developed to calculate LLY–Ricci curvature of each edge of a graph by inputting node and edge information.

[3] https://www.mas.ncl.ac.uk/graph-curvature/ On this page, you can select from the tabs in the lower left corner to calculate various types of curvatures. The tab "Lin–Lu–Yau Curvature" allows you to calculate the curvatures used in this study. Remark that these are calculated on **unweighted graphs**, and the following discussion considers **weighted graphs**.

A.2 "Ricci curvature" between the trajectory and route

According to Definition A.6, we introduce a "curvature" between the trajectory and each route. It is this quantity that determines which routes the trajectory is closer to.

The denominator $d(x, y)$ used in the Definition A.6 (i.e. Definition A.2) was the distance between the two vertices x, y of the target. In this setting, the target to be measured is not two vertices but two "sets of vertices". Therefore, it is necessary to first consider what the quantity corresponding to $d(x, y)$ should be in this setting. In conclusion, we adopt $W_1(\mu_{\mathbf{p}}, \nu_A)$ (with respect to the route A) as described in the midterm presentation. This is because $W_1(\mu_{\mathbf{p}}, \nu_A)$ was quantifying the distance between the trajectory and route using only the location information of the trajectory. We next need to consider the amount of the numerator of Definition A.2. In the idea of Definition A.6, they perturbed the Dirac measures[4] at x, y by ε along the graph structure. In the setting of Wasserstein method (§ 5.2), the vertices of the local road network are only the trajectory and the divided points $\{a_1, \ldots, a_m, b_1, \ldots, b_m\}$ of each route A and B, and thus it is a problem of **how to introduce the edges**. Note that we define the transition probabilities for routes as **combinatorial graphs**. Then, we consider introducing **weighted edges** using the input speed and direction at each trajectory point and we set up and defined them as described in § 5.3.

Now that ε-perturbation measures have been defined, we can define the "curvature" between the trajectory and routes as in Definition A.2 and Definition A.6. In the following, we will write about route A only, since the definition is the same for both routes A and B.

Definition A.7 (ε-Ricci curvature between the trajectory and route)**.** Let $\varepsilon \in [0, 1]$. The *ε-Ricci curvature* $\kappa(\varepsilon; \mathbf{p}, A)$ between the trajectory $\mathbf{p}$ and route A is defined as:

$$\kappa(\varepsilon; \mathbf{p}, A) \coloneqq 1 - \frac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A)}{W_1(\mu_{\mathbf{p}}, \nu_A)}. \tag{A.1}$$

We next want to prove the corresponding properties for Lemma A.3 and Theorem A.5. However, we prove only the concavity of the function $\varepsilon \mapsto \kappa(\varepsilon; \mathbf{p}, A)$ corresponding to Lemma A.4 (Proposition A.9) because the piecewise linearity of h like Theorem A.5 is not yet clearly known (see also § 5.4).

Proposition A.8 (ε-boundedness of $\kappa(\varepsilon; \mathbf{p}, A)$)**.** For any $\varepsilon \in [0, 1]$, we have

$$|\kappa(\varepsilon; \mathbf{p}, A)| \leq \frac{2\varepsilon}{W_1(\mu_{\mathbf{p}}, \nu_A)}.$$

Proof. By the triangle inequality of W_1, we obtain

$$\begin{aligned} W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A) &\leq W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \mu^0_{\mathbf{p}}) + W_1(\mu^0_{\mathbf{p},A}, \nu^0_A) + W_1(\nu^0_A, \nu^{\varepsilon}_A) = W_1(\mu^0_{\mathbf{p},A}, \nu^0_A) + 2\varepsilon = W_1(\mu_{\mathbf{p}}, \nu_A) + 2\varepsilon, \\ W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A) &\geq W_1(\mu^0_{\mathbf{p},A}, \nu^0_A) - W_1(\mu^0_{\mathbf{p},A}, \mu^{\varepsilon}_{\mathbf{p},A}) - W_1(\nu^0_A, \nu^{\varepsilon}_A) = W_1(\mu^0_{\mathbf{p},A}, \nu^0_A) - 2\varepsilon = W_1(\mu_{\mathbf{p}}, \nu_A) - 2\varepsilon. \end{aligned}$$

This implies the following:

$$-\frac{2\varepsilon}{W_1(\mu_{\mathbf{p}}, \nu_A)} \leq \kappa(\varepsilon; \mathbf{p}, A) \coloneqq 1 - \frac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A)}{W_1(\mu_{\mathbf{p}}, \nu_A)} \leq \frac{2\varepsilon}{W_1(\mu_{\mathbf{p}}, \nu_A)}.$$

□

Proposition A.9 (Concavity of $\kappa(\varepsilon; \mathbf{p}, A)$)**.** The function $h : [0, 1] \ni \varepsilon \mapsto \kappa(\varepsilon; \mathbf{p}, A) \in \mathbb{R}$ is concave.

Proof. Let $\varepsilon_1, \varepsilon_2, \varepsilon_3$ be $0 \leq \varepsilon_1 < \varepsilon_2 < \varepsilon_3 \leq 1$ and $t \coloneqq (\varepsilon_3 - \varepsilon_2)/(\varepsilon_3 - \varepsilon_1)$. Then, $\varepsilon_2 = t\varepsilon_1 + (1 - t)\varepsilon_3$ holds. We show that

$$\kappa(\varepsilon_2; \mathbf{p}, A) \geq t\kappa(\varepsilon_1; \mathbf{p}, A) + (1 - t)\kappa(\varepsilon_3; \mathbf{p}, A) \tag{A.2}$$

holds. First, we show the following.

[4] Here, notice that $d(x, y)$ can be transformed to $d(x, y) = W_1(\delta_x, \delta_y)$ and $\mu^0_x = \delta_x$, $\mu^0_y = \delta_y$. This means that the fraction Definition A.2: $W_1(\mu^{\varepsilon}_x, \mu^{\varepsilon}_y)/W_1(\delta_x, \delta_y)$ measures the fundamental probability measures δ_x, δ_y in the denominator and the ε-perturbations $\mu^{\varepsilon}_x, \mu^{\varepsilon}_y$ of them in the numerator, with W_1 between them, respectively.

Claim Let π_j be the optimal coupling between $\mu_{\mathbf{p},A}^{\varepsilon_j}$ and $\nu_A^{\varepsilon_j}$, with respect to $j = 1, 3$. Then, $\pi_2 := t\pi_1 + (1-t)\pi_3$ is a coupling between $\mu_{\mathbf{p},A}^{\varepsilon_2}$ and $\nu_A^{\varepsilon_2}$.

Proof of ***Claim***

By the definition of π_2, we obtain

$$\begin{aligned}\sum_{v_1\in V}\pi_2(v_1,v_2) &= t\sum_{v_1\in V}\pi_1(v_1,v_2) + (1-t)\sum_{v_1\in V}\pi_3(v_1,v_2) = t\nu_A^{\varepsilon_1}(v_2) + (1-t)\nu_A^{\varepsilon_3}(v_2)\\ &= t\cdot\frac{1}{m}\sum_{a\in V(A)}\nu_a^{\varepsilon_1}(v_2) + (1-t)\cdot\frac{1}{m}\sum_{a\in V(A)}\nu_a^{\varepsilon_3}(v_2), \qquad \text{(A.3)}\\ \sum_{v_2\in V}\pi_2(v_1,v_2) &= t\sum_{v_2\in V}\pi_1(v_1,v_2) + (1-t)\sum_{v_2\in V}\pi_3(v_1,v_2) = t\mu_{\mathbf{p},A}^{\varepsilon_1}(v_1) + (1-t)\mu_{\mathbf{p},A}^{\varepsilon_3}(v_1)\\ &= t\cdot\frac{1}{n}\sum_{p\in\mathbf{p}}\mu_{p,A}^{\varepsilon_1}(v_1) + (1-t)\cdot\frac{1}{n}\sum_{p\in\mathbf{p}}\mu_{p,A}^{\varepsilon_3}(v_1). \qquad \text{(A.4)}\end{aligned}$$

It is sufficient to check that the right hand side of (A.3) and (A.4) coincide with $\nu_A^{\varepsilon_2}(v_2)$ and $\mu_{\mathbf{p},A}^{\varepsilon_2}(v_1)$, respectively.

(A.3) (i) In the case of $v_2 \in V(A)$: it holds that

$$\text{(A.3)} = t\cdot\frac{1}{m}(1-\varepsilon_1) + (1-t)\cdot\frac{1}{m}(1-\varepsilon_3) = \frac{1}{m}(1-\varepsilon_2) = \frac{1}{m}\sum_{a\in V(A)}\nu_a^{\varepsilon_2}(v_2) = \nu_A^{\varepsilon_2}(v_2).$$

(ii) In the case of $v_2 \in \mathbf{p}$: it holds that

$$\text{(A.3)} = t\cdot\frac{1}{m}\left(\varepsilon_1\cdot\frac{1}{n}\right)\cdot m + (1-t)\cdot\frac{1}{m}\left(\varepsilon_3\cdot\frac{1}{n}\right)\cdot m = \frac{1}{m}\left(\varepsilon_2\cdot\frac{1}{n}\right)\cdot m = \frac{1}{m}\sum_{a\in V(A)}\nu_a^{\varepsilon_2}(v_2) = \nu_A^{\varepsilon_2}(v_2).$$

(iii) In the case of $v_2 \in V(B)$: it holds that $\text{(A.3)} = 0 = \nu_A^{\varepsilon_2}(v_2)$.

(A.4) (iv) In the case of $v_1 \in \mathbf{p}$: it holds that

$$\text{(A.4)} = t\cdot\frac{1}{n}(1-\varepsilon_1) + (1-t)\cdot\frac{1}{n}(1-\varepsilon_3) = \frac{1}{n}(1-\varepsilon_2) = \frac{1}{n}\sum_{p\in\mathbf{p}}\mu_{p,A}^{\varepsilon_2}(v_1) = \mu_{\mathbf{p},A}^{\varepsilon_2}(v_1).$$

(v) In the case of $v_1 \in V(A)$: it holds that

$$\begin{aligned}\text{(A.4)} &= t\cdot\frac{1}{n}\sum_{p\in\mathbf{p}}\left(\varepsilon_1\cdot\frac{1+w_p(v_1)}{m+1}\right) + (1-t)\cdot\frac{1}{n}\sum_{p\in\mathbf{p}}\left(\varepsilon_3\cdot\frac{1+w_p(v_1)}{m+1}\right) = \frac{1}{n}\sum_{p\in\mathbf{p}}\left(\varepsilon_2\cdot\frac{1+w_p(v_1)}{m+1}\right)\\ &= \frac{1}{n}\sum_{p\in\mathbf{p}}\mu_{p,A}^{\varepsilon_2}(v_2) = \mu_{\mathbf{p},A}^{\varepsilon_2}(v_2).\end{aligned}$$

(vi) In the case of $v_1 \in V(B)$: it holds that

$$\begin{aligned}\text{(A.4)} &= t\cdot\frac{1}{n}\sum_{p\in\mathbf{p}}\left(\varepsilon_1\cdot\frac{w_p(v_1)}{m+1}\right) + (1-t)\cdot\frac{1}{n}\sum_{p\in\mathbf{p}}\left(\varepsilon_3\cdot\frac{w_p(v_1)}{m+1}\right) = \frac{1}{n}\sum_{p\in\mathbf{p}}\left(\varepsilon_2\cdot\frac{w_p(v_1)}{m+1}\right)\\ &= \frac{1}{n}\sum_{p\in\mathbf{p}}\mu_{p,A}^{\varepsilon_2}(v_2) = \mu_{\mathbf{p},A}^{\varepsilon_2}(v_2).\end{aligned}$$

This concludes the proof of **Claim**. ■

From **Claim**, we obtain

$$\begin{aligned}W_1(\mu_{\mathbf{p},A}^{\varepsilon_2},\nu_A) &\le \sum_{v_1,v_2\in V}\pi_2(v_1,v_2)d(v_1,v_2) = t\sum_{v_1,v_2\in V}\pi_1(v_1,v_2)d(v_1,v_2) + (1-t)\sum_{v_1,v_2\in V}\pi_3(v_1,v_2)d(v_1,v_2)\\ &= tW_1(\mu_{\mathbf{p},A}^{\varepsilon_1},\nu_A) + (1-t)W_1(\mu_{\mathbf{p},A}^{\varepsilon_3},\nu_A).\end{aligned}$$

This yields the following:

$$\kappa(\varepsilon_2;\mathbf{p},A) \coloneqq 1-\frac{W_1(\mu_{\mathbf{p},A}^{\varepsilon_2},\nu_A^{\varepsilon_2})}{W_1(\mu_{\mathbf{p}},\nu_A)} \geq t\left(1-\frac{W_1(\mu_{\mathbf{p},A}^{\varepsilon_1},\nu_A^{\varepsilon_1})}{W_1(\mu_{\mathbf{p}},\nu_A)}\right)+(1-t)\left(1-\frac{W_1(\mu_{\mathbf{p},A}^{\varepsilon_3},\nu_A^{\varepsilon_3})}{W_1(\mu_{\mathbf{p}},\nu_A)}\right)$$
$$= t\kappa(\varepsilon_1;\mathbf{p},A)+(1-t)\kappa(\varepsilon_3;\mathbf{p},A).$$

This proves (A.2). □

Definition A.10 (Ricci curvature between the trajectory and route)**.** The existence of the value of the limit $\lim_{\varepsilon\downarrow 0}\kappa(\varepsilon;\mathbf{p},A)/\varepsilon$ is guaranteed by Proposition A.8 and Proposition A.9. We define this value as *Ricci curvature* $\kappa(\mathbf{p},A)$ between the trajectory and route A:

$$\kappa(\mathbf{p},A) \coloneqq \lim_{\varepsilon\downarrow 0}\frac{\kappa(\varepsilon;\mathbf{p},A)}{\varepsilon}.$$

Observation A.11. We determined the route with the smallest (5.5) to be the output as the true route. Note that if (5.8) holds, then we obtain

$$\kappa(\varepsilon;\mathbf{p},A) \coloneqq 1-\frac{W_1(\mu_{\mathbf{p},A}^{\varepsilon},\nu_A^{\varepsilon})}{W_1(\mu_{\mathbf{p}},\nu_A)} < 1-\frac{W_1(\mu_{\mathbf{p},A}^{\varepsilon},\nu_A^{\varepsilon})}{W_1(\mu_{\mathbf{p}},\nu_A)} =: \kappa(\varepsilon;\mathbf{p},B).$$

Therefore, the Wasserstein method is a method that outputs the route with the largest Ricci curvature with the trajectory as the true route.

Remark A.12. As mentioned in § 5.4, we have not shown the piecewise linearity like Theorem A.5 in LLY–Ricci curvature yet.

Remark A.13. This value may require modifications in the way the weights (Definition 5.13) are assigned, as we have not yet calculated the concrete examples.

References

[BCLMP] D. P. Bourne, D. Cushing, S. Liu, F. Münch and N. Peyerimhoff, *Ollivier-Ricci idleness functions of graphs*, SIAM J. Discrete Math. **32**(2) (2018), 1408–1424.

[BK] D. Bernstein and A. Kornhauser, *An introduction to map-matching for personal navigation assistants*, (1996).

[CK] D. Cushing and S. Kamtue, *Long-scale Ollivier Ricci curvature of graphs*, Anal. Geom. Metr. Spaces. **7**(1) (2019), 22–44.

[CKLLS] D. Cushing, R. Kangaslampi, V. Lipiäinen, S. Liu and G. W. Stagg, *The Graph Curvature Calculator and the curvatures of cubic graphs*, Exp. Math. (2019), 13pp. https://doi.org/10.1080/10586458.2019.1660740

[CXHZ] P. Chao, Y. Xu, W. Hua and X. Zhou, *A survey on map-matching algorithms*, Springer, Cham. (2020).

[FG] A. Figalli and F. Glaudo, *An invitation to optimal transport, Wasserstein distances, and gradient flows*, EMS Textbooks in Mathematics. EMS Press (2021).

[GH] GraphHopper Github Repository, https://github.com/graphhopper/graphhopper.

[JL] J. Jost and S. Liu, *Ollivier's Ricci Curvature, Local Clustering and Curvature-Dimension Inequalities on Graphs*, Discrete Comput. Geom. **51**(2) (2014), 300–322.

[Jo] J. Jost, *Riemannian geometry and geometric analysis*, 7th edn. Springer, Berlin (2017).

[KCMMN] M. Kubička, A. Cela, P. Moulin, H. Mountier and S. I. Niculescu, *Dataset for testing and training of map-matching algorithms*, In 2015 IEEE Intelligent Vehicles Symposium (IV), 1088–1093 (2015).

[LLY] Y. Lin, L. Lu and S.-T. Yau, *Ricci curvature of graphs*, Tohoku Math. J. (2) **63**(4) (2011), 605–627.

[LSG] X. Li, Y. Shi and I. Gutman, *Graph energy*, Springer, New York. (2012).

[NK] P. Newson and J. Krumm, *Hidden Markov map matching through noise and sparseness*, In Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, 336–343 (2009).

[NLGGS] C.-C. Ni, Y.-Y. Lin, J. Gao, X. D. Gu and E. Saucan, *Ricci Curvature of the Internet Topology*, In: IEEE Conference on Computer Communications. (2015), 2758–2766.

[NLLG] C.-C. Ni, Y.-Y. Lin, F. Luo and J. Gao, *Community Detection on Networks with Ricci Flow*, Sci Rep **9**, 9984 (2019), 12pp.
https://doi.org/10.1038/s41598-019-46380-9

[Ol] Y. Ollivier, *Ricci curvature of Markov chains on metric spaces*, J. Funct. Anal. **256**(3) (2009), 810–864.

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

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

[Sa] F. Santambrogio, *Optimal transport for applied mathematicians. Calculus of variations, PDEs, and modeling*, Progress in Nonlinear Differential Equations and their Applications, Birkhäuser/Springer, Cham. (2015).

[Vi] C. Villani, *Optimal Transport. Old and New*, Springer-Verlag. (2008).

[YG] C. Yang and G. Gidófalvi, *Fast map matching, an algorithm for integrating a hidden Markov model with precomputation*, International Journal of Geographical Information Science. Taylor & Francis, **32**(3), 547–570 (2018).

[YCWXCLMD] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell, *BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning*, In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2636–2645 (2020).