Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 82 (2025) 303–322

Transportation Research Procedia

www.elsevier.com/locate/procedia

World Conference on Transport Research - WCTR 2023 Montreal 17-21 July 2023

Evaluating Map Matching Algorithms for Smartphone GNSS Data: Matching Vehicle Trajectories to an Urban Road Network

Joshua Stipancic[a], Nicolas Saunier[b,*], Neda Navidi[b], Etienne B. Racine[c], Luis Miranda-Moreno[d], and Aurélie Labbe[a]

[a]*HEC Montréal, Montréal, QC H3T 2A7 Canada*
[b]*Polytechnique Montréal, Montréal, QC H3C 3A7 Canada*
[3]*Intact Insurance, Montréal, QC H3T 2A7 Canada*
[d]*McGill University, Montréal, QC H3A 0C3 Canada*

Abstract

Data from vehicles tracked using Global Navigation Satellite Systems (GNSS) can be used to monitor driving behaviour and road safety. In usage-based insurance programs, driver insurance premiums are tailored according to individual driving behaviour, often using data collected from user-owned smartphones. Due to positional noise caused, map matching algorithms must be used to spatially link GNSS observations to the road network. The purpose of this study is to evaluate the performance of several algorithms to process smartphone GNSS data for vehicular trips in urban road networks. This study evaluated five implementations, namely one topological (TMM), two probabilistic (PMM) using Hidden Markov Models (HMM), one fuzzy (FMM), and one hybrid map matching algorithm (HyMM) in terms of match accuracy and run time. Data was collected using ten smartphone devices and three applications across 12 trip scenarios in Montreal, Canada targeting the downtown, old city, highways, bridges, and tunnels. Results were compared with a series of ANOVA tests. Accuracy was not significantly different for the best performing algorithms (the Fast HMM and TMM) followed by the HyMM, with the Standard HMM and FMM algorithms performing significantly worse. Only the FMM algorithm was significantly slower than the others in terms of run time.

Keywords: ANOVA, GNSS data, map matching, smartphones, usage-based insurance

* Corresponding author. Tel.: +1-415-340-4711, ext. 4962.
E-mail address: nicolas.saunier@polymtl.ca

10.1016/j.trpro.2024.12.045

1. Introduction

The continuous monitoring and tracking of individuals are of great interest to many in the public and private sectors. The development of Global Navigation Satellite Systems (GNSS), as well as other vehicle identification and tracking technologies like license plate recognition, Wi-Fi and Bluetooth sensors, has led to a dramatic increase in the number of location-based services (LBS) used to track the location of individuals or vehicles. In North America, these LBS frequently leverage smartphones connected to satellite constellations like the Global Positioning System (GPS) and the GLObal NAvigation Satellite System (GLONASS), providing users with information, entertainment, or security in fields ranging from navigation to marketing. One example is the continuous tracking of vehicles on the road network and monitoring of driving behaviour, which has revealed important issues related to urban mobility and safety. More specifically, accurate and quantitative measures of system performance necessary for improving transportation systems (such as urban road networks) can be developed using LBS data and traffic monitoring. These performance measures are important to individuals, whose travel choices are influenced by network performance, to industry, who rely on navigation solutions for fleet or service management, and to road network operators, who require accurate information to properly monitor and manage the road network. Of particular interest to this work is the application of LBS to car insurance. Conventional insurance premiums are calculated based on rating variables that describe the client and their vehicle (Tselentis, et al., 2016). Advances in insurance telematics, technologies for "sending, receiving, and storing information from and to road vehicles" (Handel, et al., 2014), have led to the development of usage-based insurance (UBI) programs. In UBI programs, insurance premiums are tailored according to individual driving behaviour, determined using GNSS data collected using either dedicated devices with sensors connected to the on-board diagnostic (OBD) system or client-owned smartphone devices (Tselentis, et al., 2016).

(a) (b)

Fig. **1**. Typical GNSS data quality for an urban canyon (a) and bridge (b) in Montreal, Canada

GNSS data accuracy is critical to provide useful performance measures (related either to the road network or individual drivers). Yet the optimal operating conditions for GNSS receivers are rarely, if ever, studied in the field. Tracking in freeway environments has generally been successful (Quiroga & Bullock, 1998), though tall buildings and other aspects of urban environments can completely block the GNSS or create spurious signals (the urban canyon effect) (D'Este, et al., 1999). In some cases, GNSS performance can be enhanced using cellphone towers or geostationary satellites, though radio frequency-based systems remain sensitive to interference, signal blocking, jamming, and multipath errors. In general, all raw GNSS traces contain some positional variability, as demonstrated in Fig. 1, though these issues are most concerning in indoor environments, urban centers, and under dense foliage. Obviously, obtaining reliable results in difficult environments requires a method to reduce or eliminate positional noise. Although several filtering methods exist, many only smooth the data in terms of longitude and latitude. For many analyses, it is also desired to explicitly match each vehicle trip to the travelled route, spatially linking the GNSS observations to the road network (or to other networks for other travel modes). Map matching is the process of estimating a sensor position on the road network using its GNSS trip data. Though map matching is often challenging

due to variations in the density of data and noise, it remains an important step in maintaining data reliability.

For very large GNSS datasets (such as those collected in UBI programs), the selected map matching algorithm should be sufficiently accurate and computationally efficient. Considering the variation in existing map matching schemes with respect to both accuracy and efficiency, the objective of this study is to evaluate the performance of several algorithms to process smartphone GNSS data for vehicular trips in urban environments. Algorithms are compared based on accuracy (by comparing to the ground truth route data) and computational time. Reproducing algorithm performance is important, as many existing algorithms have been evaluated on few real-world trip scenarios, with some evaluated on the same scenarios used for their development. A robust validation of the algorithms on a separate dataset would be highly beneficial. Performance is compared across several types of smartphones, several data collection applications, and several environments in Montreal, Canada. Specifically, algorithms from four different map matching methodologies are considered including topological (Greenfeld, 2002; Li, et al., 2005), probabilistic (Quddus & Washington, 2015; Nayak & Narvekar, 2017; Koller, et al., 2015; Newson & Krumm, 2009; Bierlaire, et al., 2013), fuzzy (Quddus, et al., 2006), and hybrid approaches (Luxen & Vetter, 2011). Algorithms were selected because they are readily available, easily accessible, and open-source or free to use. Evaluating the reproducibility of these algorithms using consistent trip datasets provides the maximum impact to researchers or practitioners seeking to implement a map matching scheme. Based on the findings of this study, a suitable algorithm can be chosen for a given application and immediately implemented, thanks to an accompanying repository. The paper is organized as follows. First, a review of existing literature and description of the tested algorithms are presented. Next, the methodology and data collection process are presented. Finally, results, conclusions, and future work are presented.

2. Literature Review

Generally, map matching algorithms can be classified as either real time (online) or post-processing (offline). While real time algorithms have access only to previous GNSS observations when matching a given point, offline algorithms typically have information about the entire trip. For this reason, offline algorithms tend to be more accurate than their real-time counterparts, through this accuracy comes at the expense of computational resources. The rapid development of map matching algorithms in the last 10 years is well documented in the existing literature and motivates this study; a formal and up-to-date comparison of available algorithms is an important contribution to the field. Two previous surveys were conducted by Quddus *et al*. (Quddus, et al., 2007) and Hashemi *et al.* (Hashemi & Karimi, 2014), though the former is largely obsolete as many of the reported issues have since been addressed (Hu, et al., 2017; Hunter, et al., 2014). The latter, which considers 12 map-matching algorithms published between 2000 and 2013, presents the strengths and limitations of the algorithms but fails to independently verify their accuracy on the same data, instead reporting the accuracy from the original papers. This represents a major issue, as accuracy is known to depend on environment, with urban centers being the most challenging. Additionally, computational efficiency is not addressed. Though Hashemi *et al.* (Hashemi & Karimi, 2014) provided a robust review of real-time algorithms, and Singh *et al.* (Singh, et al., 2009) provide a more recent comparison of three algorithms for navigation purposes, a review of offline algorithms is needed.

2.1. Geometrical Map Matching Algorithms

As the earliest map matching procedure developed in the 1990s, geometrical map matching (GMM) algorithms remain popular (Krakiwsky, et al., 1988; White, et al., 2000; Quddus, et al., 2007; Velaga, et al., 2012). The three main GMM algorithms rely primarily on geometric information from the network to determine the set of travelled links. Point-to-point map matching (PPMM) finds nodes (i.e., those defining intersections or geometry) that are close to the GNSS positions and the set of links connecting the selected nodes. The map matched links are those on the closest path from all links identified in the previous step. While PPMM is simple and easy to implement, the map-matched results are unreliable, especially at dense intersections (Peker, et al., n.d.; Chawathe, 2007) and lead to “jumping” of consecutive map-matched positions across links. In point-to-curve map matching (PCMM), GNSS points are simply matched to the nearest road segment or link (Bernstein & Kornhauser, 1998; Quddus, et al., 2007; Pereira, et al., 2009). PCMM is more reliable than PPMM when determining true paths, though still suffers from inaccuracies in dense networks and jumping of consecutive points. Curve-to-curve map matching (CCMM) provides a slight improvement by

considering a series of points (a curve), matching the group of points to the link to which they are closest to on average (White, et al., 2000; Bernstein & Kornhauser, 1998). Although this eliminates some link jumping, accuracy still suffers in dense networks. As GMM algorithms have been largely replaced by more accurate and efficient algorithms (as described below), none are considered in this study.

2.2. Topological Map Matching Algorithms

The topology of a network is defined by its geometric features including points, lines, and polylines and the relationships between them. Topological map matching (TMM) algorithms compare the topology of the road network with characteristics of the observed trips to determine travelled routes (Velaga, et al., 2009; Quddus, et al., 2007; Yang, et al., 2013; Velaga, et al., 2012; Wang, et al., 2012; Huang, et al., 2014). TMM algorithms supplement simple geometric information used in GMM with additional information such as historical map matching data and features of the considered vehicle such as velocity and turn radius (Greenfeld, 2002; Velaga, et al., 2012). Several services for TMM are currently available. TrackMatching is a commercially available, cloud-based web map-matching software service (Marchal, 2015) that matches GNSS trip data to the OpenStreetMap (OSM) road network (OpenStreetMap, 2015). Matched links and map-matched positions (new coordinates falling directly on the road network) are identified using a combination of the Euclidean distance from the raw GNSS points to the nearest link and on network topology (Marchal, et al., 2005). The algorithm identifies the set of links closest to the initial point, with each link becoming the start of a unique path. Additional GNSS points are added to the initial link until an intersection is reached, where network topology dictates all possible paths branching from that intersection. As additional GNSS points are considered, existing paths are scored based on how likely they are to represent the true path through the network and the least likely paths are discarded. When the end of the trip is reached, the algorithm outputs the most likely path from the set of candidates.

2.3. Probabilistic Map Matching Algorithms

Probabilistic map matching (PMM) algorithms utilize statistical models to identify the series of links that are the most likely path for a given GNSS trajectory (Velaga, 2010). Most PMM algorithms utilize a Hidden Markov Model (HMM) where, in the case of map matching, i) the hidden states are the travelled links, ii) the observed variables are the GNSS positions, iii) the conditional distribution of the observed data follows the distribution of the GNSS error measurements, and iv) the transition probabilities of the next link given the current link of the vehicle. A traditional PMM algorithm (a "Standard HMM") follows four basic steps. Step 1: Candidate links are determined based on a search area around the GNSS point. Step 2: The likelihood for each candidate link is determined based on its distance from the GNSS point. Step 3: For each candidate link, the transition probability to all candidate roads for the next GNSS point is calculated by a shortest path routing algorithm. Step 4: A recursive algorithm (such as the Viterbi algorithm) is employed to find the optimal, or most probable) sequence of links. The number or shortest path calculations required for Step 3, and the recursive nature of Step 4 have inspired research into the improvement of PMM algorithms (Dogramadzi & Khan, 2021; Mohamed, et al., 2017). Specifically, this paper considers a "Fast HMM" presented by Koller *et al.* (Koller, et al., 2015), who used a Bidirectional Dijkstra algorithm to calculate the "cost" of a node and its edges only once the search reaches that node, greatly improving the run time by eliminating unnecessary calculations. The final set of travelled links is then chosen by minimizing the path cost rather than maximizing the probability as in the Viterbi algorithm. The current implementation allows for several shortest path algorithms which operate in the same basic manner; by building a set of nodes starting at the origin and sequentially adding the next closest node until reaching the destination (bidirectional algorithms start at both origin and destination simultaneously).

2.4. Fuzzy Map Matching Algorithms

Recently, more advanced map matching algorithms extended simpler geometric and topological methods by incorporating additional information such as GNSS data quality, vehicle heading, and vehicle speed along with advanced statistical, mathematical, or artificial intelligence techniques to improve performance. Techniques considered to date include dead reckoning (Zhao, et al., 2003), ant colony algorithms (Gong, et al., 2018; Yi-hu, et

al., 2007); extended Kalman filtering (Krakiwsky, et al., 1988; Xu, et al., 2010), belief theory (Nassreddine, et al., 2008), neural networks (Winter & Taylor, 2003), genetic algorithms (Nikolic & Jovic, 2017), and fuzzy inference (Quddus, et al., 2006; Kim & Kim, 2001; Syed & Cannon, 2004; Ren & Karimi, 2012). This study considers the fuzzy map matching (FMM) algorithm presented by Quddus, Noland, and Ochieng (Quddus, et al., 2006), who noted that the accuracy of urban GNSS trajectories is often low due to urban canyons. Recognizing that the number of candidate links was quite high, the authors implemented a fuzzy inference system (FIS) to explicitly handle the issues of certainty and likeliness. In FIS, discrete inputs are first "fuzzified" to represent various degrees of truth (e.g., "the speed of the vehicle is low" (Quddus, 2006)). This is accomplished using a predetermined membership function which assigns a value between 0 and 1 to represent the truth of the statement. Next, IF-THEN rules are implemented for inference (e.g., "if speed is high then the likelihood of Link A is low"), and results of multiple rules are aggregated. This result is transformed back to a discrete value (e.g., "match to Link B"). The considered algorithm uses inputs of vehicle speed, heading, current position, and previous positions along with connectivity and orientation of the road network and the contribution of satellite geometry to horizontal errors. The outputs are the map-matched positions (and their uncertainty) and link IDs.

2.5. Hybrid Map Matching Algorithms

Although the previous approaches are presented as mutually exclusive, some algorithms, known as hybrid map matching (HyMM) algorithms, combine various methodologies in an attempt to optimize the map matching process. Singh *et al.* (Singh, et al., 2009) recognize that recent research has focussed heavily on hybrid approaches, including studies from Knapen *et al*. (Knapen, et al., 2018), Sharath, Velga, and Quddus (Sharath, et al., 2019), Wu *et al.* (Wu, et al., 2020), Liu et al. (Liu, et al., 2017), and Trogh et al. (Trogh, et al., 2020). This research considers the hybrid topological/probabilistic algorithm implemented in the Open Source Routing Machine (OSRM) (Luxen & Vetter, 2011), though other combinations of schemes are possible. The OSRM uses a similar method to the Fast HMM described earlier, implementing a bidirectional Dijkstra algorithm to query a graph representing the road network. However, rather than considering all the possible nodes in the road network, OSRM utilizes contraction hierarchies (CH) to order the nodes "by some measure of importance and shortcuts them in this order" (Luxen & Vetter, 2011). Shortcutting essentially removes specific nodes and inserts shortcuts in their place. This process leads to a "convenient trade-off between preprocessing and query time" and allows queries to be run "in the order of about a hundred microseconds" (Luxen & Vetter, 2011). For full details of the OSRM implementation, readers are referred to Luxen and Vetter (Luxen & Vetter, 2011).

3. Methodology

The methodology consisted of several steps describe in the following sections, beginning with the selection of smartphone devices and applications. Next, the trip scenarios were defined, before the map matching algorithms were run, and analysis performed on their outputs.

3.1. Smartphone Devices

Data was collected using 10 smartphone devices, affixed to a single board shown in Fig. 2, and placed in the rear window of the test vehicle, an automatic transmission Toyota Corolla. While one team member drove, a second manually activated and monitored the smartphones for the duration of the scenario. Details of the 10 devices, including 5 iPhone models and 5 Android devices, are provided in Table 1.

3.2. Data Acquisition

Three different data acquisition applications were installed on all smartphone devices. TrueMotion (https://apps.apple.com/us/app/truemotion-family-safe-driving/id1121316964) is a smartphone telematics platform that forms that basis of the UBI program of Intact Insurance, and AutoMerit (https://apps.apple.com/ca/app/automerit/id1157965687) is the UBI application of Belaire Direct, an Intact subsidiary. GPSTracker (https://apps.apple.com/us/app/gps-tracker-mobile-device/id1025196806?ls=1) is a third-party

application. The primary difference between the applications is their sample rate. While the TrueMotion and AutoMerit applications sample GNSS data at a baseline rate of 1 Hz (60 samples/minute), GPSTracker has a much lower baseline rate, which varies depending on the smartphone device. For GPSTracker, the highest sampling rate achieved was for the iPhone 6s (between 6 and 14 samples per minute) while the lowest was for the iPhone 8 (between 1 and 6 samples per minute). In general, these sampling rates correspond to approximately 0.016 to 0.23 Hz. Data was extracted from the applications in .csv format or javascript object notation (json) before being converted to .xlsx and .gpx formats for map matching.

Fig. **2**. Smartphones installed on a board during the data collecting process

3.3. Trip Scenarios

To capture environments detrimental to GNSS performance, 12 scenarios were defined within Montreal, Canada. Targeted environments include the urban center (likely to suffer from the urban canyon effect), short road segments in Montreal's old city, highways with parallel service lanes and overpasses, bridges with overhanging structures, and tunnels. Ground truth trajectory for each trip was manually recorded and is presented in Fig. 3. GNSS data was collected from December 30, 2017 to January 1, 2018, with details presented in Table 2.

The urban canyon effect is represented by Scenarios 1 to 5 which cover the neighbourhoods of Griffintown and Old Port as well as Saint Catherine St in the downtown core. The total distance for these 5 scenarios, which feature particularly noisy data and partial GNSS signal blockage, was 97.4 km. Scenarios 6 and 7 cover the Jacques Cartier Bridge and the Ville Marie Tunnel respectively, where GNSS signals are either partially or completely blocked for at least a portion of the 14.8 km trip length (in the case of the bridge, blockages are caused by the steel support structure above the roadway). Trajectories 8 to 12 consider highway environments amongst dense urban networks on Highway 134, Highway 15, Highway 720, Highway 520, and Highway 25 for a total traveled distance of 290.5 km. The primary note for GNSS quality in these scenarios is that the higher speed results in a greater distance between consecutive points.

3.4. Algorithm Implementation

The code for the algorithms considered in this study are open source, except for TrackMatching as described below. Where necessary, some of the open-source code has been modified to ensure consistent results between algorithms. For each algorithm, one set of parameters was used to process all data files (specifically, the default parameters). No attempt was made to optimize results based on phone, scenario, or application. All algorithms were configured to

match data to the OSM road network. All the algorithms (along with instructions on their use), data, and additional scripts to generate the results in full are provided in a publicly available repository (https://github.com/JoshuaStipancic/mapmatching-project). Readers are encouraged to refer to the repository for more information (including additional results not provided in this paper) and to verify the findings presented herein.

Table 1. List of Smartphone Models and Related Characteristics Used in the Data Collection Process

No	Model	Operating System	Processor	GNSS Receiver	Released
1	IPhone SE	iOS 9.3.2, upgradable to iOS 11.2	Qualcomm MDM9625M LTE modem, WTR1625R RF transceiver	A-GPS, GLONASS	March 2016
2	IPhone 6S	iOS 9, upgradable to iOS 11.2	Apple A9 Dual-core 1.84 GHz Twister PowerVR GT7600	A-GPS, GLONASS, GALILEO, QZSS	September 2015
3	IPhone 7	iOS 10.0.1, upgradable to iOS 11.2	Quad-core 2.34 GHz (2x Hurricane + 2x Zephyr)	A-GPS, GLONASS, GALILEO, QZSS	September 2016
4	IPhone 7+	iOS 10.0.1, upgradable to iOS 11.2	Quad-core 2.34 GHz (2x Hurricane + 2x Zephyr)	A-GPS, GLONASS, GALILEO, QZSS	September 2016
5	IPhone 8	iOS 11, upgradable to iOS 11.2	Hexa-core (2x Monsoon + 4x Mistral)	A-GPS, GLONASS, GALILEO, QZSS	September 2017
6	LG G6	Android 7.0 (Nougat)	Quad-core (2x2.35 GHz Kryo & 2x1.6 GHz Kryo)	A-GPS, GLONASS, BDS	March 2017
7	Google Pixel2	Android 8.0 (Oreo)	Qualcomm® Snapdragon™ 835 2.35Ghz + 1.9Ghz, 64Bit Octa-Cor	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)	October 2017
8	Samsung Galaxy S5	Android 4.4.2 (KitKat), upgradable to 6.0	Quad-core 2.5 GHz Krait 400	A-GPS, GLONASS, BDS	April 2014
9	Samsung Galaxy S7	Android 6.0 (Marshmallow), upgradable to 7.0 (Nougat)	Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)	A-GPS, GLONASS, BDS	March 2016
10	Samsung Galaxy S8	Android 7.0 (Nougat)	Octa-core (4x2.3 GHz & 4x1.7 GHz) - EMEA	A-GPS, GLONASS, BDS, GALILEO	April 2017

Topological Map Matching

The TMM implementation used in this study is TrackMatching (https://mapmatching.3scale.net/), a web-based map matcher that is offered as software-as-a-service (SaaS). Although paid subscriptions are available, smaller volumes of data can be processed free-of-charge, including the entirety of this project's dataset. TrackMatching can be implemented using a simple curl command from the command line. In this project, the TMM results are processed by a shell script to handle both pre- and post-processing of the data.

Probabilistic Map Matching

As discussed earlier, two different PMM algorithms are tested in this study. The first is a conventional HMM utilizing the Viterbi algorithm (the Standard HMM) available in the open-source Python library map_matching (https://github.com/mapillary/map_matching). This algorithm requires the road network be stored in the spatial database PostGIS before being run using a simple Python command. This study makes use of a developed Python script to handle data processing. The second PMM algorithm (the Fast HMM) is implemented using Graphhopper Version 0.10 (https://github.com/graphhopper/graphhopper/) as well as Version 0.10 of Graphhopper's map matching algorithm from another repository (https://github.com/graphhopper/map-matching/). The Fast HMM is implemented

using a developed Java script to implement the necessary Graphhopper classes and handle data processing. For this study, four shortest-path routing algorithms are considered, namely Dijkstra, bidirectional Dijkstra, A*, and bidirectional A*.

Fuzzy Map Matching

This study's FMM algorithm utilizes a modified version of the R library *fuzzyMM* (https://github.com/ngort01/fuzzyMM). Additional R scripts were written to implement R packages and handle data processing.

Hybrid Map Matching

The HyMM algorithm in this study was modified from Version 5.23 of the OSRM backend (https://github.com/Project-OSRM/osrm-backend). The compiled C++ algorithm is called from a curl command, run from a shell script developed for parallel data processing.

Table 2. Definition of Predefined Trip Scenarios

No	Scenario	Distance (km)	Duration (min)	GNSS Quality
1	Downtown Area 1	24.4	83	Noisy, partial blockage
2	Downtown Area 2	26.8	107	Noisy, partial blockage
3	Downtown Area 3	10.3	40	Noisy, random values
4	Downtown Area 4	25.1	95	Noisy, random values
5	Downtown Area 5	10.8	70	Noisy, random values
6	Bridge	10.5	28	Partial blockage
7	Tunnel	4.3	17	Complete blockage
8	Highway 1	47.9	63	High speed, spatially sparse
9	Highway 2	57.8	75	High speed, spatially sparse
10	Highway 3	68.1	85	High speed, spatially sparse,
11	Highway 4	51.4	71	High speed, spatially sparse
12	Highway 5	65.3	91	High speed, spatially sparse,

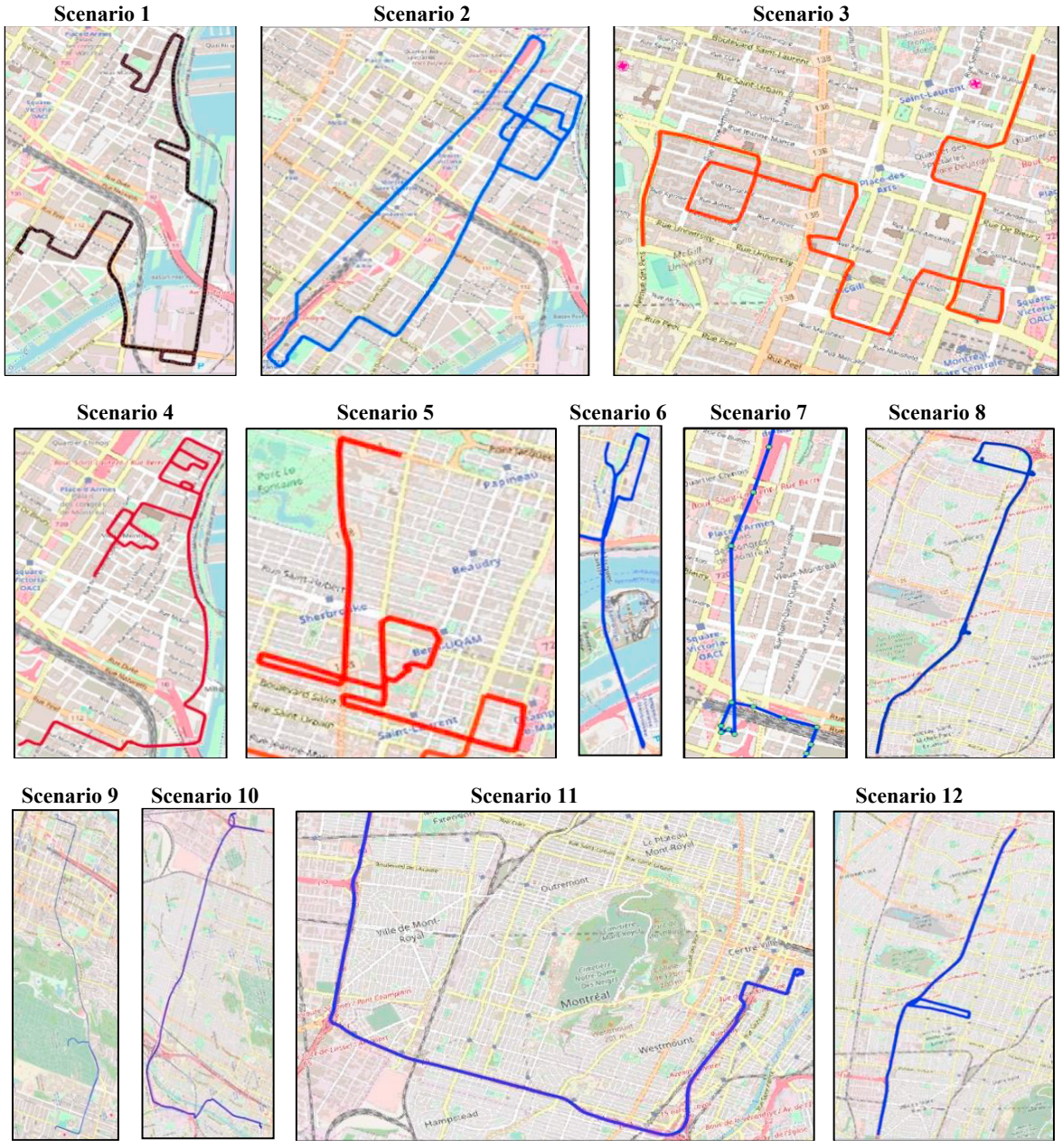

Fig. **3**. Maps of the predefined trip scenarios generated using ground truth data

3.5. Analysis

Performance Measures

The performance of the map matching algorithms is evaluated using four measures, the first three of which are related to match performance, or how closely the map matched results match the actual travelled routes. The sequence of travelled links (defined by their unique OSM ID) identified by the algorithms (the "map-matched data") was compared to ground truth data prepared manually (the "ground truth data") using the *SequenceMatcher* class from the Python library *difflib* to identify the matching blocks of consecutive link IDs. Based on this comparison, three measures were developed to quantify the overall accuracy, false negatives, and false positives. First, the overall accuracy for map-

matched output in scenario j when processed with algorithm i was calculated according to

$$accuracy_{ij} = 2\,C_{ij}/\left(G_j + M_{ij}\right) \quad (1)$$

where C_{ij} is the number of correct link matches, G_j is the number of links of the ground truth route, and M_{ij} is the number of links of the map-matched output by algorithm i (the multiplication by 2 allows the indicator to be normalized with a maximum of 1). Next, the false negative percentage in the map-matched data (links present in the ground truth that are missed by the map matching algorithm) is computed according to

$$missed_{ij} = \left(G_j - C_{ij}\right)/G_j \quad (2)$$

Finally, the false positive percentage in the map-matched data (links falsely identified by the map matching algorithm) is computed using

$$false_{ij} = \left(M_{ij} - C_{ij}\right)/M_{ij} \quad (3)$$

Computing false positives and negatives in addition to an overall accuracy measure provides more insight into the performance of the algorithms and is useful in applications where either may be particularly harmful. Some authors (Newson & Krumm, 2009) have suggested that accuracy measures be distance-weighted, so that mistakes related to longer links (which represent a relatively longer portion of the total trip) have a relatively larger impact on the measured performance. While distance-weighted equivalents of the above measures (weighted by length of the OSM IDs) were tested, they made no significant difference to the results.

One final measure was computed related to run time performance. Computational time was defined as the time required to run the map matching algorithm, not including other elements such as reading and writing files or formatting data and was normalized according to input trip length. Therefore, runtime was computed according to

$$time_{ij} = 1000\,T_{ij}/O_j \quad (4)$$

where T_{ij} is the computational time as defined above for algorithm i applied to input GNSS data for scenario j, and O_j is the number of observations in the input GNSS data for scenario j. To ensure consistency in the time measurements, all algorithms were run on the same computer (2017 MacBook Pro, 2.5 GHz Intel Core i7, 8 GB RAM, macOS Mojave 10.14.6) with no other programs running simultaneously.

ANOVA and Data Visualization

The stated purpose of this study is to compare the performance of several algorithms on smartphone GNSS data in urban environments. Specifically, the goal of this analysis is to determine which factors influence map matching algorithm performance. In order to determine the factors influencing performance, a series of ANOVA tests were performed. Tests were performed on all four performance measures, with data grouped according to a) algorithm, b) scenario, c) phone, and d) application. Additionally, as algorithm performance is the primary interest in this work, interactions between algorithm and the other grouping variables were considered (the interaction between algorithm and scenario, for example). This yielded 28 unique ANOVA tests (16 independent tests and 12 first-order interactions). In the results below, only results statistically significant at 95% confidence are presented for brevity. Note that because 28 tests are conducted simultaneously, this study considers that p-values must be less than an experimental threshold of 5%/28 (or 1.8×10^{-3}, according to the Bonferroni correction). Additionally, if the results for the independent tests were significant in addition to a significant result for their interactions, then only the results including the interaction are presented. For example, consider if both "scenario" and "algorithm" significantly influence "accuracy", but the interaction variable "algorithm x scenario" is also significant. In this case, not only is the effect of "algorithm" significant, but that effect varies significantly across scenarios (and vice versa). Therefore, the results of the test with the interaction are needed to draw all relevant conclusions. If the results of one independent test were not statistically significant, then no results are presented even if the interaction term was shown to be statistically significant (i.e., the variables must be independently significant for the results of the test on their interaction to be considered valid). For

some of the significant tests, additional pairwise testing was performed using Tukey's range test to determine which means were significantly different from each other. Following the ANOVA tests, a series of line, box, and scatter plots are used to illustrate the results of the statistical testing and to further describe performance.

4. Results

4.1. Descriptive Statistics

Table 3. Descriptive Statistics of Algorithm Performance

		min	max	mean	median	std dev
TMM	Accuracy (%)	1.0	100.0	**92.1**	**93.8**	9.35
	Missed (%)	0.0	98.7	10.1	5.5	12.1
	False (%)	0.0	99.2	**4.5**	**2.6**	7.8
	Time (s)	0.8	1346.0	27.0	1.4	152.1
Std. HMM	Accuracy (%)	0.7	87.1	52.8	52.8	14.5
	Missed (%)	0.0	98.7	17.8	15.4	13.7
	False (%)	7.4	99.58	57.6	60.8	18.8
	Time (s)	1.2	11799.8	86.4	3.7	715.7
Fast HMM	Accuracy (%)	5.0	100.0	87.0	91.4	14.1
	Missed (%)	0.0	93.2	**10.2**	**6.2**	11.6
	False (%)	0.0	96.1	13.7	7.7	17.0
	Time (s)	0.0	3578.0	27.0	1.0	233.8
FMM	Accuracy (%)	1.2	65.4	31.8	31.1	13.8
	Missed (%)	0.0	96.0	28.3	26.1	18.4
	False (%)	33.3	99.3	76.5	79.2	14.0
	Time (s)	21.5	275416.7	2853.6	149.7	20190.6
HyMM	Accuracy (%)	21.7	98.4	80.6	81.6	11.1
	Missed (%)	0.0	87.8	24.5	25.3	14.9
	False (%)	0.0	75.0	10.8	8.2	10.6
	Time (s)	0.4	1936.0	**18.3**	**0.9**	152.0

Table 3 presents the descriptive statistics for the map matching algorithms, with the best means and medians highlighted and bolded. For the Fast HMM, results for accuracy, missed, and false links were identical regardless of the shortest path routing algorithm used. Results for run time were similar, with the bidirectional shortest path algorithms being slightly faster. Therefore, in this table and elsewhere, the results for the Fast HMM are specifically for the bidirectional A* algorithm unless explicitly noted. Based only on these descriptive statistics, it appears that three of the algorithms (TMM, Fast HMM, and HyMM) are superior to the other two in terms of accuracy, with these three having a mean accuracy of between 80% and 90%. Additionally, these algorithms have a median run time of around 1 second, compared to a slightly longer median run time of 3.7 seconds for the Standard HMM and a much longer median time for the FMM of 149.7 seconds. The poorer performance of both the Standard HMM and the FMM can be attributed to a higher proportion of false links (exceeding 50% on average for both algorithms). This result implies that the TMM, Fast HMM, or HyMM are a superior choice to the other two, although more conclusive results are revealed through the following statistical testing.

4.2. ANOVA Results

The results of all significant ANOVA tests are presented in Table 4. The first primary observation from these results is that groups of *algorithm*, *scenario*, and *application* all significantly affect the three match performance measures of *accuracy*, *missed links* and *false links*. Furthermore, the effect of *algorithm* varies significantly with respect to both the scenario and the application from which the data is collected for all three match performance measures. This result is discussed in more detail in the following section. In addition to these effects two additional significant effects are observed. First, the mean proportion of missed links is significantly different for at least one of the considered phones. It should be noted that the choice of phone does not significantly affect any other performance measures, so this observation may have few practical implications. Finally, *algorithm* was the only factor to have a significant impact on the matching run time per 1000 observations. These additional significant effects are described in detail further below.

Table 4. Results of All Significant ANOVA Tests

Comparison Group	F	p-value
ACCURACY		
Algorithm x Scenario	9.09	5.42E-88
Algorithm x Application	21.29	2.73E-52
MISSED		
Algorithm x Scenario	13.19	1.75E-135
Algorithm x Application	18.26	2.97E-44
Phone	3.24	6.48E-04
FALSE		
Algorithm x Scenario	8.74	7.94E-84
Algorithm x Application	25.78	4.58E-64
TIME		
Algorithm	6.49	1.31E-07

4.3. Effects of Algorithm, Scenario, and Application on Match Performance

Plots illustrating the statistically significant ANOVA results are provided in Fig. 4. These plots show the average of each match performance measure (*accuracy*, *missed*, and *false*) for each algorithm (denoted as coloured lines) across each scenario or application. First, considering the plots for the average accuracy, three of the algorithms outperform the other two. Specifically, the Fast HMM, TMM, and HyMM algorithms score significantly higher than the Standard HMM, which in turn outperforms the FMM algorithm in terms of accuracy (this result was confirmed by additional pairwise testing). Considering the results by scenario, not only does the choice of algorithm significantly affect accuracy, but that effect varies by scenario. For example, while the average accuracy for Scenario 10 is lower than for Scenario 9 for most of the algorithms, Scenario 10 accuracy was actually higher than that in Scenario 9 when using the TMM algorithm. While other similar observations could be made about other scenarios, it is generally concluded that the overall accuracy does not increase or decrease uniformly by scenario for each of the algorithms. Considering the results by application used for data collection, there is negligible difference between results obtained using either TrueMotion or AutoMerit, which both collect data at 1 Hz. However, the accuracy of the GPS Tracker application, which collects data at a lower frequency, is significantly different. For the Standard HMM and FMM algorithms,

overall accuracy was higher for the GPS Tracker data compared to the other applications, while for the TMM algorithm it was slightly lower and for the remaining algorithms it was approximately equal.

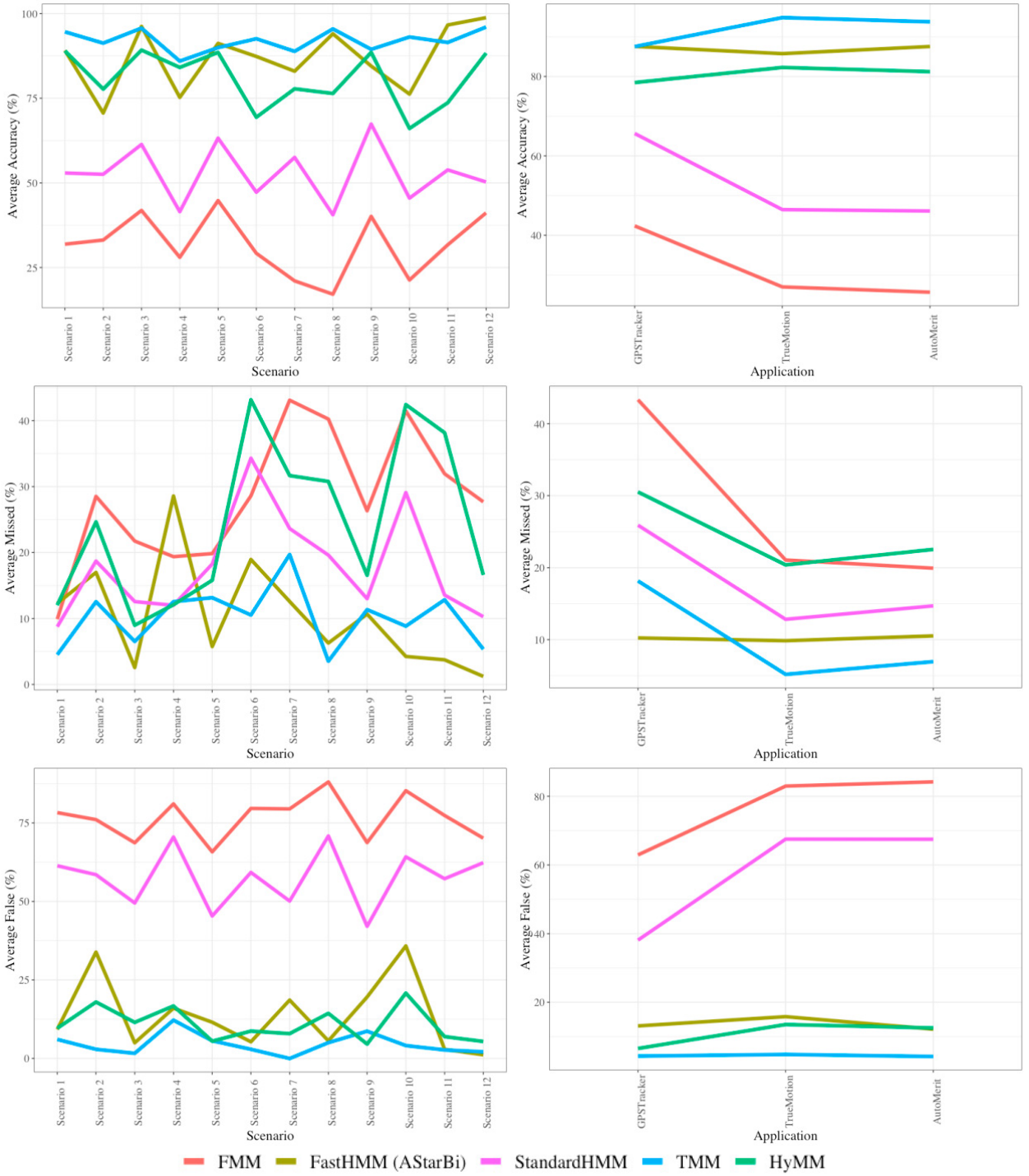

Fig. **4**. Plots of average accuracy, average missed links, and average false links for each algorithm by scenario and application

Next, considering the results for missed links, the difference in performance across algorithms is much smaller than for overall accuracy, though again the effect clearly changes across scenarios. For example, consider the Fast HMM: the average percentage of missed links is much higher in Scenario 4 relative to the other scenarios, which was not the case for the other algorithms. One key conclusion from this analysis is that on average, the proportion of missed links is higher for the bridge, tunnel, and highway scenarios than for the downtown scenarios, especially for the FMM, HyMM and Standard HMM algorithms. For the tunnel, this is expected as GNSS communication is interrupted leading to missing observations. For the other highway environments, this can likely be explained by small links present at exits, underpasses, and overpasses, which due to high speeds are more likely to be missed in the raw GNSS data. Considering the average percentage of missed links by application, the FMM and HyMM algorithms appear to have slightly poorer performance than their counterparts. Interestingly, using the GPSTracker data always resulted in more missed links on average except for the Fast HMM algorithm. Results for false links are similar to overall accuracy, with the FMM and Standard HMM algorithms performing significantly worse than the other algorithms. Additionally, for those two algorithms, the percentage of false links was much lower for the GPS Tracker data than for the other two applications. This is mainly because the lower frequency of the data yields less opportunity for incorrect links to be mistakenly added to the results.

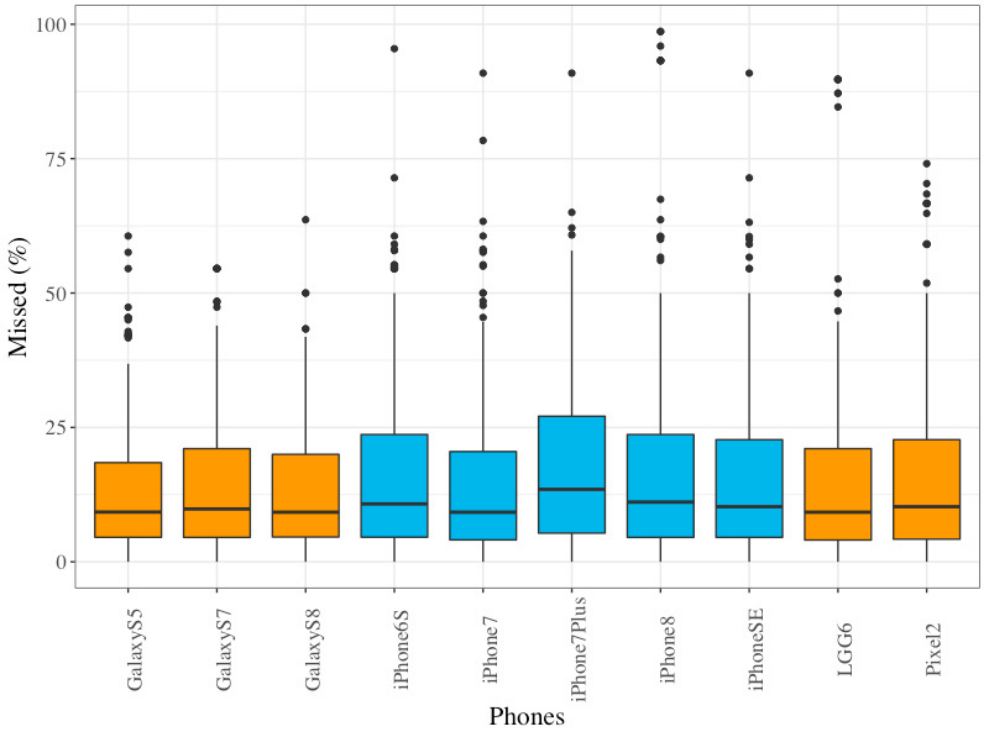

Fig. **5**. Box plot of the percent of missed links by phone, with iOS devices in blue and Android devices in orange

4.4. Other Significant Effects

A boxplot illustrating the distribution of missed links by phone is presented in Fig. 5. Although the ANOVA results indicate that there is a significant difference in the mean percentage of missed links by phone, this plot shows very little relative difference between the phones. According to further pairwise testing, the only significant differences are between the Galaxy S8 and the Galaxy S7, between the Galaxy S8 and iPhone 8, between the Galaxy S8 and the iPhone 7 Plus, and between the iPhone 7 Plus and the Galaxy S5. The final significant ANOVA result is that the average run time per 1000 observations is significantly different for at least one of the algorithms, and this result is demonstrated in Fig. 6. As can be seen in the boxplot (and as was confirmed by additional pairwise testing) the FMM algorithm has significantly longer run times compared to the other algorithms. There was no statistically significant

difference in the run times of the other algorithms. It should be noted that as the TMM is run as SaaS, the map matching is executed on a server with unknown characteristics. It therefore may appear to run faster (or slower) relative to the other algorithms.

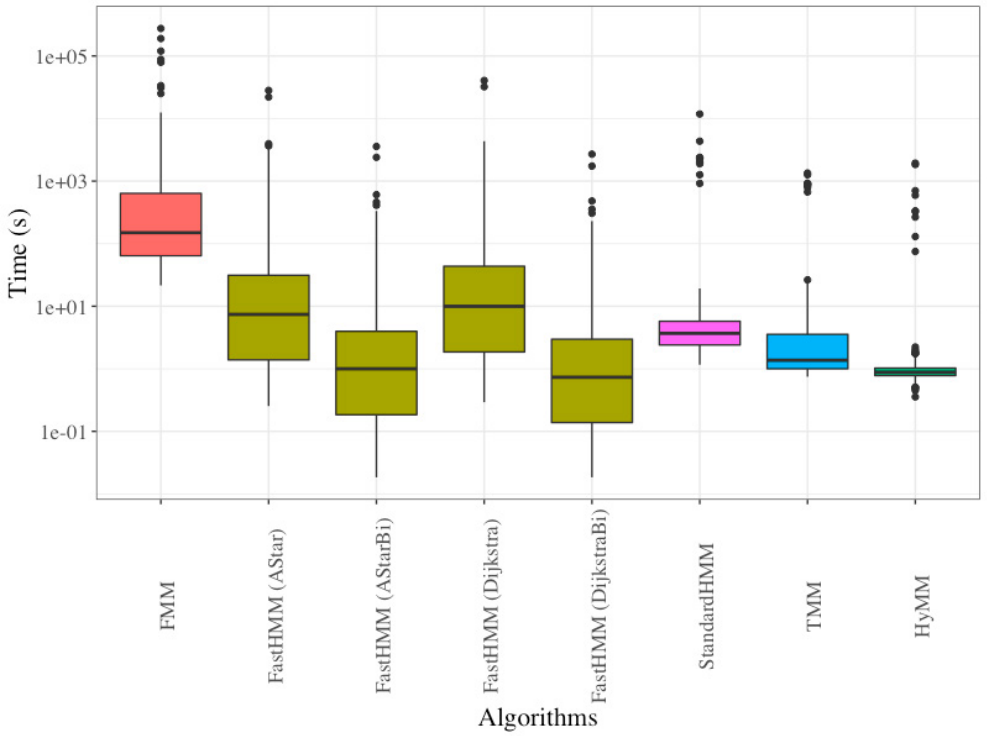

Fig. **6**. Box plot of matching run time per 1000 observations by algorithm (on a logarithm scale)

4.5. Overall Performance by Algorithm

The final step is to determine which algorithm or algorithms are superior from a practical perspective. This is done based on the two plots contained in Fig. 7. First, the percentage of missed links and false links are compared for every result (i.e., each data point represents exactly one input GNSS trajectory for each scenario, application and phone processed by one map matching algorithm). An exact match would appear in the top right corner of the graph, where the percentage of both missed and false links is 0 %. This plot confirms earlier findings from the descriptive statistics. Although the difference between algorithms in terms of missed links is relatively small (with most map matched results containing fewer than 25 % missed links), the Standard HMM and the FMM generate a far higher share of false links than the other algorithms, contributing significantly to their poorer overall performance. With manual review, this can be largely attributed to map matched points “jumping” between either parallel link (such as along divided arterials and highways) and perpendicular links (such as those connected at intersections). Even though the link IDs match those in the ground truth data, the unnecessary repetition in the IDs presents as false links.

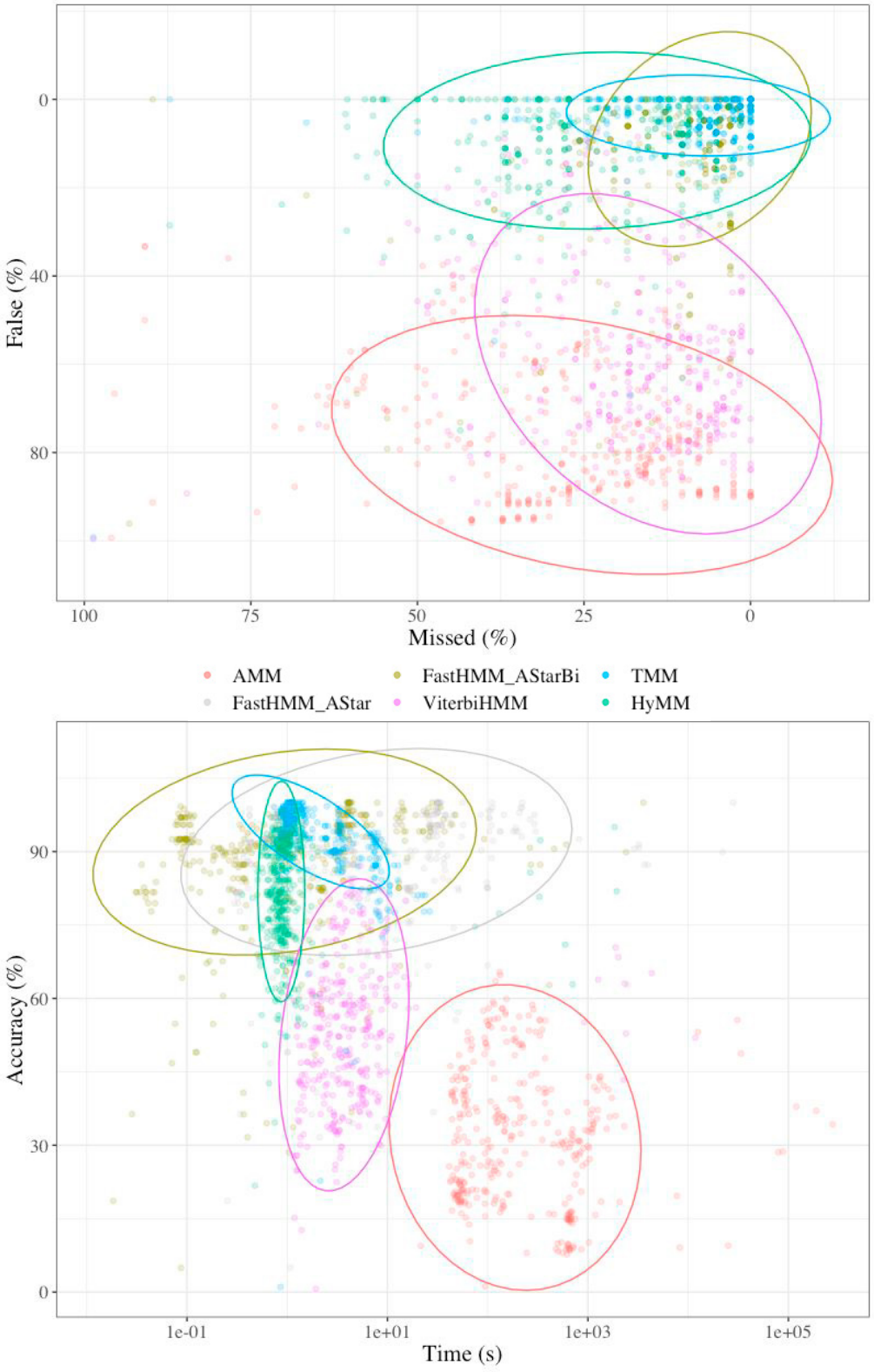

Fig. 7. % missed vs. % false (top) and accuracy vs. run time (bottom) with dispersion elipses for all considered algorithms

The second plot in Fig. 7 demonstrates the overall accuracy plotted against the run time, again for every input trajectory processed by every algorithm. In this plot, an ideal result would appear in the top left corner (high accuracy and low time per 1000 observations). This plot again clearly demonstrates the poorer performance of the Standard HMM and FMM. As discussed earlier, the only significant difference in terms of run time is for the FMM, which is significantly slower than the other algorithms. In fact, the FMM is both significantly slower and significantly less accurate. While there is no statistically significant difference in run time for the Standard HMM or the HyMM, there seems to be one visually and these algorithms are significantly less accurate on average than the Fast HMM and TMM algorithms. These results were confirmed by additional pairwise testing, which showed that the only pair of algorithms for which there was no significant difference in mean accuracy were the TMM and Fast HMM, which are the most accurate (followed by the HyMM, then the Standard HMM, then the FMM). In other words, this study found no significant difference in either accuracy or run time for the TMM and Fast HMM. All other algorithms were either significantly less accurate, slower, or both.

5. Conclusions

Analyzing large GNSS datasets requires accurate and efficient map matching procedures to reduce positional noise and link GNSS records to the road network. The objective of this study was to evaluate the performance of several algorithms on smartphone GNSS data in urban environments, by identifying the factors impacting performance and isolating the situations where one algorithm may outperform the others. Data was collected across 12 scenarios in Montreal, Canada using 10 smartphones and 3 data collection applications. This research considers five algorithms, namely one topological, two probabilistic, one fuzzy, and one hybrid. The performance of these algorithms was quantified using performance measures of overall accuracy, percent missed links, percent false links, and run time per 1000 observations.

Beginning with ANOVA testing, one primary observation is that the choice of algorithm significantly impacts the three matching performance measures: accuracy, missed links, and false links. Furthermore, the effect of the algorithm varies significantly across scenarios and applications. Using data visualization and pairwise testing, it was determined that the Fast HMM, TMM, and HyMM had significantly better overall accuracy than either the Standard HMM or FMM algorithms. This is largely due to the increased number of false links introduced into the matched results of those algorithms due to consecutive points jumping between parallel or perpendicular links. Considering the results by scenario, the only consistent observation made was that the bridge, tunnel, and highway scenarios tended to have more missed links than the downtown scenarios. This is logical as the higher speeds (and therefore larger distances between consecutive points) make it easier to miss smaller segments. Considering the results by application, GPSTracker (with a much lower observation frequency) tended to have more missed links overall, with fewer false links and poorer overall accuracy for the Standard HMM and FMM algorithms. Other algorithms were less sensitive to the choice of application. Two additional significant effects were uncovered by the ANOVA testing. The choice of phone has a significant impact on the number of missed links, though, as overall accuracy is not affected, this may have few practical implications. Pairwise testing also demonstrated the significant differences only held for few combinations of phones. Finally, the algorithm was shown to have a significant impact on run time. Based on the data visualization and further pairwise testing, it was determined that the FMM algorithm was significantly slower than the others, with no significant difference in the run times for any of the other considered algorithms.

A final comparison was made between the overall accuracy and the run time, as these measures are of most importance to the end user. This comparison again clearly demonstrates the poorer performance of the Standard HMM and FMM. Compared to the other algorithms, the FMM algorithm is both significantly slower and significantly less accurate (it should be noted that this is the only algorithm implemented in R). While there is no statistically significant difference in run time for either of the other four algorithms, there were additional differences in accuracy. Pairwise testing showed no significant difference in accuracy for the TMM and Fast HMM, though the HyMM was significantly less accurate and the Standard HMM was less accurate still. This study found no significant difference in either accuracy or run time for the TMM and Fast HMM. All other algorithms were either significantly less accurate, significantly slower, or both. Furthermore, both algorithms are relatively insensitive to scenario, phone, or application. The only major difference between them is that the TMM is offered as SaaS (free for small datasets, paid subscription for larger ones) while the Fast HMM implementation, Graphhopper, is open source.

For perspective, both the code and data for the entire project is provided for anyone to replicate this work and improve upon it. The provided data with its ground truth constitutes a public benchmark for map-matching algorithms.

This is particularly attractive as new map-matching algorithms will be developed and can be compared to the state of the art on this benchmark. Future research can complement this study with GNSS data for other modes, for less studied modes like active modes. Cycling and walking may have their own characteristics, with the lower speed providing more observations per link, and challenges in terms of their respective networks. Cycling facilities may be particularly difficult to correctly identify when parallel with roads. Foot paths and more importantly open two-dimensional spaces like parks and plazas require redefining the problem itself. GNSS data changes with the addition of new satellite constellations such as the Chinese BeiDou system will bring improvements in terms of positional accuracy that should be quantified for map matching.

References

Bernstein, D. & Kornhauser, A., 1998. *An introduction to map matching for personal navigation assistants.* s.l.:s.n.
Bierlaire, M., Chen, J. & Newman, J., 2013. A probabilistic map matching method for smartphone GPS data. *Transportation Research Part C: Emerging Technologies,* Volume 26, pp. 78-98.
Chawathe, S. S., 2007. *Segment-based map matching.* s.l., s.n., pp. 1190-97.
D'Este, G. M., Zito, R. & Tayler, M. A., 1999. Using GPS to Measure Traffic System Performance. *Computer-Aided Civil and Infrastructure Engineering,* Issue 14, pp. 255-265.
Dogramadzi, M. & Khan, A., 2021. Accelerated Map Matching for GPS Trajectories. *IEEE Transactions on Intelligent Transportation Systems ,* pp. 1-10.
Gong, Y.-J.et al., 2018. AntMapper: an ant colony-based map matching approach for trajectory-based applications. *IEEE Transactions on Intelligent Transportation Systems,* Volume 19, pp. 390-401.
Greenfeld, J. S., 2002. *Matching GPS observations to locations on a digital map.* Washington DC, s.n., pp. 164-173.
Handel, P. et al., 2014. Smartphone-Based Measurement Systems for Road Vehicle Traffic Monitoring and Usage-Based Insurance. *IEEE Systems Journal,* 8(4), pp. 1238-1248.
Hashemi, M. & Karimi, H. A., 2014. A critical review of real-time map-matching algorithms: Current issues and future directions. *Computers, Environment and Urban Systems,* Volume 48, pp. 153-165.
Huang, J., Liu, C. & Qie, J., 2014. *Developing map matching algorithm for transportation data center.* s.l., s.n., pp. 167-170.
Hu, G. et al., 2017. If-matching: Towards accurate map-matching with information fusion. *IEEE Transactions on Knowledge and Data Engineering,* Volume 29, pp. 114-127.
Hunter, T., Abbeel, P. & Bayen, A., 2014. The path inference filter: model-based low-latency map matching of probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems,* Volume 15, pp. 507-529.
Kim, S. & Kim, J.-H., 2001. Adaptive fuzzy-network-based C-measure map-matching algorithm for car navigation system. *IEEE Transactions on industrial electronics,* Volume 48, pp. 432-441.
Knapen, L., Bellemans, T., Janssens, D. & Wets, G., 2018. Likelihood-based offline map matching of GPS recordings using global trace information. *Transportation Research Part C,* Volume 93, pp. 13-35.
Koller, H., Widhalm, P., Dragaschnig, M. & Graser, A., 2015. *Fast hidden Markov model map-matching for sparse and noisy trajectories.* s.l., s.n., pp. 2557-2561.
Krakiwsky, E. J., Harris, C. B. & Wong, R. V., 1988. *A Kalman filter for integrating dead reckoning, map matching and GPS positioning.* s.l., s.n., pp. 39-46.
Liu, X., Liu, K., Li, M. & Lu, F., 2017. A ST-CRF Map-Matching Method for Low-Frequency Floating Car Data. *IEEE Transactions on Intelligent Transportation Systems,* 18(5), pp. 1241-1254.
Li, X., Lin, H. & Zhao, Y., 2005. A connectivity based map-matching algorithm. *Asian Journal of Geoinformatics,* Volume 5, pp. 69-76.
Luxen, D. & Vetter, C., 2011. *Real-time routing with OpenStreetMap data.* Chicago, Illinois, s.n., pp. 513-516.
Marchal, F., 2015. *TrackMatching.* [Online]
Available at: https://mapmatching.3scale.net/
[Accessed 1 May 2015].
Marchal, F., Hackney, J. & Axhausen, K. W., 2005. Efficient Map Matching of Large Global Positioning System Data Sets. *Transportation Research Record,* Issue 1935, pp. 93-100.
Mohamed, R., Aly, H. & Youssef, M., 2017. Accurate Real-time Map Matching for Challenging Environments. *IEEE Transactions on Intelligent Transportation Systems,* 18(4), pp. 847-857.
Nassreddine, G., Abdallah, F. & Denœux, T., 2008. *Map matching algorithm using belief function theory.* s.l., s.n., pp. 1-8.

Nayak, S. & Narvekar, M., 2017. *Real-time vehicle navigation using modified A∗ algorithm.* s.l., s.n., pp. 116-122.
Newson, P. & Krumm, J., 2009. *Hidden Markov map matching through noise and sparseness.* s.l., s.n., pp. 336-343.
Nikolic, M. & Jovic, J., 2017. Implementation of generic algorithm in map-matching model. *Expert Systems with Applications,* Volume 72, pp. 283-292.
OpenStreetMap, 2015. *About.* [Online]
Available at: http://www.openstreetmap.org/about
[Accessed 11 May 2015].
Peker, A. U., Tosun, O. & Acarma, T., n.d. *Particle filter vehicle localization and map-matching using map topology.* 2011, s.n., pp. 248-253.
Pereira, F. C., Costa, H. & Pereira, N. M., 2009. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review,* Volume 1, pp. 107-124.
Quddus, M. A., 2006. *High Integrity Map Matching Algorithms for Advanced Transport Telematics Applications,* London, UK: s.n.
Quddus, M. A., Noland, R. B. & Ochieng, W. Y., 2006. A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems,* Volume 10, pp. 103-115.
Quddus, M. A., Ochieng, W. Y. & Noland, R. B., 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies,* Volume 15, pp. 312-328.
Quddus, M. & Washington, S., 2015. Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies,* Volume 55, pp. 328-339.
Quiroga, C. A. & Bullock, D., 1998. Travel time studies with global positioning and geographic information systems: an integrated methodology. *Transportation Research Part C,* Issue 6, pp. 101-127.
Ren, M. & Karimi, H. A., 2012. A fuzzy logic map matching for wheelchair navigation. *GPS solutions,* Volume 16, pp. 273-282.
Sharath, M. N., Velaga, N. R. & Quddus, M. A., 2019. A dynamic two-dimensional (D2D) weight-based map-matching algorithm. *Transpoartion Research Part C,* Volume 98, pp. 409-432.
Singh, J., Saravjeet, S., Singh, S. & Singh, H., 2009. Evaluating the performance of map matching algorithms for navigation systems: an empirical study. *Spatial Information Research,* 27(1), pp. 63-74.
Syed, S. & Cannon, M. E., 2004. *Fuzzy logic-based map matching algorithm for vehicle navigation system in urban canyons.* San Diego, CA, s.n., pp. 26-28.
Trogh, J. et al., 2020. Map Matching and Lane Detection Based on Markovian Behavior, GIS, and IMU Data. *IEEE Transactions on Intelligent Transportation Systems,* pp. 1-15.
Tselentis, D. I., Yannis, G. & Vlahogianni, E. I., 2016. Innovative insurance schemes: pay as/how you drive. *Transportation Research Procedia,* Volume 14, pp. 362-371.
Velaga, N. R., 2010. *Development of a weight-based topological map-matching algorithm and an integrity method for location-based ITS services,* s.l.: s.n.
Velaga, N. R., Quddus, M. A. & Bristow, A. L., 2009. Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies,* Volume 17, pp. 672-683.
Velaga, N. R., Quddus, M. A. & Bristow, A. L., 2012. Improving the performance of a topological map-matching algorithm through error detection and correction. *Journal of Intelligent Transportation Systems,* Volume 16, pp. 147-158.
Wang, M., Bao, X., Zhu, L. & Bao, Y. L., 2012. A map-matching method using intersection-based parallelogram criterion. *Advanced Materials Research,* pp. 2746-2750.
White, C. E., Bernstein, D. & Kornhauser, A. L., 2000. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies,* Volume 8, pp. 91-108.
Winter, M. & Taylor, G., 2003. *Modular neural networks for map-matched GPS positioning.* s.l., s.n., pp. 106-111.
Wu, Z., Xie, J., Wang, Y. & Nie, Y. (., 2020. Map matching based on multi-layer road index. *Transportation Research Part C,* Volume 118.
Xu, H., Liu, H., Tan, C.-W. & Bao, Y., 2010. Development and application of an enhanced Kalman filter and global positioning system error-correction approach for improved map-matching. *Journal of Intelligent Transportation Systems,* Volume 14, pp. 27-36.

Yang, H., Cheng, S., Jiang, H. & An, S., 2013. An enhanced weight-based topological map matching algorithm for intricate urban road network. *Procedia-Social and Behavioral Sciences,* Volume 96, pp. 1670-1678.
Yi-hu, W., Ning, L. & Zheng-wu, W., 2007. Application of ant colony algorithm in vehicle route guide system. *Systems engineering,* Volume 25, pp. 27-31.
Zhao, L., Ochieng, W. Y., Quddus, M. A. & Noland, R. B., 2003. An extended Kalman filter algorithm for integrating GPS and low cost dead reckoning system data for vehicle performance and emissions monitoring.. *The Journal of Navigation,* 56(2), pp. 257-275.