OBD-Data-Assisted Cost-based Map-matching Algorithm for Low-Sampled Telematics Data in Urban Environments

Patrick Alrassy, *Member, IEEE,* Jinwoo Jang, *Member, IEEE,* and Andrew W. Smyth, *Member, IEEE*

***Abstract*—A myriad of connected vehicles collects large-scale telematics data throughout cities, enabling data-based infrastructure planning. To truly benefit from this emerging technology, it is important to integrate pervasive telematics data with map data to produce more tractable and readable information for traffic flows and safety. Map-matching algorithms enable the projection of noisy trajectory data onto map data as a means of integrating telematics data. However, map-matching poses challenges due to higher levels of positioning errors and complex road networks. The authors propose a novel map-matching algorithm that can fuse in-vehicle data with trajectory data to improve the efficiency and accuracy of the algorithm. The proposed algorithm combines the probabilistic and weight-based map-matching frameworks. The novelty of the proposed algorithm includes (i) an adaptive segment candidate search mechanism based on in-vehicle speed information, (ii) adaptive matching parameters to reflect the variations in the Global Positioning System (GPS) noise levels, (iii) a novel transition probability that uses in-vehicle speed data, and (iv) a backend data query system for the shortest routes. Map-matching results were validated based on ground-truth data collected using an in-vehicle sensing device developed by the authors, as well as comparing with a commonly-used off-the-shelf map-matching platform. The proposed algorithm is proven to be robust, with an accuracy of 97.45%, particularly where map data are denser and GPS noise is high.**

***Index Terms*—Map-matching, Telematics, Trajectory data, Connected Vehicle, Smart Cities.**

I. Introduction

ONE of the novel technologies for emerging smart cities, particularly those cities dealing with traffic congestion and public safety, is connected-vehicle technology [1], [2]. Connected vehicle technology enables sharing information among vehicles, the infrastructure, and personal communication devices through safe and interoperable networked wireless communications. Detailed overviews of the concept, connectivity, and architecture of connected vehicles can be found in the literature [3]–[9].

Connected-vehicle technologies can be grouped into three categories based on inter-vehicle communications: Vehicle-to-Vehicle (V2V), Vehicle-to-Roadside Infrastructure (V2I), and Vehicle-to-Broadband Cloud (V2B). Examples of connected-vehicle applications include traffic management systems [10], [11], parking spot locator systems [12], lane marking localization systems [13], collision warning systems [14]–[17], road surface monitoring systems [18]–[20], and driver volatility estimation [21], [22]. The applications of V2V communications, in general, focus more on exchanging useful information between vehicles that are traveling along the same road. The V2I applications aim to provide the right information at the right time, such as road surface conditions. The V2B communications aim to create a large-scale monitoring data center, which opens a new door for many data-intensive applications [23]. In particular, large-scale connected vehicles will enable data-based infrastructure planning and management. For example, a large number of taxis, public transport, utility vehicles, and private vehicles are collecting big trajectory data across cities with valuable traffic information about real-time and network-wide traffic conditions. These data sets will significantly contribute to the improvement of transportation systems and mobility.

The main component of telematics vehicle data is the trajectory consisting of recorded data of the vehicle's position over time. Different types of localization sensors are currently being used to estimate the vehicle position with respect to time, such as the Global Navigation Satellite System (GNSS) [24], Wi-Fi [25], and cellular tower networks [26]. The GNSS trajectory data can estimate travel time and vehicle speed, supporting traffic operations monitoring, incident detection, and route guidance applications.

However, it is widely known that the quality of GNSS data is significantly affected by measurement noise. GNSS positioning error is primarily due to the multipath problem associated with buildings and infrastructure, which interferes with the direct path between

P. Alrassy and A. W. Smyth are with the Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, NY, 10027 USA, (e-mail: pa2492@columbia.edu; smyth@civil.columbia.edu).

J. Jang is with the Department of Civil, Environmental and Geomatics Engineering, Florida Atlantic University, FL, 33431, (e-mail: jangj@fau.edu).

GNSS receiver and the constellation of satellites in the sky [27]. Since the global positioning system operates on the trilateration concept, a GNSS receiver must communicate with a minimum of four visible satellite clocks before determining its true position [28]. It is noteworthy that GNSS positioning errors can propagate to the estimation of the speed and heading of vehicles when GNSS sensors are primary used to obtain speed and orientation data. Furthermore, the level of GNSS measurement noise, in general, significantly increases in urban environments, posing challenges for various V2B applications in metropolitan areas. Therefore, the implementation of novel algorithms that can deal with GNSS measurement noise is a vital component in leading to the success of V2B applications.

Moreover, telematics data include important vehicle-centric data, collected from extra sensor modules (e.g., On-Board Diagnostics (OBD-II) scanners, inertial measurement unit (IMU) sensors, Carbon Dioxide sensors, etc.). Vehicle-centric information can be collected from the Controller Area Network (CAN bus) [29]–[33] and augmented sensor hardware. A portion of CAN bus data has standardized protocols and can be accessed through OBD ports [34]. Examples of OBD data include engine RPM, vehicle speed, and fuel system status. By combining OBD data with data from extra sensor modules attached to in-vehicle sensor networks, data availability can be easily customized to meet the requirement of various applications. Augmented sensor modulus enable various large-scale connected vehicle applications, such as environmental monitoring systems [35]–[37], street-asset data collection systems [38]–[41], and public safety [42], [43].

It is important that telematics data must be translated into more tractable and readable formats. Telematics data include spatio-temporal data (e.g., trajectory data) and non-spatial data (e.g., OBD data). Importantly, trajectory data are represented in a coordinate system (typically, in a geographic coordinate system) and are used to associate non-spatial data with the correct streets of road networks. In addition, large-scale connected vehicle applications require to summarize valuable information based on map data. For example, it is more intuitive to know traffic information, vehicle speed, and street asset conditions by street. In other words, trajectory data do not directly indicate the locations of vehicle-centric data to the street without proper data integration schemes. Therefore, data integration is critically important for large-scale V2B applications.

Map-matching techniques are promising approaches to deal with both GNSS positioning errors and map-based data integration. The following section provides an overview of map-matching algorithms. Map-matching algorithms aim to match a set of observed noisy vehicle position data with the sequence of road segments, summarizing meaningful traffic flow and safety metrics based on map data. Map-matching applications in urban environments become more challenging due to denser road segments and relatively-higher level of GNSS error. The performance of map-matching algorithms can vary based on the accuracy of GPS positioning data, the quality of the map data, and the tuning of the parameters [44].

This paper presents a novel map-matching technique to perform the data fusion of in-vehicle sensor network data, map data, and trajectory data. In particular, the vehicle speed, which is directly measured from the vehicle itself, is used to improve the accuracy and efficiency of the map-matching. The authors challenge complex V2B applications that have low-sampled telematics data and denser road networks. The authors collected real-world telematics data sets to validate the performance of the algorithm using their own telematics device.

The map-matching problem addressed in this study is more challenging as it is tested in New York City (NYC) dense road network with the relatively high level of positioning error caused in its urban canyons. Thus, the addressed application represents a prototype for the future of map-matching in smart cities. As smart cities are often envisioned to have denser road networks, with streets populated with tall buildings, the proposed algorithm balances simplicity, accuracy and performance through fusing GPS with in-vehicle speed data, and vertically scaling the algorithm as explained in Section IV-H, unlike sophisticated algorithms that use advanced sensor data that often are not available, trading high accuracy with low performance and utilization. The performance of the proposed map-matching algorithm is compared to a commonly-used map-matching algorithm: BMW car IT Barefoot [45].

The remainder of this paper is structured as follows: Section II provides an overview of the current map-matching algorithms. Section III defines a map-matching problem. Section IV-A-F describes the proposed map-matching algorithm. Section IV-G discusses database management, graph partitioning, and query-optimization techniques that support the implementation of our map-matching algorithm. Section IV-H documents a scalable map-matching system architecture. Section V includes validation results. Finally, future research directions and conclusions are provided in Section VI.

II. OVERVIEW OF MAP-MATCHING ALGORITHMS

Map-matching algorithms follow either (i) a probabilistic approach, (ii) a weight-based, or (iii) a machine learning approach. The latter includes Kalman filter and artificial neural network techniques which require more

input data, learning, and computational effort, compared to the first two approaches [46].

Recent map-matching algorithms harness various sensing data obtained from multiple sensors to improve their map matching accuracy. However, those additional sensing data might not be available in many applications that deal with typical vehicles that are not autonomous. Furthermore, most of the recent algorithms rely on machine learning techniques, which require parameter learning processes. Toledo-Moreo et al. [47] creates a particle-filter-based algorithm that hybridizes measurements from a GNSS receiver, a gyroscope and an odometer to solve the map-matching problem at the lane-level. Similarly, Szottka et al. [48] presents a particle-filter-based algorithm that incorporates camera detections data of the lane markings along with commercial map data. Tao et al. [49] build a localization solver, based on Kalman filtering, that leverages GPS data, vehicle data and observations from a video camera along with lane markings embedded in digital navigation maps. Gu et al. [50] integrates multiple sensor measurements and a 3-dimensional (3D) map to build a robust localization system in urban canyons. The 3D map is used to perform a signal ray tracing process to rectify the vehicle positioning. Shunsuke et al. [51] developed a particle filtering vehicle localization system at the lane-level for autonomous driving that integrates GNSS data, Inertial Navigation system (INS) and camera observations. Kuhnt et al [46] and Rabe et al. [52] introduce an approach on self-vehicle localization using sensors to detect the object positions in the neighborhood of the vehicle. The object's position and direction of movement along with an odometer sensor are used to localize the vehicle on a digital map. Zheng et al. [53] proposes a machine learning segmentation and classification algorithm for lane-change detection using steering angle and vehicle speed data extracted from CAN-buses.

The probabilistic map-matching approach, in general, exploits the Hidden Markov Model (HMM) to find the most probable path because of its power in assessing different combinations of roads which the vehicle could have taken for the purpose of finding the most probable path [54]–[56]. The sequence of projected points are the hidden states in the Markov model. The raw GPS data points are the emitted observed elements. Different algorithms propose different transition probability distributions to determine the likelihood of traversing a certain candidate road segment given that the vehicle has already passed on a road segment. Most algorithms use the Gaussian distribution to describe the emission probability, the probability of emitting the noisy data point given that the true match is a certain candidate point. HMM-based map-matching leverages the Viterbi algorithm defined in [57] to find the matched sequence of roads. In contrast, the work of Knapen et. al. [58] adopts the probabilistic approach but from a different angle. In fact, they use the GNSS trace to minimize the unlikelihood of existing candidates using only the spatiotemporal information contained in the input data without any added additional assumption related to the shortest path which is the basis of most HMM transition probability equations.

Unlike the probabilistic approach, a weighted map-matching approach assigns a cost to different candidate paths and uses different selection methods to find the surviving path [59]. Lin et al. [60] present a Dijkstra-based selection map-matching algorithm to estimate the correct sequence of roads. They define a virtual directed graph based on a physical graph (e.g., map data), whose nodes include a set of candidate points (e.g., map-projected GPS data points) and edges denote the transition probability from one candidate point to another. For clarity, each edge of a virtual graph may include multiple edges of a physical graph. The edges of a virtual graph are assigned with a cost; and then the shortest path problem is solved to find the least-cost path. However, the cost function for the edges is only a function of the shortest path from a candidate point associated with previous GPS data points to one corresponding to the current GPS point, without any consideration of how proximate candidates points are to GPS data points.

Moreover, there was a notable effort to improve the run–time performance for HMM-based map-matching in the literature. Koller et al. [61] combine the probabilistic and weighted approaches to leverage HMM-based map-matching, but replace the Viterbi algorithm with Dijkstra. This is achievable by converting total probabilities at the last stage of the HMM to costs. The blended approach improves the run-time of HMM-based map-matching by decreasing the computational overhead of the unnecessary transition probability calculations [61].

Most map-matching algorithms that were developed in a programming language and released to the community rely in their backend on one of the commercial routers. These routers perform the route calculations using shortest path algorithms on a certain road network that is saved in a relational database (SQL) or graphical database (NoSQL) [62]. Some are limited to a specific type of map data structure such as Open Source Routing Machine (OSRM). The others, such as pgRouting (pgRouting Contributors), perform the shortest path calculations live and may have significant overhead when copying the network database into RAM at each route execution [63]. Google Snap to Road, a map-matching API, has a 300-meter limit between two GPS data points to maintain accuracy and avoid false snapping [64], which could be problematic at low sampling-rates. It also works only on Google maps.

Similarly, TrackMatching [65], a commercially available, cloud-based web map-matching software service, performs the map-matching based only on OpenStreet maps (OSM). However, city planners support their own map data for their road network studies.

III. Map-matching problem statement

A. Data Structure

This section summarizes the data structure and map-matching problem that is considered in this study. A raw GPS trajectory is the given sequence of N noisy data points $P = (p_i|i = 1, \ \ldots, \ N)$ ordered by timestamps. The time interval between two consecutive points does not exceed a certain threshold Δt , which is the sampling rate. The sampling rate of the NYC data considered in this study is 30 seconds. It is noteworthy that a time difference between two consecutive data points can be bigger than the sampling time interval because it may take more than 30 seconds to obtain the next available positioning data due to a weak communication with the satellites. Each data point p_i has the following parameters: (i) longitude, latitude and altitude values, (ii) timestamp, (iii) the number of satellites that are visible at the location of the sampled point, and (iv) OBD-II speed information. The speed data represent the maximum speed value between the previous data point and the current data point (typically over 30 second time intervals). Also, the data collected by the authors for the evaluation purpose, contain the average OBD-II traveled speed. The illustration of the GPS trajectory data structure is shown in Figure 1.

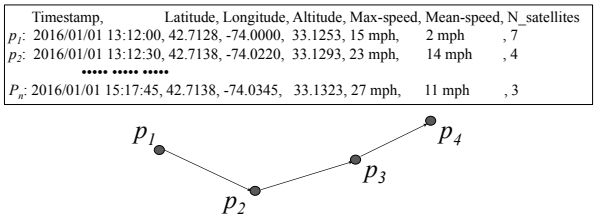


Fig. 1: Raw trajectory data formed of N noisy GPS points $P = (p_i|i = 1, \ \ldots, \ N)$ ordered by a timestamp field, along with the number of satellites and the OBD-II speed information.

A digital road network (map data) is a directed graph $G(V, E)$, where the road edges (road segments) E are connected by a set of nodes V. The map data used in this study have 102,489 nodes and 159,253 road segments, covering the NYC road network. Every edge has the following parameters: (i) the length of the road segment, (ii) the traffic direction (one-way or two-way), (ii) other topological constraints such as road level information to keep track of the edges and nodes that may be overlapping in the 2D map so that they will be separated prior to the map-matching problem, (iv) the node from, (v) the node to, (vi) the road segment index, (vii) the speed limits, (viii) a list of intermediate points that describes a road segment as a polyline and ix) the street names. New York's road network is characterized by a high road density, concentrated in Manhattan, the Bronx, and Queens [66]. The average road width is 8.88 meter; the density of arterial roads is 0.74 km/km^2; and the average block size is 0.067 km^2 [67]. The typical length of a north-south road segment in Manhattan runs approximately 80 meters and the typical distance between avenues is roughly 230 meters. The density of roads results in considering an average of 20-40 projected points on neighboring road segments per GPS point.

B. Problem Statement

A candidate point c_i^j is defined as the projected point of the GPS data point p_i onto a neighboring road segment j as depicted in Figure 2. Each data point p_i can have more than one corresponding candidate point c_i^j $(j = 1, \ldots, n_i)$. The number of the candidate points n_i can vary based on the density of neighboring road segments, the search radius, and the location of p_i. Among the candidate points, only one candidate point is selected and used to represent the projected data point of point p_i on the road network. Therefore, the outcome of the map-matching algorithm is a sequence of projected data points (selected candidate points), representing a reconstructed path that the driver could have taken in a chronological order. The intermediate path between every pair (c_{i-1}^k, c_i^j) is labeled as $P(c_{i-1}^k, c_i^j)$ when reconstructing the full path of the vehicle.

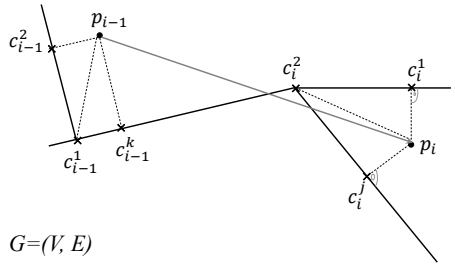


Fig. 2: Projection of the raw GPS points (p_{i-1}, p_i) onto the neighboring segments of the digital map, resulting in a set of candidate points: c_{i-1}^k $(k = 1, \ldots, n_{i-1})$ and c_i^j $(j = 1, \ldots, n_i)$

IV. Proposed OBD-data-fused map-matching algorithm

Most map-matching algorithms in the literature are designed to be used with high GPS sampling rate

applications which are essential for most GPS-based services (navigation and road guidance, and distance-based road pricing) [68]. However, many real-time applications require data collection at a low sampling rate. Such practice is often adopted in order to reduce power consumption as well as communication costs [69]. Most algorithms use road connectivity and heading restrictions [59]. These two pieces of information are misleading in low sampling rate data because within a 30-second interval multiple heading changes occur and the arc-skipping problem exists [70].

Therefore, the authors propose an offline map-matching algorithm that processes a set of trajectory data to compute the most probable sequence of roads. It produces more accurate map-matching results for trajectory data with a low sampling rate and can handle the scalability of map-matching systems with an efficient router system. Improved map-matching is achieved by the following unique features:

1) The proposed transition probability fuses trajectory data with OBD data to improve the map-matching accuracy, especially when high GPS noise, denser road networks, and low GPS sampling rates are present. The transition probability function harnesses an actual travel distance between two GPS points, calculated based on OBD speed data. The inclusion of an actual travel distance in the transition probability improves the estimation of a sequence of road segments based on given trajectory points. Furthermore, the transition probability is non-parametric; therefore, it does not require any pre-learning of parameters from the map and trajectory data.
2) An adaptive local search algorithm is designed to improve the performance of the candidate road segment selection process and trust-region filtering. This algorithm utilizes additional sensor information to indicate the level of GPS accuracy and adjust based on the local searching grid. This feature overcomes one of the notable challenges in map-matching mentioned by Hashemi and Karimi [71]: "narrowing the entire road network to a limited number of road segments."
3) The usage of connected sliding windows in the cost-based selection of the most probable path is developed to mitigate a challenging map-matching problem, where priority roads have parallel service roads.
4) This work proposes an efficient shortest path query system that can minimize repeated shortest path calculations for map-matching problems, and stores a robust subset of pre-calculated shortest paths determined based on GPS data.

A. Basic Flow of the Algorithm

The map-matching system described in this paper adopts the blended map-matching approach that is based on HMM-techniques, but leverages the Dijkstra algorithm described in the work of Dreyfus [72] for matching a GPS trajectory to a path for the above mentioned benefits. It consists of two sub-algorithms. Algorithm 1: *Candidate road segments selection and Candidate Graph*, responsible for choosing the set of candidate paths between two GPS data points and assignment of a cost for each path ; Algorithm 2: *Dijsktra least–cost Path Map-Matching*, takes Algorithm 1 as an input and runs the Dijkstra algorithm to determine the most probable path of the vehicle.

Candidate road segments selection: Having the digital road network with directionality information, topology and connectivity, a set of candidate road segments are selected for each GPS data point. The selection procedure takes into account a trust region that is built based on the OBD-II maximum speed information and filter out candidate points that fall outside the trust-region.

Candidate Graph: Each pair of candidate points that belong to two consecutive GPS data points constitute a path. We compute the emission and transition probability of the Hidden-Markov-Model and we assign a cost value for taking that path. We then construct a virtual graph where the candidate points of the GPS raw points are the nodes and the intermediate path for every pair of candidate points are the edges with the calculated cost.

Dijsktra Least–Cost Path Map-Matching: After building the candidate graph, we run a Dijkstra shortest path algorithm and save the sequence of candidate points of the most likely path.

The pseudocode of the proposed OBD-data-fused map-matching algorithm is formulated as follows. Algorithm 1 processes the trajectory points ($\mathbf{P} : p_i,\ i = 1,\ \ldots, N$) in parallel, as separate tasks for each central processing unit (CPU) available and returns the HMM calculations, which are the input for Algorithm 2. As soon as Algorithm 1 executes, Algorithm 2 outputs the matched path by creating a virtual directed graph $G_V(N_V, E_V)$ and running the shortest path Dijkstra algorithm. N_V includes a set of candidate points for each GPS data point; and E_V includes a set of edges that connect two neighboring candidate points.

B. Candidate Road Segment Selection

As our algorithm is based on the HMM approach [54], [55], the first step is to form a set of candidate road segments e_i^j ($j = 1,\ \ldots,\ n_i$) and the corresponding candidate projected data points c_i^j ($j = 1,\ \ldots,\ n_i$) around each data point p_i within a given radius r_i. An example is shown in Figure 3. n_i is the number of

Algorithm 1: Candidate road segment selection and candidate graph generation.

Input: 1) Raw GPS trajectory as sequence of N noisy data points, $\mathbf{P} = (p_i | i = 1, \ \ldots, \ N)$. 2) Digital directed road network (map data), $G(V, E)$, where the road segments E are connected by a set of nodes V.

Output: returns the Dijkstra cost C_i^{kj} for each pair of candidate points (c_{i-1}^k, p_i) at each consecutive pair of GPS points (p_{i-1,p_i})

foreach $p_i \in \mathbf{P}$ **do**
- Execute *Adaptive Trust-region Search*
- Obtain a set of candidate road segments e_i^j
- **for** $j \in e_i^j$ **do**
  - Obtain the candidate point c_i^j for each segment
  - Compute the emission probability $p(p_i|c_i^j)$
  - Add a virtual node c_i^j in $G_V, c_i^j \in N_V$

foreach *pair of* c_i^j *and* c_{i-1}^k **do**
- Compute the transition probability $p(c_i^j|c_{i-1}^k)$
- Create a virtual edge connecting $c_{i-1}^k \rightarrow c_i^j$
- Assign $p(c_i^j|c_{i-1}^k)$ to a virtual edge $c_{i-1}^k \rightarrow c_i^j$

Algorithm 2: Dijkstra least-cost map-matching.

Input: $G_V(N_V, E_V)$, $c_i^j \in N_V$; $p(c_i^j|c_{i-1}^k)$; $p(p_i|c_i^j)$

Output: return the most likely path

foreach *set of 20 GPS points* $P = (p_i | i = 1, \ \ldots, \ 20)$ **do**
- Run *Dijkstra shortest path algorithm*
- Save the resulting sequence of candidate points of the most likely path
- Set $C_i^{kj} = 0$ for the last five matched edges and set $i = i - 5$.

candidate road segments associated with the data point p_i. It is noteworthy that the number of candidate road segments n_i can vary based on the density of the digital map around the point p_i and the candidate search radius r_i. In order to find the n_i candidate road segments for each GPS data point, two things need to be defined: the distance measure between the data point p_i and a set of neighboring road segments; and an efficient candidate search radius r_i.

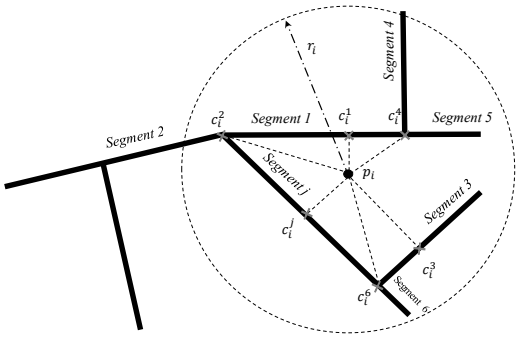


Fig. 3: Candidate projected points on neighboring road segments and Search Radius.

The GPS data points are projected onto road segments in such a way that the distance between the GPS points p_i and the candidate point c_i^j on the edge e_i^j s minimized. Therefore, each road segment e_i^j has one candidate data point c_i^j calculated as follows:

$$c_i^j = \arg\min \text{dist}(p_i, c_i^j) \qquad \forall c_i^j \in e_i^j. \tag{1}$$

Figure 3 shows an example of a candidate search. The search radius r_i should reflect the level of GPS position error changes. Typically, GPS positioning is accurate to about 15 meters; and the accuracy depends on many factors such as the number and position of the satellites and the design of the receiver [73]. In the work of Lou et. al. [55], the GPS positioning error is assumed as a Gaussian distribution, and its standard deviation is set to be 20 meters.

Unlike other methods that use a fixed radius search mechanism, the novelty of the proposed map-matching algorithm is that the selection of candidate road segments can adaptively adjust to the level of GPS positioning error. Our method uses two data sources that can be indicative of the level of the positioning error: the number of satellites visible at the data point p_i; and the altitude value. We expect that when a GPS sensor has a limited number of visibile satellites, the accuracy of the positioning data is low. Furthermore, the positioning error can propagate into positioning data in all directions, not limited to the latitude and longitude. Therefore, the unrealistic value of the altitude can be an indicator of poor accuracy of the positioning data.

Data analytics on raw trajectory points reveal that unrealistic altitude values mostly occur in dense regions with no clear open sky. Unrealistic altitude value from a GPS sensor can be detected when it significantly differs from the known elevation at a point on the map. In fact, since the road network is in New York City, which is a relatively flat area, one is able to easily judge whether the GPS receiver returns a false altitude value. We define a false altitude value for New York City area for every

Fig. 4: Density of data points with unreasonable altitude value.

value above 100 meter, a value close to the highest natural point in the five boroughs of New York City. Hence, if it fails in determining a reasonable vehicle altitude, it probably provides erroneous longitude and latitude estimates. Figure 4 shows the density distribution of the erroneous altitude value of the sampled points and thus, it supports the previous assumption. We can clearly tell that false altitude values are concentrated around Manhattan midtown and downtown towers, as well as around elevated subway and roadway structures, where GPS reflections are high.

In this study, the baseline of the candidate search radius is set to be 30 meters, which is double the typical GPS positioning error found in the literature. In other places, the search radius can reach 210 meters. This is because the map-matching problem addressed in this study can be more challenging due to the denser road network and the relatively high level of positioning error caused in New York City's urban canyons. The search radius for candidate points is calculated as:

$$r_i = k_i \times \sigma_i \tag{2}$$

where the scaling coefficient k_i controls the envelop of confidence interval; and σ_i represents the assumed standard deviation of the GPS positioning error. In this study, the values of the scaling coefficient k_i and the standard deviation are intuitively defined as follows:

$$\sigma_i = \begin{cases} 30 \text{ meters}, & \text{if } N_{\text{sat},i} \geq 6 \\ 70 \text{ meters}, & \text{otherwise} \end{cases} \tag{3}$$

$$k_i = \begin{cases} 2, & \text{if Altitude}(p_i) \leq 100 \text{ meters} \\ 3, & \text{otherwise,} \end{cases} \tag{4}$$

where $N_{\text{sat},i}$ is the number of visible satellites at the data point p_i. The rationale behind determining these values is that the positioning errors of around 60 meters are frequently observed within the trajectory data set; and the data point p_i with relatively-poor GPS communication (i.e., $N_{\text{sat}} \leq 5$) can have GPS positioning errors up to 140 meters. Then, we define the two ranges as a 95% confidence interval (i.e., $2\sigma_i = 60$ or 140 meters based on the number of visible satellites.) Furthermore, if the altitude of the data point p_i becomes unrealistic, the 99.7% interval (i.e., $3\sigma_i$) is used for a local candidate search.

C. Trust-Region Candidate Filtering Based on The OBD-II Speed Information

An adaptive trust-region search is applied to each data point p_i. This adaptive trust-region filters out some of the candidate segments selected in Part B, that could not be reached by a vehicle due to its low speed during congestion. It efficiently adjusts a search radius based on directly measured in-vehicle speed data. Between two consecutive GPS data points p_{i-1} and p_i, the maximum vehicle speed is obtained from an OBD-II connection. The vehicle speed data point $v_{\text{max,i}}$ is the maximum vehicle speed when a vehicle travels from the current point p_{i-1} to the next data point p_i. Since the maximum vehicle speed $v_{\text{max,i}}$ and time interval $\Delta t_i = t_i - t_{i-1}$ between two points are given, it is possible to define a more reliable search radius R_{max}, as shown in Figure 5, for the selection of neighboring road segment candidates. R_{max} would vary based on the actual vehicle speed from the OBD connection $v_{\text{max,i}}$, and is defined as follows:

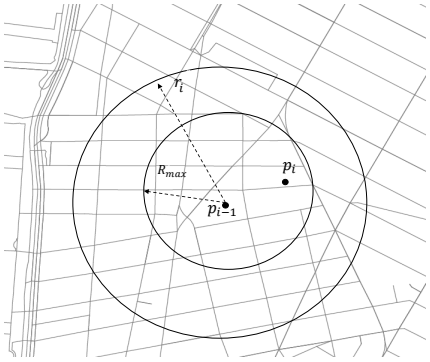
$$R_{\text{max}} = v_{\text{max,i}} \times \Delta t_i \times s \tag{5}$$

When a vehicle is stuck in traffic or moves slowly, the adaptive search radius is relatively small. In the example shown in Figure 5, r_i will be equal to 2×70 meters = 140 meters, however, R_{max} = 2 mph $\times$ 0.447 (m/s)/(mph) $\times$ 30 sec $\times$ 1.2 = 32 meters. Therefore, only candidate segments that fall within R_{max} will be considered.

When the vehicle speed increases, the adaptive search radius becomes large enough to cover possibly-visited road segments. The slack variable s provides an extra margin for a search radius to accommodate GPS positioning error. In this study, the value of the slack variable s is defined as 1.2, which means the adaptive search radius is increased by 20%.

D. Emission Probability

Each of the candidate projected points is considered to be a hidden state in the Markov model and has an emission probability $p(p_i|c_i^j)$, which is the likelihood of emitting the noisy data point p_i given that the true match is c_i^j. Instinctively, we favor candidates that are closer to p_i. It has generally been adopted in the literature that GPS errors have a zero-mean Gaussian behavior.



Timestamp	, •••	, Max-speed, Mean-speed
p_{i-1}: 2016/01/01 13:12:00	, •••	, 2 mph , 1 mph
p_i: 2016/01/01 13:12:30	, •••	, 1 mph, , 0.5 mph

Fig. 5: Case when the adaptive trust-region $R_{\max}$ less than the default search radius r_i.

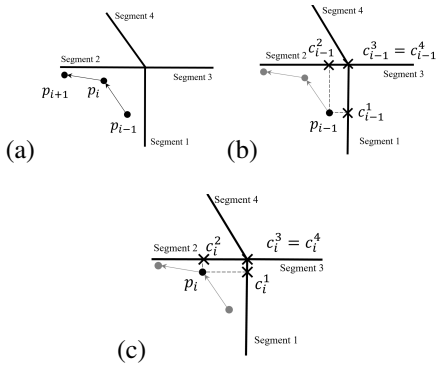


Fig. 6: Map-matching problem at road intersections.

Therefore, in our algorithm we define the emission probability as follows:

$$p(p_i|c_i^j) = p(d_j) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-d_j^2}{2\sigma_i^2}\right), \quad (6)$$

where $d_j = ||p_i c_i^j||$ is the distance between p_i and c_i^j, and σ_i is the assumed standard deviation of GPS positioning error. It is noteworthy that σ_i used in the emission probability (Eq. 6) is the same as the one used in the adaptive candidate search radius r_i (Eq. 2). The emission probability only uses the geometric information of a road network. Therefore, it fails to consider the GPS point's location context within an entire trajectory [55].

E. Transition probability

The transition probability $p(c_i^j|c_{i-1}^k)$ uses the topological information of the road network and evaluates the probability that a vehicle travels from the projected point c_{i-1}^k to c_i^j when it moves from data point p_{i-1} to p_i. The estimation of the transition probability can be assumed based on map data and GPS data points.

In the work of Koller et al. [61], which adopts the Dijkstra-based map-matching technique described in Section II, each transition path (c_{i-1}^k, c_i^j) is given a cost rather than a probability. (i.e., the higher the cost, the lower the transition probability is). The cost C_i^{kj} is estimated based on the route distance, calculated by the shortest path algorithms, and the distance between two GPS data points. The cost function is defined as follow:

$$C_i^{kj} = \beta * \frac{w(c_{i-1}^k, c_i^j)}{d(p_{i-1}, p_i)}, \quad (7)$$

where β is a pre-learning parameter for each road network, which in most applications may be impossible to determine, when ground truth trajectory data is unavailable. $w(c_{i-1}^k, c_i^j)$ is the route distance and $d(p_{i-1}, p_i)$ is the distance between the consecutive data points. This transition probability function may lead to incorrect matching at road intersections as demonstrated in Figure 6. Figure 6(a) shows a sequence of trajectory points for a vehicle approaching an intersection from South to West. Figures 6(b) and 6(c) depict respectively the candidate projected points of p_{i-1} and p_i. Eq. 7 tends to favor a pair of c_{i-1}^3 and c_i^3 with zero cost $C_i^{33} = 0$. The real matching pair (c_{i-1}^1, c_i^2) with $C_i^{12} > 0$ will never be chosen by Dijkstra when a different candidate pair with zero cost exists.

Similarly, Lou et al. [55] uses the transition probability, defined as follows:

$$p(c_i^j|c_{i-1}^k) = V(c_{i-1}^k, c_i^j) = \frac{d(p_{i-1}, p_i)}{w(c_{i-1}^k, c_i^j)}, \quad (8)$$

which might lead to the similar matching problem at an intersection since the shortest path $w(c_{i-1}^3, c_i^3)$ equals to zero and $V(c_{i-1}^3, c_i^3) = \infty$. It is noteworthy that the above-mentioned transition probability functions are reasonable enough to visualize the matched path of a vehicle, if the user is only interested in knowing the correct trajectory of a vehicle or minimizing GPS positioning errors. However, when it is important to link GPS data points with the correct road segments (e.g., probe-vehicle based traffic condition analysis and street-asset information collection), it is critical to find correctly-matched road segments, in addition to minimizing GPS positioning error and estimating matched trajectories.

Therefore, we propose a novel transition probability that can improve the accuracy of finding correct road segments, defined as follows:

$$p(c_i^j|c_{i-1}^k) = \frac{1}{d_d} \quad (9)$$

where d_d is the distance discrepancy estimation for each pair of from-to road segments. It represents the absolute difference between the shortest route from c_{i-1}^k to c_i^j and

the distance between p_{i-1} and p_i, which is calculated as follows:

$$d_d = \left| w(c_{i-1}^k, c_i^j) - d(p_{i-1}, p_i) \right| \tag{10}$$

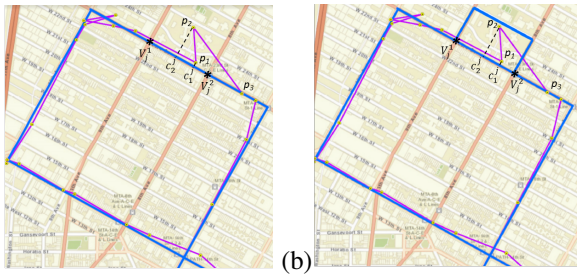
where $w(c_{i-1}^k, c_i^j)$ is the length of the shortest path between the projected points c_{i-1}^k and c_i^j ; and $d(p_{i-1}, p_i)$ is the distance $\|p_{i-1}p_i\|$ between data points p_{i-1} and p_i. It was proven in the work of Newson and Krumm [74] that these two distances are highly correlated. In this paper, we propose in Eq. 11 using the OBD-II vehicle average speed between p_{i-1} and p_i to estimate $d(p_{i-1}, p_i)$, in an attempt to minimize the distance discrepancy d_d and thus accurately evaluate each pair of from-to road segments.

$$d_d = \min \begin{cases} \left| w(c_{i-1}^k, c_i^j) - v_{\text{mean},i} \times \Delta t_i \right| \\ \left| w(c_{i-1}^k, c_i^j) - \|p_{i-1}p_i\| \right| \end{cases} \tag{11}$$

The OBD-II average speed better represents the actual traveled distance especially at intersections and low-speed areas where the GPS noise is high. The proposed transition probability becomes more robust for finding the correct sequence of road segments at an intersection. For example, in Figures 6(b) and 6(c), the real matching pair (c_{i-1}^1, c_i^2) is better weighted when comparing the shortest path $w(c_{i-1}^1, c_i^2)$ with $v_{\text{mean},i} \times \Delta t_i$ rather than $\|p_{i-1}p_i\|$. Also, the transition probability that a vehicle switches from segment 3 to segment 3, $p(c_i^3|c_{i-1}^3)$, becomes low since $w(c_{i-1}^3, c_i^3)$ is zero; and therefore the discrepancy d_d is high. Unlike the transition probability that requires parameter tuning [61], [74], the proposed transition probability does not require any parameter learning process. Due to the simplicity of the computation, the transition probability of Eq. 9 is not normalized to the total probability $\sum_{j=1}^{n_j} p(c_i^j|c_{i-1}^k)$, as the authors noticed that the omission of the normalization does not affect the algorithms' performance in Section V.

Due to the GPS measurement noise, a certain GPS data point p_i can fall behind a previous data point p_{i-1}. This loop-creation phenomenon, which is a common problem in map-matching, is frequently observed in metropolitan cities, in particular low-speed, congested regions with high GPS noise. Examples are shown in Figure 7. In Figure 7(b), the algorithm will disregard the candidates c_1^j and c_2^j by comparing the direction of $\overrightarrow{c_1^j c_2^j}$ to the traffic direction of segment j which is in this case in the direction $\overrightarrow{V_j^1 V_j^2}$, and thus having $\overrightarrow{c_1^j c_2^j} \cdot \overrightarrow{V_j^1 V_j^2} < 0$, $w(c_{i-1}^1, c_i^2)$ is set to ∞.

The suggested transition probability in Eq. 9 is robust to extreme noise in dense urban environments like Manhattan, since computing the inverse of the difference of two distances returns a large number. This makes the weights of each candidate pair (c_{i-1}^k, c_i^j) differ greatly from each other. Figure 8 shows an example of a vehicle driving on 5th Avenue in Manhattan. One can clearly see how, despite the fact that the candidate pair (c_{i-1}^2, c_i^2) is far in distance from $\|p_{i-1}p_i\|$, (c_{i-1}^2, c_i^2) was successfully chosen by our map-matching algorithm over (c_{i-1}^1, c_i^1) since $d_{d,2} = |w(c_{i-1}^2, c_i^2) - d(p_{i-1}, p_i)|$ is less than $d_{d,1} = |w(c_{i-1}^1, c_i^1) - d(p_{i-1}, p_i)|$ and therefore $p(c_i^2|c_{i-1}^2) > p(c_i^1|c_{i-1}^1)$.



(a) (b)

Fig. 7: (a) map-matching at closely spaced points (b) map-matching at closely spaced points with loop formation.

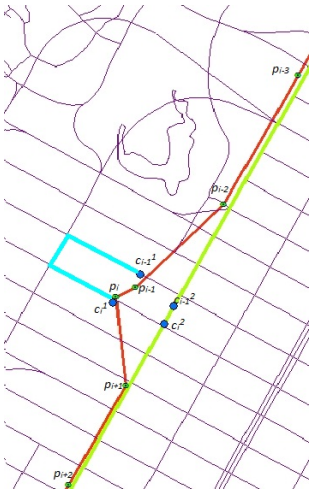


Fig. 8: Map-matching in an extremely noisy region: The green path is chosen by the map-matching algorithm over the blue path.

F. Most likely Path Selection

Following the Markov assumption, a model state c_i^j depends only on the previous state c_{i-1}^k. And the likelihood of emitting p_i depends only on the current state. Using Bayes' theorem and the Markov chain assumption, the probability $p(c_i|p_{0:i})$ of matching the data point p_i to candidate c_i^j, given the position history of the data points $p_{0:i}$ can be expressed as:

$$p(c_i^j|p_{0:i}) \propto p(p_i|c_i^j)p(c_i^j|p_{0:i})$$

$$= p(p_i|c_i^j) \sum_{k=1}^{N} p(c_i^j|c_{i-1}^k) p(c_{i-1}^k|p_{0:i}) \quad (12)$$

where $p(p_i|c_i^j)$ and $p(c_i^j|c_{i-1}^k)$ are respectively, the emission and transition probabilities and can be calculated from Eqs. 6 and 9, respectively; and the term $p(c_{i-1}^k|p_{0:i})$ represents a recursive element. To initialize the probability $p(c_{i-1}^k|p_{0:i})$, this algorithm needs to obtain the first two points $p_{0:1}$.

The most likely path link can be obtained based on the maximum a posteriori (MAP) estimation, meaning that we need to select a sequence of projected data points (matched road segments) that maximize the probability $p(c_i^j|p_{0:i})$. The most likely path is identified as follows:

$$c_i^j = \arg\max_{c_i^j} p(c_i^j|p_{0:i}), \quad (13)$$

To solve this maximization problem, one can use the Viterbi algorithm defined in Forney [57] to find the sequential combinations of road segments that maximize the probability $p(c_i^j|p_{0:i})$. However, as mentioned earlier, we are altering the problem to a Dijkstra least-cost path problem by building a virtual graph $G_V(c_i^j, C_i^{kj})$ shown in Figure 9, where the candidate points corresponding to the GPS raw points are the nodes of G_V and the intermediate path $P(c_{i-1}^k, c_i^j)$ between every pair (c_{i-1}^k, c_i^j) are the edges. Since our algorithm relies on the Dijkstra algorithm to solve the shortest path problem, the complexity of the map-matching algorithm is $O(ElogV)$, where E and V are the total number of edges and vertices in the virtual graph, respectively. We propose the cost function below to compute C_i^{kj} for every edge (c_{i-1}^k, c_i^j) in G_V:

$$C_i^{kj} = \frac{1}{T_i^{kj}}, \quad (14)$$

where,

$$T_i^{kj} = p(p_i|c_i^j) p(c_i^j|c_{i-1}^k), \quad (15)$$

for every candidate pair (c_{i-1}^k, c_i^j), with $p(p_i|c_i^j)$ and $p(c_i^j|c_{i-1}^k)$ computed from Eqs. 6 and 9, respectively.

The intuition behind our cost function in Eq. 14 is that computing the inverse of a low probability value produces a high cost; therefore, the corresponding path is not likely to be selected by the shortest path algorithm.

The network graph $G_V(c_i^j, C_i^{kj})$ is built on a 20-GPS point sliding window to avoid Dijkstra's computational overhead. However, we made the windows connected in order to have consistent matching at the first few nodes of each window. To explore this practice, Figure 10 shows three consecutive windows or in other words three Dijkstra's networks. Window 1 performs the map-matching of GPS point 1 to GPS point 20. Instead of starting Window 2 from point 21, we stepped back five

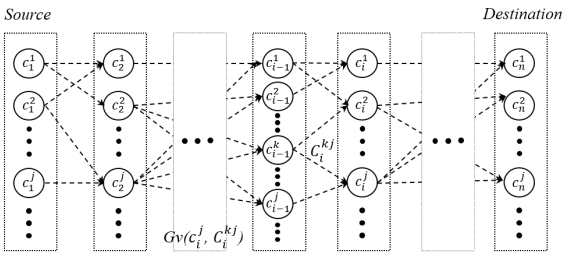


Fig. 9: $G_V(c_i^j, C_i^{kj})$ virtual graph of the Dijkstra least-cost path algorithm.

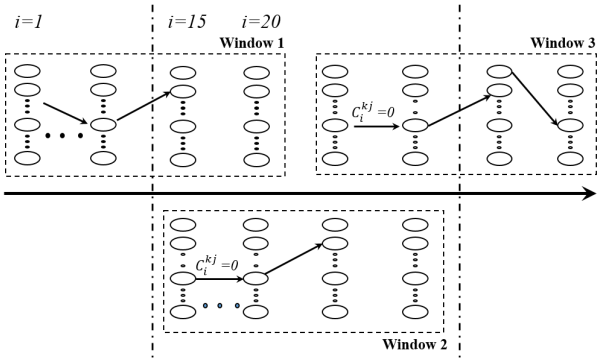


Fig. 10: Dijkstra's connected network windows.

solved GPS points and started Window 2 from point 15. We set the cost C_i^{kj} of the previously matched edges 15 to 20 from Window 1 to zero so that they will definitely be selected by the next map-matching Dijkstra's of Window 2. This connection also helps avoid incorrect mixed matching between main and parallel service roads. The second window will not switch the path of the vehicle to a service road even though the GPS data are closer to it given the edited cost of edges from the previous window.

In summary, for each pair of GPS points (p_{i-1}, p_i), the algorithm gets the neighboring candidate points from the digital map, and sends a request to a router database in Redis [75] in order to retrieve the corresponding shortest routes to compute the parameter $w(c_{i-1}^k, c_i^j)$ of Eq. 10. The procedure depicted in Figure 10 is repeated until the graph G_V is generated.

G. Building Router

In this section, the detailed design of the router is discussed. Building the router is a one-time exercise, executed before starting to feed trajectories to the map-matching algorithm. The router can be compiled on any digital map. This makes it possible for users to map their raw data and execute the patched path in their proper digital map index scheme. Since vehicles travel on the same predetermined road network, shortest path

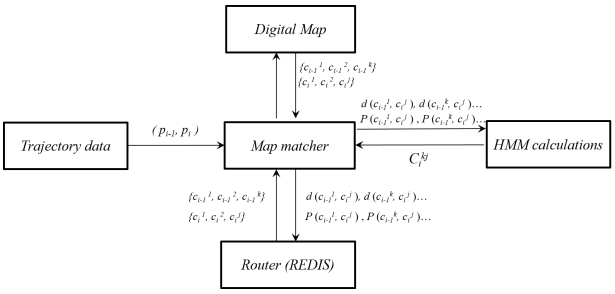


Fig. 11: Map-matching algorithm architecture.

calculations between two nodes can be precomputed and saved in a database. Dijkstra's shortest path algorithm was used as a routing tool to solve the network. In order to speed up lookup operations for the map-matching algorithm, by setting a limit on the search radius R_{router}, only routes between two nodes that are possible to visit within a 30-seconds interval (the sampling interval of the testing data) were solved and stored. For instance, there is no need to know the route of a node in lower Manhattan to a destination node in uptown Manhattan. To ensure a robust database, a limit on R_{router} was set as follows:

$$R_{\text{router}} = V_{\max} \times \Delta t \times \text{SF}, \tag{16}$$

where $V_{\max} = 70$ mph, $\Delta t = 30$ sec and SF is a safety factor = 1.2. The calculation above results in a maximum router radius of $R_{\text{router}} = 0.7$ miles. In other words, we solve the shortest path from node i to every node that is at most 0.7 miles away. A smart indexing technique is established in this paper to support faster lookup and retrieval from the Redis routing database. This technique relies on graph partitioning of the New York City road network. Each pair of noisy points are 30 seconds apart, which means that these points are a few meters away from each other. In consequence, the corresponding set of source and destination candidate points is located in the same geographical area. Therefore, it is possible to partition the graph $G = (V, E)$ into smaller components with specific properties. The borough information of every node in V that is encoded in the map GIS geodatabase, is used to partition the graph network into five smaller sub-networks as shown in Figure 12, and thus store the routes in five smaller databases (one database for each borough: Manhattan, Brooklyn, the Bronx, Staten Island, and Queens). As a result, Redis can perform the lookup operations in a smaller database every time it receives a routing request.

H. Vertical Scalability of a Map-matching System

The above map-matching algorithm and the custom-built router were combined in one map-matching system.

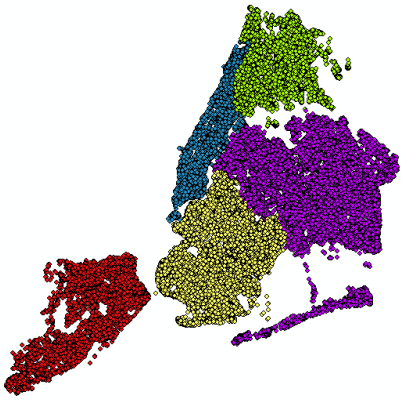


Fig. 12: The graph $G = (V, E)$ is partitioned into smaller sub-networks based on the borough information. Every node in $G = (V, E)$ is colored based on the borough it belongs to: Manhattan, Brooklyn, the Bronx, Queens and Staten Island.

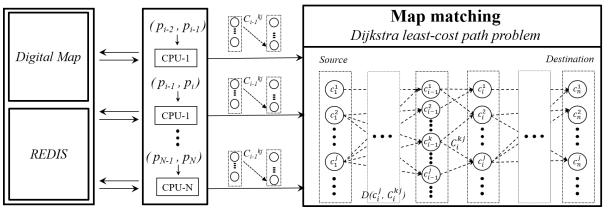


Fig. 13: Map-matching system.

Figure 13 shows the scalable map-matching system that is architected in Python. Scalability is the ability of a computer application or software to continue to be fast when it is changed in size or volume [76]. In this context, a change in size means when the road trajectory gets bigger.

In this paper, vertical scalability of the algorithm is presented, which is defined as the maximum use of the available resources on a computer as opposed to horizontal scalability which is speeding up the algorithm by forming a cluster of computing nodes [77]. The time-consuming part of the map-matching system is the generation of the virtual graph G_V. HMM parameters calculation for each pair (p_{i-1}, p_i) at the iteration i is independent of those corresponding to (p_{i-2}, p_{i-1}) at the iteration $i-1$. Also, when solving the least cost path problem using Dijkstra's algorithm, the least-cost calculations for each pair of candidate source and destination nodes are independent. These two observations made it possible for the algorithm to be vertically scalable. And thus parallel computing was applied as shown in Figure 13.

V. PERFORMANCE EVALUATION AND APPLICATION

Ground Truth: The algorithm is evaluated using ground truth trajectories that the authors collected in New York City. Each trajectory represents 15 to 30 minutes of driving. An in-vehicle sensing hardware package is developed and comprises a Raspberry PI 3 Model B+ microcomputer, a microSD 32GB SD card, an OBDCheck BLE OBD-II scanner, and a GPS module. This sensor configuration aims to collect timestamps, GPS positioning, and instantaneous speed data. A USB car charger was plugged into the cigarette lighter socket of the vehicle to provide power to the sensing hardware package. In order to evaluate the performance of the algorithm at low-sampling rate, the data collected was down-sampled to one sample every 30 seconds with a position, a maximum speed value for the past 30 seconds, and an average traveled speed value. When the maximum OBD-II speed value is less than 2 mph within the last 30 seconds, a minimum speed value of 2 mph was adopted in Eq. 5 to avoid very small trust-regions.

Evaluation Criteria: The map-matching algorithm is evaluated in terms of the matching quality. The matching quality is measured using an accuracy metric defined as follows:

$$A_{cc} = \frac{N_c}{N_t} \tag{17}$$

where N_c is the number of correctly matched GPS points and N_t is the number of total GPS points. Also, the algorithm is validated using a reliable off-the-shelf map-matching platform: BMW Car IT Barefoot library [45]. The authors would like to note that the comparison with Barefoot is made to validate the proposed algorithm with a commonly-used method to show the benefit of including in-vehicle speed data into the transition probability function, as well as computing the search trust-region using the maximum speed value. In general, a comparison between map-matching algorithms is challenging since map data are different and could be denser in one map than the other. Also, the algorithm's accuracy may vary between cities, depending on their road networks, as well as between data sets depending on the GPS sampling rate. In this study, we challenged an extreme scenario by collecting telematics data in New York City's urban canyons, with a GPS sampling rate of one sample per 30 seconds.

Barefoot Algorithm Background Information:

The Barefoot algorithm relies on Newson and Krumm [74]. The latter fits a negative exponential distribution to the transition probability V shown in Eq. 18 below:

$$V_{\text{barefoot}}(c_{i-1}^k, c_i^j) = \frac{1}{\beta} \exp \frac{-B(c_{i-1}^k, c_i^j)}{\beta} \tag{18}$$

where β is an experimental road-network parameter and $B(c_{i-1}^k, c_i^j)$ defined in Eq. 19 below, is the difference between the shortest route from c_{i-1}^k to c_i^j and the distance between p_{i-1} and p_i.

$$B(c_{i-1}^k, c_i^j) = |w(c_{i-1}^k, c_i^j) - \|p_{i-1}, p_i\| \,| \tag{19}$$

Barefoot uses Eq. 18 for their transition probability, but redefines $B(c_{i-1}^k, c_i^j)$ by using instead a time-priority route cost function detailed in Eq. 20 and defines β as the GPS sampling time interval (30 sec here).

$$B_{\text{redefined}}(c_{i-1}^k, c_i^j) = \frac{w(c_{i-1}^k, c_i^j)}{V_{\text{max}}} \times \text{PF} \tag{20}$$

where V_{max} is the road speed limit from OSM map data and PF is a road type factor that favors main roads over local roads. As mentioned earlier, when working with Barefoot, we tuned the σ value for the emission probability in the Barefoot library from the default value 5 meters to match the σ = 70 meters value used in our algorithm.

Experimental results: Table I shows the number of mismatched points out of 589 GPS data points for every algorithm.

TABLE I: Number of mismatched points for the proposed algorithm versus Barefoot algorithm.

	# of mismatched points	Accuracy(%)
Our Map-matching	15	97.45
Barefoot	25	95.76

Figure 14 shows a comparison example between our map-matching (Figure 14(b)), and Barefoot (Figure 14(c)). The novel transition probability implemented in our algorithm provides a very good balance between the GPS noise and the shortest path probabilities. Figure 14-b shows how our map-matching is robust to GPS measurement errors at (p_1, ..., p_6). The Barefoot time-priority strategy in computing the transition probability could lead to inaccurate matching at locations where raw GPS positions are close to each other due to traffic congestion as shown in Figure 14(c). In other words, Barefoot starts creating loops. However, the proposed transition probability includes the true mean speed of the vehicle assisting the algorithm in detecting that the vehicle cannot travel the entire loop within the 30-seconds interval with a low speed.

The authors acknowledge that cost functions that value road priority (Eq. 20), are essential in locations where major roads and service roads are in parallel with close

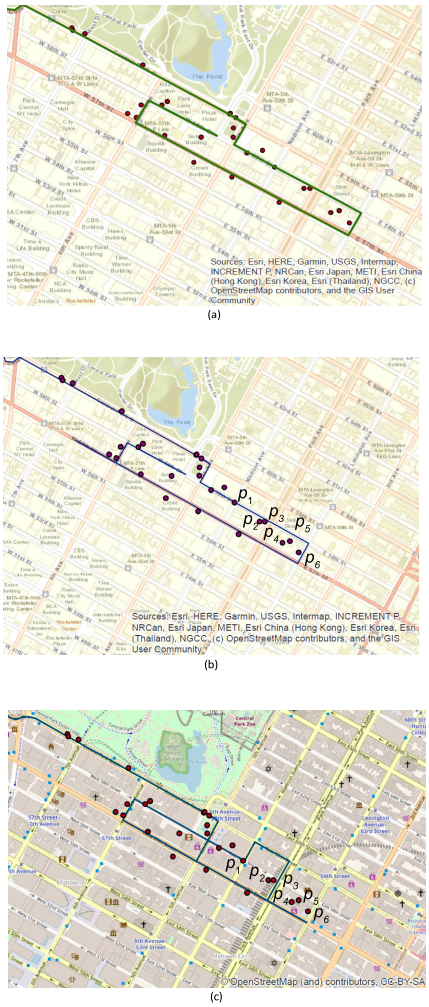


Fig. 14: Example 1: comparing our algorithm's output with Barefoot's output and ground truth data. (a) Ground truth trajectory with raw GPS points (b) our map-matching result (c) Barefoot's output.

Fig. 15: Example 2: comparing our algorithm's output with Barefoot's output and ground truth data. (a) Ground truth trajectory with raw GPS points (b) our map-matching result (c) Barefoot's output.

proximity. On the other hand, taking road priority into account could cause a mismatch at other locations where roads are proximate, but are not in a parallel configuration. In Figure 15(b), between p_7 and p_9, Barefoot went down to East 53rd St. instead of continuing on East 55th street since the former is a priority road. The proposed transition probability in Eq. 9 will assign a low probability for going down to East 53rd St., by comparing the shortest path with the average vehicle traveled distance. The road priority information is not encoded in the digital map used in our study, therefore we could not investigate the validity of including road priority information in our algorithm.

VI. CONCLUSION

As map-matching becomes critical for any traffic and driver behavior assessment relying on GPS-collected data, it is essential to have robust map-matching algorithms for urban canyons. In this paper, we described a novel OBD-data-assisted algorithm for map-matching. The algorithm is evaluated using ground truth data collected by the authors' developed telematics device. The proposed algorithm is proven to be more robust at a low sampling rate and high GPS noise with an accuracy of 97.45% when GPS noise is up to 70 meters. This scalable system executed in Python makes use of OBD-II parameters directly sampled from the vehicles, such as the maximum and mean speeds, in conjunction with GPS data to get more accurate map-matching. The map-matching algorithm uses in the backend a smart query system for the shortest routes. The map-matching algorithm can relate non-spatial safety-related driver behaviors to road networks. For example, OBD speed, hard-braking, and hard-acceleration events, do not have spatiotemporal information. However, it is practically valuable to understand these behaviors based on the road network, particularly for understanding city-scale public safety and design. Thus, the map-matching can help in creating driver behavior indexes (DBIs) for each road segment, such as vehicle speed profiles (e.g., 85th percentile speed, mean speed), and harsh driving metrics (e.g., harsh braking and acceleration rates). Road-segment-level DBIs can be further analyzed to understand their relations to crash data. The data summarization of connected vehicle data on map data can be an impactful V2B application that can potentially benefit city planners' street improvement projects and corridor safety metrics generation. As a future work, the traffic data collected by the current intelligent transportation systems (e.g., electronic toll collection systems, spot-speed radar) at selective locations, will be used to validate the accuracy of traffic flow obtained by this work from matching connected vehicle data with the road segments.

ACKNOWLEDGEMENT

We would like to thank the New York City Department of Transportation for their financial support and guidance for this study. This work was partially supported by the U.S. National Science Foundation (OAC-1948066).

REFERENCES

[1] Z. Xiong, H. Sheng, W. Rong, and D. E. Cooper, "Intelligent transportation systems for smart cities: a progress review," *Science China Information Sciences*, vol. 55, no. 12, pp. 2908–2914, 2012.

[2] S. I. Guler, M. Menendez, and L. Meier, "Using connected vehicle technology to improve the efficiency of intersections," *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 121–131, 2014.

[3] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 241–246.

[4] F. Yang, S. Wang, J. Li, Z. Liu, and Q. Sun, "An overview of internet of vehicles," *China communications*, vol. 11, no. 10, pp. 1–15, 2014.

[5] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE internet of things journal*, vol. 1, no. 4, pp. 289–299, 2014.

[6] K. M. Alam, M. Saini, and A. El Saddik, "Toward social internet of vehicles: Concept, architecture, and applications," *IEEE access*, vol. 3, pp. 343–357, 2015.

[7] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: a survey and future perspectives," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 98–104, 2016.

[8] O. Kaiwartya, A. H. Abdullah, Y. Cao, A. Altameem, M. Prasad, C.-T. Lin, and X. Liu, "Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[9] F. Cunha, L. Villas, A. Boukerche, G. Maia, A. Viana, R. A. Mini, and A. A. Loureiro, "Data communication in vanets: Protocols, applications and challenges," *Ad Hoc Networks*, vol. 44, pp. 90–103, 2016.

[10] J. Wan, J. Liu, Z. Shao, A. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, p. 88, 2016.

[11] T. T. Dandala, V. Krishnamurthy, and R. Alwan, "Internet of vehicles (iov) for traffic management," in *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*. IEEE, 2017, pp. 1–4.

[12] F. F. Kuhlman, D. H. Sarma, and A. P. Harback, "Vehicle parking spot locator system and method using connected vehicles," Mar. 8 2012, uS Patent App. 13/293,761.

[13] A. Mammeri, A. Boukerche, and Z. Tang, "A real-time lane marking localization, tracking and communication system," *Computer Communications*, vol. 73, pp. 132–143, 2016.

[14] J. H. Lemelson and R. D. Pedersen, "Gps vehicle collision avoidance warning and control system and method," Nov. 9 1999, uS Patent 5,983,161.

[15] N. Kaempchen, B. Schiele, and K. Dietmayer, "Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 678–687, 2009.

[16] X. Yang, L. Liu, N. H. Vaidya, and F. Zhao, "A vehicle-to-vehicle communication protocol for cooperative collision warning," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.* IEEE, 2004, pp. 114–123.

[17] H.-S. Tan and J. Huang, "Dgps-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 415–428, 2006.

[18] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 29–39.

[19] A. Ghose, P. Biswas, C. Bhaumik, M. Sharma, A. Pal, and A. Jha, "Road condition monitoring and alert application: Using in-vehicle smartphone as internet-connected sensor," in *2012 IEEE international conference on pervasive computing and communications workshops*. IEEE, 2012, pp. 489–491.

[20] E. P. Dennis, Q. Hong, R. Wallace, W. Tansil, and M. Smith, "Pavement condition monitoring with crowdsourced connected

vehicle data," *Transportation Research Record*, vol. 2460, no. 1, pp. 31–38, 2014.

[21] J. Liu and A. J. Khattak, "Delivering improved alerts, warnings, and control assistance using basic safety messages transmitted between connected vehicles," *Transportation research part C: emerging technologies*, vol. 68, pp. 83–100, 2016.

[22] B. Wali, A. J. Khattak, H. Bozdogan, and M. Kamrani, "How is driving volatility related to intersection safety? a bayesian heterogeneity-based analysis of instrumented vehicles data," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 504–524, 2018.

[23] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli, "Progress and challenges in intelligent vehicle area networks," *Communications of the ACM*, vol. 55, no. 2, pp. 90–100, 2012.

[24] M. Weber, L. Liu, K. Jones, M. J. Covington, L. Nachman, and P. Pesti, "On map matching of wireless positioning data: a selective look-ahead approach," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2010, pp. 290–299.

[25] K. Jones, L. Liu, and F. Alizadeh-Shabdiz, "Improving wireless positioning with look-ahead map-matching," in *2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*. IEEE, 2007, pp. 1–8.

[26] K. Perera, T. Bhattacharya, L. Kulik, and J. Bailey, "Trajectory inference for mobile devices using connected cell towers," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015, p. 23.

[27] A. El-Rabbany, *Introduction to GPS: the global positioning system*. Artech house, 2002.

[28] M. Topo, "How GPS Works," 2019. [Online]. Available: https://www.maptoaster.com/maptoaster-topo-nz/articles/how-gps-works/how-gps-works.html

[29] M. Farsi, K. Ratcliff, and M. Barbosa, "An overview of controller area network," *Computing & Control Engineering Journal*, vol. 10, no. 3, pp. 113–120, 1999.

[30] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of networked and embedded control systems*. Springer, 2005, pp. 741–765.

[31] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol Using the Controller Area Network Communication Protocol: Theory and Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer, 2012.

[32] D. Sik, T. Balogh, P. Ekler, and L. Lengyel, "Comparing OBD and CAN sampling on the go with the SensorHUB framework," *Procedia Engineering*, vol. 168, pp. 39–42, 2016.

[33] C.-M. Tseng, W. Zhou, M. Al Hashmi, C.-K. Chau, S. G. Song, and E. Wilhelm, "Data extraction from electric vehicles through OBD and application of carbon footprint evaluation," in *Proceedings of the Workshop on Electric Vehicle Systems, Data, and Applications*. ACM, 2016, p. 1.

[34] Wikipedia contributors, "OBD-II PIDs," Available: https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&oldid=848566847, Jul. 2018, [Accessed: 16- July- 2018].

[35] S.-C. Hu, Y.-C. Wang, C.-Y. Huang, and Y.-C. Tseng, "A vehicular wireless sensor network for CO 2 monitoring," in *Sensors, 2009 IEEE*. IEEE, 2009, pp. 1498–1501.

[36] S.-C. Hu, Y.-C. Wang, C.-Y. Huang, Y.-C. Tseng, L.-C. Kuo, and C.-Y. Chen, "Vehicular sensing system for CO2 monitoring applications," in *IEEE VTS Asia pacific wireless communications symposium (APWCS'09)*, 2009, pp. 168–171.

[37] W. Zeng, T. Miwa, and T. Morikawa, "Prediction of vehicle CO2 emission and its application to eco-routing navigation," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 194–214, 2016.

[38] G. Alessandroni, A. Carini, E. Lattanzi, V. Freschi, and A. Bogliolo, "A study on the influence of speed on road roughness sensing: the SmartRoadSense case," *Sensors*, vol. 17, no. 2, p. 305, 2017.

[39] J. Jang, Y. Yang, A. W. Smyth, D. Cavalcanti, and R. Kumar, "Framework of data acquisition and integration for the detection of pavement distress via multiple vehicles," *Journal of Computing in Civil Engineering*, vol. 31, no. 2, p. 04016052, 2016.

[40] J. Jang, A. W. Smyth, Y. Yang, and D. Cavalcanti, "Road surface condition monitoring via multiple sensor-equipped vehicles," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 43–44.

[41] S. T. Ng, F. J. Xu, Y. Yang, and M. Lu, "A master data management solution to unlock the value of big infrastructure data for smart, sustainable and resilient city planning," *Procedia Engineering*, vol. 196, pp. 939–947, 2017.

[42] A. J. Khattak and B. Wali, "Analysis of volatility in driving regimes extracted from basic safety messages transmitted between connected vehicles," *Transportation research part C: emerging technologies*, vol. 84, pp. 48–73, 2017.

[43] X. Zeng, K. N. Balke, and P. Songchitruksa, "Potential connected vehicle applications to enhance mobility, safety, and environmental security," Southwest Region University Transportation Center, Texas Transportation ..., Tech. Rep., 2012.

[44] F. Jiménez, S. Monzón, and J. E. Naranjo, "Definition of an Enhanced Map-Matching Algorithm for Urban Environments with Poor GNSS Signal Quality," *Sensors (Basel)*, vol. 16, no. 2, Feb. 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4801570/

[45] B. C. I. GmbH, "bmwcarit/barefoot: Java library," 2015. [Online]. Available: https://github.com/bmwcarit/barefoot

[46] N. R. Velaga, M. A. Quddus, and A. L. Bristow, "Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 6, pp. 672–683, Dec. 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X09000667

[47] R. Toledo-Moreo, D. Bétaille, and F. Peyret, "Lane-level integrity provision for navigation and map matching with gnss, dead reckoning, and enhanced maps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 100–112, 2009.

[48] I. Szottka, "Particle filtering for lane-level map-matching at road bifurcations," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 154–159.

[49] Z. Tao, P. Bonnifait, V. Fremont, and J. Ibanez-Guzman, "Lane marking aided vehicle localization," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 1509–1515.

[50] Y. Gu, Y. Wada, L. Hsu, and S. Kamijo, "Vehicle self-localization in urban canyon using 3d map based gps positioning and vehicle sensors," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2014, pp. 792–798.

[51] K. Shunsuke, G. Yanlei, and L.-T. Hsu, "Gnss/ins/on-board camera integration for vehicle self-localization in urban canyon," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2533–2538.

[52] J. Rabe, M. Hübner, M. Necker, and C. Stiller, "Ego-lane estimation for downtown lane-level navigation," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1152–1157.

[53] Y. Zheng and J. H. Hansen, "Lane-change detection from steering signal using spectral segmentation and learning-based classification," *IEEE Transactions on intelligent vehicles*, vol. 2, no. 1, pp. 14–24, 2017.

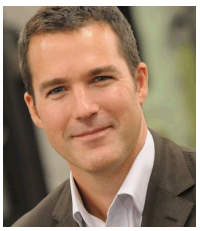
[54] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 776–781.

[55] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *Proceedings of the 17th ACM SIGSPATIAL international conference*

on advances in geographic information systems. ACM, 2009, pp. 352–361.
[56] A. Luo, S. Chen, and B. Xv, "Enhanced map-matching algorithm with a hidden markov model for mobile phone positioning," *ISPRS International Journal of Geo-Information*, vol. 6, no. 11, p. 327, 2017.
[57] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
[58] L. Knapen, T. Bellemans, D. Janssens, and G. Wets, "Likelihood-based offline map matching of GPS recordings using global trace information," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 13–35, 2018.
[59] M. Quddus and S. Washington, "Shortest path and vehicle trajectory aided map-matching for low frequency GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 328–339, Jun. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X15000728
[60] M. C.-H. Lin, F.-M. Huang, P.-C. Liu, Y.-H. Huang, and Y.-s. Chung, "Dijkstra-Based Selection for Parallel Multi-lanes Map-Matching and an Actual Path Tagging System," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2016, pp. 499–508.
[61] H. Koller, P. Widhalm, M. Dragaschnig, and A. Graser, "Fast hidden Markov model map-matching for sparse and noisy trajectories," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, 2015, pp. 2557–2561.
[62] M. Miler, D. Medak, and D. Odobašić, "The shortest path algorithm performance comparison in graph and relational database on a transportation network," *Promet-Traffic&Transportation*, vol. 26, no. 1, pp. 75–82, 2014.
[63] S. Mattheis, K. K. Al-Zahid, B. Engelmann, A. Hildisch, S. Holder, O. Lazarevych, D. Mohr, F. Sedlmeier, and R. Zinck, "Putting the car on the map: a scalable map matching system for the open source community," *Informatik 2014*, 2014.
[64] G. developers, "Snap to Roads | Roads API." [Online]. Available: https://developers.google.com/maps/documentation/roads/snap
[65] F. Marchal, "TrackMatching," 2015. [Online]. Available: https://mapmatching.3scale.net/
[66] G. Zhao, X. Zheng, Z. Yuan, and L. Zhang, "Spatial and temporal characteristics of road networks and urban expansion," *Land*, vol. 6, no. 2, p. 30, 2017.
[67] A. M. B. J. P. P. L.-H. Angel, Shlomo and N. G. Sán-chez, *Atlas of urban expansion*. New York University Urban Expansion Program; United Nations Programme for Human Settlements; Lincoln Institute of Land Policy, 2016.
[68] 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*, vol. 15, no. 5, pp. 312–328, 2007.
[69] J. Yang and L. Meng, "Feature engineering for map matching of low-sampling-rate gps trajectories in road network," *arXiv preprint arXiv:1409.0797*, 2014.
[70] J. S. Greenfeld, "Matching GPS observations to locations on a digital map," in *81th annual meeting of the transportation research board*, vol. 1, 2002, pp. 164–173.
[71] M. Hashemi and H. A. Karimi, "A critical review of real-time map-matching algorithms: Current issues and future directions," *Computers, Environment and Urban Systems*, vol. 48, pp. 153–165, 2014.
[72] S. E. Dreyfus, "An appraisal of some shortest-path algorithms," *Operations research*, vol. 17, no. 3, pp. 395–412, 1969.
[73] a. T. National Coordination Office for Space-Based Positioning, Navigation, "GPS.gov: GPS Accuracy," 2017. [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/
[74] 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*. ACM, 2009, pp. 336–343.
[75] S. Sanfilippo, "Redis," 2009. [Online]. Available: https://redis.io/
[76] A. Sivasubramaniam, A. Singla, U. Ramachandran, and H. Venkateswaran, "An approach to scalability study of shared memory parallel systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 22, no. 1, pp. 171–180, 1994.
[77] J. Pokorny, "NoSQL databases: a step to database scalability in web environment," *International Journal of Web Information Systems*, vol. 9, no. 1, pp. 69–82, 2013.

Patrick Alrassy, *Member*, IEEE received the B.E. degree in Civil Engineering from the American University of Beirut in 2015 and the M.Sc.degree in Civil Engineering from the University of Texas at Austin in 2016, and the Ph.D. degree in Civil and Engineering mechanics from Columbia University in the City of New York in 2020. He is now a Data Engineer at Optimus Ride Inc., a company building software for the self-driving vehicle technology.

Jinwoo Jang, *Member*, IEEE received his bachelor degree in Civil and Environmental Engineering from Kookmin University in South Korea and the M.Sc.degree and the Ph.D. degrees from Civil Engineering and Engineering Mechanics from Columbia University in the City of New York. He is now an Assistant Professor in the Department of Civil, Environmental and Geomatics Engineering (CEGE) at Florida Atlantic University. He is also jointly appointed with the Institute for Sensing and Embedded Network Systems Engineering (I-SENSE) as a Faculty Fellow.



Andrew W. Smyth received his Sc.B. and A.B. degrees at Brown University in 1992 in Civil Engineering and Architectural Studies respectively. He received his M.S. in Civil Engineering at Rice in 1994, an M.S. in Electrical Engineering (1997) and his Ph.D. in Civil Engineering (1998) at the University of Southern California. He is now the Robert A. W. and Christine S. Carleton Professor of Civil Engineering and Engineering Mechanics at Columbia University in the City of New York.