New geometric approaches to the map-matching problem

T. Akamatsu, G. Gress, K. Huneycutt, S. Omura
Academic Mentor: Dr. Kano
Industry Mentor: Dr. Yamazaki

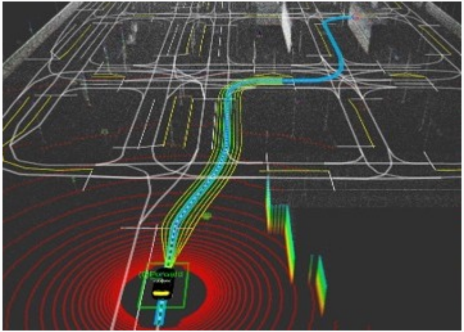
June 1, 2023

Introduction to map-matching

Map-matching

Given GPS trajectory data and a road map, **map-matching** is the process of determining the route on the map that corresponds to the trajectory data.

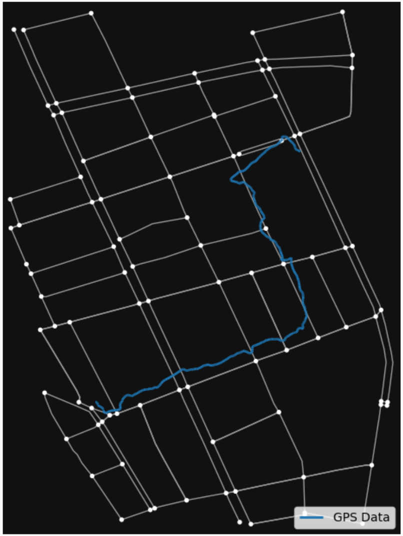
Web mapping services



Autonomous Vehicles [H]

Map-matching

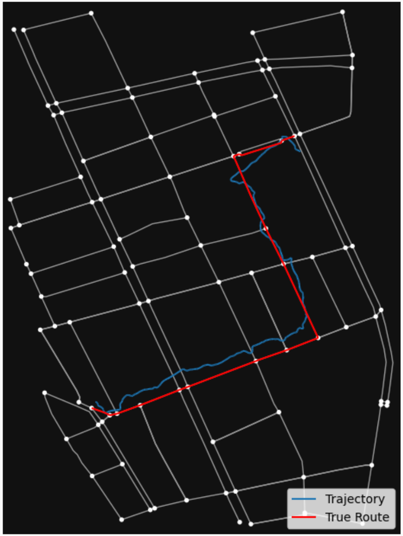
Example Trajectory in Road Network of Sendai



Map-matching

$\Longrightarrow$

Example of Successful Map Matching



How do we model this mathematically?

Mathematical Formulation of the Problem

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

Definition (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 $\text{spd}(\mathbf{p}) = (\text{spd}_1, \ldots, \text{spd}_n)$ 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).

Mathematical Formulation of the Problem

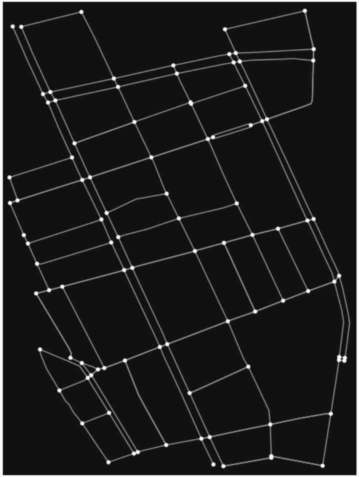
Definition (Road Network)

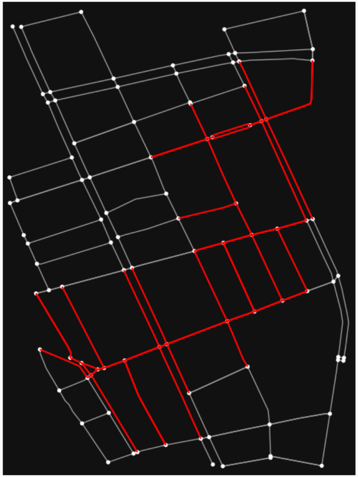
A **road network** (also known as a map) is a directed graph $G = (V, E)$ consists of the set V (resp. E) of vertices (resp. edges) with an embedding $\phi : |G| \to \mathbb{R}^N$ of the geometric realization $|G|$ of G. We will identify G and the image $\phi(|G|)$ by ϕ as long as there is no confusion.

Definition (Local Road Network)

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

(Local) Road Network





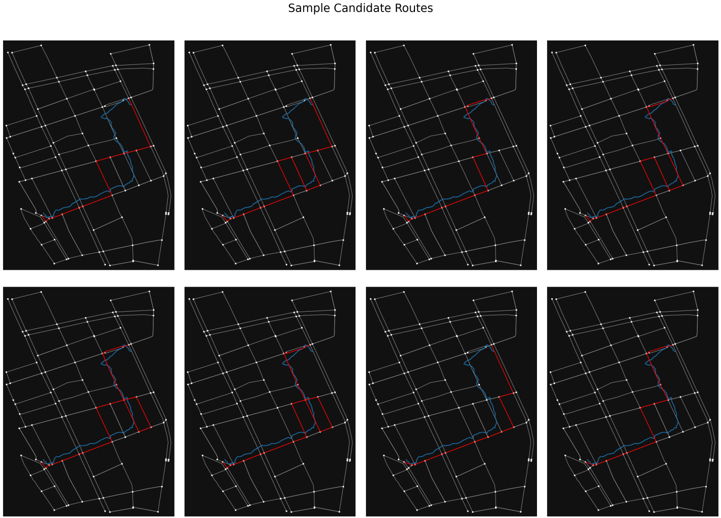
Problem Statement

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

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

Candidate Routes



Definition (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 $L : \mathcal{CR} \rightarrow \mathbb{R}^+$, called the **loss function**.

Other details

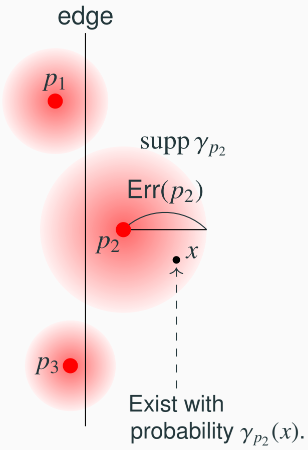
Assumption

- Give the **GPS error** as $\mathrm{Err} : \mathbf{p} \to \mathbb{R}_{\geq 0}$ and assume that the *spherically-symmetric probability measure* γ_p (e.g. *Gaussian measure*) is given such that

$$\operatorname{supp} \gamma_p = B(p; \mathrm{Err}(p)) := \left\{ x \in \mathbb{R}^N \;\middle|\; d_{\mathbb{R}^N}(x, p) \leq \mathrm{Err}(p) \right\}.$$

  We assume that $p \in \mathbf{p}$ is truly located at x with probability $\gamma_p(x)$.

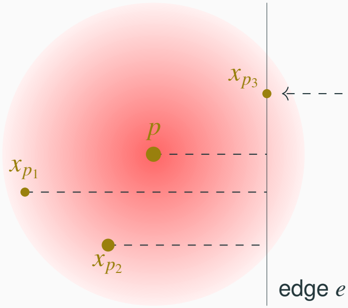
- Suppose that **there is NO error with respect to the speed and direction information.**



Definition (The “distance" with error between $p \in \mathbf{p}$ and $e \in E$)

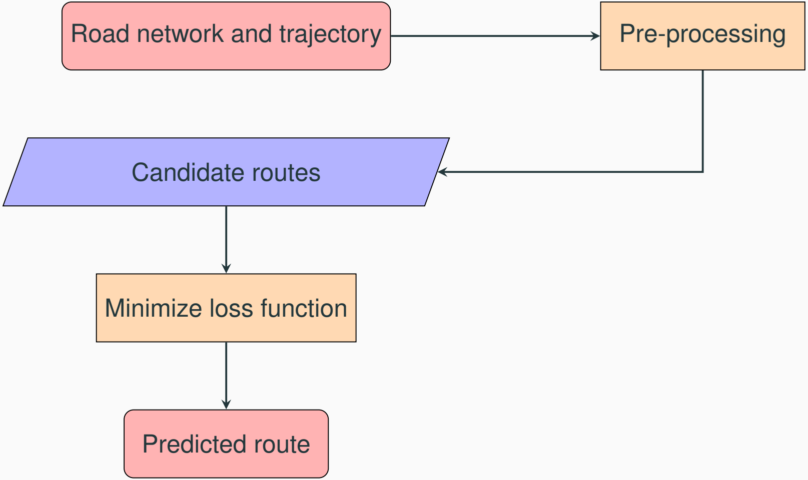
• Define the **“distance" with error** $\mathsf{d}_{\mathsf{Err}}$ between $p \in \mathbf{p}$ (*with errors*) and $e \in E$ (*without errors*) as

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



$$d_{\mathbb{R}^N}(x_{p_3}, e) := \min_{x_e \in e} d_{\mathbb{R}^N}(x_{p_3}, x_e)$$
$$= 0$$

Overview of process

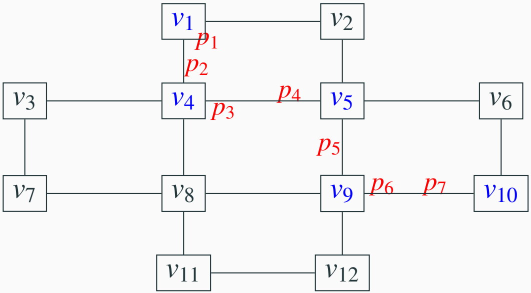


”Standard” Methods

Point-to-point

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 as:

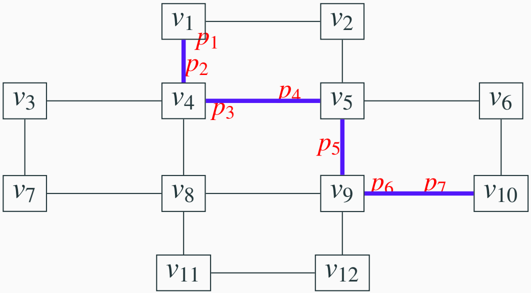
1. For each trajectory point, p_i, compute the distance from p_i to v for each $v \in V$,
2. Find $v^* \in V$ such that the euclidean distance between p_i and v^* is minimal.
3. Let the route found be the sequence of the v^*'s found for each p_i, $1 \leq i \leq n$.



Point-to-curve

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.



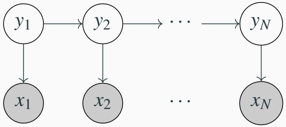
Data-driven methods

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

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

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.

- $p(x_i|y_i)$ - **emission probability**
- $p(y_i)$ - **initial distribution**



Data-driven models

This method can be used to find the sequence of events with the highest probability given a sequence of observations.

Pros:

- Low computational cost
- Adding additional points does not require a complete re-computation
- Fine-tunable

Cons:

- Sequence of events not guaranteed to be valid
- Trained models ineffective on unfamiliar data
- Difficult to incorporate auxiliary data

For our testing, we utilized an open-source Hidden Markov Model designed for map-matching called *Fast Map Matching*.

Fast Map Matching

FMM is an open source map matching framework in C++ and Python. It solves the problem of matching noisy GPS data to a road network. The design considers maximizing performance, scalability and functionality.

Geometric methods

Why geometric methods?

Geometric methods leverage the inherent geometric properties of the route to determine the most likely true route.

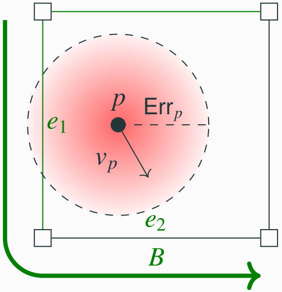
Pros:

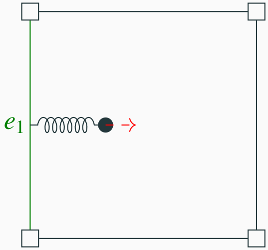
- Robust under error or missing data
- Not prone to overfitting
- Can incorporate auxiliary data (velocity, acceleration, etc.)
- Incorrect results are still "reasonable"

Cons:

- Usually computationally more expensive
- Adding additional data usually requires re-computation
- Usually requires more pre-processing

Harmonic oscillator Method



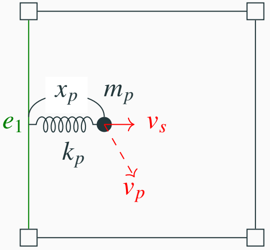


- $p \in \mathbb{R}^N$: trajectory point,
- $v_p = \mathrm{spd}_p u_p \in \mathbb{R}^N$: speed at p,
- $\mathrm{Err}(p) \in \mathbb{R}$: error of p,
- $B \in \mathcal{CR}, \quad e \in B$
- $S(p, e)$: *score* of edge e

$$:= \langle v, \vec{e} \rangle_{\mathbb{R}^N} \exp\left(-\mathrm{d}_{\mathrm{Err}}(p, e)\right)$$

- Connect trajectory point to the highest score edge by “spring”.
- Define Lagrangian of this system.

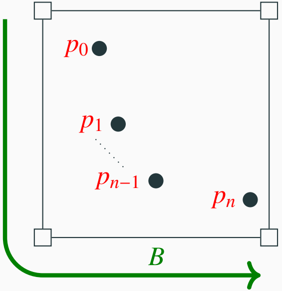
Harmonic oscillator Method

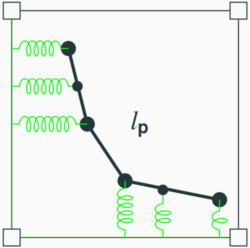


- v_s : spring direction component of v_p
- $x_p = d_{\mathrm{Err}_p}(p, e)$: “displacement” of p
- $m_p = \frac{1}{1+\mathrm{Err}(p)}$: “mass’ of p,
- $k_p = \exp(-\mathrm{Err}(p))$: “spring constant” w.r.t. p,
- $M(p) = m_p \left\| \frac{v_s}{\log(1+|v_p|)} \right\|^2$: “momentum” of p,
- $P(p) = k_p x_p^2$: “potential” of p,

$L := M(p) + P(p)$: Lagrangian.

Harmonic oscillator method





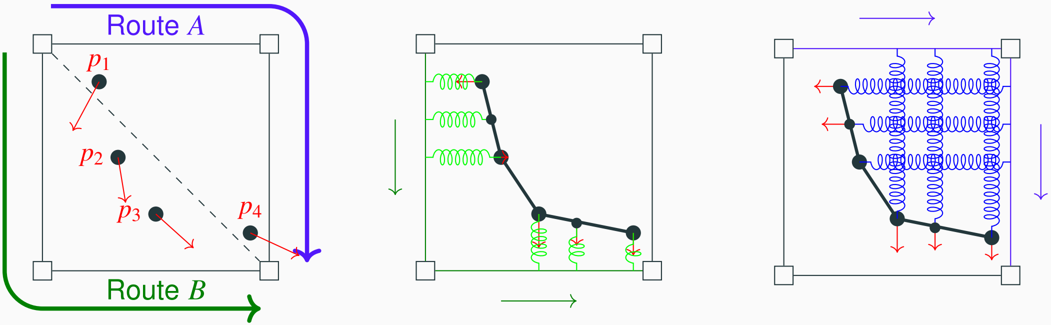
- $\mathbf{p} = (p_i)_{0\le i\le n}$: traj. points,
- $v_{\mathbf{p}} = (v_i = \mathrm{spd}_i u_i)_{0\le i\le n}$: velocity,
- $\mathrm{Err} : \mathbf{p} \to \mathbb{R}$: error,
- $B \in \mathcal{CR}$: route.

$$\downarrow$$

- $l_{\mathbf{p}} : [0,1] \to \mathbb{R}^N$: polyline,
- $v_{l_{\mathbf{p}}} : [0,1] \to \mathbb{R}^N$: velocity,
- $\mathrm{Err}_{l_{\mathbf{p}}} : [0,1] \to \mathbb{R}$: error,

$$L(t) := M(l_{\mathbf{p}}(t)) + P(l_{\mathbf{p}}(t)), \quad Act_{\mathbf{p}}(B) := \int_{[0,1]} L(t)\,dt : \text{“action” of } \mathbf{p} \text{ on } B.$$

Harmonic oscillator method

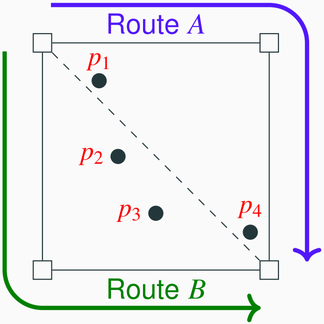


- Calculate $Act_{\mathbf{p}}(A)$ and $Act_{\mathbf{p}}(B)$.
- Choose the route that minimize action.

Harmonic oscillator method (summary)

Summary:

- Similar to least-squares methods but with additional nuances
- Simple to implement, computationally cheap
- Takes into account speed and direction



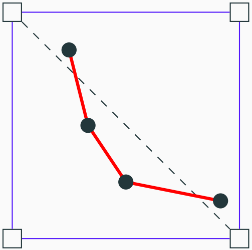


Figure 1: Connecting traj. pts. and giving charges.

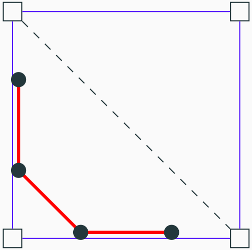


Figure 2: Moving to "closer" route.

Considers not only trajectory points, but also the entire polyline.

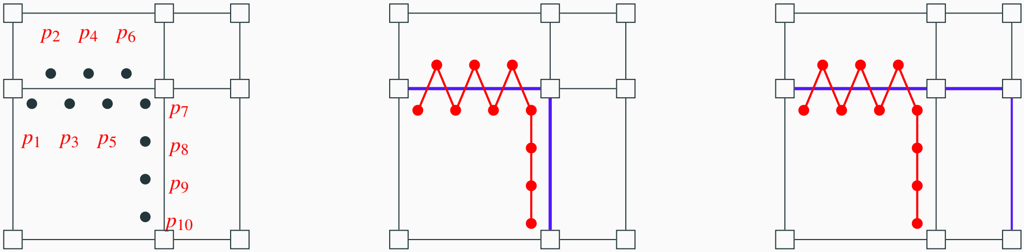
$$\int_{\text{polyline}} \int_{\text{route}} r^{-2}$$

Electric Method

Unfortunately, this method has some fundamental flaws.

- Divergence problem
- Does not take into account speed and direction information.

Even using $\int_{\text{polyline}} \int_{\text{route}} (r^2 + \varepsilon)^{-1}$, affects of intersection point is too large.



Definition ((L^1-)Wasserstein distance)
Let (X, d) be a complete and separable metric space. For probability measures μ, ν with finite supports, we define W_1 *distance* between μ and ν as

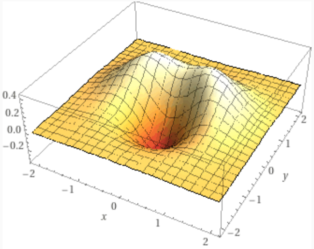
$$W_1(\mu, \nu) := \min_{\pi \in \Pi(\mu,\nu)} \sum_{x \in X} \sum_{y \in X} d(x, y)\pi(x, y),$$

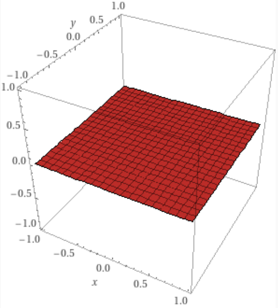
where $\pi \in \Pi(\mu, \nu) \;:\Leftrightarrow\;$ for any $x, y \in X$, $\sum_{y \in X} \pi(x, y) = \mu(x)$, $\sum_{x \in X} \pi(x, y) = \nu(y)$.

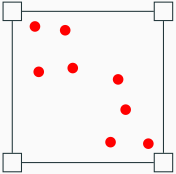
Definition (Probability measure associated with $\mathbf{p}$ and $A \in \mathcal{CR}$)

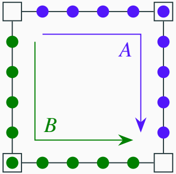
- For the trajectory $\mathbf{p}$, define $\mu_{\mathbf{p}} := (1/n) \sum_{p \in \mathbf{p}} \delta_p$.
- ▹ Devide each $A \in \mathcal{CR}$ into $m + 1$ equal parts and $V(A, m)$ denotes the set of m threshold points.
  ▹ Define $\nu_A := (1/m) \sum_{a \in V(A,m)} \delta_a$.

Wasserstein method (distance)



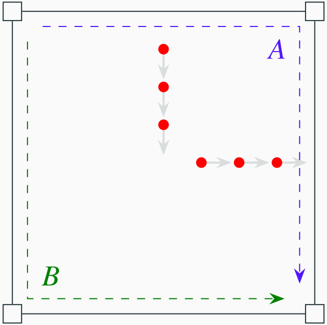






Compare $W_1(\mu_{\mathbf{p}}, \nu_A)$ with $W_1(\mu_{\mathbf{p}}, \nu_B)$.

Wasserstein method (motivation)



- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

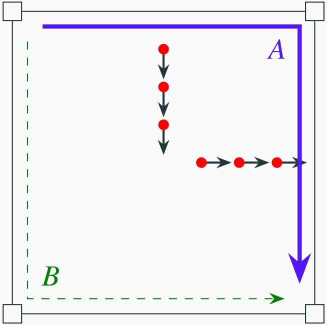
Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected.**

However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_{A} and ν^{ε}_{B} that include speed and direction information.
- Normalize with location data
  - ▹ $\dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_{A})}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_{B})}{W_1(\mu_{\mathbf{p}}, \nu_B)}$.
  - ▹ **Route B is then selected.**

Wasserstein method (motivation)



- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

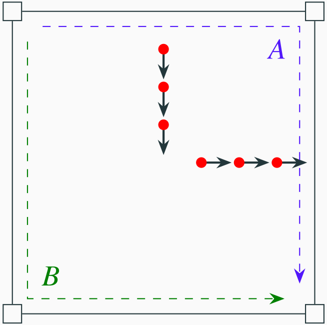
Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected.**

However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_A and ν^{ε}_B that include speed and direction information.
- Normalize with location data
  - ▹ $\dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A)}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_B)}{W_1(\mu_{\mathbf{p}}, \nu_B)}$.
  - ▹ **Route B is then selected.**

Wasserstein method (motivation)



- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

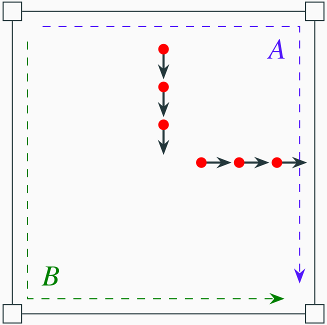
Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected.**

However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_{A} and ν^{ε}_{B} that include speed and direction information.
- Normalize with location data
  - ▹ $\dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_{A})}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_{B})}{W_1(\mu_{\mathbf{p}}, \nu_B)}$.
  - ▹ **Route B is then selected.**

Wasserstein method (motivation)



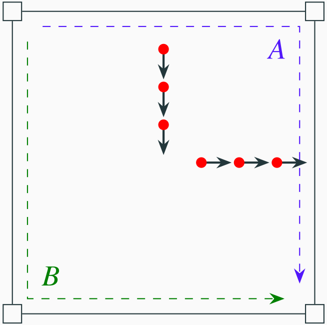
- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected.**

However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_{A} and ν^{ε}_{B} that include speed and direction information.
- Normalize with location data
  - ▹ $\dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_{A})}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_{B})}{W_1(\mu_{\mathbf{p}}, \nu_B)}$.
  - ▹ **Route B is then selected.**



- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

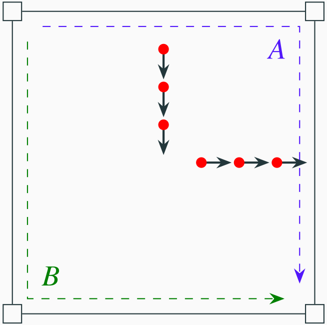
Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected.**

However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_{A} and ν^{ε}_{B} that include speed and direction information.
- Normalize with location data
  - ▹ $\dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_{A})}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_{B})}{W_1(\mu_{\mathbf{p}}, \nu_B)}$.
  - ▹ **Route B is then selected.**

Wasserstein method (motivation)



- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected.**

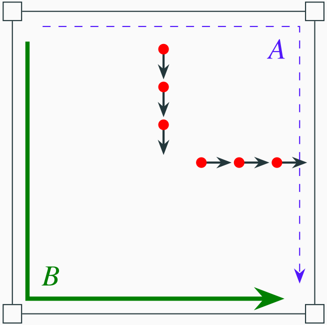
However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_A and ν^{ε}_B that include speed and direction information.
- Normalize with location data

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

▹ **Route B is then selected.**

Wasserstein method (motivation)



- Each $p \in \mathbf{p}$ is located on the vertical or parallel bisectors.
- spd(p), Err(p) are the same at each p, respectively.

Suppose we consider location only.

- ▹ $W_1(\mu_{\mathbf{p}}, \nu_A) < W_1(\mu_{\mathbf{p}}, \nu_B)$.
- **Route A is then selected**.

However, we can consider auxiliary information with the Wasserstein method.

- Introduce probability measures $\mu^{\varepsilon}_{\mathbf{p},A}$, ν^{ε}_{A} and ν^{ε}_{B} that include speed and direction information.
- Normalize with location data
- ▹ $\dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_{A})}{W_1(\mu_{\mathbf{p}}, \nu_A)} > \dfrac{W_1(\mu^{\varepsilon}_{\mathbf{p},B}, \nu^{\varepsilon}_{B})}{W_1(\mu_{\mathbf{p}}, \nu_B)}$.
- ▹ **Route B is then selected.**

Summary of method

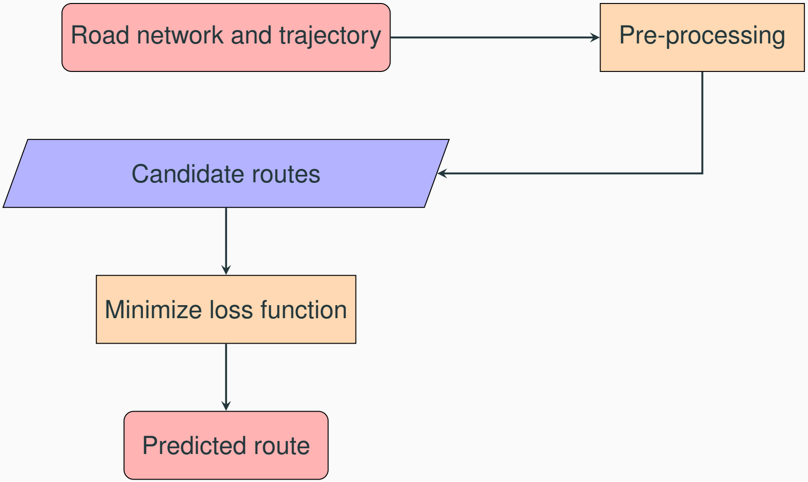
- Quantify the distance between $p \in \mathbf{p}$ and each $A \in \mathcal{CR}$ by making $\mu_{\mathbf{p}}$ and ν_A ε-pertubation according to speed and direction information.
- Conclude that **the route A with the smallest $W_1(\mu^{\varepsilon}_{\mathbf{p},A}, \nu^{\varepsilon}_A)/W_1(\mu_{\mathbf{p}}, \nu_A)$ is the true route**.

Summary:

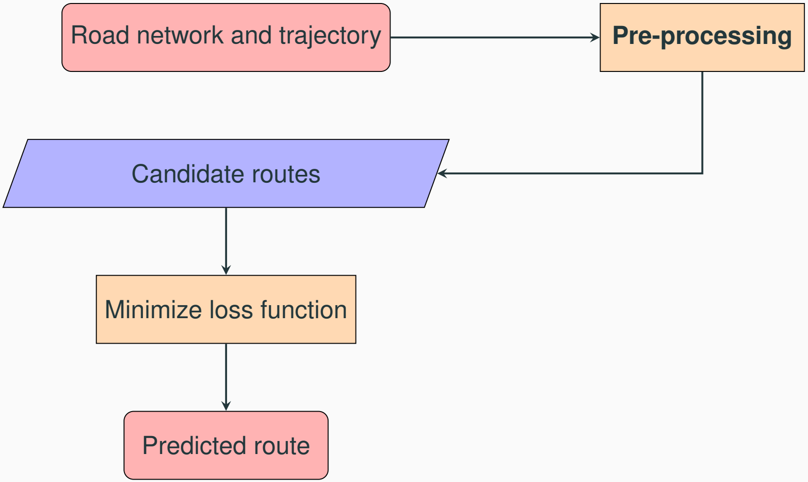
- Incorporating auxiliary information only requires modifying probability measure
- Places more weight on sensor data; more robust under high error
- Computationally expensive

Experimental Results

Map-matching pipeline



Map-matching pipeline



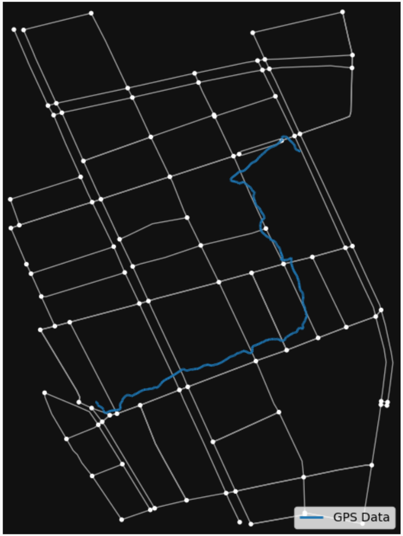
Pre-processing

One flaw of the proposed geometric methods is the reliance on a set of candidate routes. As the length of the trajectory increases, the number of possible routes increases factorially. Instead of obtaining all possible candidates, we restrict ourselves to candidates that are "reasonable". This is done via Dijkstra's algorithm.

This is still computationally expensive, but can be done a priori, and on large scales the limitation becomes data storage.

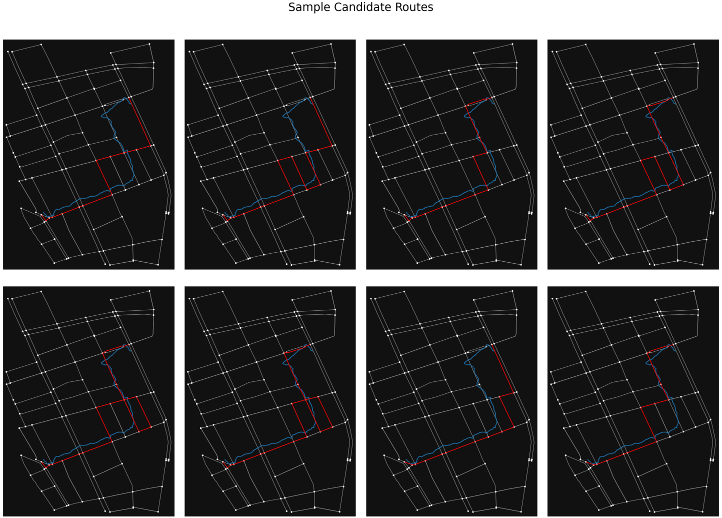
Now we revisit our original example with the new algorithms.

Example Trajectory in Road Network of Sendai



Pre-processing

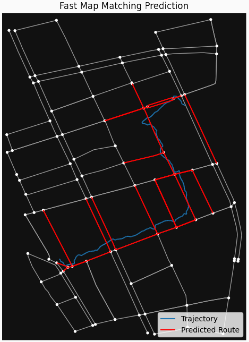
How long does it take for us to obtain our candidate route set?



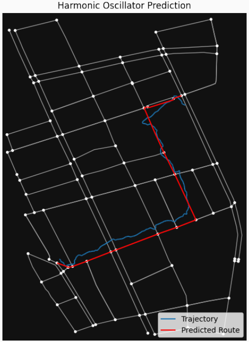
Obtaining Candidate Routes (Dijsktra's Algorithm)
CPU time: 45.9 s

Preprocessing Candidate Routes
CPU time: 12.8 s

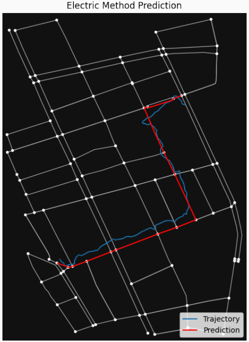
Method predictions



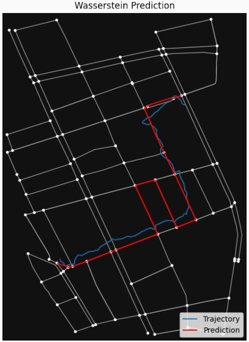
Fast Map Matching runtime
CPU time: 1.56 s



Harmonic oscillator runtime
CPU time: 1.56 s

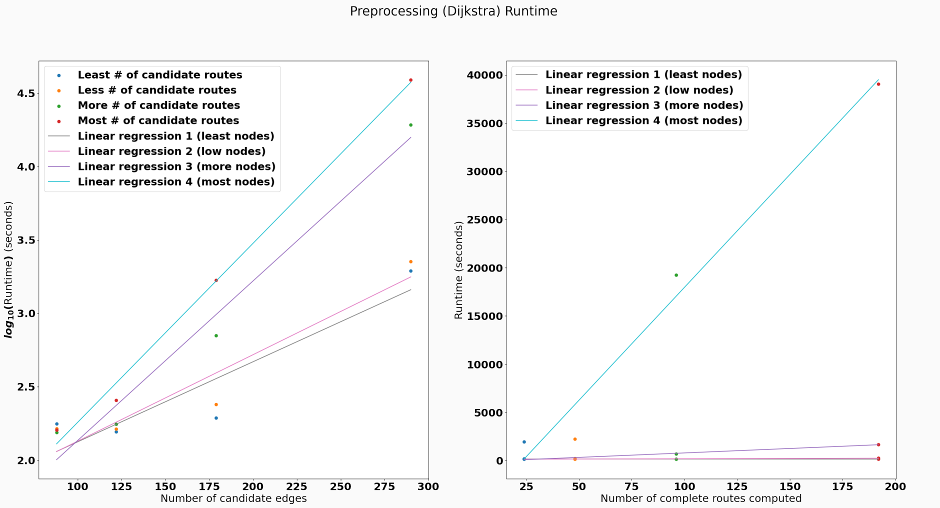


Electric method runtime
CPU time: 10.2 s



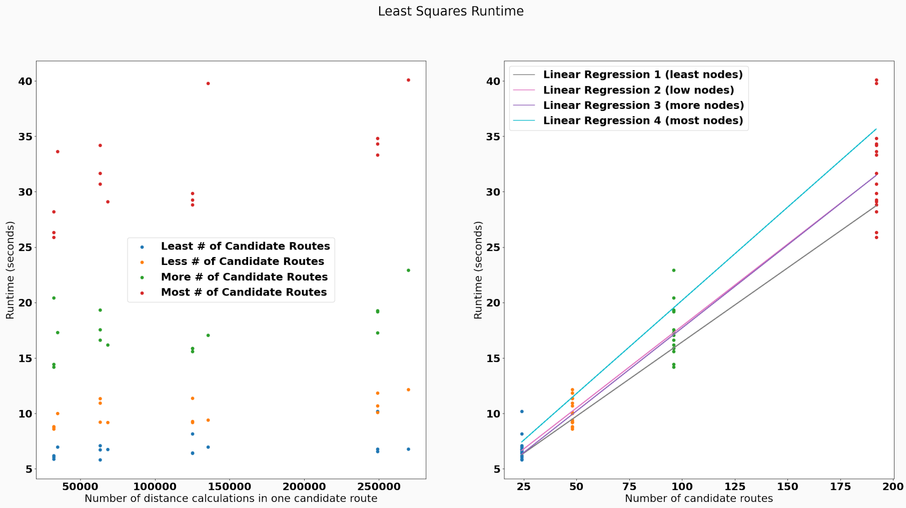
Wasserstein runtime
CPU time: 8.17 s

Computational Complexity (Preprocessing)

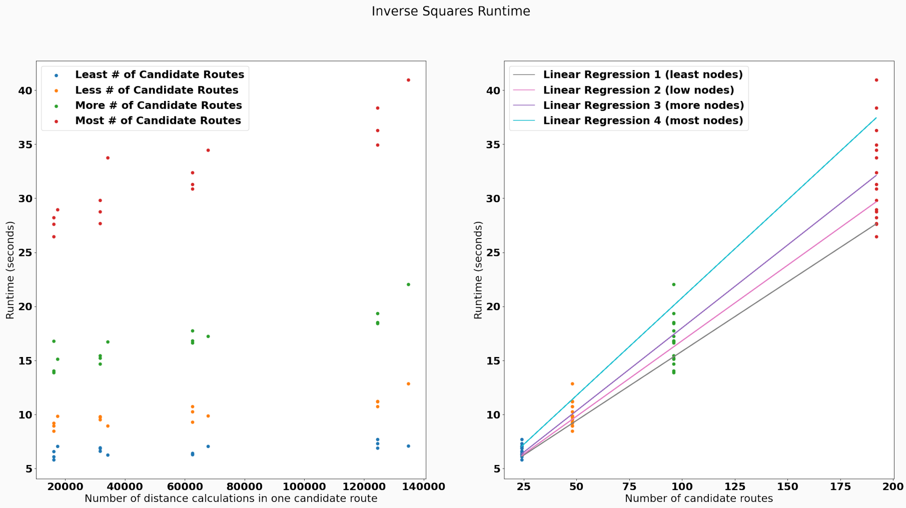


The following scatter plots demonstrates how computation grows as a function of input nodes.

Computational Complexity (Metric Calculations)



Computational Complexity (Metric Calculations)



Computational Complexity

This suggests that simple distance methods (like inverse squares and least squares) have a computational complexity of $\mathcal{O}(n)$. Due to processing power, we could not obtain enough data to infer the computational complexity of the Wasserstein method. Because it relies on linear programming, we hypothesize that the growth rate is a polynomial of degree > 1.

Some aspects of these algorithms can be improved upon– simple parallelization techniques offer a noticeable increase. However, other aspects are inherent to the method and are difficult to improve.

Deliverables

- Proposed three geometric methods with promising results
- Developed framework to prepare and clean large datasets for map-matching purposes
- Developed framework to enable the creation and implementation of algorithms with minimal programming experience
- Implemented flexible benchmarking methods compatible with common datasets

Future Work

- Improve candidate route-finding algorithm
- Adjust Wasserstein implementation
- Include IMU data in implementation
- Incomplete map-matching (determine when road exists, but isn't in the map data)

- High-assurance Mobility Control Lab. https://hmc.unist.ac.kr/research/autonomous-driving/
- 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).
- 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).
- 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).

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).

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