Speed Aware – a Mobile App Prototype for the Promotion of Responsible Driving

Francesco Cremona, Adrian Muscat, Maria Attard
adrian.muscat@um.edu.mt

Abstract

This paper addresses the promotion and awareness of responsible driving and road safety through the development of a very simple to use mobile application prototype, *Speed Aware.* This application provides users with speed limit information on roads they are travelling on, together with a journey logging feature that allows off-line self-review of driving behaviour. Tracked journeys can be displayed on a map and the trace shown as a heatmap, comparing the vehicle speed to the road speed limit. Furthermore, an audible alarm is emitted whenever the vehicle is travelling at a speed higher than the legal limit. At the heart of this app is a map matching algorithm, which matches raw Global Positioning System (GPS) data to the road network. Five map matching algorithms are implemented and compared on the basis of real-time performance and accuracy. A ground truth dataset of GPS traces in dense, urban, and sub-urban environments, together with *TraceView*, a trace visualisation and management tool, were developed. A modified version of a weight-based topological algorithm achieved accuracy of 94.9% at a GPS sampling frequency of 1Hz. This algorithm, together with three of the reviewed map matching algorithms, were implemented on a mobile device and subjectively tested for real-time performance.

Keywords: Map-matching, GPS, Vehicle speed monitor, Safe driving, Mobile app

Introduction

Car owners' choice to drive at illegal speeds is the leading cause of grievous and fatal accidents, (Hoel & Garber, 2008). Less safe road conditions lead to people preferring to use their own vehicle instead of taking greener modes of transport, resulting in less cycling and walking (Road Safety Strategy Malta, 2014). The use of personal vehicles is increasing without bounds, with an average of 71 vehicles per day registered in the third quarter of 2017 (National Statistics Office, n.d.). This paper describes the development of a mobile app designed specifically to address irresponsible driving, by discouraging speeding over the legal limits and raising awareness on safer roads among the general public.

Real time navigational aids that include road network data and speed limit information are available on the Android Google Play Store, for example Speed Cameras & Traffic Sygic (Sygic, n.d.) and Velociraptor - Speed Limits & Speedometer (Ciao, n.d.). However, these features are either not simple to use (because the main application is navigation), not freely available (since they may be in-app purchase premium add-ons), make use of information that is incomplete for local maps, or have to be overlaid on another host app. On the other hand, *Speed Aware*, the app described in this paper, is independent of any other navigational app, very simple to use, does not require any configuration and is tailor-made for the local scene. Speed Aware makes use of Open Street Maps (OSM), Global Positioning System (GPS) technology and map matching algorithms to display speed information, as well as audible and visual alerts when the driver exceeds speed limits. Additionally, the app provides a data logging option.

GPS receivers determine their position by measuring distance to multiple satellites, using triangulation to find the position of the receiver. This position determination has multiple sources of errors, mainly due to clock synchronisation issues, atmosphere phenomena and multipath effects (Langley, 1997). The latter is of particular interest in this paper and is the main source of error in matching GPS points to maps. Therefore, the app makes use of a map matching algorithm (Bernstein & Kornhauser, 1996; Quddus et al., 2007) which greatly reduces the number of errors, especially in dense urban areas. To develop and finetune the algorithm, a ground truth dataset of GPS traces in dense, urban, and sub-urban environments, together with *TraceView*, a trace visualisation and management tool, were developed.

This section introduced the topic of this paper and explained its position in the literature. The rest of the paper is organised as follows. First, map matching algorithms are reviewed, followed with a description of the implementation and experimental setup used to study a selection of map-matching algorithms. The results are then discussed, followed with an account on the development of the app, *Speed Aware*, at which point the paper is concluded.

Map matching algorithms

The mobile app *Speed Aware* makes use of GPS and road network data to retrieve road information. In addition, map-matching algorithms are used to mitigate noise in GPS data, (Quddus et al., 2007; Bernstein & Kornhauser, 1996; Velaga et al., 2009). Once coordinates are obtained, the algorithms are used to identify the correct road segment on which a vehicle is travelling on (Greenfeld, 2002; Quddus et al., 2003) and provide a more accurate vehicle location estimate by supplementing the GPS data with spatial road network data. This section reviews various map-matching algorithms, including a comparison of their performance and complexity.

The earliest map matching algorithm for car navigation in real time was based on a simple geometry-based method (Kim et al., 1996). This was not accurate at intersections and parallel roads as it was dependent on the shape of the road segment (Hashemi & Karimi, 2014). Different techniques were subsequently developed, taking advantage of topological analysis of spatial road network data and comparing GPS bearing and position readings to road direction and position. Additionally, more advanced techniques based on probabilistic theory (Zhao, 1997), Kalman filters (Li et al., 2013), fuzzy logic (Quddus et al., 2006) and belief theory (Najjar & Bonnifait, 2005) were implemented. These vary in complexity, computational efficiency and matching accuracy. Hashemi and Karimi (2014) categorise map-matching algorithms as simple, weight-based and advanced, while Quddus, Ochieng and Noland (2007) group the algorithms into geometrical, topological, probabilistic and other advanced methods.

Map matching can be carried out on-line (real time) or off-line (post-processing) (Hashemi & Karimi, 2014). In real-time map matching algorithms, also known as online, GPS points are mapped on the go (Bernstein & Kornhauser, 1996; Velaga et al., 2009; Li et al., 2013). This means that as the mobile app receives the new updated location, it is immediately processed and set to a road segment. Off-line algorithms operate by first collecting GPS data over a journey, then passing all the points and vehicle movement information as a batch through the algorithm, outputting the solution (Newson & Krumm, 2009). This obviously brings advantages over a real time system in terms of accuracy, since all points preceding and following a certain point are known, making it easier to predict the point on the road where it was located (Marchal et al., 2004). On the other hand, a real-time system has only previous points to help in predicting the correct position, thus resulting in a less accurate but a potentially more responsive experience, enabling features such as real time over-speeding checks. Therefore, this paper considers real time algorithms. A review of the algorithms follow, taking into account the classification proposed by Quddus, Ochieng and Noland (2007).

Geometric

A geometry-based map-matching algorithm makes use of geometric information of the spatial road network data, considering only the shape of roads and does not consider the way roads are connected to each other (Greenfeld, 2002; Quddus et al., 2007).

Point-to-Point Map Matching (Bernstein & Kornhauser, 1996) is the simplest of such algorithms and matches the point to the closest node or shape point of a road segment. Its implementation is fast and easy, but has the drawback of being sensitive to the way the road network data is structured, as well as inaccuracies in urban environments. Curved roads with more than a start and end node are much more

likely to be chosen as the road on which the vehicle is travelling. Therefore, these multiple shape node roads produce errors (Quddus et al., 2007). The closest point is found by selecting the node with the least Euclidean distance from the GPS point, P_t. In practice, it is not necessary to determine the distance between the GPS point and every node in the network. Instead, one can first identify those nodes that are within a certain radius using a range query and then calculate the distance to them. Range queries are carried out by pre-processing *N* points in *k*-space and storing the points in an ordered tree. This leads to faster query times but has the drawback of higher storage use (Bernstein & Kornhauser, 1996; Bentley & Maurer, 1980).

Point-to-Curve Map Matching (Bernstein & Kornhauser, 1996) is an upgrade to the point-to-point technique and attempts to find the closest link to P_t instead of the nodes. To find the closest road, the minimum distance is found from the GPS point to all the road segments forming part of each road. This is because the roads are stored as piecewise linear curves made up of nodes and links. The shortest distance is taken as the perpendicular distance from the point to the road. If the perpendicular line does not intersect with the line segment, the shortest distance between the GPS point and the two end-point nodes of the road is taken. The road segment with the smallest distance is selected as the road. This, however, gives unstable results with increased road densities, and the closest link may not always be the correct link in dense areas.

Curve-to-Curve Map Matching (Bernstein & Kornhauser, 1996; White et al., 2000) constructs a linear piecewise curve using *n* previous GPS points, where *n=3*. The distance between the constructed curve and the surrounding road curves is measured and the closest curve is then selected as the road on which the vehicle is travelling. There are multiple ways of evaluating distances between curves. One way of doing this is to find the average of the closest Euclidean distances between the GPS points and the curve. Another curve to curve approach proposed by White et al. (2000) is to make use of the distance travelled between GPS points. This consists of finding the distance between equal lengths of the two curves. The last 3 GPS points are used to create a curve. The road is segmented into three segments with equal lengths as the GPS curve. Curve to curve distance is determined by the distance between the respective nodes of the curves.

Topological Topological map matching algorithms make use of spatial and historical data to aid in the matching of GPS points. Previously matched roads, road connectivity, and directional and turn restriction information are all parameters that have been used to supplement geometric algorithms (Quddus et al., 2007).

Weight-based topological map matching (Velaga et al., 2009) splits the matching problem into three parts and defines procedures that are used in each of these different situations. These are: (a) matching the first GPS point, (b) matching when traversing a road, and (c) matching at a junction. Weights are given to candidate roads depending on heading difference and distance from the GPS reading. At an intersection, a positive or negative weight is given to roads, whether or not they are

connected to the previous road, depending on whether there is a legal restriction to cross from the previous road to the candidate road. In urban environments, heading difference is prioritised over distance, which adds heavy weighting to connected roads and is not restricted to the previous road. Suburban environments have more accurate GPS readings, therefore heading difference and distance have similar weightings due to the point being closer to the correct road. Rural areas result in accurate GPS readings so that distance is given the greatest weight. Following is a description of the three procedures.

Matching the first GPS point:

a) Determine the candidate road segments: An error region is used to determine candidate segments. All segments that are present or passing through the region are chosen as candidates.
b) Determine the correct segment by giving a weight depending on heading weight, W_h, and proximity weight, W_p, where W_h is taken to be the cosine function of the difference between the road and vehicle bearings:

$$W_h = H_W f(\vartheta_r, \vartheta_v), \qquad where \qquad f(\vartheta_r, \vartheta_v) = cos(\vartheta_r - \vartheta_v).$$

Here, H_W is the weight coefficient for the heading information's importance, ϑ_r and ϑ_v are the road and vehicle's bearings, respectively, and W_p is the weight based on the perpendicular distance from the road segments and is computed as,

$$W_p = D_w f(D), \qquad where\ f(D) = [(80\text{-}D)/80],$$

D_W is the weight coefficient for the road's proximity and D is the perpendicular distance from the point to the road. The total weight score (TWS) determines which candidate segment will be chosen and consists of the addition of the two heading and direction weights.

Matching when traversing a road:

a) If the vehicle's speed is 0, the same road segment as before is assumed.
b) If the vehicle is moving, the algorithm checks whether or not there is an intersection ahead. Two checks are put in place: (i) whether the current point is within 20m of an upcoming intersection, and (ii) whether the heading difference between the vehicle and the road is greater than the root mean square of all the previous errors on the same road. If one of these is found to be positive, then it is deduced that the vehicle is at an intersection. Otherwise, the GPS point is perpendicularly projected onto the previous road.

Matching at a junction:

This process is the same as the matching for the first GPS point using the total weight score to select a road. However, two additional weights are used: (i) If a vehicle approaches a junction and is not legally permitted to turn to one of the road segments, directional knowledge is used to then give the candidate road less turn restriction weight. The weight is positive for a legal turn, and negative for an illegal turn. (ii) A road segment is given more weight if it is directly connected to the previous road segment, known as the link connectivity weight. The weight is positive for a connected road and is negative for a road that is not connected to the previous road.

In the Hashemi and Karimi (2014) review of real-time map-matching algorithms, it was concluded, after taking into account its accuracy, simplicity and performance, that the Velaga algorithm (Velaga et al., 2009) was better than all the other examined geometric, topological and weight-based algorithms, having outputs which compete with the results obtained from advanced algorithms. The Velaga algorithm achieved a 96.71% matching accuracy when tested. Nevertheless, some drawbacks were identified. Firstly, the direction difference between consecutive GPS points is not considered. Secondly, skewed results are provided if a vehicle passes a short segment between consecutive GPS points, when illegal turns are made and in off road conditions. Lastly, the weights were derived empirically, which is less desirable than if they were derived analytically.

Advanced map matching algorithms

Advanced map matching algorithms are more complex processes that make use of concepts such as Kalman Filters (Kim et al., 2000), Fuzzy Logic (Quddus et al., 2006) and Hidden Markov Models (Newson & Krumm, 2009). Goh Dauwels, Mitrovic, Asif, Oran, and Jaillet (2012) use an HMM with a variable sliding window to compute transition probabilities on a real time basis or delay by one output if the probability calculated is below a threshold. The sliding window works by eliminating nodes in the calculation that were received before a certain timestamp, thus being able to efficiently compute probabilities with just the latest points. These advanced algorithms add a level of computational complexity and resource usage to the process, causing more battery consumption and heating up of devices when used as a mobile app.

Algorithm comparative Study

This section describes the comparative study carried out to determine which

algorithm is implemented in the app. Factors considered were accuracy of data and delay in alerting the user. Point-to-point, point-to-curve, curve-to-curve and the weight-based topological map matching algorithms were implemented. Output accuracies were examined with respect to different GPS sampling rate and urban density. In theory, geometric map-matching algorithms' accuracy is not affected by different GPS sampling rates, since only the most recent point is used. On the other hand, the accuracy of Topological map-matching algorithms is dependent on the GPS sampling rate.

Road network data

Map matching algorithms require a map data source to be able to relate GPS retrieved data to the road network. Google Maps (GM), (Google Maps Platform, n.d.), and Open Street Maps (OSM), (Open Street Map, n.d.) were considered for selecting the most suitable map data provider. Open Street Maps was chosen, since it is a crowd sourced, open source map database with the possibility of adding speed limit information. Two files were created from the OSM data. One was created for the querying of road network data by the algorithms via a SQL Spatialite database, and another in '.map' format which was used to display the map on screen. This enables the mobile app to function without the need of an internet connection.

Firstly, the algorithm queries all OSM nodes that are marked as highway or road nodes within a radius of 290m, so that it retrieves a list of all the candidate roads. This radius was determined by finding the longest single road segment within the OSM data. The radius used accommodates for the vehicle being in the middle of the longest road, at around 283m equidistant from the start and end nodes of the road. Roads are then filtered to only those within a radius of 160m, as determined by Velaga, Quddus and Bristow (2009). The algorithms were implemented in Java and SQL with the Spatialite plugin, which is an environment compatible with the Android platform.

Implementation of Map matching algorithms

1) *Point-to-Point:* An SQL Spatialite query is executed to sort road nodes around the GPS point in descending order by distance from the current point, and the shortest one is chosen. The OSM way in which the node is part of is chosen as the road on which the vehicle is travelling. Since intersection nodes have more than one way associated with them, priority is given to roads that have names over those that don't, otherwise, the way is chosen randomly. A curve is constructed using all the nodes within the way, and the GPS point is then projected onto the closest point on the road.

2) *Point-to-Curve:* A curve is constructed for each road within the 160m radius and the shortest distance from the point to the curve is calculated for each road. The distances are then ordered in ascending order and the road with the shortest distance is chosen. The GPS point is then projected to the closest point on the curve.

3) *Curve-to-Curve:* Two approaches are used to calculate the distance between two curves. The first point is matched using a point to curve technique while the second point is matched by finding the average minimum distance from the 2 points to the roads.
   a) The first approach to curve-to-curve matching is to find the minimum distance from the last 3 points to each road, and the average of these 3 values is found. The road with the shortest average distance is chosen, and the GPS point is projected to the closest point on the road.
   b) The second approach to curve-to-curve matching was implemented by finding the average of the distances between equal lengths of the curves formed from the road and the GPS trace. The line constructed from 3 GPS points is made up of two segments. A point on the line is perpendicularly projected from the oldest GPS point, and a curve is constructed starting from that point, with an equal length of the GPS trace. The distance between each respective node of the two lines is calculated and the average taken. The road with the lowest average distance is chosen.

4) *Weight-based topological map matching:* In this algorithm the heading difference weighting is calculated differently for one-way and two-way roads. The heading weighting's cosine function is only used as described when a road is one-way. When a road is two-way, the absolute value of the cosine function between the heading difference of the road and vehicle is used, due to such roads having two bearings, the one calculated and its supplementary angle. The heading differences for each matched point is saved in order to calculate the root mean square for each road. Once a point is matched to a new road, this list is cleared. In some cases, the point is matched to roads which have no intersections. In these cases, it is assumed that the point is always near a junction, since if this precaution isn't taken, then the point would remain on the road, since it is impossible for it to ever be near a junction and would therefore be continuously matched to the same road.

Visualisation Tool and Dataset

TraceView was created to visualise GPS traces. All nodes that form part of the road network are displayed as red nodes on the map. This serves as a visual aid when

comparing different map matching algorithms and the relationship between the GPS points and the surrounding road nodes. Fig.1 shows TraceView's main interface.

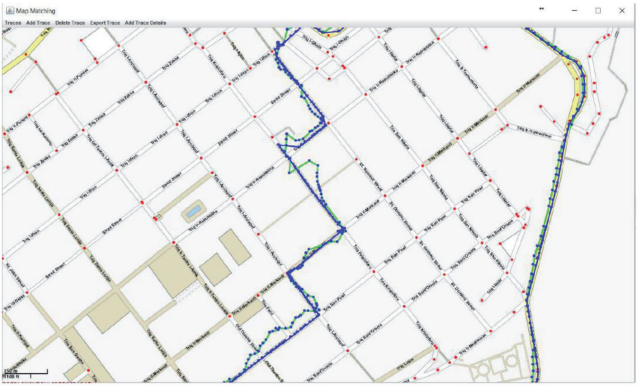


Figure 1. TraceView tool displaying traces and road nodes

An SQL table is used to store details on the GPS points and traces, with the related fields describing the trace they belong to; predicted road, expected road, latitude, longitude, time, speed, bearing, binary variable indicating if the match is correct, binary variable indicating if the point should be shown and road density around the point. TraceView also provides the option to add or export traces from/to CSV files and to annotate traces and their related GPS points. Ground truth road annotations allow for an accuracy statistic to be calculated; this is derived by comparing the ground truth road to the road predicted by the map matching algorithms. Each GPS point is manually labeled with the density of the area it is situated in, facilitating testing of the map matching algorithm in different environments. Densities are grouped in four categories; open spaces, low density rural roads, medium density residential roads and high density roads with 4+ floor buildings.

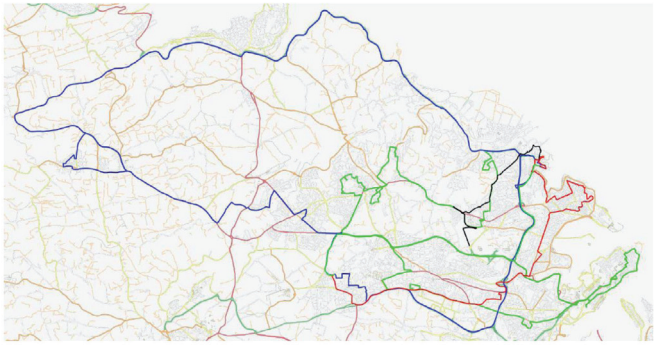


Figure 2. Example traces from the dataset

A dataset of GPS traces was developed for the comparative analysis study, where the performances of the various algorithms are compared in different urban environments. An Android app was developed to collect the GPS traces at a frequency of 1Hz. The collected 13993 GPS points were spread out across the different densities, distributed as follows: 32% in open areas, 24% in low, 27% in medium and 17% in high density areas. Using TraceView, the GPS traces were manually labeled with the ground truth road name and area density. Fig.2 depicts part of the dataset.

Results and Analysis

This section includes the testing of the implemented map matching algorithms and the comparison of their results at different sampling intervals and environments. TraceView was used to evaluate the algorithms' performance. The frequency was down sampled to time intervals of 2s, 4s and 8s by skipping GPS readings at random intervals, with interval ranges that average the required interval.

Point-to-point is the simplest algorithm, providing the lowest performance. Table 1(a) shows the accuracies of point to point at different intervals and densities. Since this algorithm, makes use of only one reading, intervals don't have a large effect on accuracy. Open areas provide the highest performance for this algorithm due to the presence of less neighbouring nodes. Low and medium density reduce the accuracy further due to increased road nodes. Denser areas result in higher GPS errors. However, in such areas, there may be shorter roads and more nodes/distance, resulting in better overall scores than Low and Medium density areas. Table 1(b) shows an example from TraceView, where the blue trace consists of the raw GPS points while the green trace is the map matched trace.

Point-to-curve, like point-to-point, makes use of the most recent GPS point, therefore the interval between readings has little effect on accuracy. Results are tabulated in Table 1(c). In an open area, the result has the highest accuracy due to better GPS accuracy combined with less roads in the area. As the density increases, both the GPS error and the number of roads in the region increase, resulting in an accuracy of 79.5% in urban areas, which is significantly better than point-to-point. Table 1(d) depicts the improvement over point-to-point matching along the road, only for point-to-curve to fail at the junction.

Curve-to-curve has reduced performance as intervals increase, Tables 1(a, b), since the vehicle could have travelled between roads while readings are being processed. The algorithm is similar to the point-to-curve algorithm, but with an error smoothing effect achieved by using multiple points. Once a vehicle has passed an intersection, the three points used to calculate the distances may be spread across different roads. This is especially the case at higher intervals. The two approaches to calculate distances in curve-to-curve matching have similar accuracies.

Table 1: (a) Point to Point, and (b) Point to curve results (% accuracy)

Int	Open	Low	Medium	High	Overall
1s	75.3	68.6	62.1	64.2	68.2
2s	75.6	67.0	62.1	64.2	68.0
4s	74.5	65.0	62.7	65.8	67.5
8s	74.5	66.3	66.9	61.4	68.2

(a)

Int	Open	Low	Medium	High	Overall
1s	94.3	93.4	89.2	79.5	90.0
2s	94.3	90.5	89.1	80.1	89.4
4s	93.8	91.7	88.9	78.8	89.3
8s	93.9	92.2	90.3	77.0	89.3

(c)

(b)

(d)

Table 2. (a) Curve to Curve (Distance between equal segments, % accuracy), (b) Curve to Curve (Closest distance, % accuracy)

Int	Open	Low	Medium	High	Overall
1s	95.1	93.7	90.2	82.8	91.2
2s	93.6	89.3	88.8	85.8	89.8
4s	90.2	84.3	83.4	81.2	85.3
8s	84.1	71.0	77.9	76.3	77.7

(a)

Int	Open	Low	Medium	High	Overall
1s	94.9	93.7	89.5	81.5	90.7
2s	94.0	89.7	88.9	84.7	89.9
4s	91.0	86.2	84.0	83.3	86.5
8s	85.7	81.4	73.1	77.1	79.7

(b)

Weight-based Topological Algorithm proposed by Velaga, Quddus and Bristow (2009) provides different weightings for urban and suburban environments. On the local dataset and in open areas, the urban weighting resulted in a 94.8% score, but accuracy degrades in dense areas, 75.2%. This is due to the number of roads in the area and the variation in road curvature. This means that at an intersection, if there is an error in the GPS heading information, then it is likely that the matching jumps to a nearby road with a similar heading. The algorithm checks if a junction is within 20m distance. As the interval between readings increases, the chance of missing a junction increases. This means that the point is projected onto the previous road. With urban weights at 1s interval the overall score is 85.3%. Suburban weighting results in a better overall score, 93.5%. Intervals have less of an effect on the accuracy, with an overall 90.7% correct matches compared to the urban weighting's 78.3% at an interval of 8s. This algorithm is the most accurate, having an overall accuracy of over 90.7% for all intervals.

In some situations where the vehicle is travelling at a relatively high speed, junctions are not detected. Therefore, when the conjunction condition (vehicle is assumed to be traversing a road, not close to a junction and the projected point is more than 160m away) is detected, it is assumed that a junction has just been passed. This change brought about a noticeable improvement in accuracy at higher intervals, shown in Tables 3(c, d), since junctions are more likely to be missed at these frequencies. Additionally, the overall frequency at 1Hz also improved from 93.5% to 94.4% for the suburban weights, and from 85.3% to 85.6% for the urban weights.

Table 3. Percentage accuracy for the Velaga Algorithm with (a) Urban weights, (b) Suburban weights, (c) Suburban weights and 160m check, (d) Urban weights and 160m check.

Int	Open	Low	Medium	High	Overall
1s	94.8	82.9	82.9	75.2	85.3
2s	94.0	81.3	86.1	79.3	86.3
4s	90.9	78.8	78.9	73.2	81.7
8s	88.1	76.5	72.4	71.8	78.3

(a)

Int	Open	Low	Medium	High	Overall
1s	91.9	97.5	94.9	89.6	93.5
2s	92.0	93.7	95.5	89.3	92.8
4s	90.6	93.1	94.7	86.1	91.4
8s	89.2	93.0	96.2	85.9	90.7

(b)

Int	Open	Low	Medium	High	Overall
1s	96.8	94.7	94.8	89.6	94.4
2s	96.5	93.9	94.3	88.9	93.9
4s	96.1	92.9	94.8	87.1	93.3
8s	95.4	93.6	94.7	88.2	93.4

(c)

Int	Open	Low	Medium	High	Overall
1s	95.6	81.3	82.5	77.9	85.6
2s	94.1	83.7	83.8	76.9	85.8
4s	93.3	80.3	85.1	77.4	85.1
8s	87.5	73.6	80.4	68.5	79.0

(d)

Modified Weight-based Topological Algorithm: The Velaga algorithm was modified to lower its computational complexity. The routine is executed as if it was always at an intersection instead of checking for a nearby intersection. The turn restriction check is removed, as this data on OSM is lacking and adds computational time for little effect. If the speed is 0, then the point is still matched to the previous road. The Suburban weights provided a higher accuracy than the Urban weights. The Urban weight gives the heading a higher weight, while the Suburban gives an equal weighting to distance and heading. For this modified algorithm, the heading weight was therefore reduced to achieve the weights given in Table 4(a). Additionally, the modified algorithm provided the best results out of all the algorithms, having negligible 1.3% accuracy loss when changing the interval from 1s to 8s.

Table 4. Modified Velaga Algorithm (a) Modified weights, (b) Percentage accuracy

Parameter	Weighting
H˙w	36.24
D˙w	44.99
C˙w	4.46
T˙w	4.31

(a)

Int	Open	Low	Medium	High	Overall
1s	96.9	94.1	95.5	91.9	94.9
2s	96.5	91.0	95.9	91.9	94.1
4s	95.5	90.9	95.6	89.1	93.3
8s	95.1	94.4	94.9	88.4	93.6

(b)

Summary: Table 5 displays the overall accuracies of all the algorithms studied. The accuracies are displayed for each frequency tested.

Design Consideration for the Mobile App

The prototype mobile app, *Speed Aware*, assists drivers in observing speed limits whilst driving. The app informs users on whether they are under-speeding, over-speeding or driving at the right speed, which arbitrarily is defined as the 10% range below the speed limit. The map matching algorithm is to be used to match the GPS point to the road network, from which the speed limit information is obtained. Fig.3 depicts the system in a flow-diagram.

Table 5: Percentage accuracies for all the algorithms

Algorithm	1s	2s	4s	8s
Point to Point	68.2	68.0	67.5	68.2
Point to Curve	90.0	89.4	89.3	89.3
Curve to Curve 1	91.2	89.8	85.3	77.7
Curve to Curve 2	90.7	89.9	86.5	79.7
Suburban weighted	94.4	93.9	93.3	93.4
Urban weighted	85.6	85.8	85.1	79.0
Modified weighted	94.9	94.1	93.3	93.6

The choice of the optimal map matching algorithm is based on accuracy and delay, which depend on sampling rate and computational complexity. The average time taken from when a GPS point is collected to the point being matched to the road network was measured (Table 6). The processing time, t_{proc}, is a function of the number of roads sampled. If the modified Velaga algorithm was to be selected as the algorithm, then the GPS interval would have to be set at 2s, since the average t_{proc} is 1504ms. The total delay is then $2s + t_{proc}$. A delay of 3.5s is not suitable for the app and therefore the point-to-curve algorithm is chosen for the app, which has the lowest t_{proc}=951ms. The time interval between GPS points is set to the processing

time of the previous GPS point, adding an extra 5% margin to prevent stalling of the app. If 105% of t_{proc} is 1000ms or lower, then the interval is set at 1000ms.

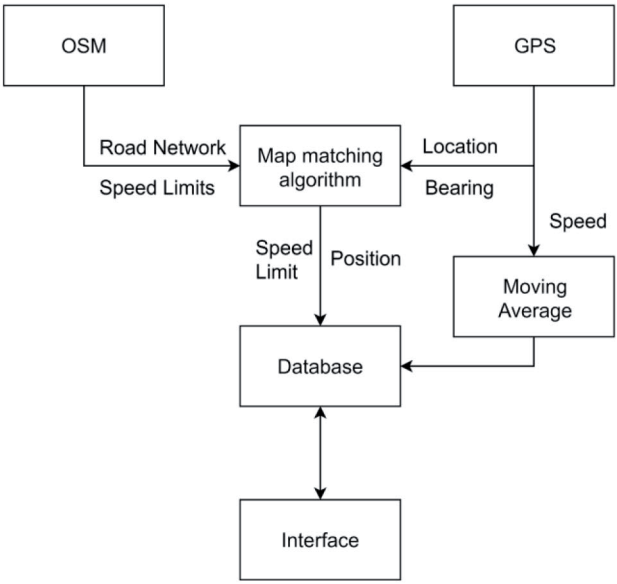


Fig 3. The system diagram for the prototype mobile app, speed aware.

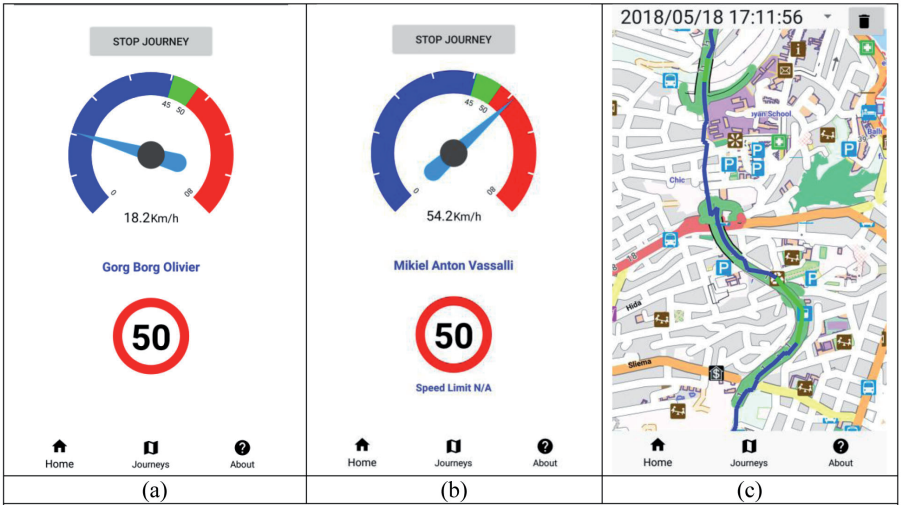
Table 6: Map Matching Algorithm delay

Algorithm	Processing delay t_{proc} (ms)
Velaga *et al.* (2009) with Suburban weights	2365
Modified Velaga *et al.* (2009)	1504
Point to curve	951
Curve to curve	2236

The mobile interface was designed to be as simple as possible, and usable by anyone. The application was split up into 3 pages; (a) Home: The home page displays speed, speed limit, a start and stop button, and the correct speed or over/under-speeding ranges. When over-speeding, a beeping alarm is sounded; (b) Journeys: The page in which the users can access their past journeys. Traces are coloured depending on driving performance compared to the speed limit, being clickable to view more detailed information on the speed and limit; (c) About: Displays terms of service, use, privacy policy and contact information. Figure 4 gives examples of the pages.

Speed limits are obtained from Open Street Maps. If the value is not found, the speed limit defaults to 50km/hr and the user is notified. Short term variations in GPS speed readings are smoothed out and, after comparing different window sizes, a steady change in speed is achieved with a 4 second window. All GPS points from the last 4 seconds are used, adapting to algorithm delay.

Fig.4. The mobile app display, (a) Home interface, (b) Notification shown when speed limit defaults to 50km/hr , (c) Saved journey viewer



Conclusion and Future Work

In this project, a review of map matching algorithms was carried out to compare various proposed geometric and topological algorithms. Geometric and topological algorithms map matching algorithms were implemented, together with a modified version of the Velaga et al. (2009) algorithm. The highest accuracy of 94.9% was obtained from the modified weight-based topological algorithm.

A visualisation and annotation tool, *TraceView*, was developed and used to assemble a labelled dataset of GPS traces obtained from different urban environments. Advanced algorithms that use Hidden Markov Models, for example, could be considered as they provide better results in terms of accuracy, at the cost of having larger computational and implementational complexity. The feasibility of running such algorithms on phones could be tested.

An Android app, *Speed Aware*, was developed, giving users road speed information in real time. Additionally, saved journeys are displayed on a map and the trace is shown as a heatmap, indicating under-speeding or over-speeding and neutral, and a beeping alarm alerts the driver when speeds exceed the limit for that

particular road. The map matching algorithms were tested in the app to determine which is the most suitable algorithm to use, that is, the one having accurate speed information with the least delay. The most accurate algorithm was found to have an average processing time of 1504ms. On the other hand, the point-to-curve algorithm has a processing time of 951ms, which translates into more timely road updates. The GPS reading interval adapts to 105% of the time taken for the previous point to be matched to the road network, to prevent the app from stalling. A smoothing filter was used to smooth out short term sharp changes in the speed reading. This filter was set to a moving window of 4s, which, at a 1Hz GPS frequency, takes 4 readings, but when the intervals increase, the number of readings decrease so that a 4 second window is never exceeded.

In the testing of the accuracy of correct road identification, the original GPS traces were manually labelled with the road name on which the vehicle was actually driving on. At intersections, there is some ambiguity on which road should the point be matched, by matching a GPS point which is in the middle of an intersection to the road which wasn't labelled. As an improvement, the road annotation feature could be changed to multi-label, so as to provide two road names at intersections, and if the matched road is one of the two, the matching is set as correct.

The algorithms were tested using a mix of Java and SQLite queries with the functionality of the Spatialite SQL plugin for geometric functions. It should be examined whether better data structures and schemas could be used to reduce the computational complexity of the algorithms.

Speed limits are derived from OSM data which are updated in a crowdsourced manner. During testing, it was determined that some road speed limits are not up to date or missing. For the app to provide more accurate information and be more reliable, complete speed limit information should be uploaded to the open source OSM database. Currently, the speed limit defaults to 50km/hr when no data is available for the road, which may be high for residential roads.

References

Bentley, J. and Maurer, H. (1980). Efficient worst-case data structures for range searching. *Departments of Computer Science and Mathematics, Carnegie-Mellon University*.

Bernstein, D. and Kornhauser, A. (1996). An introduction to map matching for personal navigation assistants. *New Jersey TIDE Center Technical Report*.

Ciao, D. (n.d.). Velociraptor - Speed Limits Speedometer. Available at https://play.google.com/store/apps/details?id=com.pluscubed.velociraptor [Accessed 25 April 2018].

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

Google Maps Platform *(n.d.).* Maps SDK for Android. Available at https://developers.google.com/maps/documentation/android-sdk/intro [Accessed 15 February 2018].

Greenfeld, J. S. (2002). Matching GPS observations to locations on a digital map.

Hashemi, M. and Karimi, H. A. (2014). A critical review of real-time map-matching algorithms: Current issues and future directions. *Computers, Environment and Urban Systems*, 48, 153–165.

Hoel, L. A. and Garber, N. J. (2008). Traffic and highway engineering, course technology. *Cengage Learning*.

Kim, J. S., Lee, J., Kang, T. H., Lee, W. Y. and Kim, Y. G. (1996). Node based map-matching algorithm for car navigation system. *Proceedings of 29th international symposium on automative technology and automation (ISATA),* 121–126.

Kim, W., Jee, G. and Lee, J. (2000). Efficient use of digital road map in various positioning for its. *IEEE Symposium on Position Location and Navigation*.

Langley, R. B. (1997). The GPS error budget. *GPS World*, 8, 51–56.

Li, L., Quddus, M. and Zhao, L. (2013). High accuracy tightly-coupled integrity monitoring algorithm for map-matching. *Transportation Research Part C: Emerging Technologies,* 36, 13–26.

Marchal, F., Hackney, J. and Axhausen, K. W. (2004). Efficient map-matching of large GPS data sets - tests on a speed monitoring experiment in Zurich. *Technical Report,* 244.

Najjar, M. E. E. and Bonnifait, P. (2005). A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Journal of Intelligent Transportation Systems*, 19(2), 173-191.

National Statistics Office (n.d.). News release: Motor vehicles: Q3/2017. Available at https://nso.gov.mt/en/News_Releases/View_by_Unit/Unit_B3/Environment_Energy_Transport_and_Agriculture_Statistics/Documents/2017/News2017_168.pdf [Accessed 19 November 2017].

Newson, P. and Krumm, J. (2009). Hidden Markov map matching through noise and sparseness. *GIS '09 Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* 336–343.

Open Street Map (n.d.). Available at https://www.openstreetmap.org/ [Accessed 15 February 2018].

Quddus, M. A., Noland, R. B. and Ochieng, W. Y. (2006). A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems,* 10(3), 103–115.

Quddus, M. A., Ochieng, W. Y. and 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,* 15(5), 312–328.

Road Safety Strategy Malta (2014), *Ministry for Transport and Infrastructure*.

Sygic (n. d). Speed cameras traffic sygic. Available at https://play.google.com/store/apps/details?id=com.sygic.speedcamapp [Accessed 25 April 2018].

Velaga, N. R., Quddus, M. A. and Bristow, A. L. (2009). Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies,* 17(6), 672–683.

White, C., Bernstein, D. and Kornhauser, A. L. (2000). Some map matching algorithms for personal navigation assistants. *Transportation Research Part C Emerging Technologies,* 8(1-6), 91-108.

Zhao, Y. (1997). Vehicle location and navigation system. *Artech House, Inc., MA.*

Bio-notes

Professor Adrian Muscat is Associate Professor at the Department of Communications and Computer Engineering, University of Malta. His research interests are in the application of pattern recognition models, discrete event simulation and optimization applied to image understanding - with special emphasis on semantic relations in images, and transport - mainly the study and development of demand responsive transport systems.

Francesco Cremona was awarded a BSc in Computer Engineering by the University of Malta. His main research interests and experiences are in the application of machine learning algorithms and the development of mobile apps. He is currently employed as a software engineer in the FinTech industry and focuses on blockchain technology.

Professor Maria Attard is Head of Geography and Director of the Institute for Climate Change and Sustainable Development at the University of Malta. She studied at the University of Malta and completed her PhD in 2006 at UCL (London) and has published in the areas of urban transport, planning and policy. She is Co-Editor of Research in Transportation Business and Management, Associate Editor of Case Studies in Transport Policy and co-editor of the Emerald Book Series on Transport and Sustainability.