g-RIPS Sendai 2022

Mitsubishi-A Group

Work Statement

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

May 29, 2023

Contents

1 Introduction **2**

2 Problem Summary and Statement **2**

3 Background **2**
3.1 Geometric Model . . . 3
3.2 Hidden Markov Model . . . 3
3.3 Mathematical Background . . . 5

4 Our Approach **5**
4.1 Wasserstein Distance Method . . . 6
4.2 Euclidean inner product method . . . 8
4.3 Energy functional method . . . 9

5 Implementation **10**
5.1 Evaluation . . . 10
5.2 Data Fusion . . . 11
5.3 Extension to Higher Dimensions . . . 11

6 Deliverables **11**

7 Timeline **11**

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. The aim of our project is 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 road map and trajectory data. We propose three novel methods: Wasserstein distance, inner product, and energy functional methods. Each of these methods has an associated loss (objective) function whose minimum is the the "matched" map. The loss function of each of the three methods employ the mathematical object for which they are named. We plan to develop these approaches into algorithms 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 $d \geq 2$ (but almost everywhere we consider the case $d = 2$).

Definition 2.1 (Trajectory)**.** A **trajectory** Tr is a sequence of points $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ where $p_i \in \mathbb{R}^d$ for $1 \leq i \leq n$ equipped with

- a sequence $t(\mathbf{p}) = (t_1, \ldots, t_n)$ such that $t_i \in \mathbb{R}^+$ for $1 \leq i \leq n$ and $t_1 < t_2 < \cdots < t_n$, called the **time stamp** of $\mathbf{p}$,
- a sequence $\mathrm{spd}(\mathbf{p}) = (\mathrm{spd}_1, \ldots, \mathrm{spd}_n)$ such that $\mathrm{spd}_i \in \mathbb{R}^+$ for $1 \leq i \leq n$, called the **speed** of $\mathbf{p}$ (optional),
- a sequence $u(\mathbf{p}) = (u_1, \ldots, u_n)$ such that $u_i \in \mathbb{R}^d$ and $\|u\| = 1$ for $1 \leq i \leq n$, 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 $\phi : |G| \to \mathbb{R}^d$ 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 (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 2.4 (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 $L : R \to \mathbb{R}^+$, called the **loss function**.

The main goal of this project is to find a suitable loss function such that $\mathcal{MR}_G(Tr)$ is the route which is minimal distance from the actual route traveled by a vehicle or pedestrian taken with respect to a chosen metric (see § 5.1 for details on chosen metrics).

3 Background

In this section, we will review two existing map-matching methods, the point-to-point 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]. In addition, we will present an introduction to discrete optimal transport theory which will be necessary for the explanation of one of our proposed map-matching algorithms.

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 algorithms is the point-to-point method (for this method only, consider a route to instead be a sequence of vertices in the road network). The procedure for the point-to-point method is given in [BK] to be

1. Compute the distance between each trajectory point p_i to v for each $v \in V$,
2. Match trajectory point p_i to the vertex $v \in V$ for which the euclidean distance between p_i and v is minimal.

Geometric methods like the point-to-point method are easily implemented and have low computational time [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 1 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, this method, and other 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 2 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.

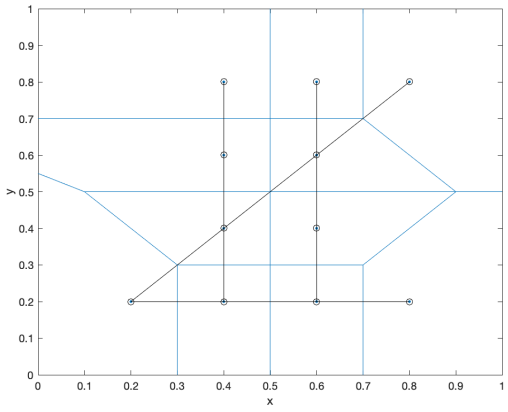

Figure 1: Voronoi Example

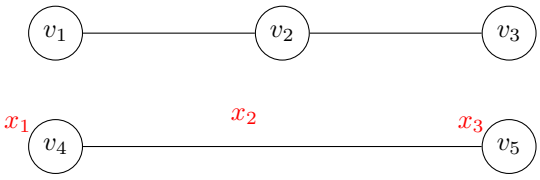

Figure 2: Geometric Method Error

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].

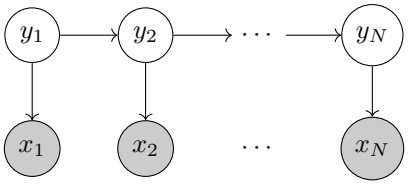

Figure 3: Hidden Markov Model (HMM)

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. 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 a sequence of observed x_i's.

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 4:

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

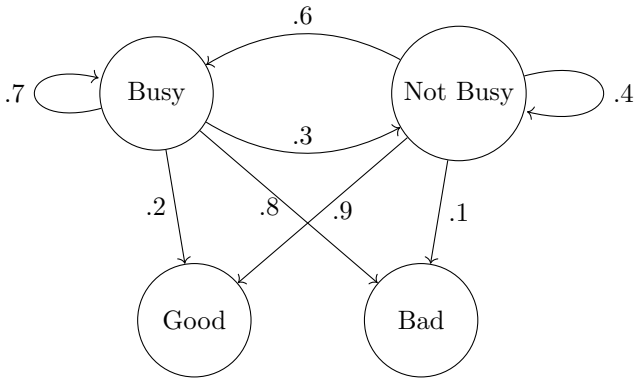

Figure 4: Transition and Emission probabilities for Example 3.1.

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 likelihood of the route containing some edge, given the previous edge, but the emission probabilities describe the probability that of observing a trajectory point given some edge in the road network. The choice of these parameter 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. 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.

3.3 Mathematical Background

A brief introduction of (discrete) optimal transport theory

We will briefly describe discrete optimal transport theory (see also [FG, Sa, Vi], etc.). Consider transporting sand from a sand pile at $x_1, \dots, x_n$ to a hole at $y_1, \dots, y_m$. Note that $n, m \in \mathbb{N}$ are independent. Suppose that each sand pile $x_1, \dots, x_n$ has $\mu(x_1), \dots, \mu(x_n)$ mass of sand, respectively, and each hole $y_1, \dots, y_m$ can contain $\nu(y_1), \dots, \nu(y_m)$ 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:

$$\sum_{i=1}^{n} \mu(x_i) = \sum_{j=1}^{m} \nu(y_j). \tag{3.1}$$

For simplicity, we normalize both sides of (3.1) 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.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} d(x_i, y_j)\pi(x_i, y_j) & & \\ \text{subject to} \quad & \pi(x_i, y_j) \geq 0 & & \text{for } i = 1, 2, \dots, n,\ j = 1, 2, \dots, m, \\ & \mu(x_i) = \sum_{j=1}^{m} \pi(x_i, y_j) & & \text{for } i = 1, 2, \dots, n, \\ & \mu(y_j) = \sum_{i=1}^{n} \pi(x_i, y_j) & & \text{for } j = 1, 2, \dots, m. \end{aligned} \tag{3.2}$$

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

$$W_1(\mu, \nu) \coloneqq \inf_{\pi \in \Pi(\mu,\nu)} \sum_{i=1}^{n}\sum_{j=1}^{m} d(x_i, y_j)\pi(x_i, y_j). \tag{3.3}$$

Under the appropriate conditions, W_1 is a distance function on the probability measure space. A probability measure space often contains geometrical information about its underling space. Therefore, it is useful to analyze the probability measure space using Wasserstein distance to understand the geometrical structure of its underlying space.

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 close set. Hence, the right hand side of (3.3) attains the minimum value.

4 Our Approach

In this section, we describe our approaches. We introduce the three methods each employing one of the following mathematical tools

- Wasserstein Distance,
- Euclidean inner product,
- Graph energy functional.

The first two methods are based on the approach of considering the road network graph as a set of lines in $\mathbb{R}^2$, and the last method is based on the approach of focusing on the graph structure.

4.1 Wasserstein Distance Method

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

We assume that the input data on trajectory is only its coordinates and timestamps. We first consider the case where the edges of the road network graph form a square, and the trajectory points are located around in a neighborhood of the diagonal of the square. We consider this to be the simplest case where is it is difficult to determine the true route. Suppose that the trajectory points $p_1, \ldots, p_n$ is observed near the diagonal as shown in Figure 5. Although the trajectory points are drawn in a zigzag pattern in Figure 5, 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 judge from this trajectory alone 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 method of route determination based on the optimal transport theory described in § 3.3. We can apply this method in theory even in general $\mathbb{R}^d$ and in the situation where each edge of the graph is a curve.

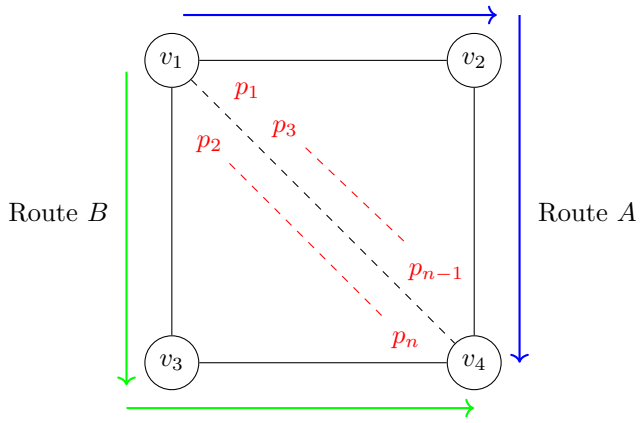

Figure 5: The case where the trajectory points $p_1, \ldots, p_n$ is observed near the diagonal of a square road network.

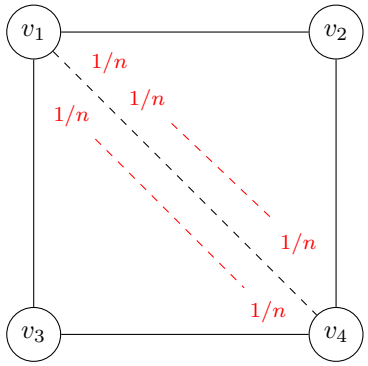

Figure 6: A probability measure $\mu_{\mathbf{p}}$ on the trajectory points. The weight $1/n$ is placed on the trajectory points $p_1, \ldots, p_n$ respectively.

We reduce the problem of route determination to the framework of an optimal transportation problem by introducing the following three types of probability measures: probability measures associated with the trajectory points and routes A and B. In other words, our idea is to quantify the distance between the trajectory points $\mathbf{p} = \{p_1, \ldots, p_n\}$ and each route by measuring Wasserstein distance between probability measures associated with the trajectory points and route.

First, we introduced the probability measure $\mu_{\mathbf{p}}$ associated with the trajectory points $\mathbf{p}$.

Definition 4.1 (A probability measure associated with the trajectory points)**.** We define a probability measure $\mu_{\mathbf{p}}$ as by putting weights of $1/n$ on each points in $\mathbf{p}$ (see also Figure 6):

$$\mu_{\mathbf{p}} \coloneqq \frac{1}{n}\delta_{p_1} + \cdots + \frac{1}{n}\delta_{p_n}.$$

Then, we introduced the probability measures associated with the route A, B by dividing each routes into $m+1$ equal parts and putting weights by $1/m$ on the points that are its thresholds.

Definition 4.2 (The probability measures associated with the route A and B)**.** We divide the route A into $m+1$ equal parts along the edges, and denote the threshold points as $a_1, \ldots, a_m$, starting from the point closest to v_1. We define a probability measure $\nu_{A,m}$ as by putting weights of $1/m$ on each threshold points on the route A (see also Figure 7):

$$\nu_{A,m} \coloneqq \frac{1}{m}\delta_{a_1} + \cdots + \frac{1}{m}\delta_{a_m}.$$

We define $\nu_{B,m}$ in exactly the same way (see also Figure 8):

$$\nu_{B,m} \coloneqq \frac{1}{m}\delta_{b_1} + \cdots + \frac{1}{m}\delta_{b_m}.$$

Remark 4.3. In the above definitions, n and m are independent, while m, the number of supports of $\nu_{A,m}$ and $\nu_{B,m}$, are both equal: for any $m \in \mathbb{N}$, $\#\operatorname{supp}\nu_{A,m} = \#\operatorname{supp}\nu_{B,m}$ $(= m)$. 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.

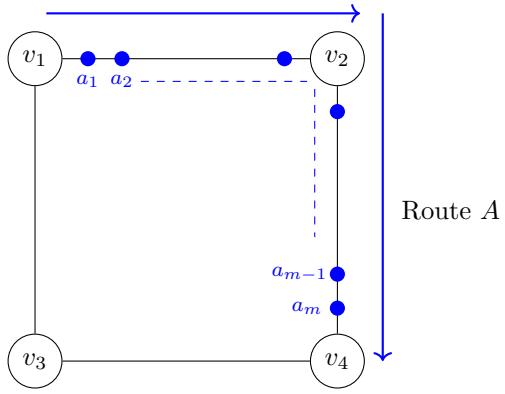

Figure 7: A probability measure $\nu_{A,m}$ associated with the route A. The edges (v_1, v_2) and (v_2, v_4) forming the route A are divided into $m+1$ equal parts, each with $1/m$ weight on its threshold points $a_1, \ldots, a_m$.

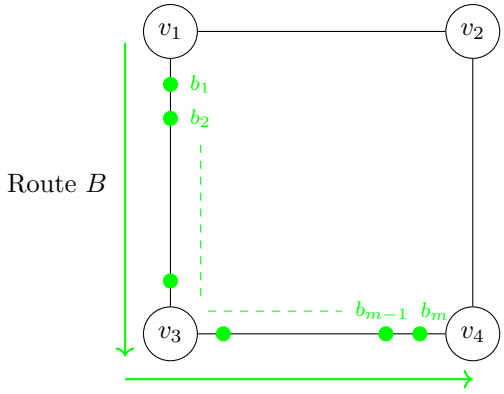

Figure 8: A probability measure $\nu_{B,m}$ associated with the route B. The edges (v_1, v_3) and (v_3, v_4) forming the route B are divided into $m+1$ equal parts, each with $1/m$ weight on its threshold points $b_1, \ldots, b_m$.

Definition 4.4. For $\mu_{\mathbf{p}}$, $\nu_{A,m}$, $\nu_{B,m}$, we define

$$\varphi(A, m) \coloneqq W_1\big(\mu_{\mathbf{p}}, \nu_{A,m}\big), \qquad \varphi(B, m) \coloneqq W_1\big(\mu_{\mathbf{p}}, \nu_{B,m}\big).$$

As m is increased, $\operatorname{supp}\nu_{A,m}$ and $\operatorname{supp}\nu_{B,m}$ approach route A: $(v_1, v_2) \to (v_2, v_4)$ and route B: $(v_1, v_3) \to (v_3, v_4)$ respectively. Our idea is that we can judge which route is closer to the trajectory points $\mathbf{p}$ by computing $\varphi(A, m) - \varphi(B, m)$ for a sufficiently large $m \in \mathbb{N}$. Then, we need to show the following.

Problem 4.5. There exists a $\widetilde{m} \in \mathbb{N}$ such that $\varphi(A, m) - \varphi(B, m)$ is a monotone function for any $m \in \mathbb{N}$ such that $m \geq \widetilde{m}$.

We can judge which route the trajectory points is closer to by checking the sign of $\varphi(A, m) - \varphi(B, m)$ for a sufficiently large m if we can prove Problem 4.5. Hence, if $\varphi(A, m) < \varphi(B, m)$ holds, then we can conclude that the trajectory points is closer the route A, which is the true route.

Remark 4.6. Although $\varphi(A, m)$ and $\varphi(B, m)$ take finite values for any $m \in \mathbb{N}$, they might not converge as $m \to \infty$. In fact, 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 4.7. Even if all trajectory points are on the route A, $\varphi(A, m) \neq 0$ 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 4.8. Note that the trajectory points $\mathbf{p}$ is a finite set, and the 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 actually a "point-to-(point on curve)" iteration. In contrast, our method transports weights from trajectory points $\mathbf{p}$ to a candidate route, in other words, it is a "(trajectory points)-to-route" process. The effect of outliners should be small because we deal with all trajectory points at the same. In this sense, we want the method to be robust to observation errors.

Incorporating speed and direction data

We consider to develop the above discussion in the case where we also have information on speed and direction. In calculate $\varphi(A, m)$, we consider

$$\sum_{i=1}^{n}\sum_{j=1}^{m} d(p_i, a_j)\pi(p_i, a_j) \tag{4.1}$$

as loss function and solve a linear programming problem (see (3.2) in § 3.3). Therefore, we can apply the method using Wasserstein distance in this setting if we can modify this loss function (4.1) so that associate with the trajectory speed and direction information.

Problem 4.9. How to include the trajectory speed and direction information in the loss function (4.1)?

In the case of reflecting the trajectory speed and direction information, there are different future developments depending on whether the probability measure or the distance (or both) is modified. The term to be modified is different this is because $d(p_i, a_j)$ and $\pi(p_i, a_j)$ represent the distance between two points and the transport mass, respectively in (4.1). At the present stage, it seems that the distance between the two points should be modified, i.e. we introduce a flow locally within $\mathbb{R}^2$ that takes into account the trajectory speed and direction information.

4.2 Euclidean inner product method

We consider making the geometric method more accurate by taking into account not only the position of each trajectory point but also the vector connecting each point, calculating the inner product of that vector and the edge of the route, and measuring how close the oriented directions are. This method shares similarities with "topological methods" described in [QOZN] but we believe that our method has advantages over this one.

Specific procedure

We assume that the image of a given road network $\phi(|G|)$ for $G = (V, E), \phi : G \to \mathbb{R}^d$ is a piecewise linear curve and have $\phi(V) =: \{V_\alpha\}_\alpha$ as vertices of that. Also we assume that input data on trajectory is only its coordinate. We first list all candidate routes by point-to-point method and point-to-curve method. That is, we determine the closest vertex and edge of road network for each trajectory points and list routes that are composed of those vertices and edges. Then we define score for each edge contained in the candidates and adopt the one which has the highest score as the route. The factors that determine the score are the inner product of the line segment connecting the trajectory points and the edge, and the closeness of their positions.

The specific procedure for defining score is as follows. We define vectors $v_i = p_{i+1} - p_i$ and $v_{\alpha\beta} = V_\beta - V_\alpha$ for a pair of trajectory points p_i, p_{i+1} and each edge $e_{\alpha\beta}$ which connects vertices V_α and V_β, contained in candidates routes, as shown in Figure 9.

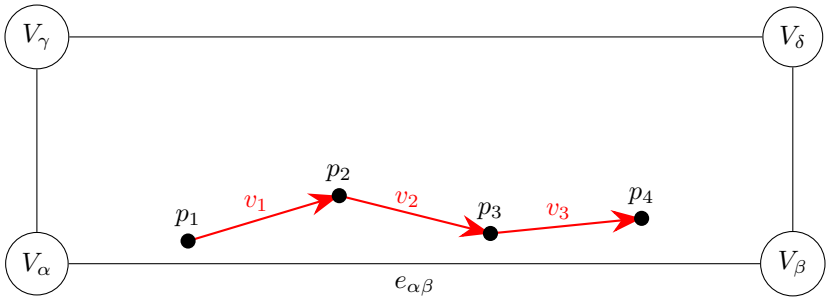

Figure 9: Vectors defined by connecting trajectory points

Then we define $P(v_i, v_{\alpha\beta}) := \left\langle \frac{v_i}{|v_i|}, \frac{v_{\alpha\beta}}{|v_{\alpha\beta}|} \right\rangle$ where $\langle -, - \rangle$ denotes the inner product on the Euclidean space $\mathbb{R}^d$. Let $L_{i,e}$ be the distance between the trajectory point p_i and the point on the edge e which is closest to p_i. We define the score

$$S(e) := \sum_i \frac{P(v_i, v_{\alpha\beta})}{1 + L_{i,e}}$$

for each edge.

We are also considering a methods that we use vectors representing the speed spd_i and direction u_i at the trajectory point p_i instead of v. In this case, we can calculate the score of edge $e_{\alpha\beta}$ as follows:

$$S(e) := \sum_i \frac{\mathrm{spd}_i \langle u_i, v_{\alpha\beta} \rangle}{1 + L_{i,e}}$$

In addition, we also consider taking average v_i and $\mathrm{spd}_i \cdot u_i$ to average out the errors in the position, velocity, and direction data.

Remark 4.10. This method has some points in common with the method called the "topological method" [QOZN], but it is considered to be a simpler method because, unlike the topological method, it does not require the calculation of angles and a classification according to their magnitudes.

Problem 4.11.

1. How to modify this method to one for the case when the image of a given road network is not a piecewise linear curve but Strongly curved curve?
2. How to improve this methods for the case when we also have information on velocity and angle?

4.3 Energy functional method

We would like to define an energy functional on the set of routes so that we can determine the most suitable route as the one that minimizes the value of the functional as we wrote in the end of Section 2. At this time, we want to define the energy functional as reflecting the weights given to each routes and find a quantity that is a necessary condition for minimizing the energy functional and is particularly easy to compute.

Example 4.12. Let us consider about energy of a curve in Riemannian geometry. Take a Riemannian manifold (M, g) and a smooth curve $c : [0, 1] \to M$, the energy of c is defined by $E(c) := \int_0^1 g(\frac{dc}{dt}, \frac{dc}{dt})dt$. This energy functional is defined on a set consists of all curves on M and critical points of this functional are geodesic, i.e., for Levi-Civita connection ∇ of M, curves which satisfy the equation $\nabla^{c^* TM}_{\frac{dc}{dt}} \frac{dc}{dt} = 0$. To satisfy the geodesic equation means that c is "straight" in M. As we see bend of a curve has a close relation with and characterize length. See also [Jo].

Making use of graph energy

We think that the notion of graph energy in graph theory may be useful for our plan described above. A transition matrix is a probabilistic way to describe the movement between vertices on a graph. Therefore, we take one transition matrix by determining the probability of movement considering information such as the speed of a given trajectory points. In general, the random walk normalized Laplacian can be defined from the transition matrix. As a natural energy associated with the graph structure, we can consider the Laplacian energy which is defined as the sum of absolute values of eigenvalues of the graph Laplacian.The graph energy described here is a potential candidate for a suitable energy functional. See also [LSG].

Remark 4.13. In the case of using graph energy or its variants, we only use information about structures of graph so we can apply this method to settings in arbitrary dimensions.

5 Implementation

After formulating the proposed mathematical methods into robust map-matching algorithms, we will implement them in python to evaluate their performance numerically. We will evaluate them their performance on one of the following open source data sets

- *Dataset for testing and training of map-matching algorithms* [KCMMN] (GPS only, has ground truths),
- The BDD100K open data set provided by Berkeley [YCWXCLMD] (for GPS and IMU data, no ground truths).[1]

We will also compare the performance of our methods to a geometric method, such as point-to-curve, and HMM method, such as an extended Kalman filter (EKF) or Fast Map-Matching [YG].

5.1 Evaluation

How do we measure the accuracy of our prediction? There isn't a consensus in the literature, so we will evaluate our predictions using the following error calculations:

- Newson and Krumm presented the formula:

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

  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 10 [NK] for a visual explanation.

- Fréchet distance
- Euclidean distance

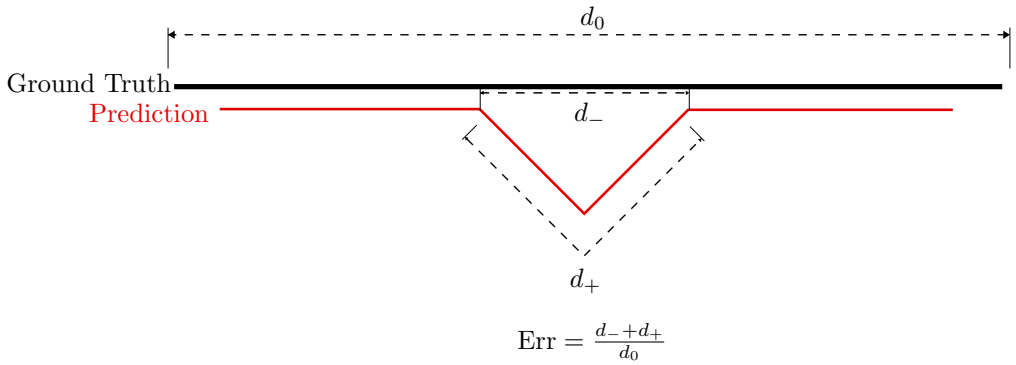

Figure 10: Error Formula by Newson and Krumm

[1] Because there are no public annotated ground truths, we compare our predictions with the standard EKF approach. This evaluation method is flawed but unavoidable.

5.2 Data Fusion

Furthermore, the Jupyter notebooks will develop a framework for implementing and testing other algorithms against data sets. One issue that pervades this field of research is the lack of standard formatting–namely, there are several different GIS data types which differ slightly, and in particular none of these formats are well-suited for containing IMU data. Our notebooks will provide a rudimentary fusion method to align asynchronous data and incorporates IMU data into GPX/GeoDataFrame in a sensible manner. In particular, this method will subsample discrete-time data such as speed, accelerometer, and gyroscope, and merge it with the GPS data sequence.

The benefit these data types provide is being simple (human-readable and easy to convert from/to) and lightweight (quick to access and manipulate, and simple to compress). Moreover, it is easy to convert from these data types to others, unlike proprietary formats such as ESRI ShapeFile. Testing algorithms against data sets then reduces to formatting your inputs/outputs properly. This is designed for accessibility, so other students or researchers can experiment with their own ideas and obtain results more easily.

The code written for the project can be found at: https://github.com/gjgress/G-RIPS-2022-Mitsubishi-A.

5.3 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 region.

Fortunately, the Python framework developed is quite modular, and 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 public.

6 Deliverables

- Three novel approaches to map-matching
- Comparison of these methods with current method
- Python scripts and Jupyter notebooks for reproducibility

7 Timeline

The following is our proposed timeline for the project.

Week 1-2

- Write project proposal
- Develop Python framework to manipulate map structures and set up data sets

Week 3-4

- Expand preliminary ideas into robust map-matching algorithms
- Implement the algorithms
- Prepare and deliver midterm presentation (15/07)

Week 5-6

- Evaluate performance of proposed map-matching methods by
  - deriving theoretical estimates for accuracy
  - applying our algorithms and preexisting algorithms to the same map-matching data set and comparing accuracy

Week 7-8

- Summarize and present results by
  - preparing and giving final presentation (08/08)
  - writing final report (due 09/08)

References

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

[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.

[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).

[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).

[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).