 https://doi.org/10.1109/TITS.2020.3031080

# Map Matching and Lane Detection Based on Markovian Behavior, GIS, and IMU Data

Jens Trogh, Dick Botteldooren, Bert De Coensel, Luc Martens, Wout Joseph, and David Plets
Department of Information Technology, Ghent University - IMEC, Belgium, jens.trogh@ugent.be

***Abstract*—This paper presents a fast, memory-efficient, and worldwide map matching algorithm based on raw geographic coordinates and enriched open map data with support for trajectories on foot, by bike, and motorized vehicles. The proposed algorithm combines the Markovian behavior and the shortest path aspect while taking into account the type and direction of all road segments, information about one-way traffic, maximum allowed speed per road segment, and driving behavior. Furthermore, a self-adapting lane detection algorithm based solely on accelerometer readings is added on top of the map matching algorithm. An experimental validation consisting of 30 trajectories on foot, by bike, and by car, showed the efficiency and accuracy of the proposed algorithms, with an average F1-score and median error of 99.5% and 1.89 m for the map matching algorithm and an average F1-score of 86.7% for the lane detection algorithm, which resulted in the correctly estimated lane 93.0% of the time. Moreover, the proposed technique outperforms existing state of the art techniques with accuracy improvements up to 45.2%.**

***Index Terms*—Map Matching, Lane Detection, GPS, Accelerometer, Sensor, Data Fusion**

## I. Introduction

Map matching is the problem of how to link recorded geographic coordinates to a road network of the real world, which is usually represented by a geographic information system (GIS). A typical example of map matching is to determine the trajectory of a moving object based on GPS data due to the nearly ubiquitous availability of the GPS signal [1]. The moving object can be a vehicle, cyclist, or runner, and the input GPS signals can be augmented with the data of an inertial measurement unit (IMU) to improve the accuracy or to lower the GPS update frequency and hence reduce the power consumption [2]. Other applications of map matching are, e.g., satellite navigation, freight tracking, activity recognition, road usage patterns, intelligent transportation systems (ITS), or urban traffic modeling. An example of the latter is the detection of problematic traffic points to optimize traffic flow in a city or to build statistical models of traffic delays that can be used to give suggestions to avoid traffic jams by finding the time-optimal driving route.

Map matching algorithms can be divided in real-time (online) and non-time critical (offline) algorithms. Real-time systems map the location to a road network during the recording process, whereas offline techniques are used to map geographic coordinates to a road network after all data has been recorded, which generally results in a better accuracy at the cost of a delay.

The most basic approach is to map each geographic input coordinate to the nearest point on the road network (snap-to-road), but due to measurement noise this can easily result in a wrong reconstruction. The noise in GPS data is usually caused by the urban canyon effect, tunnels, or terrestrial features that affect the GPS signal, e.g., hills, forests, or valleys. Figure 1 shows an example of the typical problem with a basic nearest neighbor algorithm that maps the GPS point to the closest grid point on a road network, resulting in an unrealistic and physically impossible trajectory. Note that due to the urban canyon effect, the error between the raw GPS points and ground truth trajectory is up to 53 m in this real-world example.


Figure 1: Map matching problem: nearest grid point

This basic approach is known as point-to-point mapping if the road network is discretized in a grid [3]. Note that it is called point-to-curve mapping if the road network is represented by a continuous set of lines. A disadvantage of these techniques is that historical data is not taken into account. Curve-to-curve map matching is a geographic approach that uses a sequence of input coordinates to form a smoothed curve that is matched to a road segment with a similar geometry [4]–[6]. Disadvantages are its sensitivity to measurement noise and the sampling rate, e.g., if only once every minute a GPS sample is available to limit the power consumption in devices with limited battery capacity, then the curve based on the input coordinates can highly deviate from the real trajectory unless a user is driving on a straight road segment. Another problem is shown in Figure 2, the reconstruction is physically possible but the ground truth trajectory is more likely from

a typical driver's point of view. However, the reconstructed trajectory can be the most likely in a map matching algorithm, e.g., if the objective is to minimize the Euclidean distance between the geographic input coordinates and the mapped trajectory. Merely minimizing a distance-based metric can result in unnecessary loops, U-turns, and overall weird driving behavior.

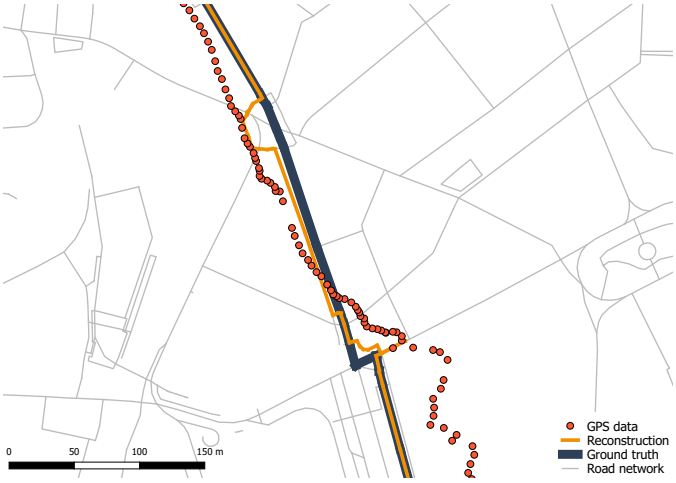



Figure 2: Map matching problem: minimizing a distance-based metric

The map matched geographic coordinates can be enriched with driving lane information if additional data is available, e.g., monocular or stereo vision [7], [8], LIDAR (light detection and ranging) [9], or data from an IMU [10]. This is useful for accurate road surface monitoring or to obtain lane-specific statistical models about driving behavior. Note that the IMU data also allows estimating human drivers' behavior, e.g., speeding, swerving, hard braking, or maneuvers, which is useful for, e.g., insurance companies, mobility operators, or to prevent potential car accidents [11], [12]. Furthermore, the map matching is the first step in a lane detection algorithm, since the type of road, number of available lanes, and passage of highway ramps are necessary for techniques that are not based on vision.

The remainder of this work is structured as follows, Section II describes related work of map matching and lane detection techniques. Section III discusses the grid, road segments, and proposed map matching algorithm and Section IV introduces the proposed lane detection technique. Section V describes the evaluation trajectories and performance metrics, and discusses the results of the simulations and experimental validation. Finally, in Section VI, conclusions are provided.

## II. Related work

### A. Map matching

Advanced map matching algorithms are often based on the Hidden Markov Model (HMM) [1], [13], [14]. These systems can model the road infrastructure and take into account measurement noise and many different path hypotheses simultaneously. A prerequisite for HMMs is to model the observation and transition probabilities, i.e., to select candidate roads for the GPS observations. An HMM-based map matching solution that uses a route choice model estimated from real-world drive data to evaluate the paths generated by the HMM is presented in [14]. The route choice model is based on the concept of drivers' route and is used to avoid unreasonable paths generated by HMMs for highly noisy geographic data. The path-related parameters for this model are estimated based on a 1000 drives.

A conditional random field (CRF) is a type of undirected graphical model that is used to encode known relationships between observations, e.g., to segment and label sequential data [15]. In [16], a CRF is used to map match GPS trajectories at a low sample rate based on floating car data (FCD) with GPS trajectories of 70 taxis from one day. The dataset is split in training and test data (ratio 7:3) to select features, fine-tune hyper parameters, and evaluate the approach. A disadvantage of these data-driven techniques is that they depend on the quantity and quality of available historical data to improve the map mapping accuracy and hence will fail if parts of the input trajectory are not present in the dataset. A multi-track technique that simultaneously matches a collection of trajectories to a map is presented in [17]. This multi-track map matching is based on the observation that human drivers show a high degree of temporal and spatial regularity [18]. They exploit the regular structure in a large set of GPS traces by enforcing a set of partially overlapping trajectories to coincide their mapped paths in the intersection regions.

The basic assumption underlying HMM-based map matching algorithms is that the true mobility is Markovian [19]. However, in [20], it is argued that mobility is non-Markovian based on the Chapman-Kolmogorov equation [21] and a dataset with thousands of real taxicab rides spanning several weeks. The results show that there is a strong connection between the shortest path and the real trajectory traversed by the moving objects. Their proposed technique relies exclusively on shortest path computations and could achieve an improvement of 20% in accuracy over HMM-based approaches but this is largely due to the nature of the dataset, i.e., the moving objects (taxicabs) have intent to reach a specific destination in a timely manner, which is inherently suited for shortest path algorithms.

Furthermore, most map matching algorithms are evaluated by calculating the overlap between the ground truth path and the estimated path. This means that even if both paths completely overlap, the exact time at which a vehicle is at each road segment can be unknown or differ due to traffic delays, different road speed limits, unpredictabilities, or randomness in the driver behavior. This is disadvantageous for applications that need the exact location on the road network at each time instance, e.g., autonomous driving, precise registration of telematics data [22], pothole identification [23], or lane detection [10].

### B. Lane detection

The problem of lane detection is usually tackled with vision-based techniques. These vision-based lane detection algorithms are categorized into feature-based and model-based. A feature-based algorithm uses low-level features, e.g., the

solid or dashed painted lines on public roads, and image segmentation [24], deep learning [24], [25], or sensor fusion [26] to detect the lanes. A model-based approach uses a few parameters to represent the lanes, e.g., straight lines or parabolic curves, these parameters can be estimated by a Hough transformation [27], [28] or a likelihood function [29], [30]. A lane-detection method that extracts the lane marks based on color information in traffic scenes with moving vehicles is presented in [31]. Typical difficulties of vision-based lane detection algorithms are the large diversity in color and width of lane markings, image clarity, e.g., due to nearby vehicles, headlight glare, low sun angles, strong reflections, or faded lane markings, and the illumination problem at night or in bad weather, e.g., due to haze, fog, or rain [32], [33].

A possible solution is to use inertial sensor data to facilitate lane detection in all conditions. Note that these inertial sensors are also useful to detect potholes or to derive driving behavior statistics. A probabilistic lane estimation algorithm that uses an unsupervised crowdsourcing approach to learn the position and lane-span distribution of the different lane-level anchors based on accelerometer, gyroscope, and magnetometer, is presented in [34]. The lane anchors are based on empirical assumptions, e.g., vehicle stops occur on the right most lane, U-turns occur on the left lane, and potholes typically span only one lane. A GNSS/INS integration system, i.e., global navigation satellite system / inertial navigation system, that utilizes ray tracing and a 3D building map to rectify ranging errors caused by multipath or non line-of-sight (NLoS) scenarios is presented in [35]. Experiments in an urban canyon demonstrated half-lane width errors 60% of the time. A lane-level positioning system based on crowdsourced location estimates of roadway landmarks and vehicular sensors is presented in [36]. The position estimates of other cars at these landmarks are combined with odometry and bearing information from the vehicular sensors in a sequential Monte Carlo (SMC) method.

### C. Contributions

In this work, a method is proposed that combines the Markovian behavior and the shortest path aspect while taking into account the road speed limits and driving behavior. The proposed technique is a fast, memory-efficient, and worldwide map matching algorithm based on geographic coordinates and open map data, with support for lane detection that self-adapts to the driving behavior. Usually map matching is aimed solely at car rides [16] but this algorithm is compatible with walks, bikes, and car rides, and is evaluated on real GPS data for a varying levels of measurement noise and temporal sparsity, based on precision, recall, and location accuracy on a per point basis instead of merely the overlap between the ground truth and estimated path. The total validation dataset is 17.4 hours, covers 423 km, consists of 58k GPS points, and 402k accelerometer samples. Furthermore, neither the map matching technique nor the lane detection algorithm depend on large training sets [16], [17] or crowdsourced measurement campaigns [34], [36]. The proposed techniques are offline algorithms, i.e., the geographic coordinates are mapped to a road network and the driving lanes are assigned after all data has been recorded.

## III. Map matching

The goal of the proposed map matching algorithm is to output a continuous trajectory, i.e., connected road segments, based on timestamped geographic data as input.

### A. Grid and road segments

The road segments are based on publicly available OpenStreetMap data, consisting of line segments enriched with metadata about the type of road, e.g., sidewalk, bike path, or highway, information about one-way traffic, relative layering, street name, number of driving lanes, and maximum allowed speed. Note that a default value is used when the maximum speed for a line segment is unknown or invalid, e.g., 120 km/h for highways, 90 km/h for primary roads, 70 km/h for secondary roads, and 50 km/h for all other road types. In our approach the number of driving lanes is assumed to be available through OpenStreetMap metadata but this information can also be automatically extracted from crowdsourced GPS data if it is distributed across multiple traffic lanes [37]–[39]. To take into account cars that are speeding causing location estimations to lag behind, the allowed speed limit (for the reconstructed trajectory) can be increased by, e.g., 20% for each road segment. The straight line segments are further divided into equal pieces, i.e., road segments, based on the grid size. Note that line segments smaller than the grid size also occur, e.g., at roundabouts, and are automatically included without further separation. The begin and end points of all road segments constitute the grid. For Belgium, this results in 13.4 million road segments and 12.2 million grid points for a grid size of 10 m. Figure 3 shows a sample of these road segments and grid points.

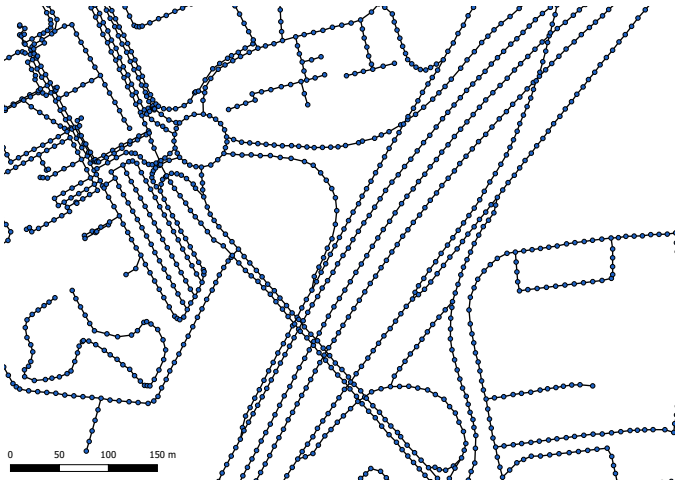



Figure 3: Grid and road segments based on OpenStreetMap data and a grid size of 10 m.

Note that the outputted map matched locations are not limited to the grid points but can lie anywhere on the road segments because of the interpolation phase (discussed in Section III-B). The grid and road segments are organized in square area blocks of 10 km$^2$ and calculated once based on the Winkel tripel projection [40], which is also the standard projection for world maps made by the National Geographic

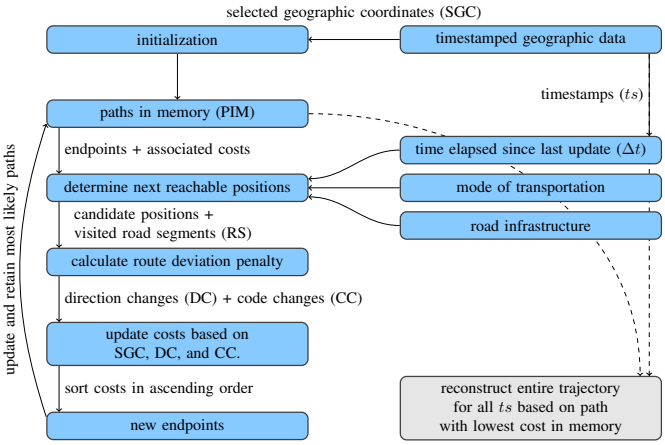



Figure 4: Flow graph of the map matching algorithm. The dashed lines are only executed at the end after all timestamped geographic data are processed. SGC: selected geographic coordinates, PIM: paths in memory, RS: road segments, DC: direction changes, and CC: code changes.

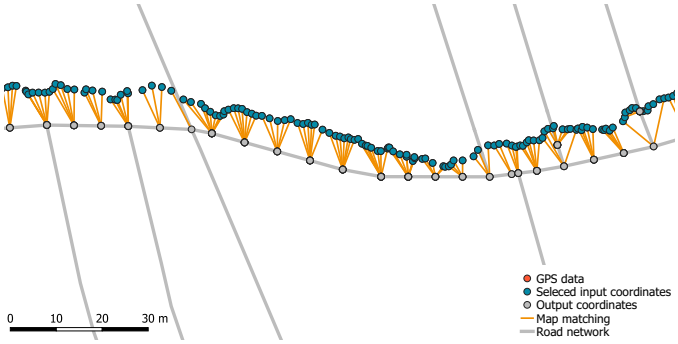
Society [41], [42]. The OpenStreetMap data for the whole world has a size of 40.6 GB (June, 2019) [43]. The map matching starts with a scan of the geographic input data to load only the necessary area blocks and reduce the required memory. Furthermore, the proposed map matching algorithm is online available through a web service [44].

### B. Algorithm

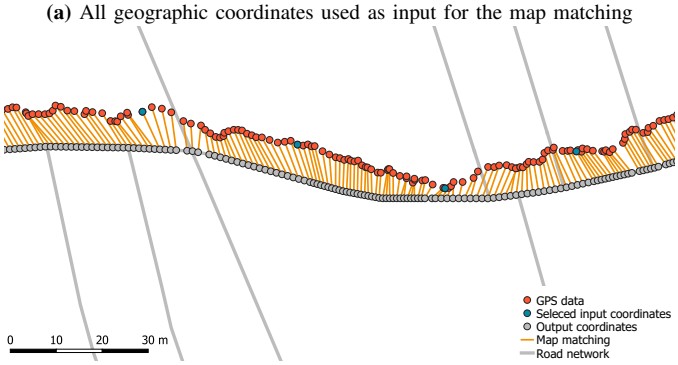
*1) General:* The proposed map matching algorithm is based on the Viterbi path, a technique related to hidden Markov models [45], [46]. The algorithm's input are raw locations, e.g., unmatched GPS data, the time between two location updates, the road network, and a variable maximum speed per road segment. By processing all available data at once, previous estimated locations can be corrected by future measurements (similar to backward belief propagation [47]). Naturally, this is only possible if the intended application tolerates a certain delay. Figure 4 shows a flow graph of the map matching algorithm, which ensures realistic and physically possible paths.

*2) Geographic data selection:* The proposed map matching algorithm starts with a selection of the geographic input coordinates (*initialization* in Figure 4) based on a minimum update distance and standing-still detection. This avoids the map matching problem of Figure 2 as a result of the shortest path effect between the selected points combined with a direction and code change penalty. The geographic input coordinates, that are not included in this selection, are map matched based on interpolation (Section III-B3).

The minimum update distance is the required traveled distance, based on the raw input data, before the next geographic coordinates are added to the selection. This is to establish a smoothing effect in the map matching, e.g., in the extreme case of only the begin and end point of a trajectory, the output is the shortest path between these two points. Figure 5 shows an example of the map matching based on all and a selection of the geographic input coordinates with a grid size of 10 m.




**(a)** All geographic coordinates used as input for the map matching




**(b)** Selection of geographic input coordinates + interpolation

Figure 5: Geographic data selection.

The direction change penalty is the absolute sum of direction changes along the visited road segments between the estimations of two selected geographic coordinates, multiplied by a weight ($\beta$). The code change penalty is the total amount of road code changes along these visited road segments, multiplied by a weight ($\gamma$). Note that the road code indicates the type of road, e.g., highway, primary, secondary, or tertiary roads, residential area, sidewalks, or bicycle path. This direction change penalty and code change penalty are introduced to encourage the reconstructed trajectory to go straight on its current road segment and to diminish the weird route effect caused by noisy geographic data. For example, the unnecessary detour in Figure 2 caused by solely minimizing a distance-based metric can be avoided by adding a direction change penalty and code change penalty. The first will penalize the four additional turns that are used to get closer to the original GPS data and the second will penalize the back and forth switch between main road to footpath.

The default minimum update distance is 50 m, and the default weights for the distance-based penalty ($\alpha$), direction change penalty ($\beta$), and code change penalty ($\gamma$) are 1 per meter, 10 per turn of 90°, and 10 per code change, respectively, which results in a unitless cost. The explanation behind these default values and the effect of these four parameters on the map matching accuracy are discussed in Section V-C4. Note that the default minimum update distance has no effect if the sample rate of the geographic coordinates is low and the tracked object is on the move, e.g., a car driving at a rate of 120 km/h and a sample interval of 5 s already results in 167 m per sample, and hence all geographic coordinates will be selected, i.e., are used as input in the map matching

algorithm (*SGC* in Figure 4).

The interpolation between two selected geographic coordinates results in significant location errors if the tracked person or object is barely moving for a certain period between these two coordinates. Therefore, a standing-still technique detects the start and end of intervals where there is no movement, e.g., a car waiting at a red traffic light. It is labeled as standing-still if the moving average of the raw GPS data is below a threshold, e.g., 5 m with a window length of 5 samples. The begin and end of these intervals are included in the selection to ensure a correct interpolation.

*3) Mapping and interpolation:* The pseudo-code of the map matching method is shown in Algorithm 1 and the variables and steps are discussed in the text below.

After a selection of geographic coordinates is made based on the minimum update distance and standing-still detection, the mode of transportation ($MoT$) is estimated based on the 95th percentile value of the reported driving speeds by the GPS with standing-still periods filtered out (to avoid picking an outlier). It is labeled as walking if it is below 10 km/h, as cycling if it is between 10 km/h and 40 km/h, and otherwise as driving a motorized vehicle. For our dataset, which encompassed 30 trajectories, this always resulted in the correct mode of transportation. If it is labeled as on foot or by bike, the highway road segments are discarded from the grid and if it is labeled as by car the sidewalks and bicycle road segments are discarded from the grid. Note that an incorrect label does not always deteriorate the performance, e.g., a runner with a 95th percentile value of 12 km/h would be labeled as a biker but this has no influence on the performance; on the contrary, if he or she would be labeled as *walking*, the reconstructed path would lag behind the real trajectory. However, the reconstructed trajectory of a car labeled as *walking* would avoid highways but changes are small that the 95th percentile value would be smaller than 10km/h for a trajectory on the highway.

The map matching algorithm is initialized with the first geographic coordinate (GC) as current position ($GC_0$), e.g., a GPS data point. Then, a predefined number of other locations ($MP$) are selected around this first position and their cost is initialized to zero, e.g., the 1000 closest grid points to the current position. This ensures that the map matching algorithm can recover from initially noisy GPS data, e.g., 1000 grid points and a grid size of 10 m resulted in covered surfaces between 18 and 75 hectares in the experimental validation of Section V (the exact area depends on the density of the road network). The initialization forms the starting point of all possible paths that are kept in the memory of the location tracking algorithm ($pathsInMem$).

Next, for the subsequent geographic coordinate ($GC$), all reachable positions ($RGP$) starting from the path's current last grid point ($PGP$: parent grid point) are determined for all *paths* in memory by making use of the surrounding road network, the time elapsed since last location update ($\Delta t$), the mode of transportation ($MoT$), and OpenStreetMap metadata, i.e., maximum speed, type of road, and one-way information. These reachable positions, which are also grid points, are the candidate positions for the next location update. The transitions between grid points are limited by the road infrastructure.

**Algorithm 1:** Map matching technique.

**Data:** timestamped geographic data (TGD)
**Result:** map matched trajectory (MMT)

$SGC \leftarrow$ selected geographic coordinates based on update distance and standing-still detection
$MoT \leftarrow$ (estimated) mode of transportation
$GC_0 \leftarrow$ first geographic coordinates
$t_{prev} \leftarrow$ first timestamp
$MP \leftarrow 1000$ `// maximum paths in memory`
$pathsInMem \leftarrow$ list with $MP$ grid points closest to $GC_0$ initialized with cost 0
`// iterate over all geographic coordinates in SGC`
**for** $GC \in SGC$ **do**
  $t \leftarrow$ current timestamp
  $\Delta t \leftarrow t - t_{prev}$
  $pathsTemp \leftarrow$ empty list
  **for** $path \in pathsInMem$ **do**
    $cost \leftarrow$ current cost of $path$
    $PGP \leftarrow$ current endpoint of $path$ (parent grid point)
    $RGP \leftarrow$ reachable grid points along roads with $MoT$ within time span $\Delta t$ starting from $PGP$
    `// calculate new path cost for each candidate position (CP) based on distance, direction, and code change penalty`
    **for** $CP \in RGP$ **do**
      $RS \leftarrow$ road segments between $PGP$ and $CP$
      $ED \leftarrow eucl\_dist(CP, GC)$ `// Euclidean distance`
      $DC \leftarrow$ penalty due to direction changes along $RS$
      $CC \leftarrow$ penalty due to code changes along $RS$
      $path_{new} \leftarrow path + RS + CP$
      $cost_{new} \leftarrow cost + \alpha \cdot ED + \beta \cdot DC + \gamma \cdot CC$
      add $(path_{new}, cost_{new})$ to $pathsTemp$
  $pathsInMem \leftarrow$ retain $MP$ paths from $pathsTemp$ based on lowest cost
  $t_{prev} \leftarrow t$
*MMT* $\leftarrow$ reconstruct trajectory along $path$ with lowest cost in $pathsInMem$ for all timestamps $t$ in $TGD$ based on interpolation

Each candidate position ($CP$) retains a link to the parent grid point ($PGP$), a list with visited road segments $RS$, and a cost that represents this new branch along the road network. This new path (branch) and updated cost are added to the temporary list ($pathsTemp$) as a tuple (($path_{new}, cost_{new}$)). The updated cost is a weighted sum of a distance ($ED$), direction change ($DC$), and code change ($CC$) penalty. The distance penalty is the Euclidean distance between $CP$ and $GC$, and the $DC$ and $CC$ penalty is calculated based on the directions and codes of the visited road segments $RS$, i.e., a physically possible path between $CP$ and $PGP$.

Lastly, the $MP$ paths with lowest cost are retained to serve as input for the next iteration. After all timestamped geographic data is processed, the entire trajectory of the path with lowest cost in memory is reconstructed for all timestamps in the geographic input data. The interpolation between the selected geographic coordinates is based on the visited road segments and a fixed or variable speed depending on the mode of transportation, i.e., for walks and bike rides this is fixed and for car rides this is the maximum road speed. The latter increases the accuracy if the road speed limit changes between two selected geographic coordinates, e.g., if once every minute a GPS sample is available and the car drives 30 s on a 90 km/h road and 30 s on a 30 km/h road, using regular interpolation would map too many locations to the first road. Note that this does not affect the precision, recall, or $F_1$ score (Section V-B) but solely the estimation accuracy in meter (Section V-C1).

## IV. Lane detection

The map matched GPS coordinates can be enriched with information about the current driving lane, which is useful to map potholes to their exact location or to derive driving statistics per lane on a highway. The number of driving lanes per road segment are included in the open map data and are added as metadata to the road segment (Section III-A). Figure 6 shows a detail of the number of available lanes per road segment for an area surrounding a driveway exit.


Figure 6: Available driving lanes per road segment based on open map data.

### A. Lane changes

The lane change detection algorithm is based on pattern recognition with accelerometer data. Note that the sensor coordinate system of the accelerometer must be transformed to the car coordinate system if they are not aligned, e.g., based on inertial measurements or knowledge of the sensor placement. In the experimental validation (Section V), the sensor placement was known and the coordinate systems of the accelerometer and car are equal (Figure 7), i.e., the x-axis is aligned with the driving direction (longitudinal axis), the y-axis is directed towards the left lane (lateral axis), and the z-axis is aligned along the direction of gravity (vertical axis).

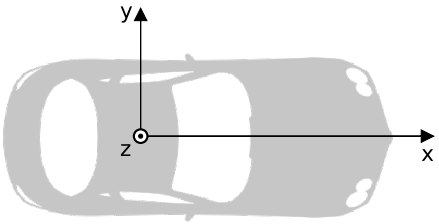



Figure 7: Alignment of sensor and car coordinate systems.

First, accelerometer values are low-pass filtered to reduce the influence of random noise with a one-second time window, which was found to be a good compromise between suppressing noise and distinguishable patterns.

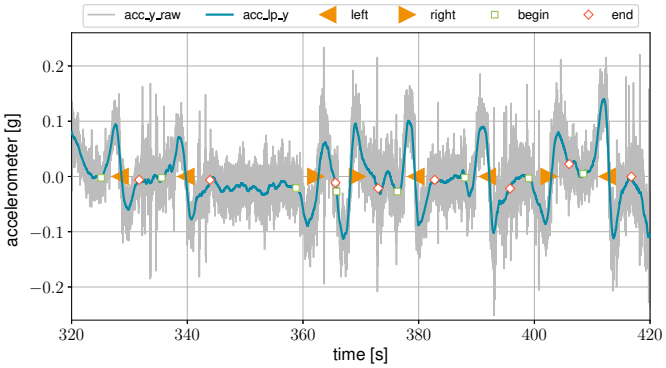
$$acc_t^{lp,\{x,y,z\}} = \frac{1}{W} \sum_{i=-\frac{W}{2}}^{\frac{W}{2}} acc_{t+i}^{\{x,y,z\}} \quad (1)$$

$acc_t^{\{x,y,z\}}$ and $acc_t^{lp,\{x,y,z\}}$ are the raw and low-pass filtered accelerometer values for the x-, y-, and z-axis of sample $t$, and $W$ is the window size, e.g., for a sample rate of 100 Hz and a one second time window $W$ is equal to 100.

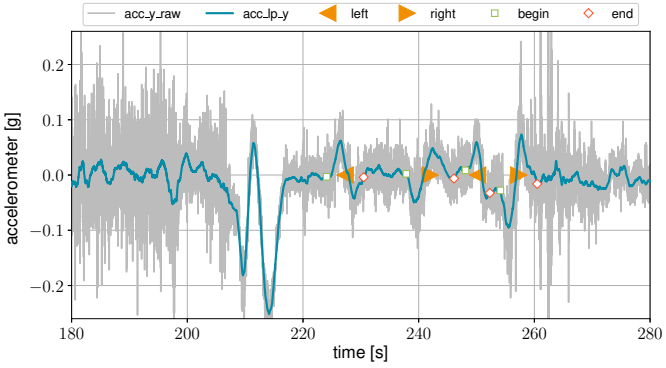
*1) Peak detection:* Figure 8 shows a sample of the raw and low-pass filtered sensor values for the lateral axis with indication of the begin (green square) and end (red circle) of left and right lane changes for two trajectories with a different car and driver (Section V-A).

It is clear from this real example that left and right lane changes show unique and similar patterns on the lateral acceleration and that the low-pass filtering is necessary to distinguish these patterns. A lane change to the left starts with a positive peak followed by a slightly lower negative peak along the lateral axis, for a right lane change this is the other way around. Figure 8a shows a sample of 100 s on the highway with five lane changes to the left and three to the right. The three peaks around the 210 s time mark in Figure 8b are due to driving on a roundabout and taking the first exit. The small negative and positive peaks before the 210 s time mark in Figure 8b are caused by speed ramps and road bumps on a tertiary road with one lane. Peaks are detected with following formulas:

$$t_{peak_{pos}} = \left\{ t \;\middle|\; acc_t^{lp,y} > acc_{t+i}^{lp,y},\ |acc_t^{lp,y}| > \delta_{acc},\ |i| \leq \tfrac{N}{2},\ i \neq 0 \right\} \quad (2)$$




**(a)** Trajectory 1




**(b)** Trajectory 2

Figure 8: Sample of raw and low-pass filtered sensor values for the lateral axis of the accelerometer with indication of left and right lane changes for both trajectories.

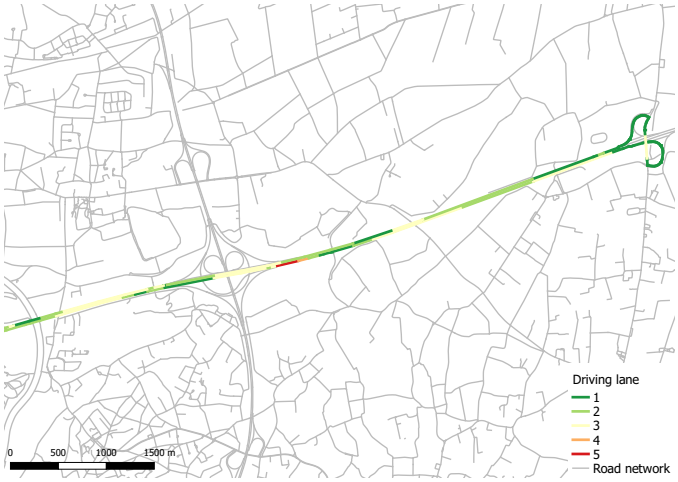
$$t_{peak_{neg}} = \left\{ t \;\middle|\; acc_t^{lp,y} < acc_{t+i}^{lp,y},\ |acc_t^{lp,y}| > \delta_{acc},\ |i| \leq \tfrac{N}{2},\ i \neq 0 \right\} \quad (3)$$

$t_{peak_{pos}}$ and $t_{peak_{neg}}$ are the timestamps of all positive and negative peaks, $acc_t^{lp,y}$ is the low-pass filtered lateral accelerometer value of sample $t$ (Equation 1), $\delta_{acc}$ is the noise floor for peak detection and is set at 0.02 g, and $N$ is the peak detection window and is set at a value that matches an interval of 2 s, e.g., 200 if the sample rate is 100 Hz. In the remainder of this section, $t_{peak_{pos}}$ and $t_{peak_{neg}}$ are referred to as (positive and negative) peaks.

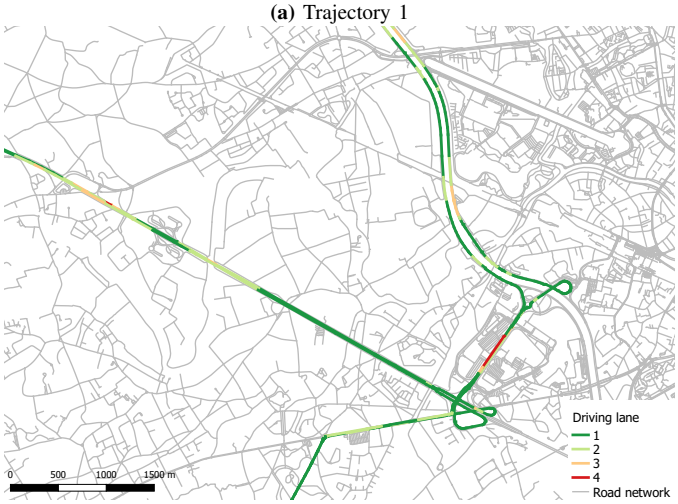
Figure 9 shows a sample of the actual driving lane at each road segment for both trajectories (the parts on the highway are traveled in both ways).

Table I summarizes the mean ($\mu$), standard deviation ($\sigma$), 75th percentile value, minimum, and maximum values of the positive peaks ($\text{peak}_{\text{pos}}$), negative peaks ($\text{peak}_{\text{neg}}$), peak-to-peak values, and time between two peaks associated with a lane change ($\Delta t$), for both trajectories.

Note that the absolute values of the negative peaks are used to simplify the comparison. Obviously, the driving style has an influence on the lateral acceleration patterns. Higher peak-to-peak values indicate fiercer lane changes, which can also be deducted from the shorter lane change times $\Delta t$ for the first trajectory. The lane changes in the second trajectory are milder and slower, i.e., an average peak-to-peak and $\Delta t$ value




**(a)** Trajectory 1




**(b)** Trajectory 2

Figure 9: Sample of the actual driving lane at each road segment for both trajectories.

| Trajectory | LC | Type | $\mu$ | $\sigma$ | 75th | Min | Max |
|---|---|---|---|---|---|---|---|
| 1 | 86 | $\text{peak}_{\text{pos}}$ [g] | 0.10 | 0.03 | 0.12 | 0.03 | 0.20 |
| | | $\text{peak}_{\text{neg}}$ [g] | 0.10 | 0.03 | 0.12 | 0.02 | 0.15 |
| | | peak-to-peak [g] | 0.20 | 0.04 | 0.21 | 0.11 | 0.35 |
| | | $\Delta t$ [s] | 2.14 | 0.43 | 2.37 | 1.27 | 3.30 |
| 2 | 67 | $\text{peak}_{\text{pos}}$ | 0.06 | 0.02 | 0.07 | 0.02 | 0.10 |
| | | $\text{peak}_{\text{neg}}$ | 0.07 | 0.02 | 0.08 | 0.03 | 0.12 |
| | | peak-to-peak | 0.12 | 0.03 | 0.15 | 0.08 | 0.22 |
| | | $\Delta t$ [s] | 2.93 | 0.87 | 3.21 | 1.48 | 4.87 |

Table I: Lane change peak statistics based on lateral accelerometer data. LC: amount of lane changes.

of 0.12 g and 2.93 s vs. 0.20 g and 2.14 s for the first trajectory, making them harder to detect because the patterns are less distinct (Section V-D1). To cope with different driving styles, a lane change detection algorithm based on a variable and bounded threshold, is proposed.

*2) Lane change detection:* First, the peaks are filtered based on the map matched data, i.e., all peaks that are detected while only one lane is available are discarded, as these peaks cannot be caused by a lane change but are due to, e.g., bad road conditions or driving maneuvers. These peaks are filtered

again based on a lower ($peak_{min}$) and upper bound ($peak_{max}$) threshold, i.e., all peaks outside these boundaries are discarded as well because these are most likely due to turns, noise, or small deviations within the driving lane. The remaining peaks are visited chronologically and two peaks (a positive and a negative peak for a left lane change and vice versa for a right lane change), are paired and marked as a left or right lane change if two conditions are met: their peak-to-peak value is above a threshold ($p2p_{min}$) and the time between both peaks ($\Delta t$) is below $t_{max}$. This $t_{max}$ value is fixed at 5 s, which is larger than the maximum duration of any lane change in the experimental validation over a total of 153 lane changes (LC in Table I).

The other threshold values ($peak_{min}$, $peak_{max}$, and $p2p_{min}$) are estimated per trajectory based on the low-pass filtered lateral accelerometer values on the road segments with at least two lanes. To estimate these three threshold values only straight road segments where the car was driving at least 50 km/h are taken into account to remove the influence of left or right turns and traffic jams. The reason for this is that the accelerometer values during turns and traffic jams substantially deviate from normal driving behavior, which would otherwise skew the threshold values. Note that these selected road segments are used to estimate the threshold values but afterwards all lateral accelerometer values on road segments with at least two lanes, are processed. The negative peaks that have a neighboring positive peak within $t_{max}$ are taken into account (and vice versa for the positive peaks). The minimum and maximum absolute value of these selected peaks are used as $peak_{min}$ and $peak_{max}$, and the minimum peak-to-peak value of these negative and positive peaks is taken as $p2p_{min}$.

### B. Driving lane allocation

After the detection of lane changes based on the lateral accelerometer data, the driving lane can easily be updated by adding one to the current driving lane if it is a lane change to the left and subtracting one if it is to the right. Lane one is used for the slow lane. The far left lane, i.e., fast lane, depends on the number of available lanes and could be anywhere from 2 to 8. Note that in left-driving countries the fast lane is the far right lane. Therefore, in the remainder of this section a right-driving country is assumed to avoid confusion. Only the initial driving lane is unknown and although it seems plausible to start at lane one, this is not always the case because, e.g., after a highway ramp a driver can go immediately to the second or third lane without being noticed by the lane change detection algorithm. Another source of errors are the addition of lanes on the right, e.g., when two highways merge, the current driving lane changes while the car was going straight.

The first problem is solved by accounting for a penalty for each possible starting lane when a driver goes from a road segment with one lane to a road segment with multiple lanes, e.g., when passing a highway ramp or when going from a small local road to a secondary road with two lanes. The penalties are calculated based on all future lane changes until the next road segment with only one lane. Each time an impossible lane change occurs, i.e., a left lane change when driving on the outside lane or a right lane change when driving on the first lane, this penalty is increased by one. The starting lane with the lowest penalty is taken as most likely initial lane. Note that the detection of two lane changes at once, when a driver is already on the highway, is possible by taking into account the time between two peaks, e.g., flag as two lane changes if it is between 4 and 6 seconds. However, in the experimental data only one such case is present.

The second problem is solved by monitoring the number of available lanes when the road segment changes, e.g., if this number increases and a road is merged from the right side, the current driving lane is increased by one otherwise it is left unaltered. Note that for left-driving countries this is the other way around.

## V. Experimental validation

### A. Trajectories

*1) Map matching:* The experimental validation encompasses 30 trajectories on foot, by bike, and by car. Table II summarizes the total distance, total duration, average speed, and number of GPS data points for all trajectories per mode of transportation and environment.

| MoT | Environment | #routes | Distance [km] | Duration [min] | Speed [km/h] | #GPS points |
|---|---|---|---|---|---|---|
| Walk | Urban | 8 | 30.95 | 5.57 | 5.56 | 17040 |
| | Rural | 2 | 11.27 | 2.27 | 4.96 | 7325 |
| Bike | Urban | 6 | 60.53 | 3.33 | 18.19 | 11654 |
| | Rural | 4 | 25.47 | 1.20 | 21.19 | 4212 |
| Car | Urban | 4 | 107.22 | 2.43 | 44.16 | 8616 |
| | Rural | 2 | 43.27 | 0.85 | 50.94 | 3049 |
| | Highway | 4 | 144.24 | 1.75 | 82.26 | 6292 |
| Total | | 30 | 422.95 | 17.40 | 82.95 | 58188 |

Table II: GPS dataset details per mode of transportation (MoT) and environment.

The trajectories are recorded at a sample rate of 1 Hz with a GPS logging application on a smartphone. The smartphone was put in the dashboard holder for the car rides, which were done by four different drivers, and carried in the pocket for the trajectories on foot and by bike. The total dataset is 17.4 hours, covers 423 km, consists of 58188 geographic coordinates (GPS points). The total distance on foot, by bike, and by car are 42 km, 86 km, and 295 km, respectively. The total distance in urban and rural areas, and on the highway are 199 km, 80 km, 144 km, respectively. The trajectories pass through rural and urban areas, on primary and secondary roads, sidewalks, bicycle paths, and highways. Note that the environment has a strong influence on the performance of a map matching algorithm, e.g., in rural areas the road network is usually sparser, which reduces the chance to select a wrong road segment and there are less tall buildings that can cause additional noise on the GPS signals.

*2) Lane detection:* Accelerometer data (Analog Devices ADXL345 [48]) is available for two car rides on the highway with a lot of forced lane changes to have a sufficient amount of data to validate the lane detection, i.e., around four and two lane changes per minute. Note that these are the two trajectories from Section IV-A (Figures 8 and 9). The driving lane ground truth is manually annotated based on video

recordings from a dashcam. A script was written to facilitate this process, i.e., a video can be watched at an adjustable speed and is annotated by pushing the left or right button when the car makes a left or right lane change. The driving lane details, i.e., the number of lane changes (LC), the average lane changes per minute where lane changes are possible (LCPM), total duration, time spent in each driving lane, and accelerometer samples, are summarized in Table III for the two car rides.

| Trajectory | LC | LCPM | Duration [min] | Time per lane [min] | | | | | #samples |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 1-av | 1 | 2 | 3 | 4 | |
| 1 | 86 | 4.0 | 24.6 | 3.3 | 5.9 | 8.1 | 6.7 | 0.6 | 143229 |
| 2 | 67 | 2.1 | 43.4 | 11.3 | 18.5 | 10.3 | 3.0 | 0.2 | 258606 |

Table III: Driving lane details per trajectory based on dashcam video recordings. LC: amount of lane changes, LCPM: average number of lane changes per minute, 1-av: only one lane available, and number of accelerometer samples.

### B. Performance metrics

The quality of the map matching and lane detection algorithm are each validated with two performance metrics. The map matching algorithm is evaluated with the $F_1$ score (also F-score or F-measure) and the average error (Euclidean distance) between the ground truth and estimated location. The $F_1$ score is the harmonic average of the precision and recall, where an $F_1$ score reaches its best value at 1 (perfect precision and recall) and worst at 0. Broadly speaking, the precision, also called positive predictive value, is the fraction of relevant instances among the retrieved instances and the recall, also known as sensitivity, is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Precision can be seen as a measure of exactness or quality, whereas recall is a measure of completeness or quantity.

$$\text{precision} = \frac{tp}{tp + fp} \tag{4}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{5}$$

$tp$ are the true positives or hits, $fp$ are the false positives or false alarm (Type I error: asserting something that is absent), $fn$ are the false negatives or misses (Type II error: failing to assert what is present), and $tn$ are the true negatives or correct rejections but these are abundant to calculate the precision or recall.

In the context of map matching, this results in following formulas:

$$F_1^{mm} = 2 \cdot \frac{\text{precision}_{\text{mm}} \cdot \text{recall}_{\text{mm}}}{\text{precision}_{\text{mm}} + \text{recall}_{\text{mm}}} \tag{6}$$

$$\text{precision}_{\text{mm}} = \frac{L_{corr}}{L_{mm}} \tag{7}$$

$$\text{recall}_{\text{mm}} = \frac{L_{corr}}{L_{gt}} \tag{8}$$

The *mm* in $F_1^{mm}$, *precision*$_{mm}$, and *recall*$_{mm}$ refers to *map matching* so that it can be distinguished from the recall and precision definition for the lane detection. $L_{gt}$ is the length of the ground truth trajectory, $L_{mm}$ is the length of the map matched trajectory, and $L_{corr}$ is the length of the (correct) overlapping segments between the map matched and ground truth trajectory. Most map matching algorithms use this metric to evaluate the approach because it suffices for a broad range of applications, e.g., road usage patterns or urban traffic modeling.

The ground truth trajectories, i.e., continuous sequence of road segments and individual locations on these segments, are constructed based on the shortest path between manually indicated points where the route did pass at certain timestamps. Note that the trajectories were known beforehand and indication points are added as long as the constructed paths were not completely correct. Figure 10 shows the raw GPS input data with a sample rate of 1 Hz and the constructed ground truth segments and locations.

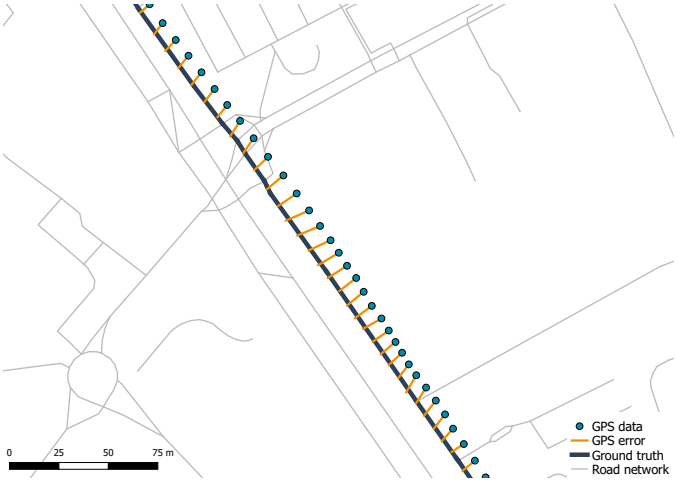



Figure 10: Raw GPS input data with ground truth construction.

The individual timestamped locations are used to calculate the average error of the map matched trajectories. This metric is generally ignored but is important for applications that map events or other sensor measurements to an exact location.

The performance of the lane detection is evaluated with the $F_1$ score based on the precision and recall in the classification context.

$$F_1^{ld} = 2 \cdot \frac{\text{precision}_{\text{ld}} \cdot \text{recall}_{\text{ld}}}{\text{precision}_{\text{ld}} + \text{recall}_{\text{ld}}} \tag{9}$$

$$\text{precision}_{\text{ld}} = \frac{tp_{\text{ld}}}{tp_{\text{ld}} + fp_{\text{ld}}} \tag{10}$$

$$\text{recall}_{\text{ld}} = \frac{tp_{\text{ld}}}{tp_{\text{ld}} + fn_{\text{ld}}} \tag{11}$$

The *ld* subscript refers to *lane detection* so that it can be distinguished from the $F_1$ score, recall and precision definition for the map matching. $tp_{\text{ld}}$ are the true positives, i.e., a left or right lane change is correctly identified. $fp_{\text{ld}}$ are the false positives, e.g., a driver stays in his lane but the algorithm detects a lane change due to a bump or maneuver. $fn_{\text{ld}}$ are

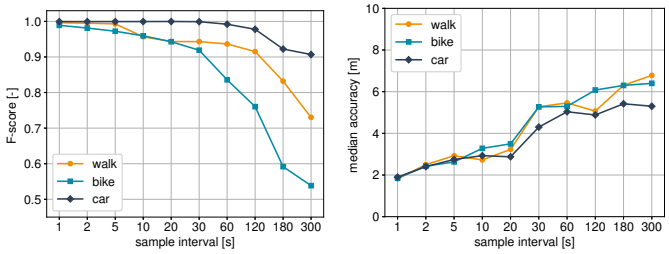
the false negatives, i.e., failing to identify a left or right lane change.

The second performance metric for the lane detection is the amount of time the correct lane is estimated based on the detected lane changes. Incorrectly predicted driving lanes are further divided into 1-off and 2-off when the predicted lane is one or two lanes off, i.e., the absolute difference between the ground truth and detected lane (three lanes off did not occur in our experimental validation).

### C. Map matching accuracy

The accuracy of the map matching is evaluated as a function of the sample interval, GPS noise, and three algorithm parameters: update distance, direction change, and code change penalty.

*1) GPS sample interval:* Lowering the GPS sample rate saves battery power and reduces the communication cost by limiting the bandwidth usage but increases the computational cost per location update as a larger area needs to be considered. Note that since GPS devices need a lock on the available satellites it is not possible to turn the device completely off between samples because the startup time with a cold or warm start is too high [49]. However, a rapid acquisition of satellite signals is enabled in standby (hot) mode because the receiver already has valid time, position, almanac (approximate information on all the other satellites), and ephemeris data (detailed orbital information). The influence of the GPS sample interval is mimicked by downsampling the input data, i.e., discarding GPS samples to acquire the intended sample rate, and each simulation was repeated ten times for averaging. Figure 11 shows the $F_1^{mm}$ score and the median error between the estimated and ground truth locations, averaged over all simulations and trajectories per mode of transportation, as a function of the sample interval.

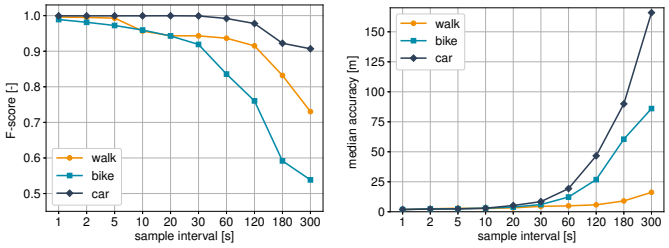



**(a)** $F_1^{mm}$ score **(b)** Median error

Figure 11: $F_1^{mm}$ score and median error as a function of the sample interval per mode of transportation.

As expected, the highest accuracy is obtained with the highest sample rate, i.e., an $F_1^{mm}$ score and median error of 99.6% and 1.82 m, 98.9% and 1.85 m, and 99.9% and 1.99 m, for the trajectories on foot, by bike, and by car, respectively. Note that these are averaged over 30 trajectories (10 trajectories per mode of transportation). This results in an average $F_1^{mm}$ score and median error of 99.5% and 1.89 m. Contrarily, increasing the sample interval results in lower $F_1^{mm}$ scores and a larger median error. For the car rides, the $F_1^{mm}$ score and median error are still 90.7% and 5.31 m even if only once every 5 min a GPS sample is available. Increasing the sample rate to once every minute, which can still be considered as a power saving mode, improves the average $F_1^{mm}$ scores and median errors: 93.7% and 5.46 m, 83.6% and 5.29 m, and 99.2% and 5.04 m, for the trajectories on foot, by bike, and by car, respectively.

Although the median errors are similar for the three considered modes of transportation, the $F_1^{mm}$ score is higher for the car rides because these have more restrictions on the road segments than a walk and bike ride, e.g., the reconstruction of a person on foot has access to all type of roads (except highways) in both ways because one-way streets are usually not applicable to pedestrians and a separate sidewalk is not always included in the open street map data. A typical error for the walks and bike rides is to select a parallel road that is very close to the correct road, which has a small impact on the location error but decreases the $F_1^{mm}$ score because the trajectories do not overlap. Note that the median error is based on the (downsampled) geographic input coordinates, e.g., once every 5 minute, which does not reveal much about the exact location accuracy of the points along the reconstructed trajectory (based on the timestamps). Figure 12 shows the same plot but with interpolation, e.g., for a sample interval of 5 min and interpolation at 1 Hz, the algorithm outputs all road segments and 300 estimated locations with every update, i.e., once every second instead of only one estimated location and the road segments to get there.




**(a)** $F_1^{mm}$ score **(b)** Median error

Figure 12: $F_1^{mm}$ score and median error as a function of the sample interval per mode of transportation with interpolation at 1 Hz.

Naturally, the $F_1^{mm}$ score is exactly equal and the median error is larger, e.g., with a sample interval of 1 min, the median errors of all estimated locations during this minute are 4.94 m, 12.3 m, and 19.32 m, for the trajectories on foot, by bike, and by car, respectively. Note that a car driving at 120 km/h travels a distance of 2 km during a one minute interval. The location error along the interpolated points starts to increase rather fast for the bike and car rides if the sample rate is further decreased because the traveling speed can be variable due to traffic lights, road speed limits, or oncoming traffic. This is detrimental for the interpolation, e.g., with a sample interval of 5 min there is no way of knowing if a car is standing-still for a minute or just driving slowly. The reason for the lower location error of the trajectories on foot is because a walking speed is more constant than a biking or driving speed due to accelerations and slowing down.

*2) OSRM comparison:* The well-known Open Source Routing Machine (OSRM [50]) method is used to validate our proposed map matching technique. Since OSRM does not output continuous road segments but only separate coordinates, only the average error (Euclidean distance between the ground truth and estimated location) can be compared. Table IV summarizes the median and 75th percentile value of the average error per mode of transportation.

| Algorithm | walk | | bike | | car | |
|---|---|---|---|---|---|---|
| | 50th [m] | 75th [m] | 50th [m] | 75th [m] | 50th [m] | 75th [m] |
| OSRM | 2.36 | 6.37 | 1.95 | 5.48 | 2.36 | 4.53 |
| Proposed | 1.82 | 3.49 | 1.85 | 3.55 | 1.9 | 3.51 |
| Improvement [%] | 22.9 | 45.2 | 5.1 | 35.2 | 19.5 | 22.5 |

Table IV: Accuracy comparison of proposed technique with OSRM per mode of transportation.

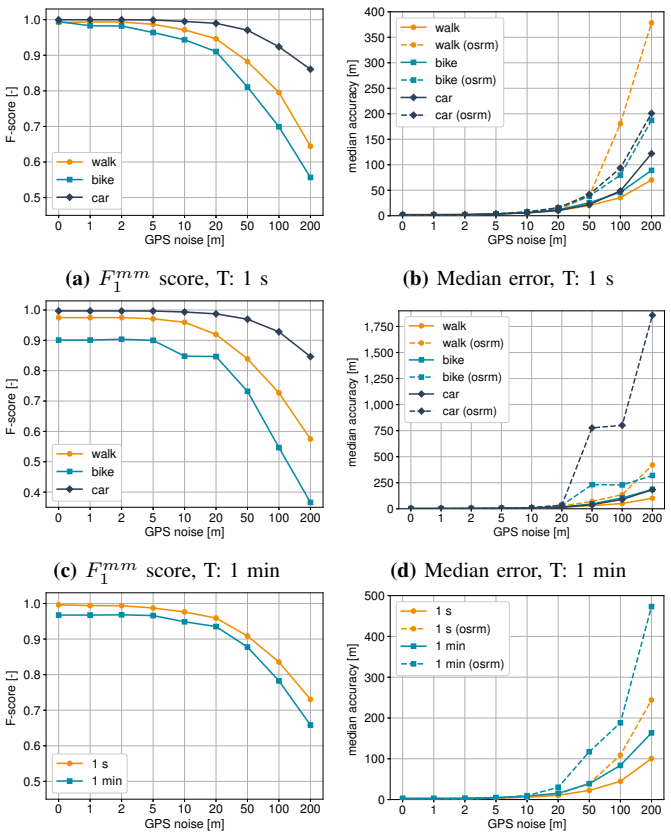
The proposed technique performs better than OSRM technique, with improvements up to 45.2% (averaged over the 10 walks). In absolute values, mainly the maximal errors are reduced. An additional advantage of our approach is the ability to interpolate along the road network, e.g., when only one sample per minute is available, and that it outputs a realistic trajectory, which is more robust against noisy input data compared to the nearest road approach of OSRM (discussed in Section V-C3).

*3) GPS noise:* The noise in GPS measurements can be modeled as zero-mean Gaussian [51]. To assess the influence of noise on the proposed map matching algorithm, geographic input data is simulated by taking the ground truth locations from Section V-A and adding Gaussian noise with a varying standard deviation. In the remainder of this section, the standard deviation of the added Gaussian noise is referred to as *noise level*. Figure 13 shows the $F_1^{mm}$ score and the median error between the estimated and ground truth locations, as a function of the added noise, for two sample intervals $T$ (1 s and 1 min).

Obviously, increasing the noise level results in lower $F_1^{mm}$ scores and a larger median error. The $F_1^{mm}$ scores and median errors, for a sample interval of 1 s and noise levels of 2 m and 20 m, start at 99.3% and 2.26 m, 98.2% and 2.22 m, and 100.0% and 2.33 m, and drop to 94.6% and 10.0 m, 91.0% and 11.57 m, and 99.0% and 10.34 m, for the walks, bike rides, and car routes, respectively.

Increasing the sample interval from once every second to once every minute degrades the performance but has a limited effect on the median error for low noise levels (Figure 13f). The $F_1^{mm}$ score is shifted by an average value of 3.5% for all noise levels. The average $F_1^{mm}$ scores and median errors for a noise level of 10 m start at 95.9% and 10.56 m for a sample interval of 1 s, and drop to 93.5% and 15.34 m for a sample interval of 1 min, although the amount of available data is reduced by a factor of 60.

For completeness, the accuracy of the OSRM technique (Section V-C2) is added in the median error plots (Figures 13b, Figures 13b and 13f). The proposed technique always outperforms OSRM, especially for higher noise levels with similar relative improvements for both sample intervals, e.g., a median




**(a)** $F_1^{mm}$ score, T: 1 s **(b)** Median error, T: 1 s

**(c)** $F_1^{mm}$ score, T: 1 min **(d)** Median error, T: 1 min

**(e)** $F_1^{mm}$ score per sample interval **(f)** Median error per sample interval

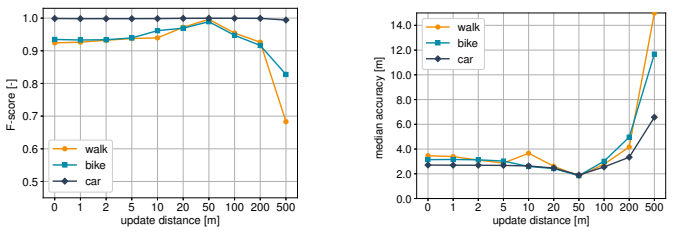
Figure 13: $F_1^{mm}$ score and median error as a function of the GPS noise per mode of transportation for a sample interval $T$ of 1 s (**a**–**b**) and 1 min (**c**–**d**). Overall $F_1^{mm}$ score and median error as a function of the GPS noise per sample interval $T$ (**e**–**f**)

accuracy improvement of 58.7% and 65.4% for noise level of 200 m and a sample interval of 1 s and 1 min.

*4) Sensitivity analysis:* This section discusses the influence of four parameters of the proposed map matching technique (Algorithm 1): update distance, weight of the distance metric ($\alpha$), weight of the direction change penalty ($\beta$), and weight of the code change penalty ($\gamma$). The update distance is used to establish a smoothing effect in the map matching by making a selection of geographic input coordinates (Section III-B). The $\alpha$ parameter is used to assign more weight to the geographic input coordinates and the $\beta$ and $\gamma$ parameters are used to discourage deviations from the current road or driving direction. Figure 14 shows the $F_1^{mm}$ score and the median error between the estimated and ground truth locations, as a function of these four parameters.

An update distance of 0 m, i.e., every geographic coordinate is used as input, results in an average $F_1^{mm}$ score and the median error of 95.2% and 3.10 m. Increasing the update distance to 50 m results in an average $F_1^{mm}$ score and the median error of 99.5% and 1.86 m.

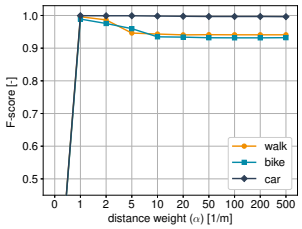
A distance weight of 0 gives poor results because the geographic data is not taken into account. Best performance is obtained with a distance weight of 1 and increasing this weight



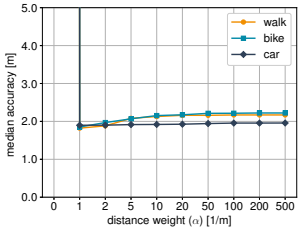

**(a)** $F_1^{mm}$ score



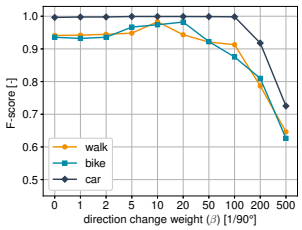

**(b)** Median error



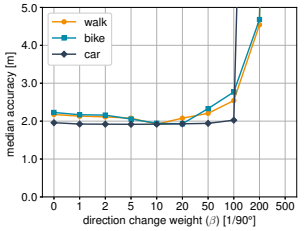

**(c)** $F_1^{mm}$ score



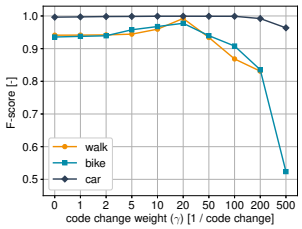

**(d)** Median error



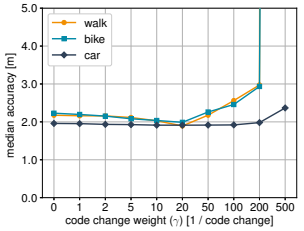

**(e)** $F_1^{mm}$ score




**(f)** Median error




**(g)** $F_1^{mm}$ score




**(h)** Median error

Figure 14: $F_1^{mm}$ score and median error as a function of the update distance (**a**–**b**), distance weight $\alpha$ (**c**–**d**), direction change weight $\beta$ (**e**–**f**), and code change weight $\gamma$ (**g**–**h**).

slowly deteriorates the $F_1^{mm}$ score (Figure 14c). Including the direction change penalty slightly improves the accuracy for weights up to 10 per 90°, i.e., the $F_1^{mm}$ score improves from 95.8% to 98.5% (Figure 14e). Including the code change penalty slightly improves the accuracy as well for weights up to 20, i.e., the $F_1^{mm}$ score improves from 95.8% to 99.0% (Figure 14g). The best global accuracy, averaged over all 30 trajectories on foot, by bike, or by car, is obtained with an update distance of 50 m, $\alpha$ of 1 per meter, $\beta$ of 10 per 90°, and $\gamma$ of 10 per code change. Note that $\gamma$ is set to zero to evaluate the influence of the direction change penalty and $\beta$ is set to zero to evaluate the influence of the code change penalty. Furthermore, the considered parameter values were on a logarithmic scale to avoid overfitting on the dataset of this experimental validation.

*5) Execution time:* The experiments are run on a laptop with an Intel Core i7 2.70 GHz processor, 16.00 GB DDR3-SDRAM and 64 bit operating system. The execution time, averaged over all trajectories per mode of transportation and calculated as the total duration divided by the number of input GPS points, is 1.03 ms, 5.75 ms, and 8.56 ms, for the walks, bike rides, and car routes, respectively. These differences per mode of transportation stem from the difference in maximum speed, i.e., more candidate routes and positions have to be calculated given a certain time interval. For comparison, the execution times reported in recent literature vary between 39 ms and 5.62 s, depending on the algorithm's settings and the input sample interval [52]–[54].

### D. Lane detection accuracy

The lane detection is evaluated by the accuracy of the lane change detection algorithm, the driving lane estimation, and the influence of the sample rate of the accelerometer.

*1) Lane changes:* The precision, recall, and $F_1^{ld}$ score of the lane change detection algorithm is summarized in Table V for both trajectories.

| Trajectory | Precision [%] | Recall [%] | $F_1^{ld}$ score [%] | tp [-] | fp [-] | fn [-] |
|---|---|---|---|---|---|---|
| 1 | 100.0 | 96.5 | 98.2 | 83 | 0 | 3 |
| 2 | 85.0 | 67.2 | 75.1 | 45 | 8 | 22 |

Table V: Precision, recall, and $F_1^{ld}$ score for the lane change detection. tp: true positives, fp: false positives, fn: false negatives.

For the first trajectory, the proposed algorithm correctly labels 83 lane changes, detects no false positives (precision 100.0%) and fails to detect only three lane changes (recall 96.5%). This results in an $F_1^{ld}$ score of 98.2%. For the second trajectory, the proposed algorithm correctly labels 45 lane changes, detects 8 false positives (precision 85.0%) and fails to detect 22 lane changes (recall 67.2%). This results in an $F_1^{ld}$ score of 75.1%. The difference in performance is due to different driving styles and making the detection harder by shifting lanes while turning, carrying out small maneuvers, or changing lanes in a slow manner.

*2) Driving lane:* The amount of time the lane detection algorithm estimates the correct and wrong lane are summarized in Table VI for both trajectories. These amounts of time are expressed as a percentage with respect to the total duration of a trajectory.

| Trajectory | Correct lane [%] | | | Wrong lane [%] | | |
|---|---|---|---|---|---|---|
| | 1-av | match | total | 1-off | 2-off | total |
| 1 | 14.8 | 84.4 | 99.2 | 0.7 | 0.1 | 0.8 |
| 2 | 21.6 | 65.2 | 86.8 | 12.3 | 0.9 | 13.2 |

Table VI: Accuracy of the lane detection algorithm as a percentage of the complete trajectory. 1-av indicates that there is only one lane available, and 1-off or 2-off that the estimated lane is one or two lanes next to the correct lane.

The driving lane is estimated correctly 99.2% and 86.8% of the time for the first and second trajectory, which is a logical consequence of the lane change detection performance. Note that undetected or erroneous lane changes do not automatically lead to a wrongly estimated driving lane because of the limited

number of available lanes per road segment and the most likely starting lane technique (Section IV-B). Furthermore, there was only one lane available during 14.8% and 21.6% of the time, which automatically results in the correct lane because the lane change detection and driving lane allocation are disabled when the map matching algorithm estimates a road segment with only one lane. A wrongly estimated lane is nearly always only one lane off, two lanes off occurs only in 0.1% and 0.9% of the time, and three lanes off does not occur. This is better than the IMU-based lane detection technique that is proposed in [34], where an average accuracy of 80% is achieved.

*3) Accelerometer sample rate:* The sample rate of the accelerometer is simulated by discarding samples in the input data. Similarly to the GPS sample interval (Section V-C1), lowering the sample rate of the accelerometer saves battery power and reduces the communication cost by limiting the bandwidth usage, e.g., if the data is collected or processed centrally. Unlike with the GPS data, the computational cost decreases because less samples need to be examined during the peak detection. However, this can affect the accuracy of the lane change detection algorithm and hence driving lane estimation. Figure 15 shows the precision, recall, and $F_1^{ld}$ score as a function of the accelerometer sample rate averaged over both trajectories.

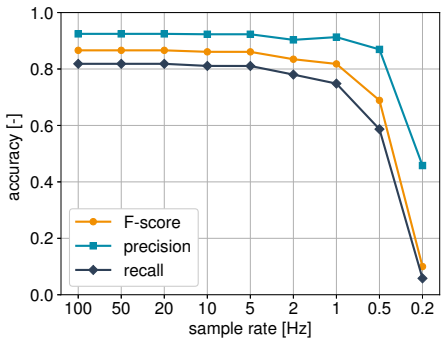



Figure 15: Precision, recall, and $F_1^{ld}$ score as a function of the accelerometer sample rate over both trajectories.

Lowering the sample rate results in a negligible performance loss up to 5 Hz, i.e., a reduction by a factor of 20. Increasing the sample interval to 5 s results in a recall near zero because all lane changes happen within 5 s. The lane changes that are still detected are because the lateral accelerometer values at the selected time instances are still above or below the thresholds values from Section IV-A.

## VI. Conclusion

In this work, a fast, memory-efficient, and worldwide map matching algorithm with support for trajectories on foot, by bike, and motorized vehicles, is presented. The input for the map matching algorithm are raw geographic coordinates and enriched open map data. The proposed algorithm combines the Markovian behavior and the shortest path aspect while taking into account the type and direction of all road segments, information about one-way traffic, maximum allowed speed per road segment, and driving behavior. Furthermore, a lane detection algorithm based on accelerometer readings and traffic lane information from the open map data, that self-adapts to the driving behavior, is added on top of the map matching algorithm. An experimental validation consisting of 30 trajectories on foot, by bike, and by car, showed the efficiency and accuracy of the proposed algorithms. The total dataset is 17.4 hours, covers 423 km, consists of 58k GPS points, and 402k accelerometer samples. The average F1-scores and median errors of the map matching algorithm, if all GPS samples are used, were an $F_1^{mm}$ score and median error of 99.6% and 1.82 m, 98.9% and 1.85 m, and 99.9% and 1.99 m, for the walks, bike rides, and car routes, respectively. The performance remained adequate if the input data was downsampled to only one sample every minute, i.e., the average F1-scores were 93.7%, 83.6%, 99.2%, for the walks, bike rides, and car routes, respectively. Moreover, the proposed technique outperforms the well-known OSRM method with accuracy improvements up to 45.2%. Two trajectories with accelerometer data were used to evaluate the lane detection algorithm with F1-scores of 98.2% and 75.1% for the lane change detection, which resulted in the correctly estimated lane 99.2% and 86.8% of the time.

## Acknowledgment

This work was executed within MobiSense, a research project bringing together academic researchers and industry partners. The MobiSense project was co-financed by imec (iMinds) and received project support from Flanders Innovation & Entrepreneurship.

## References

[1] B. Hummel, "Map matching for vehicle guidance," in *Dynamic and Mobile GIS*. CRC Press, 2006, pp. 196–207.

[2] H.-J. Chu, G.-J. Tsai, K.-W. Chiang, and T.-T. Duong, "Gps/mems ins data fusion and map matching in urban areas," *Sensors*, vol. 13, no. 9, pp. 11 280–11 288, 2013.

[3] D. Bernstein, A. Kornhauser *et al.*, "An introduction to map matching for personal navigation assistants," 1996.

[4] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation research part c: emerging technologies*, vol. 8, no. 1-6, pp. 91–108, 2000.

[5] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 853–864.

[6] A. Dewandaru, A. M. Said, and A. N. Matori, "A novel map-matching algorithm to improve vehicle tracking system accuracy," in *2007 International Conference on Intelligent and Advanced Systems*. IEEE, 2007, pp. 177–181.

[7] D. C. Andrade, F. Bueno, F. R. Franco, R. A. Silva, J. H. Z. Neme, E. Margraf, W. T. Omoto, F. A. Farinelli, A. M. Tusset, S. Okida *et al.*, "A novel strategy for road lane detection and tracking based on a vehicle's forward monocular camera," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–11, 2018.

[8] W. Song, Y. Yang, M. Fu, Y. Li, and M. Wang, "Lane detection and classification for forward collision warning system based on stereo vision," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5151–5163, 2018.

[9] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks," in *2017 ieee intelligent vehicles symposium (iv)*. IEEE, 2017, pp. 1019–1024.

[10] M. M. Atia, A. R. Hilal, C. Stellings, E. Hartwell, J. Toonstra, W. B. Miners, and O. A. Basir, "A low-cost lane-determination system using gnss/imu fusion and hmm-based multistage map matching," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3027–3037, 2017.

[11] J. Yu, Z. Chen, Y. Zhu, Y. J. Chen, L. Kong, and M. Li, "Fine-grained abnormal driving behaviors detection and identification with smartphones," *IEEE transactions on mobile computing*, vol. 16, no. 8, pp. 2198–2212, 2016.

[12] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, "Estimating driving behavior by a smartphone," in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 234–239.

[13] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2009, pp. 336–343.

[14] G. R. Jagadeesh and T. Srikanthan, "Online map-matching of noisy and sparse location data with hidden markov and route choice models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2423–2434, 2017.

[15] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[16] J. Yang and L. Meng, "Feature selection in conditional random fields for map matching of gps trajectories," in *Progress in Location-Based Services 2014*. Springer, 2015, pp. 121–135.

[17] Y. Li, Q. Huang, M. Kerber, L. Zhang, and L. Guibas, "Large-scale joint map matching of gps traces," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 214–223.

[18] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *nature*, vol. 453, no. 7196, p. 779, 2008.

[19] C. A. V. Campos, D. C. Otero, and L. F. M. de Moraes, "Realistic individual mobility markovian models for mobile ad hoc networks," in *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No. 04TH8733)*, vol. 4. IEEE, 2004, pp. 1980–1985.

[20] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map matching: Facts and myths," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 484–487.

[21] R. Metzler, "Generalized chapman-kolmogorov equation: A unifying approach to the description of anomalous transport in external fields," *Physical Review E*, vol. 62, no. 5, p. 6233, 2000.

[22] J. Wahlström, I. Skog, and P. Händel, "Smartphone-based vehicle telematics: A ten-year anniversary," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2802–2825, 2017.

[23] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers," in *2011 International conference on distributed computing in sensor systems and workshops (DCOSS)*. IEEE, 2011, pp. 1–6.

[24] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 286–291.

[25] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 690–703, 2016.

[26] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 540–555, 2013.

[27] K. Ghazali, R. Xiao, and J. Ma, "Road lane detection using h-maxima and improved hough transform," in *2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation*. IEEE, 2012, pp. 205–208.

[28] X. Wei, Z. Zhang, Z. Chai, and W. Feng, "Research on lane detection and tracking algorithm based on improved hough transform," in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*. IEEE, 2018, pp. 275–279.

[29] W. Liu, H. Zhang, B. Duan, H. Yuan, and H. Zhao, "Vision-based real-time lane marking detection and tracking," in *2008 11th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2008, pp. 49–54.

[30] M. Nieto, A. Cortés, O. Otaegui, J. Arróspide, and L. Salgado, "Real-time lane tracking using rao-blackwellized particle filter," *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 179–191, 2016.

[31] H.-Y. Cheng, B.-S. Jeng, P.-T. Tseng, and K.-C. Fan, "Lane detection with moving vehicles in the traffic scenes," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 4, pp. 571–582, 2006.

[32] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[33] J. Son, H. Yoo, S. Kim, and K. Sohn, "Real-time illumination invariant lane detection for lane departure warning system," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1816–1824, 2015.

[34] H. Aly, A. Basalamah, and M. Youssef, "Lanequest: An accurate and energy-efficient lane detection system," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2015, pp. 163–171.

[35] Y. Gu, L.-T. Hsu, and S. Kamijo, "Gnss/onboard inertial sensor integration with the aid of 3-d building map for lane-level vehicle self-localization in urban canyon," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4274–4287, 2015.

[36] Y. Jiang, H. Qiu, M. McCartney, G. Sukhatme, M. Gruteser, F. Bai, D. Grimm, and R. Govindan, "Carloc: Precise positioning of automobiles," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 253–265.

[37] Y. Chen and J. Krumm, "Probabilistic modeling of traffic lanes from gps traces," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, 2010, pp. 81–88.

[38] E. Uduwaragoda, A. Perera, and S. Dias, "Generating lane level road data from vehicle trajectories using kernel density estimation," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 384–391.

[39] L. Tang, X. Yang, Z. Dong, and Q. Li, "Clric: collecting lane-based road information via crowdsourcing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2552–2562, 2016.

[40] "Winkel tripel projections," http://www.winkel.org/other/Winkel%20Tripel%20Projections.htm, accessed: 2018-09-01.

[41] J. P. Snyder, *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[42] B. Vsavric, B. Jenny, D. White, and D. R. Strebe, "User preferences for world map projections," *Cartography and Geographic Information Science*, vol. 42, no. 5, pp. 398–409, 2015.

[43] "Openstreetmap data extracts," http://download.geofabrik.de/, accessed: 2019-06-01.

[44] "Map matching tool," https://www.waves.intec.ugent.be/exposure-tool/map-matching-tool, accessed: 2018-10-22.

[45] J. Trogh, D. Plets, L. Martens, and W. Joseph, "Advanced real-time indoor tracking based on the viterbi algorithm and semantic data," *International Journal of Distributed Sensor Networks*, vol. 501, p. 271818, 2015.

[46] J. Trogh, D. Plets, A. Thielens, L. Martens, and W. Joseph, "Enhanced indoor location tracking through body shadowing compensation," *IEEE Sensors Journal*, vol. 16, no. 7, pp. 2105–2114, 2016.

[47] J. Ziniel, L. C. Potter, and P. Schniter, "Tracking and smoothing of time-varying sparse signals via approximate belief propagation," in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*. IEEE, 2010, pp. 808–812.

[48] "Analog devices adxl345," https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf.

[49] G. M. Djuknic and R. E. Richton, "Geolocation and assisted gps," *Computer*, vol. 34, no. 2, pp. 123–125, 2001.

[50] "Open source routing machine (osrm)," http://project-osrm.org//, accessed: 2020-03-09.

[51] F. V. Diggelen, "System design & test-gnss accuracy-lies, damn lies, and statistics-this update to a seminal article first published here in 1998 explains how statistical methods can create many different," *GPS world*, vol. 18, no. 1, pp. 26–33, 2007.

[52] Y.-J. Gong, E. Chen, X. Zhang, L. M. Ni, and J. Zhang, "Antmapper: An ant colony-based map matching approach for trajectory-based applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 390–401, 2017.

[53] L. Luo, X. Hou, W. Cai, and B. Guo, "Incremental route inference from low-sampling gps data: An opportunistic approach to online map matching," *Information Sciences*, vol. 512, pp. 1407–1423, 2020.

[54] S. Singh, J. Singh, and S. S. Sehra, "Genetic-inspired map matching algorithm for real-time gps trajectories," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 2587–2603, 2020.