Australasian Transport Research Forum 2024 Proceedings
27-29 November, Melbourne, Australia
Publication website: https://australasiantransportresearchforum.org.au/

Map-matching large streams of vehicle GPS data into bespoke networks

Pedro Camargo[1], Renata Imai[1]

[1]Outer Loop Consulting

Email for correspondence (presenting author): pedro@outerloop.io

Abstract

In this paper, we present *MapMatcher*, an open-source Python package capable of map-matching large amounts of GPS data onto user-defined networks. The map-matching algorithm selects links likely to be traversed by a vehicle and verifies whether the points in the vehicle trajectory are close to this path. In case it doesn't, it iteratively adds waypoints to build a more suitable path. We assessed *MapMatcher*'s capabilities on a dataset containing more than 250,000 trip records and their GPS pings in Australia. The results obtained for the valid trips in the research dataset suggest that more than 95% of the trips have a match quality equal to or higher than 90%, and more than 96% of the map-matched traces have less than 50% variation in the path length.

1 Introduction

Access and availability of large streams of GPS data are getting more common, allowing its usage in the inference of vehicle route parameters and for route choice analysis, which are key components in the development of freight models.

Although GPS data has been a fixture in transport modelling and planning for at least 15 years, there is still a substantial disconnect between transport modelling software, transport models, and those analysis tools used to extract insights from this type of data. This disconnect is particularly obvious when it comes to map-matching, as there are no widely available tools for ingesting these datasets into the network models maintained by planning agencies.

Despite a wealth of studies focused on incorporating vehicle GPS data into planning models (Bernardin et al., 2015; Camargo and Tok, 2014; Kuppam et al., 2014; Liao, 2009; Sahin et al., 2024), we are not aware of any studies that resulted in tools that transcended those particular research projects, perhaps with one study on the validation of truck route choice enumeration (Camargo et al., 2017) being the only exception.

When it comes to map-matching, the disconnect is manifested by the difficulty of utilizing user-defined networks and the poor computational performance of the underlying algorithm implementations of existing tools. These problems became apparent when we were recently presented with the challenge of map-matching over 250,000 GPS different vehicle traces over a custom-built cadastral network covering one of the largest states in Australia, leading to the development of an open-source map-matching tool for large GPS databases.

This research was therefore conducted with the ultimate objective of map-matching a large stream of vehicle GPS data on a very large statewide network maintained by the state's transportation department. In the context above, the agency has agreed to have the development done with a more general architecture and as open-source software to serve as a public good, plugging this market gap and also benefitting from continuous updates and future improvements.

2 Context

There is very little freight data available in Australia, so the relative importance of any data grows significantly, including vehicle GPS data. The lack of data is accompanied by a lack of tools and frameworks for incorporating the available data sets into the existing analysis pipelines used by most transport agencies, further increasing the effort required for freight planning tasks within these agencies.

As mentioned in the introduction, one of the most obvious instances where the lack of appropriate tools increases the friction for data ingestion is the pervasive use of the map-matching software Valhalla by GPS data providers and the lack of available alternatives.

In summary, Valhalla (https://valhalla.github.io/valhalla/) is an excellent Open-Source tool for route computation on OpenStreet Maps networks. It also provides related tools, such as map matching, although its excellent performance comes at the cost of a rather complex setup process.

Despite being a powerful piece of software, Valhalla's design around Open-Street Maps means that, in most cases, all map-matching information for vehicle GPS data is provided in terms of Open-Street maps link IDs, which bear no relationship to the official networks maintained by most Australian planning agencies.

On the other end of the spectrum, there is a range of general open-source packages dedicated to map-matching GPS data, such as PyTrack, Leuven-MapMatching, and MappyMatch. However, all of them are designed for map-matching small quantities of GPS data, which is incompatible with the large streams of data normally processed in transportation planning applications.

In that setting, data providers in Australia and abroad resort to using Valhalla as their map-matching workhorse, forcing their agency clients to conflate Open-Street networks with their own networks in order to incorporate these results into existing analysis pipelines. The complexity and imprecision of network conflation, added to the substantial work required in these data translations, result in the incorporation of more imprecision into the data, as well as limits the instances where agency teams choose to deploy the data due to budgetary/timeline constraints.

3 Algorithm

Two approaches to map-matching have been tested during implementation. In the first one, the algorithm computes all likely stops along a route and builds the route by re-connecting these stops, while the second attempts to reproduce the route by computing the shortest path between the first and last GPS pings, and recursively adds waypoints to portions of the path that have been missed. The first approach was presented in previous research done in the USA (Camargo et al., 2017) and provided the base software structure used in this research, which implements the second approach.

Algorithm 1, presented below, describes the map-matching of a vehicle trace. This assumes that the map-matched vehicle traces occur between the vehicles' true origins and destinations, and do not record multiple trips between a sequence of stops.

Algorithm 1: Map-matching for vehicle trace

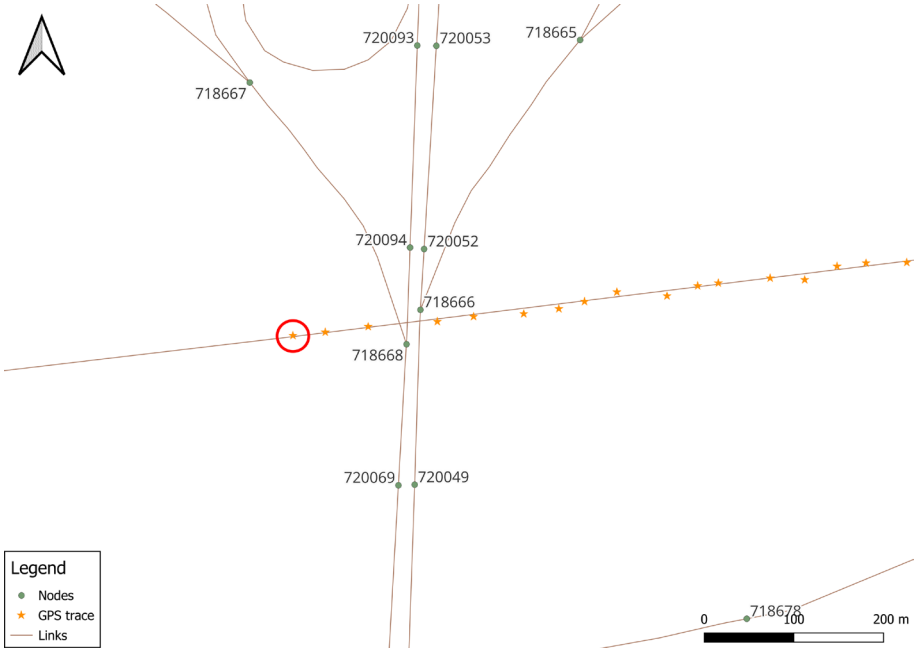
Algorithm 1:	
1	**procedure**
2	*link_candidates* ← lists all links likely traversed by the vehicle through Discount *link_candidates* on the graph
3	*stop_list* ← list[node from *link_candidates*[0], node from *link_candidates*[n-1]]
4	*rebuilt_path* ← shortest_path[*stop_list*[0], *stop_list*[1]]
5	*uncovered_pings* ← overlay(gps_trace, *rebuilt_path*)

```
2       While uncovered_ pings >min_coverage & waypoints<max_waypoints do:
3         waypoints ← waypoints + 1
4         stop_list ← stop_list  + waypoint WP to largest uncovered section
          rebuilt_path  ← empty
6         For stop1, stop2 in paired_sequences(stop_list) do:
7           rebuilt_path  ← rebuilt_path + shortest_path[stop1, stop2]
8           uncovered_pings ← overlay(gps_trace, rebuilt_path)
9           If uncovered_pings has not diminished
10            Remove WP from stop_list
18        end for
28      end while
30    end procedure
```

Generating the list of links likely traversed by a vehicle is the result of a geo-processing stage, where all links within a certain buffer from any GPS point in the vehicle's trajectory are selected, and further refined by requiring that the vehicle's heading at that point falls within a certain tolerance around the link's heading, which avoids matching irrelevant links when GPS pings fall close to an intersection.

This list is called link candidates, is used both to augment the network by discounting the graph in a way to incentivise shortest-path computations to use those links, and also so that we can use the start/end nodes of the link as inputs for shortest path computations. Surprisingly enough, deriving nodes from the identified links instead of identifying the closest links to GPS pings is critical to algorithm performance. A simple example is provided on Figure 1.

Figure 1: Trace starting in the middle of a link



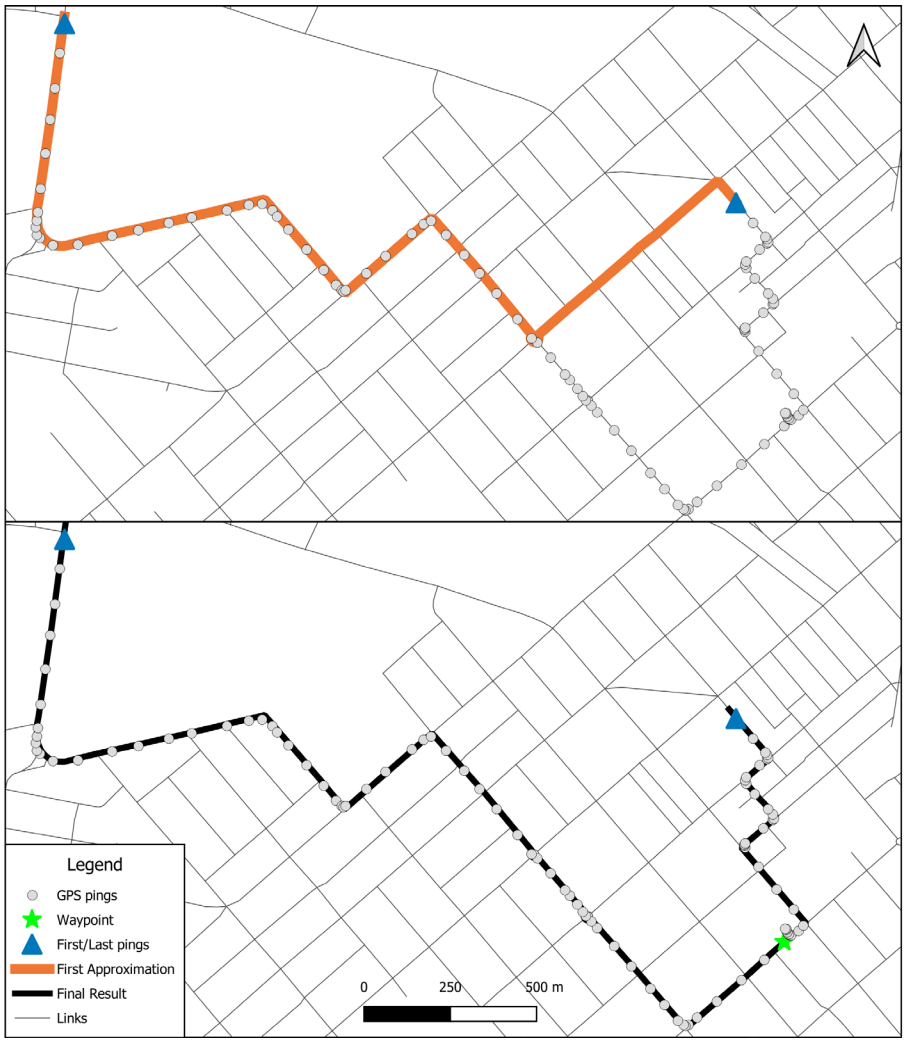
GPS trace is represented by stars with the first ping circled in red, and it is easy to see that we would be wrong to select the closest point to the first ping, namely node 718668 as the start of the path, as that node belongs to links that run perpendicular to the direction of the GPS trace.

The actual map-matching of the GPS trace starts with computing the shortest path between the first and last nodes on the list of link nodes identified when *link_candidates* were built. This is the first approximation of the path. After this first approximation, we verify whether a sufficient portion of the GPS pings has been adequately captured by the path approximation. In case it has not, the algorithm moves to its iterative stage.

The iterative stage of the algorithm consists of adding waypoints to the largest uncovered section of the path to a stop list. Then, the shortest path is calculated for each pair in the stops list, and the resulting path is overlaid on the existing GPS pings. If the addition of a waypoint does not improve the quality of the solution, it is removed, thus reducing the number of path computations required in future iterations. When a stopping criterium is met, the procedure finishes, returning the route that best connects the GPS pings in the vehicle trace.

Figure 2 exemplifies the iterative process of adding waypoints. Notice that the orange line that represents the first path approximation between the first and last GPS pings (represented by blue triangles) does not cover several GPS pings in the bottom right of the figure. However, as a waypoint (represented by a magenta diamond) is added between the existing stops, the shortest paths between each pair of waypoints approximate better the path registered by the GPS pings, leaving fewer or no pings uncovered.

Figure 2: Iterative waypoint addition



MapMatcher requires three different data inputs: A valid network, GPS data, and an AequilibraE Graph (Camargo, 2015). A valid network is provided as a GeoPandas GeoDataFrame (Van den Bossche et al., 2024) and has the link IDs corresponding to those in the AequilibraE Graph. The GPS data has to present at least columns containing the

trace_id, latitude, longitude, and timestamp. In its turn, the AequilibraE Graph holds the information necessary for path computation and can be easily built from the network itself.

A series of data-quality assurances is made to ensure the GPS data is adequate for map-matching before the process begins. The overall quality is assessed by the minimum number of pings, the minimum number of pings within the expected distance from any network link, and the minimum bounding box covered by all records. The vehicle speed is also checked, as vehicles can sustain high speeds only for an acceptable length of time. Lastly, we check for any inconsistencies related to location and timestamps, as GPS traces can contain records at the same instant but in different positions (i.e., data "jitter").

There are 5 built-in parameters for data quality and another 5 for the map-matching process included in the algorithm, all of them presented in Table 1These have default values and can be fine-tuned to better represent particular cases. The default parameters included in the *MapMatcher* package are those found to be optimal for the data source utilized in this research.

Table 1: *MapMatcher* procedure parameters and default values

Parameter	Description	Default value	Unit
Minimum pings	Minimum number of pings that the vehicle needs to have to be considered valid	15	n/a
Minimum coverage	Minimum diagonal of the Bounding box (m) defined by the GPS pings in the trace	500	m
Max speed	Maximum speed value considered realistic (130km/h default)	36.1	m/s
Max speed time	Maximum amount of time that speeds above the default maximum speed is considered acceptable. Above this time, GPS trace is considered problematic	120	S
Maximum jittery	Maximum "jitter" acceptable in the model	1	M
Cost discount	Link cost reduction ratio for links likely to be used	0.1	n/a
Buffer size	Buffer around the links to capture links likely used.	20	M
Minimum match quality	*Match_quality* expected to be reached	0.99	n/a
Maximum waypoints	Number of middle waypoints added in the algorithm	20	n/a
Heading tolerance	Tolerance to be used when comparing a link's direction with the link it seems to be associated with	22.5	degree

The procedure's output has four built-in quality measures, computed automatically during the map-matching process. The *match_quality* corresponds to the share of the GPS pings that are within the defined buffer distance from the map-matched path result but only considers GPS pings that have at least one network link closer than the buffer distance. The *match_quality_raw* represents the share of the GPS pings within the defined buffer distance from the map-matched path result. The number of waypoints required to reproduce the final path, which were the waypoints that, when added, improved the overall match of the solution are represented by *middle_points_required*. Finally, the distance_ratio, which is the total map-matched length divided by the trace's distance, is computed as straight lines between all points.

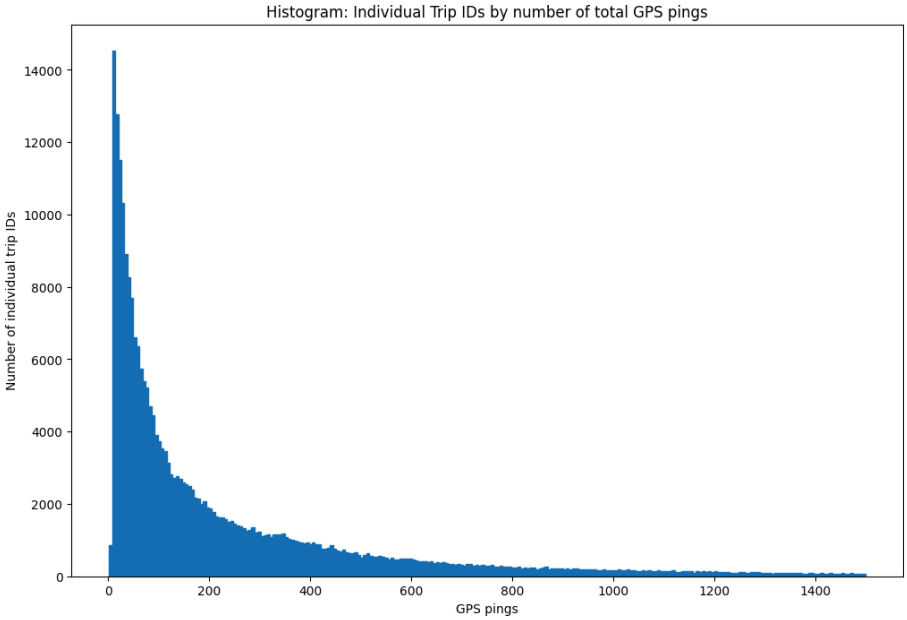
Since map-matching for cold data is an embarrassingly parallel problem, *MapMatcher* has been implemented with support for parallelisation through the Python multi-processing package. It defaults to using all CPU cores available. The software allows the user to set a specified number of threads to be used, but it is recommended that the user provide as much computational power as possible to run map-matching tasks to reduce run times.

4 Use case

The map-matching procedure was applied to the dataset provided by the client. This dataset included a total of 259,947 individual trace IDs, represented by 73,908,183 GPS records, resulting on an average of 284 records per trip. The network used in the procedure included 245,000km of centreline roads, over 338,000 links and 273,000 nodes, translating into over 620,000 directed links.

Figure 3 presents the distribution of pings per trace. For visualisation purposes, the distribution presented is only related to trips with 1,500 pings or less, which represent 97% of the traces. Two trips have over 9,000 pings, although it is easy to see that the majority, or over 62%, of all traces have less than 200 pings, which is an unexpectedly high share.

Figure 3: Ping distribution per trace/ trip ID



As a result of the data quality parameters used for this exercise, over 77,300 traces (29.7%) were not map-matched for violating the default parameter values. Five different data quality errors for the dataset are presented in Table 2. The most common errors were the distance between the GPS ping and a network link exceeding 20 metres, traces with less than 15 pings, and traces at unrealistic speeds. Although some of these traces could have potentially been map-matched, it should be noted that most of these traces were excluded for corresponding to movements outside the road network (2/3 of error records), so proper map-matching would be effectively impossible given the network used in this project.

Table 2: *MapMatcher* data quality errors

Error	Number of Traces	Error (%)
No GPS ping within 20m of a network link	53,342	20.5%
With less than 15 pings	17,662	6.8%
Too fast (130km/h for more than 2 minutes)	4,466	1.7%
Trace covering less than 500m	1, 837	0.7%
All pings matching to a single network point	1	0.0%
Total	77,308	29.7%

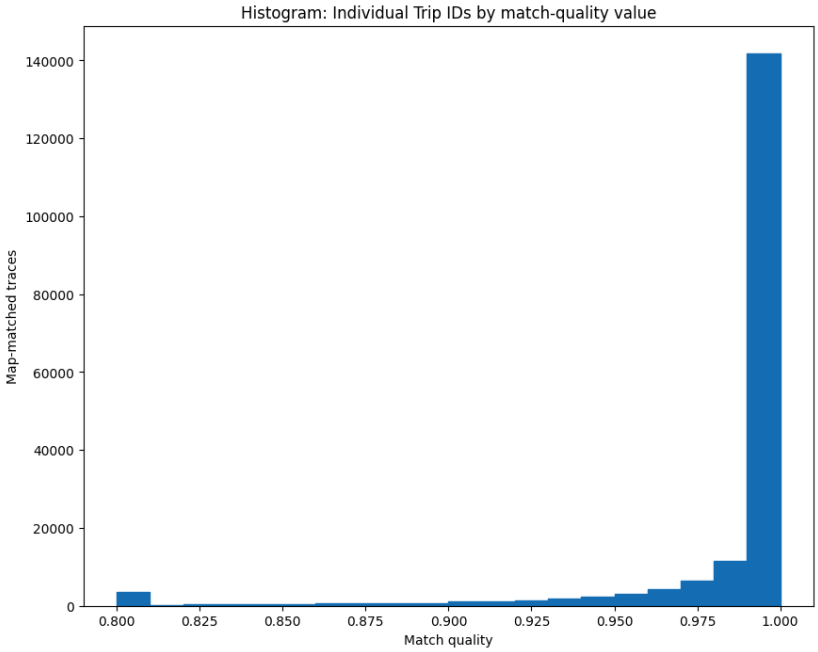
Looking in more detail at the remaining 182,639 traces, we have verified that nearly 15% of all its GPS pings (9,553,622 out of 64,198,273) were considered unmatchable, as they are more than 20m away from any link in the network. This adds to the data condition verified on Table 2.

5 Results and conclusions

For the aforementioned dataset, *MapMatcher* was run on a 32-core Threadripper 3970X workstation, resulting in an approximate 8h30min run time, or around 0.16s per GPS trace.

The results regarding the quality of the map-matching are illustrated in Figure 4, where we can see that there is a non-insignificant number of trips that have not been well map-matched.

Figure 4: Map match quality – individual trip ids



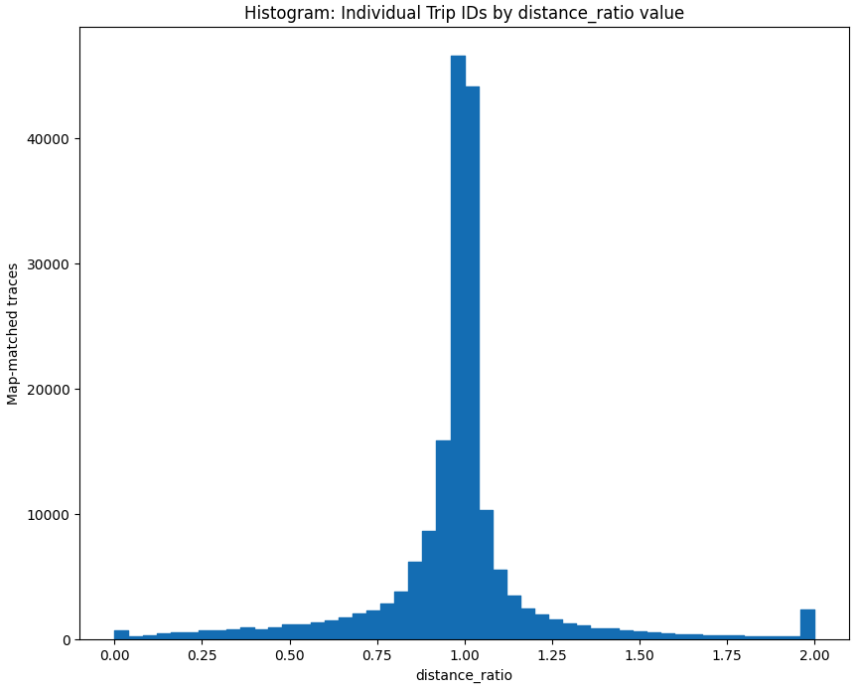
The key statistic, however, is that over 95% of the trips have had a match quality index of 90% or over, with 77.5% of the traces having had a match quality of 99% or higher.

Two considerations should be made when we analyse these results. The first one regards data quality, particularly the fact that the raw data contained an unquantified amount of GPS traces for vehicles apparently using busways and other limited-access infrastructure. The second consideration pertains to the intended use of the map-matching results, which are all centred around transport modelling tasks such as the estimation of route-choice algorithms, and for which much smaller samples are required.

We have also looked at the distance ratio between the map-matched distance versus the straight-line distance covered by each trace, and the results are shown on Figure 5 are of a tight distance ratio of around 1.0, pointing to consistent results. The figure also indicates that 87.6% of the map-matched traces have less than 10% variation in path length, while 96.7% have less than 50% variation in path length.

Although most traces were tightly map-matched, we see a small proportion of traces with a metric value that suggests further manual investigation of the data and the network may be required, particularly the roughly 2,200 traces with a distance ratio equal to or larger than 2.

Figure 5: Distance ratio for final matching result



In summary, *MapMatcher* produced excellent matching results for the large use case presented in this research, despite substantial issues with data quality. The tools developed during this research have been released as a well-supported open-source Python package available in all major platforms, with documentation available at https://aequilibrae.github.io/mapmatcher/.

6 Next steps

During the final stages of this research, 6 releases of *MapMatcher* have been published, incorporating several performance improvements and user control. However, two outstanding issues remain to be resolved.

The main issue is the pending incorporation of algorithms for the identification stops along GPS traces, allowing for the map-matching of full-day and multi-day GPS traces. Procedures described in more established literature are good candidates for this improvement (Camargo et al., 2017; Taghavi et al., 2019), while computational techniques can be borrowed from existing Open-Source packages (Graser, 2019).

The second pending improvement is a comprehensive comparison between *MapMatcher* and other map-matching solutions, particularly around software capabilities and their computational performance, although the literature suggests that Valhalla's performance might be comparable (Saki and Hagen, 2022).

7 References

Bernardin, V.L., Trevino, S., Short, J., Board, T.R., 2015. Expanding Truck GPS-Based Passive Origin-Destination Data in Iowa and Tennessee. p. 12p.

Camargo, P.V. de, Tok, A., 2014. Validating truck route enumeration algorithms using GPS data, in: 93rd TRB Conference Proceedings. p. 14.

Camargo, P., Hong, S., Livshits, V., 2017. Expanding the Uses of Truck GPS Data in Freight Modeling and Planning Activities. Transportation Research Record 2646, 68–76. https://doi.org/10.3141/2646-08

Graser, A., 2019. MovingPandas: Efficient Structures for Movement Data in Python. GI_Forum 2019, Volume 7, 54–68. https://doi.org/10.1553/giscience2019_01_s54

Kuppam, A.R., Lemp, J., Beagan, D.F., Livshits, V., Vallabhaneni, L., Nippani, S., 2014. Development of Tour-Based Truck Travel Demand Model Using Truck GPS Data, in: Transportation Research Board, 93rd Annual Meeting. Washington, DC, p. 27.

Liao, C.-F., 2009. Using Archived Truck GPS Data for Freight Performance Analysis on I-94/I-90 from the Twin Cities to Chicago. University of Minnesota Center for Transportation Studies.

Sahin, O., Stinson, M., Shen, Hui, Ismael, Abdelrahman, 2024. Analysis of Urban Freight Flows and Retail Goods Movement Using GPS Trajectory and Land Use Data. The 15th International Conference on Ambient Systems, Networks and Technologies Networks (ANT) and The 7th International Conference on Emerging Data and Industry 4.0 (EDI40).

Saki, S., Hagen, T., 2022. A Practical Guide to an Open-Source Map-Matching Approach for Big GPS Data. SN COMPUT. SCI. 3, 415. https://doi.org/10.1007/s42979-022-01340-5

Taghavi, M., Irannezhad, E., Prato, C.G., 2019. Identifying Truck Stops from a Large Stream of GPS Data via a Hidden Markov Chain Model, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). Presented at the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 2265–2271. https://doi.org/10.1109/ITSC.2019.8917156