INF
Informatik

Bachelor-Thesis
Media and Communication Informatics

Evaluation of map matching approaches for vehicle trajectories on incomplete or incorrect street network data

Jonas Dominik Müller
Matriculation No.: 765428

First examiner: Prof. Dr. Benjamin Himpel

Second examiner: Prof. Dr. rer. nat. Uwe Kloos

Project supervisor: Timofey Khokhlovskiy

Submission date: 30.06.2022

Hochschule Reutlingen
Reutlingen University
Alteburgstraße 150 • 72762 Reutlingen • Deutschland • T. +49 (0)7121 271-0 • F. +49 (0)7121 271-1101 • www.reutlingen-university.de

Abstract:
A crucial step of transforming raw vehicle location measurements into usable data for many software systems is map matching, a process which combines them with knowledge of the street network to infer the traveled route. Since errors in street network data are common enough to be an issue for this process, several approaches for extending existing map matching algorithms have been proposed to handle them robustly.

This work presents how a selection of these proposals can be implemented into an existing commercial software system, taking the specific requirements of the use case at hand into account. The developed prototypes are evaluated using state of the art metrics and ground truth collection methods. A detailed comparison is then made between the prototype results and a standard Hidden Markov Model map matching implementation and the results critically evaluated.

Zusammenfassung:
Map matching ist ein wichtiger Schritt für viele Softwaresysteme bei der Verarbeitung von rohen Positionsmessungen von Fahrzeugen zu nutzbaren Daten. Hierbei werden bekannte Straßennetzdaten verwendet, um aus den Messungen die wahre Fahrzeugroute zu schließen. Fehler in diesen Daten kommen jedoch häufig genug vor, dass sie ein Problem für diesen Prozess darstellen. Daher gibt es bereits mehrere Ansätze, um bestehende Map matching Algorithmen zu erweitern und diese Problematik robust zu behandeln.

Diese Arbeit stellt vor, wie eine Auswahl solcher Ansätze für ein bestehendes kommerzielles Softwaresysteme umgesetzt werden können. Hierfür werden die spezifischen Anforderungen des Anwendungsfalls definiert und miteinbezogen. Die entwickelten Prototypen werden mithilfe anerkannter Metriken und Methodiken zur Erhebung von Referenzdaten evaluiert. Daraufhin werden die Ergebnisse der Prototypen mit einer nicht erweiterten Hidden Markov Model Map matching Implementierung verglichen und kritisch betrachtet.

Acknowledgements

This Bachelor-Thesis was created as part of the Media and Communication Informatics degree programme at Reutlingen University. The goal of the thesis is to independently solve a relevant problem using scientific methods within a fixed amount of time.

I want to thank my first examiner, Prof. Dr. Benjamin Himpel, for his suggestions on how to define the off-road classification problem.
Additionally, I want to thank both of my examiners for their support concerning this thesis.

For their valuable feedback on my concepts and ideas, I want to thank the product and development team at vialytics.
For helping me choose the final topic, as well as support on the company-side of the thesis, I want to thank Timofey Khokhlovskiy.

This thesis was written in cooperation with vialytics GmbH between March 1st 2022 and June 30th 2022.

Contents

List of Figures

List of Tables

1 Introduction

Location data is at the heart of many modern software systems, either as a way of tracking objects and users, for example in navigation software, or as an important part of giving meaning to other measurements.
Often, location data is recorded using consumer-grade global positioning system (GPS) receivers, such as ones integrated into smartphones. These receivers suffer from limited accuracy, signal noise, and often perform drastically worse in unfavorable conditions, such as when being surrounded by tall buildings.
As a result of this, raw location measurements often require additional processing, before being used further in an application.

Map matching is such a processing step, utilizing street network data to infer the actual route taken by a vehicle on roads. Even though the availability of such geographical data is wider than ever, errors and gaps in it cannot be ruled out. This then poses a problem for traditional map matching (MM) approaches, resulting in incomplete and inaccurate matching results.
This work aims to present, implement, and evaluate a selection of state-of-the-art proposals to address this issue by extending existing MM implementations.

1.1 Context

This Bachelor-Thesis was written in cooperation with *vialytics GmbH*. As such, a core motivation of this effort is the continued improvement of vialytics' map matching algorithm for processing collected road condition data.

Vialytics offers customers, usually municipalities and their building yards, a digital road management system, at the core of which lies automatic road condition assessment. Currently, this service is offered for local roads and bike lanes. For this purpose, customers are equipped with a smartphone, which they can mount to cars or service vehicles in order to record condition data as they drive on the roads in question. The smartphone is preinstalled with a custom inspection app, which records an image, timestamp, positional data, and accelerometer data at

a regular interval. A batch of these recordings is termed an *inspection* and contains at most one thousand recordings, termed *points*, before another has to be started.
Inspections are then uploaded by the customer, at which point they are processed by a pipeline of various steps to filter, clean, anonymize and analyze the data. As these include multiple machine learning (ML) steps, as well as other compute-intensive processes, it usually takes up to an hour for an inspection from upload to the finished result. These results are regularly inspected by quality assurance employees to ensure their validity despite the wide range of factors influencing inspections, such as weather condition at time of recording, mounting height of the smartphone and road type. The finished result, alongside all other uploaded inspections, is then available to the customer in a web-based management and geographic information system (GIS), an example of which can be seen in Figure 1.1. This displays an overall grade for the inspected street network, grades for individual streets and individual points, as well as an identification of which categories of road damages were detected.

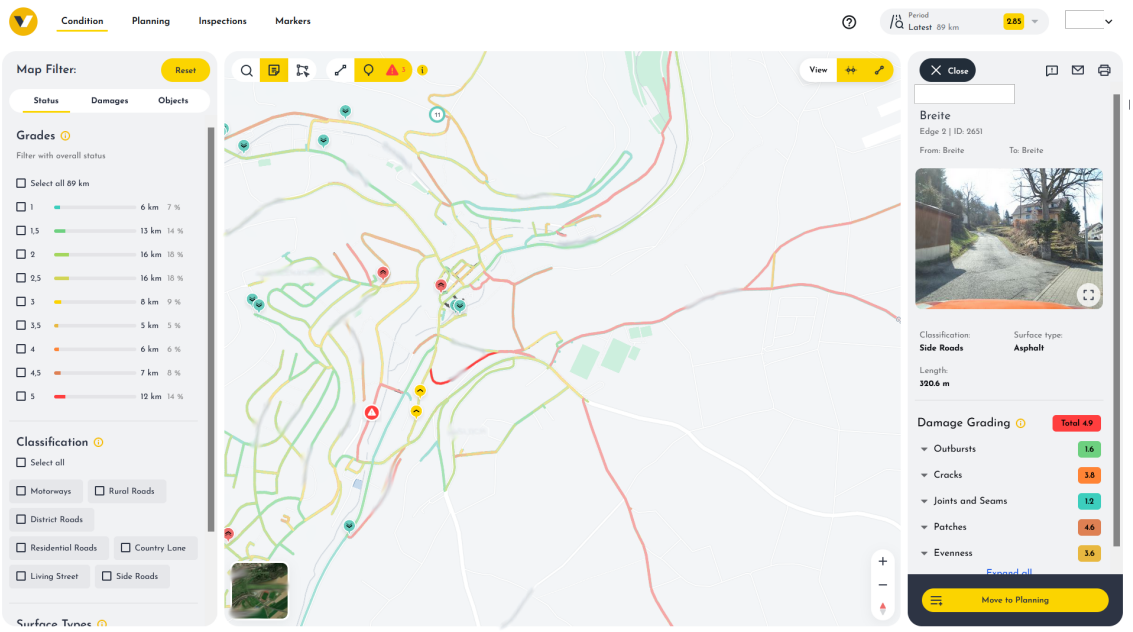


Figure 1.1: The condition view in the web-based management application of an example customer, showing the results of several inspections as colored-coded road segments.

Because of the nature of this inspection process, all points should optimally be mapped onto a road, as they are directly relevant to the condition of a road. Due to measurement errors discussed in further detail in Section 2.1, the raw location measurements do not fulfill this requirement by themselves, making MM a necessary step of the inspection processing pipeline.

1.2 Motivation

While unreliable street network data may be an issue for almost all GIS including vialytics' product, it is more noticeable for some use cases than others. A missing connecting road between residential blocks, shown in Figure 1.2, may only result in a minor miscalculation of travel times for route planning applications, but represents a major amount of inspection data for vialytics, which cannot be integrated into the existing road condition map and therefore not shown to the customer.

Figure 1.2: An example of an inspected street not being in OpenStreetMap (OSM). The left side shows the OSM base map, contrasted with Google Earth satellite imagery on the right. Overlaid are the measured vehicle location points in red and gray.

Additionally, demand has risen for inspecting cycle tracks as well as unpaved roads, such as dirt and gravel roads through fields. Both these categories have proven challenging for the existing MM implementation due to them generally being more affected by unreliable street network data. In addition to this, cycle tracks often run alongside regular streets as bike lanes and can have complex topology, especially in cities and large intersections. This exacerbates issues with the location measurement of vehicles driving on cycle tracks.

As a result of this, new solutions need to be found to fulfill the existing requirements to map matching at vialytics in the context of this increased demand for inspections on streets with less reliable geographical data.
These solutions should aim to make the existing MM implementation more robust to these inspections in a testable manner.

1.3 Research questions

Based on the context and motivation of this thesis, the following research questions are posed as the core of the research effort of this thesis:

RQ 1: How effective are extensions of existing map matching approaches for matching vehicle location data on streets with missing or erroneous street network data?
RQ 1.1: Which evaluation methods can be used to reproducibly compare map matching algorithms?
RQ 1.2: How can map matching approaches be extended to detect street network errors?
RQ 1.3: Which actions can a map matching algorithm take when detecting a street network error to produce a better result?

2 Foundations

The following chapter will lay the groundwork for the concepts discussed in Chapter 3:
Section 2.1 describes the nature of location measurement and the inherent uncertainties when using GPS devices. Section 2.2 defines the model used to describe street network data, which concrete data source will be used for the evaluation and which quality issues it may have. Lastly, Section 2.4 will relate the last two topics to the topic of map matching and highlight several approaches to solve it.

2.1 Vehicle location measurement

Location measurement can be achieved using a multitude of technologies such as using cell tower signals, known Wi-Fi hotspots and Bluetooth. These methods are often used jointly, for example in smartphones, and have advantages and disadvantages based on the use case and environment. Due to its prevalence in the field of vehicle tracking, GPS will be the basis for this section. However, the foundational concepts described here are shared by all of the aforementioned methods.

When measuring vehicle location, individual measurements are taken at a fixed or variable interval (*sampling rate*). In the case of GPS, the location determined using the time coded signals of at least four satellites is specified as coordinates, latitude and longitude, in a geographic coordinate system. A time series of these measurements, or *trajectory*, of a vehicle can then be represented by a polyline consisting of these measurements linearly connected.
When taking such measurements using GPS receivers, their accuracy must be taken into account in addition to the sampling rate when evaluating the quality of the overall trajectory.
While the sampling rate is generally known, the characteristics of the measurement error of a GPS device can vary. A widely used and effective assumption however, is that GPS errors are Gaussian [12, 2, 20] and that their horizontal error distribution is circular [2].

As a result of this, GPS accuracy is often stated as a standard deviation σ and is typically measured to be between 2 and 8 meters [2].

When tracking the route of a moving vehicle on the street network, the uncertainty added by the sampling interval between consecutive measurements can be visualized as an error ellipse. For determining which roads were actually traveled, one has to consider all streets within this error ellipse [2]. This ellipse, shown in Figure 2.1, has two sampled positions as focal points and a major axis of $2a = v_t * i$, where $v_(t-1)$ is the measured speed at time $t-1$ and i is the sampling interval. This reflects the fact that, with only the knowledge of the speed at t and assuming no acceleration, the vehicle can travel a distance of $v_t * i$. The distance of the focal points $2c$ is simply the great circle distance between the two positions. Therefore, the minor axis $2b$ can be calculated from the ellipse equation $b^2 = a^2 - c^2$.

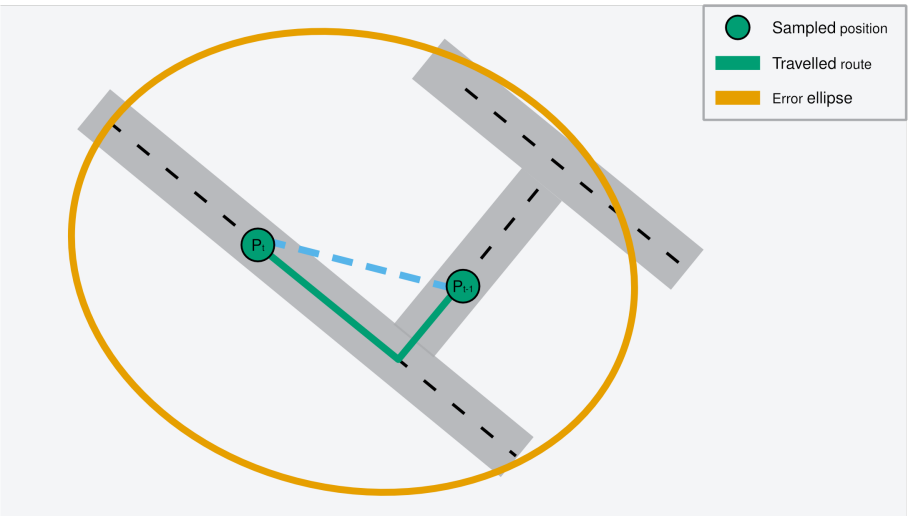


Figure 2.1: A depiction of a trajectory error ellipse, as defined by [2]. Scales of ellipse and distances are for illustrative purposes and do not follow the defined equations.

2.2 Street networks

Modern sources of map data encompass a variety of geographic information, such as buildings, political borders, and points of interest, however for the purposes of this thesis we will focus on the street network.

Street networks, as discussed in the following sections, are modeled as a set of polylines, consisting of a sequence of points, which in turn are defined by their

latitude and longitude. Both points and polylines can be associated with additional attributes, such as the road type they represent. Despite most street networks not being planar due to overpasses, tunnels, and other elevation changes, they are often simplified to being two-dimensional for analysis [1]. For purposes such as map matching, the elevation of polylines and points is only relevant at intersections. In these cases, the vertical relationship of intersecting lines can be represented by categorizing them into layers, where the ground-level line is assigned the layer 0, overpasses are incrementally assigned integers above $(1, 2, 3...)$ and tunnels incrementally below $(-1, -2, -3...)$.
The source for the implementation and comparison of map matching approaches used in this thesis will be the OSM project. OSM is a volunteered geographic information (VGI) project, meaning its data is not necessarily collected by experts, but rather by a heterogenous community of contributors. More specifically, it is a map-based and explicit VGI project, as it aims to collect map data and contributors explicitly submit data for this purpose, as opposed to implicit VGI like geographically tagged social media posts [14]. It integrates official map sources from government authorities [1] and community tools exist for automatic and manual quality control [2].
Fundamentally however, any source of street network data cannot be assumed to always be correct and VGI projects such as OSM may have additional issues compared to authoritative data. These include a lack of top-down quality insurance and knowledge of skill and precision of measurement tools of contributors [9].
Of the various quality parameters by which geographic information can be measured, this thesis will focus on the issues a lack of positional accuracy and completeness can cause for map matching. Here, *positional accuracy* refers to how close a mapped point is to a reference point, while *completeness* is a measure of what percentage of actual geographic features is included in the map data [9]. As intersections in the previously discussed street network model are defined by two or more polylines containing the same point, positional accuracy can not only cause misalignment of individual polylines, but also false identification of intersections.

[1] "Contributors - OpenStreetMap Wiki" https://wiki.openstreetmap.org/wiki/Contributors (accessed 21 June 2022)
[2] "Quality assurance - OpenStreetMap Wiki" https://wiki.openstreetmap.org/wiki/Quality_assurance (accessed 21 June 2022)

2.3 Street network inference

In the context of street network data and VGI, it is important to mention inference of streets and intersections. Compared to manually mapping features using GPS measurements, map inference methods can automatically generate street networks for entire cities based on aerial imagery [17].
An example of how this can be achieved is by using a convoluted neural network (CNN) trained on labelled images, such as SpaceNet training data [3], to perform segmentation and create a street mask [6]. This mask is then refined and ultimately used to extract the graph structure that is commonly used in projects such as OSM.

However, map inference based on aerial imagery also has accuracy concerns: Due to limitations in positioning of the aerial vehicle and map projection, images can be offset from their true position. When looking at global image services, such as Google Earth or NASA World Wind, similar quality issues to street network data collection arise when combining data from different satellites and aerial image sources. This can result in heterogenous positional accuracies depending on the country, region or even city, due to image stitching, a process where overlapping images need to be combined into one [7].
A study by Goudarzi and Landry found the distance of their ground truth GPS points in Montreal to be between 0.13 and 2.7 meters away from their matching points in Google Earth. Additionally, even with accurate imagery, complex intersections and other street structures with limited visibility remain an issue for inference algorithms [6].

As such, street network inference based on aerial imagery generally does not replace GPS mapping, but it is a useful tool for extending and refining street network data in projects such as OSM.

2.4 Map matching

Generally speaking, map matching matches the measured location data with the known street geometry in order to find the most likely corresponding route on the road network. The result of this process is often a list of street segments in the source map data that was traveled [12], but depending on the use case can also be a polyline of the route [8] or a set of points on the route that best match each GPS measurement [11]. A basic illustration of a measured trajectory and its map matched points can be seen in Figure 2.2.

[3] "Automated Road Network Extraction and Route Travel Time Estimation from Satellite Imagery" https://spacenet.ai/sn5-challenge/ (accessed 21 June 2022)

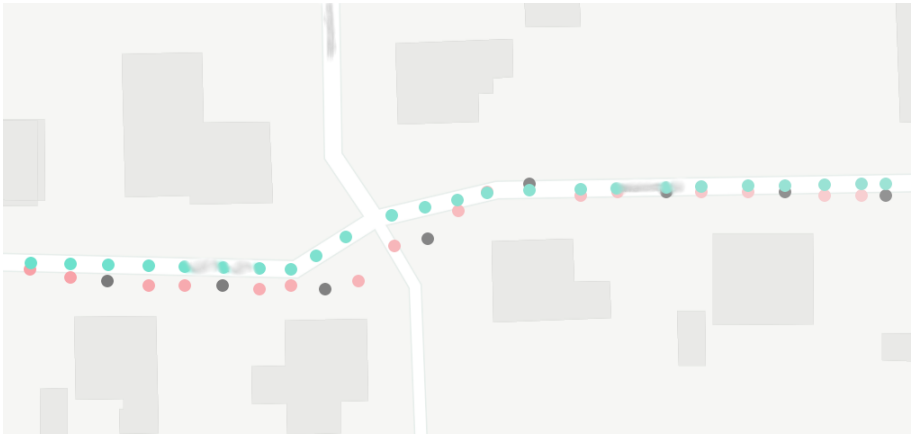


Figure 2.2: An example of matching GPS measurements (red circles) to on-road counterparts (green circles).

The most immediate approach to solving the map matching problem, especially with a high GPS sampling rate, would be to simply find the closest point on a road for each measured point. This idea, called *point-by-point nearest road matching* by Newson and Krumm, quickly results in inaccurate paths because of the inherent noise in commercial GPS receiver measurements [12]. Even errors on the scale of 5 to 10 meters are enough to cause matches on entirely different roads for measurements within seconds of each other. This problem is worsened by unfavorable GPS conditions and areas with dense road networks, where highways and roads can often be located in close proximity and even overlap without having actual intersections.
A key idea to handling these complications more robustly is the consideration of context when matching a location measurement, meaning taking previous and future measurements into account. Whether only previous measurements or the entire route trajectory is used for map matching determines if and algorithm is categorized as *on-line* map matching in the former, or *off-line* map matching in the latter case.

This thesis will focus on the off-line map matching category, of which three examples will be highlighted in the following sections with regards to the use case of map matching for vehicle trajectories:

i. A geometric solution, using *Fréchet distance* as a measure to find matching paths.

ii. A solution based on *Hidden Markov Models (HMMs)*, taking a more probabilistic approach.

iii. A solution based on using the concept of *shortest path*.

2.4.1 Global geometric distance

While the previously discussed nearest road matching might be considered a basic geometric approach for map matching, much more sophisticated solutions have been proposed by matching geometries.

One commonly cited approach was developed by Brakatsoulas et al. with the goal of developing a map matching algorithm that produces accurate travel times, rather than accurate individual matches. As such, they focus on matching the entire vehicle trajectory at once to possible route curves on the street network. While their paper also describes a greedy, local algorithm, we will be focusing on the two *global*, and therefore off-line, MM algorithms. Said algorithms compare candidate curves on the street network, with the measured trajectory using the *Fréchet distance* and *weak Fréchet distance* respectively [2]. The curves in this context refer to polylines, consisting of a finite number of discrete points. A frequently used intuitive definition for the Fréchet distance between two curves can be described like so:
Suppose a person is walking his dog, the person is walking on one finite curved path and the dog on another. Both are allowed to control their speed in order to keep the leash short, but they are not allowed to go backwards. Then the Fréchet distance of the curves is the minimal length of leash that is necessary for both to walk their curves.

The formal definition for two curves $f, g : [0,1] \to \mathbb{R}^2$ is as follows:

$$\delta_F(f,g) := \inf_{\alpha,\beta:\,[0,1]\to[0,1]} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\| \tag{2.1}$$

Where, in the case of the regular Fréchet distance, α and β range over continuous and non-decreasing reparameterizations with $\alpha(0) = \beta(0) = 0$ and $\alpha(1) = \beta(1) = 1$ only. For the weak Fréchet distance the same conditions apply, except the reparameterizations are allowed to be decreasing [2]. In the terms of the intuitive definition this means allowing the person and dog to backtrack in order to keep the leash short.

Compared to other distance measures between curves, such as the Hausdorff distance, this method takes the continuity of the curves into account. Whether the start of a vehicle trajectory is close to the end of a matched route is not nearly as useful for making MM decisions as whether the curves are continuously close. As such, Brakatsoulas et al. claim the Fréchet distance is a good fit as a metric for MM.
While the weak Fréchet distance can lead to vastly different results for specific, theoretical curve configurations, in the experiments of Brakatsoulas et al., they

produced identical results for their real-world test set of the Athenian street network and their recorded GPS trajectories. However, as weak Fréchet distance can be calculated using a less time complex algorithm ($O(mn\ log\ mn)$ compared to $O(mn\ log^2\ mn)$ [2]) it may be more efficient for MM applications.

When evaluating their results, Brakatsoulas et al. choose not to compare against a ground truth, but instead once again use the Fréchet distance of the matched route and the vehicle trajectory to compare their algorithms with each other. Additionally, the average Fréchet distance is compared, as by its nature of taking the maximum of a set of distances, the Fréchet distance is sensitive to outliers. This sensitivity is also brought up by Newson and Krumm as a weakness of geometrical MM approaches.

2.4.2 Hidden Markov Models (HMMs)

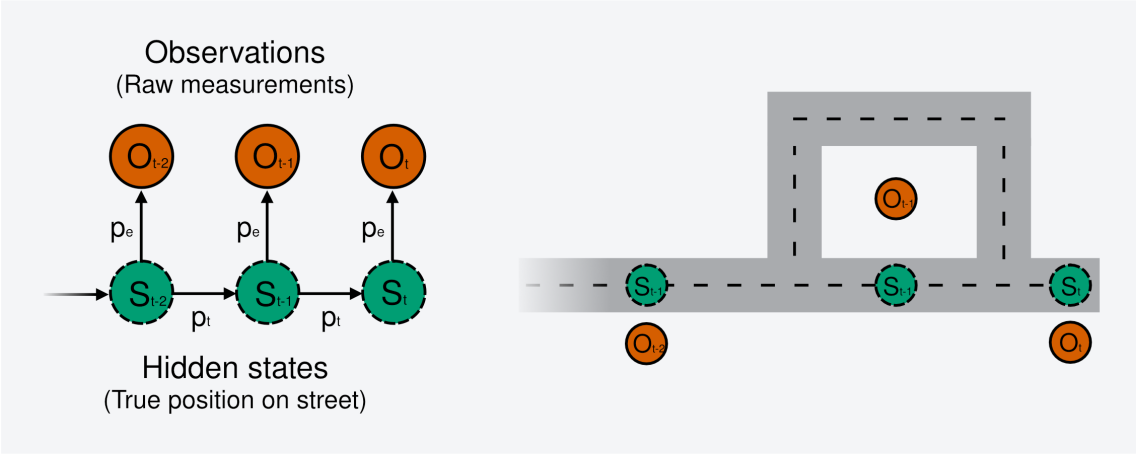


Figure 2.3: A visualization of the HMM graph for the map matching problem, where p_e is the emission probability and p_t the transition probability.

HMMs aim to describe a system with evolving states, which cannot be directly observed, called *hidden states*. Instead, *observations* are indirectly related to hidden states via probabilistic connections. The most likely path of hidden states that lead to the given observations can then be calculated using the Viterbi algorithm.
An HMM is often imagined as a graph with the states as nodes, connected by the probabilities as edges (Figure 2.3). At the core of evaluating the likeliness of a possible hidden state in the HMM model are *emission* and *transition* probabilities.

- *Emission probability* reflects the likeliness of an observation being caused, or emitted, by a hidden state.

- *Transition probability* describes the likeliness of a hidden state to transition to a different hidden state.

In the case of map matching vehicle trajectories, the true car trajectory can be understood as the unobservable hidden states and the series of GPS and other location measurements as the observations. This modeling of the MM problem allows a consideration of all possible paths through the street network given a set of possible hidden states or *match candidates* [12].
The selection of match candidate is left open in the HMM approach, but the commonly cited algorithm of Newson and Krumm places several restrictions on the process, such as only selecting candidates less than two hundred meters away from the GPS measurement and ignoring candidates with no reasonable routes from the measurement.
In the context of map matching, the emission probability of a match candidate corresponds to how likely it would be for the recorded GPS measurement to have occurred, had the vehicle been at the position of a match candidate. How likely it is for a vehicle to have moved from a previous position on the street network to a given next position is modeled by the transition probability. Both probabilities can be calculated in several different ways for the map matching problem and many different metrics have been proposed [18]. Similarly, whether the probability of consecutive matches should be multiplied or summed can change based on the implementation.
As an example for emission probabilities, Newson and Krumm utilize the great circle distance between a match candidate and a given observation, weighted by a probability density function based on the measured standard GPS error.
For transition probabilities, Newson and Krumm determine that, in general, the great circle distance between sequential observed vehicle positions and the shortest path distance on the road network between corresponding match candidates should be equal. Therefore, another probability density function can be formulated weighing the difference of these distances based on an experimentally measured expected distance.

Once match candidates have been determined and these probabilities calculated for each of them, the most likely path of matches and therefore the vehicles′ route on the street network can be retrieved using the Viterbi algorithm.

2.4.3 Shortest path

Because of its proven usefulness and adaptability, HMMs have become a popular way of modeling the map matching problem. This development is being questioned by Srivatsa et al., who first argue that the basic assumption behind using HMMs for map matching may be wrong and present a different solution based on

a shortest path approach.
Using GPS data from taxicabs in Shanghai, San Francisco, and Stockholm they calculate an error metric with respect to how much the data differs from expected Markovian behavior [15]. The resulting findings show a deviation from this expected behavior:
Especially when the taxicab was occupied with a passenger, the movement more closely followed the shortest path between its current location and the destination. Since HMM-based algorithms do not necessarily optimize for the shortest overall route and instead choose whichever match candidate might be the most likely based on the defined probabilities, they can predict overly long and complicated routes for datasets with low sampling rates.

Srivatsa et al. propose a solution which instead selects k shortest paths between trip endpoints and picks the most likely by summing the distance from each observation to a path and selecting the path with the smallest total. They test their algorithm on the taxicab datasets and compare it to a standard HMM map matching implementation, resulting in a 20% improvement in accuracy and a significant improvement in performance over it [15]. Notably, the higher k was set, the less error was measured and despite this introducing a longer compute time, a k of eight still resulted in a much shorter runtime and more accurate results than the HMM approach.

However, it is important to emphasize the nature of the used dataset:
Taxicab movements may be more likely to follow a shortest-path than drivers in other use cases. Their drivers have a clear incentive and the experience to navigate optimally between start- and endpoints, which can also be clearly determined and labelled for taxicab trips.

3 Related Work

In this chapter, we will explore the concepts relevant to the research questions of this thesis and which build upon the topics discussed in Chapter 1.

Firstly, to evaluate different map matching approaches, suitable metrics need to be found by which to measure and compare them, which is discussed in Section 3.1. Secondly, state of the art approaches for map matching with missing map data are presented in Section 3.2.

3.1 Comparing map matching algorithms

Evaluating the quality of map matching algorithm results is a complex issue for multiple reasons. Algorithms can have substantially different requirements they aim to fulfill, both in what data they are designed to take in and what qualities the result is supposed to have.

On the input side, while GPS accuracy can generally be considered to be within similar ranges, sampling rates depend entirely on given use cases. Algorithms designed for intervals of 10 seconds often choose different matching methods from algorithms designed for intervals in the minutes and often do not perform well on significantly lower or higher sampling rates [19].
On the output side, most MM approaches aim to generate a more accurate representation of the vehicle trajectory, but by what this is measured, differs. Some aim to create a route that most accurately represents the travel times on specific street segments [2, 12], meaning the sequence of street segments traveled and their entry and exit times. Others think of MM more as ′snapping′ all individual GPS measurements to streets [11, 13].
As such, multiple different metrics are chosen to compare the results of different algorithms, some focusing on the overall matched route, like the total difference in length between matched route and trajectory [13] or Fréchet distance [2], and others more on individual measurements, such as the average distance between GPS points and the matched route. Due to its comparatively simple implementation and computational complexity, the discrete Fréchet distance is

also a good fit for calculating a similarity measure between polylines based on location measurements and results in a good approximation of the continuous Fréchet distance [5].

In addition to comparing MM results with each other, some researchers establish a source of ground truth, against which the MM prediction is compared [18, 12, 3]. This however, raises the question of how to define and collect ground truth for map matching problems, as the ′actual′ trajectory of the vehicle can never be perfectly known. One approach is to collect both a high-quality and low-quality dataset of a vehicle trajectory [13], with quality in this case usually referring to sampling rate and location accuracy. This can be achieved by either using measurement devices with different accuracies, or by artificially adding noise to a collected dataset. The latter option having the benefit of requiring little to no manual work and therefore allowing larger test sets. In either case, the low-quality data is used as input data for the MM algorithm and the result is compared to the high-quality data, taking the role of ground truth.
While useful, this methodology may not be sufficient for use cases aiming for a high-quality at high sampling rates, as the ground truth in these cases is still a potentially noisy GPS trace. The accuracy of these traces can be kept consistent by pre-planning a route and using that to verify the measurements [19], but this introduces additional manual work and may result in less natural trajectories for a given use case.
A different approach is manual map matching [10, 13, 12]. Using the assumption that humans can map a set of GPS points to streets with a high degree of accuracy, given a high enough sampling rate and visualization aids [10]. Naturally, collecting ground truth using manual matching is costly, but good tooling and aids such as already providing the human matchers with a machine-matched route that they can use as a baseline, can increase the speed and efficiency of this method.

No matter how it is collected, ground truth data allows the use of well-known evaluation methods such as precision and recall to evaluate the performance when mapping measurements to specific road segments. One approach used by Wei et al. and Chao et al. compares the length of road segments shared by the ground truth data and the matched route [19]:

$$recall_l = \frac{||Truth \cap M||}{||Truth||}, \quad precision_l = \frac{||Truth \cap M||}{||M||} \tag{3.1}$$

Where $Truth$ is the ground truth set of road segments, M is the set of road segments in the matched route and $||*||$ sums the length of the segments of a set.

Circling back to research question 1.3: Given that we will be evaluating algorithms using the same inputs and outputs, a method of collecting ground truth data can be chosen and applied to create a test set. This can then be used to

calculate metrics which allow for comparisons between algorithms based on the similarity of their generated routes compared to the measured trajectory and the ground truth route.

3.2 Map matching with missing street network data

As discussed in Chapter 2.4, there are a variety of map matching approaches, but all mentioned examples implicitly assume the given street network data to be complete and correct. A key idea to tackle the issue of incomplete street network data is therefore to loosen the assumption that the map matched route has to be entirely contained in it.

In this chapter, we examine how state of the art approaches extend existing MM approaches based on this idea and how they decide when to consider such *off-road* matches.
All the here mentioned approaches use HMM map matching as their basis. This is not the result of an intentional selection for this thesis and instead a reflection of the available literature.

3.2.1 Using location measurements as match candidates

Haunert and Budig extend an HMM-based map matching algorithm by adding an additional *off-road* match candidate to each GPS point in the trajectory [8]. This off-road candidate has the same position as the GPS point itself and allows the algorithm to use the measured positions for the map matched path when *on-road* candidates are unlikely.
The crucial issue to solve with adding off-road candidates is calculating their probabilities for the HMM. Using the standard emission probability measure of great circle distance between match candidate and GPS point is no longer useful, as both are in the same location. Because of their nature, there is also no route on the street network leading from a preceding or following match candidate to a given off-road candidate, so the shortest path distance cannot be used for the transition probability.
While Haunert and Budig do not describe an adjustment to the calculation of emission probabilities for off-road candidates, they do introduce a method with several adjustable parameters to calculate the cost of a transition:

- Off-road path sections have a separate cost factor per unit traveled.

- If transitioning between two off-road candidates, the path length is the great circle distance between the two locations.

- If transitioning between an off-road candidate to an on-road candidate, or vice-versa, the path length is the combination of the on- and off-road segments connecting the two with minimum cost.

In addition to this, a factor ϕ is conditionally applied to the probability of transitions between on-road candidates while a different factor ψ is applied when transitioning from an off-road to an on-road candidate. Both factors so $\phi > \psi$ is true, reflecting the assumption that most trajectories will be contained in the given street network data.

While the evaluation of this approach is limited to a small number of experiments and non-systematic evaluation of the quality of the matched route, it does show results which would not be possible with a regular MM approach. Several parts of their tested route are on streets not present in the used OSM dataset, meaning the trajectory could not have been matched as a whole without adding off-road candidates. Additionally, Haunert and Budig show that the introduced parameters affecting the likeliness of choosing off-road candidates enable a degree of adjustment depending on the use case.
A drawback of this method is that the resulting matched route in part includes raw and potentially noisy GPS data. The approach is also limited in the sense that it describes no method by which future MM processes could make use of the missing streets recognized in a previous process.

3.2.2 Replacing missing roads with simplified GPS geometry

In a similar approach to extending HMM map matching, Torre et al. propose viewing the problem of map matching with missing road data as a hybrid between map matching and map building [16]. Instead of always considering the GPS position of observations as match candidate, their algorithm only uses GPS points when it detects a missing segment. This is achieved by setting a fixed cutoff distance, where if there are no candidates within a radius of d around the current observation, it is considered to be an error in the street network data and the algorithm will start adding new road geometry:
First, the start of the new street segment to be added is selected by finding the last recorded observation which was within a standard deviation s of the GPS error away from any on-road candidate. In a similar fashion, the end of the segment is picked by finding the first following observation within s of an on-road candidate. This ensures the street segment will be connected to the existing

street network. Once the endpoints are determined, the street segment is constructed by connecting the GPS measurements in between in sequence. As this segments′ polyline then consists of raw measurements, Torre et al. suggest simplifying the segment geometry before adding it to the street dataset, but do not mention a concrete method to accomplish this. Once the new geometry has been added, the algorithm starts over at the first observation where the new segment would contain match candidates and proceeds.

Torre et al. evaluate their algorithm using 128 GPS tracks of bike-focused VGI street network data for Minneapolis. To evaluate the street segment reconstruction, the matching algorithm is run once with the whole street network dataset. Then random segments of the dataset which appeared in the matched path are removed and the algorithm is run again. As such, the removed street segments serve as a kind of ground truth against which the results of the second run are compared, as the algorithm should optimally recreate the missing segments.
Only a limited amount of evaluation is shown using this comparison and only the quantity of the generated street segments is considered, not the quality.

Despite the shortcomings of its evaluation, the approach developed by Torre et al. is a relevant proof of concept of how to view missing road network data not just as individual points, but as geometries with connections to the existing street networks. This allows for a more complex analysis of the problem and brings up the application of methods such as geometry simplification to reduce measurement noise.
Additionally, it demonstrates how map matching results can be used to enhance the existing street network dataset.

3.2.3 Combining multiple models

In an effort to refine previous efforts like the ones in sections 3.2.1 and 3.2.2, Murphy et al. propose applying a semi-interacting multiple model filter (sIMM) consisting of both an on-road and an off-road map matching model [11]. Their solution is formulated in a generalized way, allowing combination of any two models suitable for map matching and free-space filtering, but for implementation and testing purposes they chose an HMM map matching algorithm solution as the on-road and a closed-form Kalman filter as the off-road model. As they propose a semi-interacting model for performance reasons, only the Kalman filter can influence the HMM and not vice versa.

In practice, this means the output of the Kalman filter for a given observation can be used as a match candidate in the map matching process. This bears some similarities to the previously mentioned approaches, with one core difference:

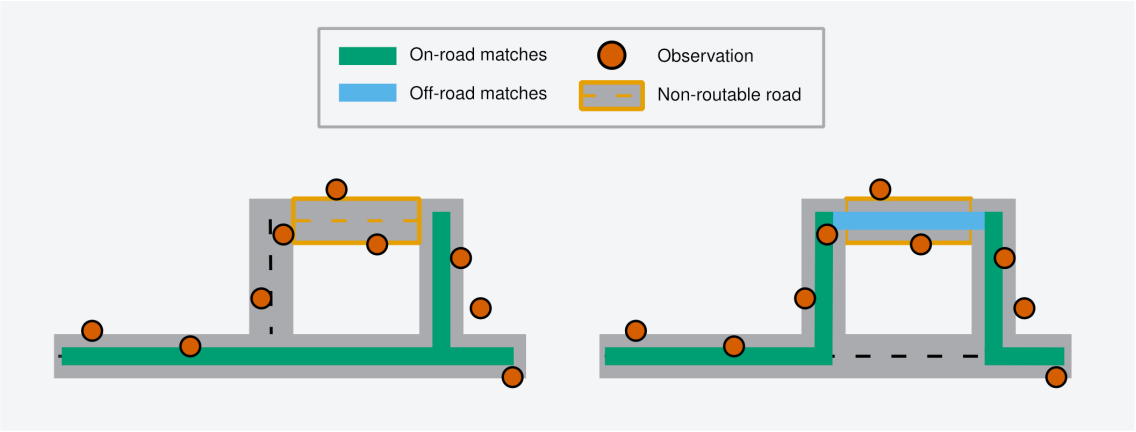


Figure 3.1: Standard HMM (left) and sIMM implementation results (right) in a situation with one road wrongly marked as non-routable. (Example from [11]).

Instead of directly using the raw GPS measurements (3.2.1) or applying a simplification algorithm to their trajectory (3.2.2), this approach continuously uses the off-road model to track the vehicle trajectory and generate predictions to be used as off-road candidates. Similar to the method of Haunert and Budig the transition probability calculations proposed by Newson and Krumm are adjusted to function with off-road candidates.

Murphy et al. do not provide any quantitative test results for their implementation. Instead, they include a selection of pictures comparing the MM results of the sIMM with the output of a regular HMM implementation working with an OSM dataset. Figure 3.1 shows one of these examples in a simplified manner: In this case, the standalone HMM implementation cannot compute a trajectory over the segment wrongly marked non-routable, as it is designed with the assumption that the given street network data is entirely correct, resulting in it matching to the wrong route. On the other hand, the sIMM implementation matches using its on-road model up until the missing segment and then switches to the now more likely off-road model trajectory, before switching back to on-road matches once close enough to a routable road segment.

In a different example, they apply the algorithm to a large set of driver location data collected from a ride-for-hire app, tracking the number of times and location, where the output of the off-road model was used. Using this analysis approach, Murphy et al. claim to have identified several hotspots with a high-density of off-road traces, pointing towards errors in the used OSM street network data.
This, once again, highlights how map matching and map-building are related, as well as how an analysis of multiple trajectories on the same roads can result in clearer indication of errors.

4 State analysis

To get an overview of MM issues at vialytics, several database queries were made to count the number of inspections in specific MM categories. This was done

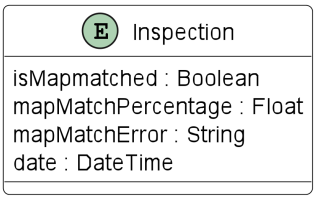


Figure 4.1: Entity-relationship diagram of the inspection entity at vialytics.

using the *isMapmatched* and *mapMatchError* attributes of *Inspection* entities (Figure 4.1):

- *isMapmatched* is true if the MM process produced any matches.
- *mapMatchError* is true if the MM process encountered issues or could not find a match for any point.

Which are used to build the inspection categories listed in Table 4.1.

Category	isMapmatched	mapMatchError
Fully matched	true	false
Partially matched	true	true
MM Failed	false	true
Not processed	false	false

Table 4.1: Inspection MM categories.

The *Not processed* category refers to inspections that exist in the system but have not yet passed the steps before map matching in the pipeline and will not be discussed further.

In total, 9423 inspections were selected by their creation date in the database from the following three 30-day periods (ISO 8601 dates):

- From 2021-12-25 to 2022-01-24.
- From 2022-01-25 to 2022-02-24.
- From 2022-02-25 to 2022-03-27.

The resulting proportions of inspections in each category can be seen in Figure 4.2 and show that seven hundred, or around 7 %, of the inspections could not be map matched without issues. While there are some failure reasons which do not need quality assurance (QA) involvement, such as the inspection not having the minimum number of points, this is a significant number of inspections which need to be manually reviewed.

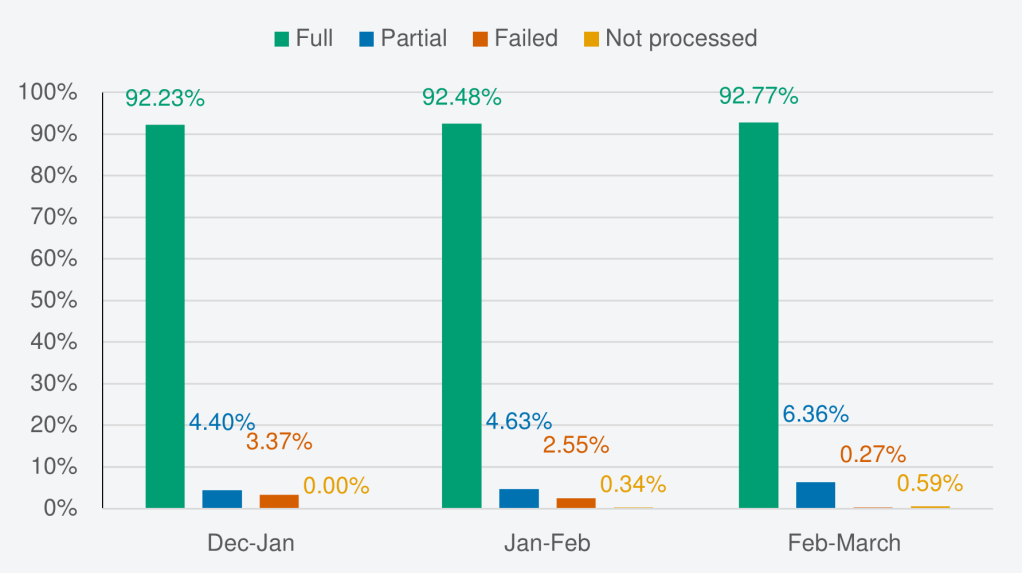


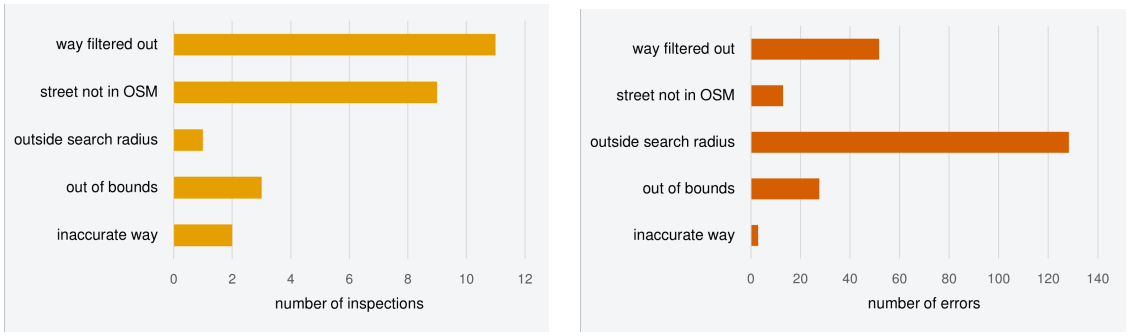
Figure 4.2: Percentage of inspections per category per time frame.

As these results merited further investigation, the next step taken was to manually review a smaller set of inspections with MM errors in order to categorize the failure reasons. In total, 328 inspections were selected by their creation date in the database from the three-day period of 2022-03-26 to 2022-03-29. Of those, 26 were partially matched and zero had failed map matching. These 26 inspections were then divided into the following categories:

- *way filtered out*: A matching OSM street can be identified, but it is of a type which vialytics did not store.

- *street not in OSM*: A matching street can be identified on satellite imagery, but it is not in OSM.
- *out of bound*: The inspection was partly or completely conducted outside the predetermined customer bounds.
- *inaccurate way*: A matching street can be identified on satellite imagery, but it is inaccurately mapped in OSM.
- *outside search radius*: A matching street can be identified, but it is further away from the measured point than the 30 m search radius.

The results in Figure 4.3 show that street network data *completeness* issues, whether caused by a lack of data in OSM or lack of OSM data saved in the vialytics database, was the main source of errors in the reviewed set. This is true for both how often it was the main error reason as well as how many points could not be matched for each of their occurrences. While the *out of bounds* error affected the most points per occurrence, they were due to a now deprecated additional geo-fence filter which was applied in a previous pipeline step and is not relevant for the purposes of this thesis.
Issues of *positional accuracy*, captured by the *inaccurate way* category, were a comparably minor issue alongside overly restrictive search radii for matches.



(a) Inspections categorized by main MM error reason

(b) Number of points affected per error occurrence

Figure 4.3: Results of manual review of 26 partially matched inspections.

While this analysis was limited in scope and depth, it gave enough confidence in the core motivation of this thesis to justify going ahead with the exploration of map matching approaches which are designed to recognize incomplete street network data.

5 Requirements analysis

Having laid out the context and foundations for this thesis, we will now describe the requirements which are relevant for the development of both the evaluation and the map matching prototypes.

As these two topics encompass different responsibilities, they will be considered as two separate, but related software systems.

5.1 Prototype implementation

To limit the scope of the implementation and variables needing to be considered in evaluation, all algorithm implementations will be embedded in the same core HMM map matching application. This application, itself based on the existing MM module at vialytics, will contain logic for querying all necessary data, including on-road match candidates, an extensible HMM implementation and functions for tracking and calculating evaluation metrics.
Prototypes, implementing approaches for taking off-road candidates into account will then be built on top of this foundation. The prototype modules are allowed to differ in how they select off-road candidates, how they score candidates and how they handle breaks in the HMM, such as when no candidate could be found within a certain range. Table 5.1 lists these requirements formally.

ID	Requirement
P.F.1	The prototypes must be implemented as modules of the same base MM application.
P.F.2	The prototypes must include off-road candidates in their model.
P.F.3	The base MM application must be responsible for all on-road HMM MM functions.
P.F.4	The base MM application must allow off-road candidate selection, scoring and HMM breaks to be changed by the prototype modules.

Table 5.1: Functional prototype implementation requirements.

5.2 Data structure

An important consideration when developing a map matching implementation is how the necessary input data can be accessed and transformed into a workable format for the MM algorithm. Due to factors outside the scope of this thesis, the vialytics system generally stores customer data, which is actively being worked on in a MongoDB, while largely read-only customer data is stored in a separate PostgreSQL database. An entity-relationship depicting the simplified data structure of inspection data at vialytics can be seen in Figure 5.1.

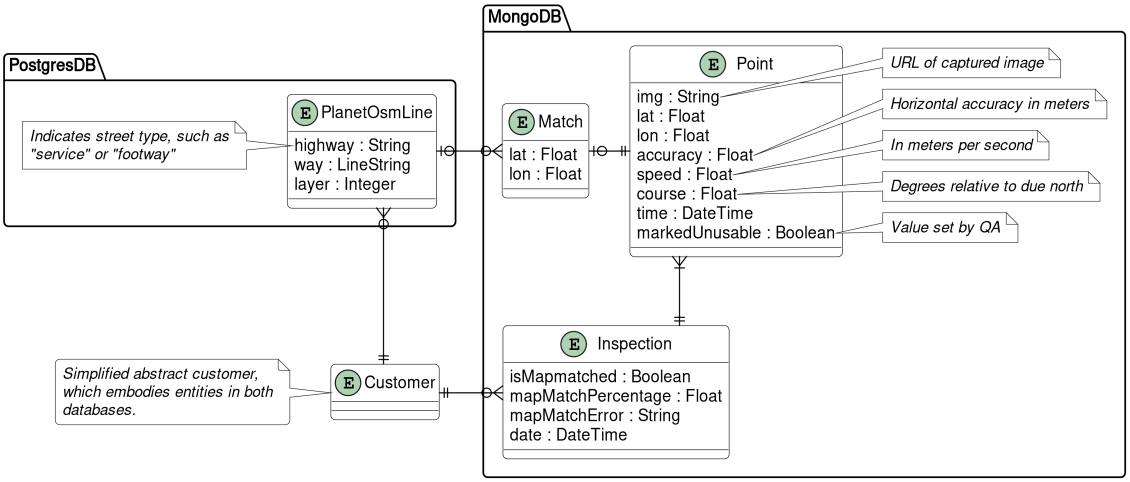


Figure 5.1: Entity-relationship diagram of inspection data at vialytics.

This data separation is relevant when map matching an inspection, as the process requires data to be fetched and related across both databases.
Match candidates for each *observation* can be queried using functions provided by PostGIS [1], a PostgresDB extension which allows for geographic queries. However, to select *match candidates* using the correct street network data, the customer to which the *Inspection* entity belongs must first be queried to then query candidates from *PlanetOsmLines* of the appropriate schema in the PostgresDB. Due to this chain of queries across different databases, the computational complexity of the *match candidates* geographic query and the transmission delays associated with each query, especially when running them locally from a development machine while prototyping, caching the required data for the purposes of the thesis is preferable. Keeping a local copy of all necessary data also ensures that no potential changes to the remote data during evaluation can affect the results.

To fit into the existing inspection pipeline, the output of all MM prototypes must

[1] "About PostGIS | PostGIS" https://postgis.net/ (accessed 21 June 2022)

be a set of *Match* entities which map exactly one point to one *PlanetOsmLine* entity.

Thus, we define functional requirements for the prototypes listed in Table 5.2 and for the evaluation framework listed in Table 5.3.

ID	Requirement
P.F.5	The base MM application must be able to collect and use all relevant MM data from the vialytics databases.
P.F.6	The base MM application and therefore all prototypes must produce results in the form of a list of *Match* entities.

Table 5.2: Data structure and access functional requirements for prototypes.

ID	Requirement
E.F.1	The system must be able to store all relevant MM data locally.
E.F.2	The system must be able to run all prototypes using only local data.

Table 5.3: Data structure and access functional requirements for evaluation.

5.3 Development process

As the focus of this thesis is to prototype and evaluate, being able to iterate quickly upon implementations and having access to debug information as needed is key. However, the previously existing map matching implementation and QA tools at vialytics were developed as modules of existing monolithic services. While this allowed the reuse of database connection logic, it poses various difficulties for prototyping and the development team decided that creating a separate service with the sole responsibility of map matching is preferred (Table 5.4).

Alongside this map matching service, a separate frontend project should be created for the purposes of evaluating the prototypes, as this would be outside of the scope of the existing QA tools (Table 5.5).

ID	Requirement
P.F.7	The system must be implemented as a service whose sole responsibility is map matching and to provide data for evaluation.

Table 5.4: Development process functional requirements for prototypes.

ID	Requirement
E.F.3	The system must be implemented as a service whose sole responsibility is evaluating the map matching prototypes.

Table 5.5: Development process functional requirements for evaluation.

5.4 Location measurements

The location data which needs to be map matched is recorded by a smartphone (iPhone 12 or newer) mounted to the windshield of the inspection vehicle. To ensure image stability and safety, drivers are instructed to drive between 15 and 60 km/h and the smartphone app aims to store an image alongside location data every three meters. Individual measurements are stored as *point* entities in the database, as can be seen in Figure 5.1.
As the speed of the vehicle is not constant and may be outside of the recommended range during turns, the sampling rate of points varies. However, this does not mean the location data is queried at this rate. Instead, the app is notified by the operating system when a location change event is triggered, which can happen at a different varying rate of every 1 to 5 seconds, depending on GPS signal strength and operating system power management logic. In order to still get a location for each captured image, their position, speed, and accuracy is linearly interpolated in the app between consecutive location measurements using the timestamps of when they were taken.
In addition to GPS, the smartphones make use of cell signal, WiFi and Bluetooth in order to determine device location, speed and an estimate of the location accuracy given in a radius of an uncertainty circle around the given measurement.

Taking this variability into account, we define the functional requirements for the evaluation listed in Table 5.6.

ID	Requirement
E.F.4	The system must store accuracy and speed for inspections.
E.F.5	The system must calculate and display accuracy and speed distribution for inspections.

Table 5.6: Location measurement functional requirements for evaluation.

And we define the functional requirements for the prototype implementations in Table 5.7.

ID	Requirement.
P.F.8	The system must not expect a constant sampling rate.
P.F.9	The system must be able to take variable accuracy into account.

Table 5.7: Location measurement functional prototype requirements.

5.5 Street network data

As mentioned before, the relevant street network data source used for map matching is OpenStreetMap.
Data for supported countries is downloaded on a weekly basis from an OSM server in order to be filtered and stored in a dedicated database for further use. During the filter step, only features with a specific type are kept. This is done using the *highway* attribute which all OSM line entities have and classifies them as one of several street types, such as *footway*, *service*, or *cycleway*. The filter step is taken to limit the amount of storage needed, as the OSM source file contains more geographic information than needed, such as buildings and land use information.

One of the first steps when a customer uploads a new inspection, is a process which fetches relevant street network data of the matching country from the database. This is accomplished by querying all streets which intersect with an elliptic buffer of a set radius of sixty meters surrounding the GPS trajectory of the inspection as well as all streets intersecting with those streets. The resulting streets are then stored in the PlanetOsmLine table of the customers' database

schema, as seen in Figure 5.1.
Both this and the filtering step greatly limit the amount of data that may need to be traversed for queries during further inspection processing steps, but also constitutes a limit of what data is available to a map matching process and has to be taken into account.

5.6 Test set creation

The prototypes will be evaluated using a set of inspections for which ground truth data will be collected. In order to be relevant to the research question 1.3, these inspections should contain cases with and without map matching issues related to missing street network data. The latter set will be used as a regression test to evaluate the prototypes' performance on inspections where the existing MM implementation is already sufficient.
Collecting a truly representative sample for the vialytics use case is challenging due to the data collection method as well as the customer base changing with time. Additionally, preparing ground truth data for each inspection is a time-consuming task, especially within the time constraints of this work.
Therefore, the state analysis of map matching of recent inspections in Chapter 4 should be combined with the experience of QA and customer success (CS) experts to curate a limited, but relevant sample of inspections for the test set (Table 5.8).

ID	Requirement	Verification
E.NF.1	The inspection set should represent relevant MM issues at vialytics.	Inspection set has been collected by interviewing QA and CS employees.

Table 5.8: Test set nonfunctional requirements for evaluation.

Due to the context of the evaluation and the limitations of other methods of collecting ground truth as described in Section 3.1, ground truth data should be able to be created from existing inspection data using manual matching. Additionally, the evaluation application should simplify the creation of the ground truth set. As such, it should display the information which is key to map matching: The measured vehicle locations and the relevant OSM lines saved in the vialytics database. Since on-road ground truth matches need to include a reference to the OSM line they are on, matches should snap to nearby lines when moving them on the map. Additionally, it should be possible to visualize the segments

of the matched route by their associated OSM line in the application. This would help to visually verify that there are no accidental outliers, as may be possible at intersections. However, as off-road sections should also be identified, this information should be displayed on a satellite map, allowing an estimation of the vehicle route independent of the known street network by positioning matches on visible tracks.
To ease ground truth creation for on-road sections, the system should allow the matches generated by a MM algorithm to be used as preliminary ground truth matches. These can then be manually verified, adjusted, and enriched with off-road matches when necessary. Following this, we define the requirements in Table 5.9.

ID	Requirement
E.F.6	The system must have a view for creating ground truth for an inspection.
E.F.7	The system must display location measurements and known OSM lines in ground truth view.
E.F.8	The system must display a satellite map in ground truth view.
E.F.9	The system should allow using MM results as a starting point for ground truth.
E.F.10	The system must allow creation and moving of matches for all location measurements in ground truth view.
E.F.11	The system must allow snapping of ground truth matches to nearby OSM lines while moving. This adds the lines' OpenStreetMap ID to the match information.
E.F.12	The system should visualize the matched route divided up by segments of matches with the same OSM ID.

Table 5.9: Test set creation functional requirements for evaluation.

5.7 Result evaluation

As discussed in Section 3.1, there are multiple approaches for comparing MM results depending on one's access to ground truth data and which aspects are of interest.
For the purposes of this thesis, we are interested in how well the prototypes can detect sections of the vehicle trajectory outside of the known street network and how this knowledge affects the matched route compared to the ground truth route. Additionally, the prototypes are to be evaluated as potential replacements

or extensions for existing MM implementations. Therefore, it is relevant whether this off-road detection impacts the prototypes' ability to match sections on the street network compared to regular HMM map matching.
Table 5.10 defines these requirements.

ID	Requirement
E.F.13	The system must evaluate the prototypes' ability to detect off-road sections.
E.F.14	The system must compare the matched route holistically to ground truth.
E.F.15	The system must compare prototype results to a regular HMM MM implementation.

Table 5.10: Result evaluation functional requirements.

6 Concept

As described in the requirements analysis, two systems with separate responsibilities will be developed.

Having defined the key requirements for the MM prototypes and their evaluation, this chapter will present concepts for both.
Firstly, the concept for the evaluation application is described in Section 6.1, followed by the prototypes and the MM base application in Section 6.2.

6.1 Evaluation

The evaluation application concept is split into two key parts. Section 6.1.1 defines the metrics by which the prototypes will be evaluated in order to answer the main research question RQ.1, while Section 6.1.2 defines the core evaluation process to be implemented, from gathering data to visualizing results.

6.1.1 Metrics

Implementing the requirements defined in Table 5.10, we define the following metrics to be evaluated for each prototype (Table 6.1):

Description	Symbol
Matched points count	n_m
OSM ID matching accuracy	a_l
Discrete Fréchet distance of matched route to ground truth	δ_{mg}
Discrete Fréchet distance of matched route to measured trajectory	δ_{mt}
Computation time	Δt_c
Average point to match great circle distance	$\overline{d_{mt}}$
Off-road precision	$precision$
Off-road recall	$recall$

Table 6.1: MM result metrics.

Where the OSM line matching accuracy is defined as

$$\frac{|\{matches\ with\ correct\ osm\ line\}|}{n_m} \tag{6.1}$$

and $\{matches\ with\ correct\ osm\ line\}$ are all matched points where the MM implementation assigned the same OSM ID as the respective ground truth point. Important to note is that $n_m < n_p$ may be the case for the *base* prototype on off-road inspections. This is due to the fact that it has a limited search radius for finding on-road candidates and if a point has no candidates, it will be skipped and have no match. As the required MM result is a list of matches (P.F.6), we change the definitions for $precision$ and $recall$ from the length-based measurement of 3.1 to counting individual matches that have been classified correctly or incorrectly according to Table 6.2.

$$recall = \frac{TP}{TP + FP}, \quad precision = \frac{TP}{TP + FN} \tag{6.2}$$

To put these metrics into context, attributes of the inspections in the test set which are relevant to map matching, such as reported location measurement accuracy and speed should also be tracked.
Using ground truth, data we can also count the number of off-road points in a

		Algorithm match	
		Off-road	On-road
GT match	Off-road	TP	FN
	On-road	FP	TN

Table 6.2: Off-road classification confusion matrix, where matches are off-road if they have no OSM ID.

given inspection, calculate the Fréchet distance between the measured trajectory and the ground truth route and determine the precision and recall when classifying positions as off-road and on-road. However, instead of calculating the continuous Fréchet distance, the discrete Fréchet distance [5] will be used. The complete list of inspection metrics is detailed in Table 6.3.

Description	Symbol
Average reported horizontal location accuracy	$\overline{\sigma}$
Standard deviation of reported horizontal location accuracy	SD_{σ}
Average reported speed	$\overline{v}$
Inspection length	l
Point count	n_p
Off-road point count	n_o
Discrete Fréchet distance of ground truth to trajectory	δ_{gt}
Average point to ground truth match great circle distance	$\overline{d_{gt}}$
Average time difference between consecutive points	$\overline{\Delta t_p}$

Table 6.3: Inspection metrics.

6.1.2 Evaluation process

In order to fulfill the evaluation requirements defined in Chapter 5, a concept was developed for a frontend application to visualize and interact with MM data, as

well as the necessary components in the MM service to support it.

The main processes of the evaluation can be grouped into three tasks (Figure A.1):
Firstly, during *data collection* all data necessary for map matching and the further evaluation steps must be fetched from vialytics services and these *RichInspection* entities stored in the frontend application to fulfill requirement E.F.1. These include all points of the inspection, all nearby OSM lines that are considered for match candidates as well as match candidates for each point. This stored data is then visualized and displayed to the tester for *ground truth creation* as described in the requirements 5.9.
To facilitate a consistent process, the following rules are applied to the ground truth creation:

i. An on-road ground truth match should be at the closest point to the measured location on the appropriate OSM line.

ii. If there is an offset between satellite imagery and map data, align ground truth matches using map data.

iii. If there is no fitting street in the map data, off-road ground truth matches are positioned with the help of satellite imagery where possible.

iv. If there is neither map data nor fitting satellite imagery, the off-road ground truth matches have the same position as the measured locations.

The additional ground truth data is then saved directly into the respective RichInspection entity and represents a complete test set entry.
Lastly, in the *prototype evaluation* step the MM service is called with a complete RichInspection and executes all prototypes to be evaluated, requiring no further queries to other services. The metrics defined in the previous section are then calculated and returned by the MM service alongside the matching results to the frontend application, where they can be presented to the tester.

A sequence diagram of the entire process is depicted in Figure A.1.

6.2 Map matching service

The concept for the map matching service encompasses the base application, which serves as the framework for the prototypes and includes its own HMM map matching implementation used for reference, as well as the concrete prototype modules.

6.2.1 Base application

While the previously existing MM implementation at vialytics already fulfills requirements P.F.3, P.F.5 and P.F.6, it is integrated as part of a different service with other responsibilities and does not allow the integration of the prototype modules. Therefore, a new base application will be implemented, porting the reference MM logic into a more extensible framework.
Although the base application is auxiliary to the MM aspects which are to be evaluated, it provides much of the functionality which is necessary for the general functioning of the service. Accordingly, it also represents an equal test bench for the prototype modules to be evaluated upon.

Defining the functionality and data model more concretely, the MM service will implement an HMM and Viterbi algorithm which can work with the input and output data as defined by the requirements. Following this, the appropriate map matching entities are defined as shown in Figure 6.1.

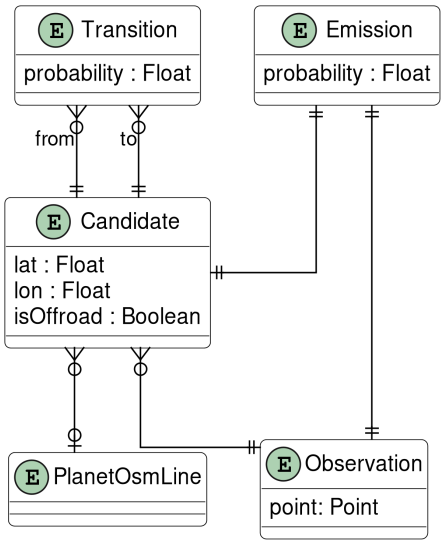


Figure 6.1: Entity-relationship diagram of map matching data in the context of the MM service.

Figure 6.2 shows an overview of the MM service architecture. As depicted, it will only contain modules necessary for the MM logic, as well data access to the necessary inspection data via the PostgreSQL DB and Core API. Both data access and map matching are exposed through representational state transfer (REST) endpoints for the evaluation application to request all required inspection data, store it locally and request any prototype to run using it. In the production use case, the *MM controller* can directly access the inspection data, as first requesting and storing it in the client is not necessary.

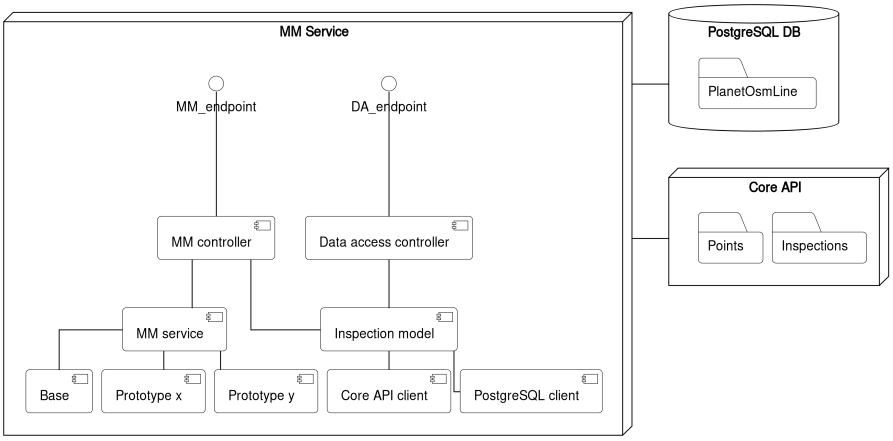


Figure 6.2: Concept for MM service.

6.2.2 Off-road map matching

Section 3.2 detailed three different approaches for extending existing MM algorithms to consider off-road matches in their calculations. Due to the defined requirements and time limitations of this thesis, as well as limited implementation details in the source papers, those approaches cannot be implemented exactly as proposed. Instead, core ideas from the proposals will be taken and applied incrementally on top of the extensible base application.

The first prototype, termed *direct off-road*, will include an off-road candidate for each measured point, as outlined by Haunert and Budig and Murphy et al.. This off-road candidate will have the same position as the measured point but will not have an OSM ID set. Without further modification, this would lead to the off-road candidate getting chosen as the match for every point, as the emission and transition probability calculations are based on the match locations being different than their respective trajectory points. Therefore, the distance calculations must be adjusted, an issue for which we will use the solution of Haunert and Budig and implement their conditional factor to apply to the transition probability.

As previously discussed, a potential issue with this approach is the usage of raw location measurements for match candidates. To evaluate whether using a different model for supplying off-road candidates, as suggested by Murphy et al., is useful in our use case, the second prototype will implement a Kalman filter and is therefore termed *kalman off-road*. The filter will be run on the measured trajectory before the HMM step and its predictions for each measured time step used as the location for an off-road candidate. These off-road candidates will be scored using the same adjusted calculations as the first prototype.

Figure 6.3 depicts the key steps of the map matching sequence, how *off-road candidate handling* and *transition probability handling* are modular and can include different steps depending on the chosen approach. This behavior can be implemented using a behavioral software design pattern such as the strategy pattern, where the two modular steps are encapsulated into an interface or abstract class.

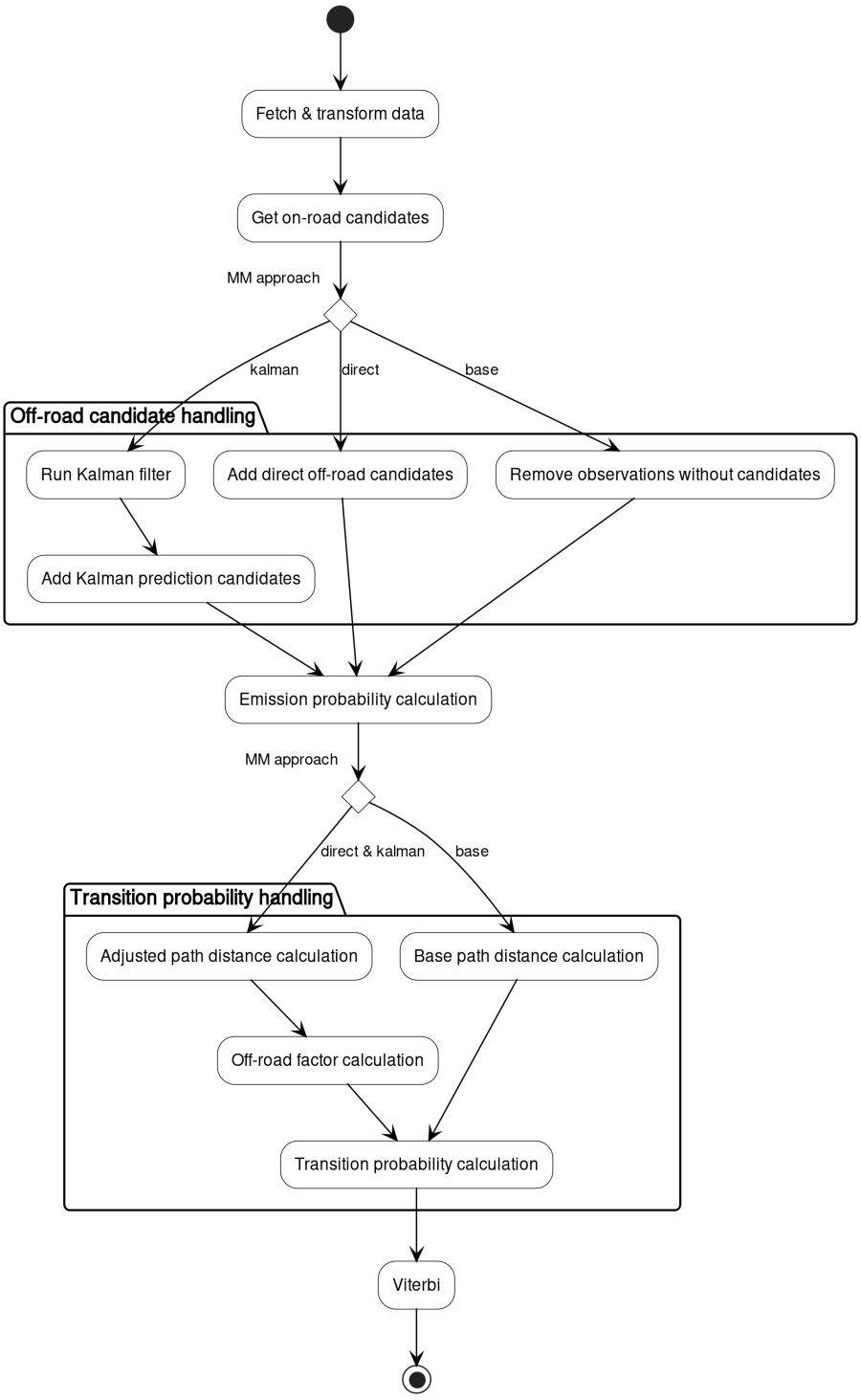


Figure 6.3: Concept for modular process in MM service.

7 Implementation

This chapter presents key details and choices made while implementing the concepts for the evaluation application and MM service.

Python-based pseudocode will be used to succinctly depict implemented algorithms while keeping enough concrete detail to allow comparison.

7.1 Evaluation

The evaluation application is implemented as a browser-based ReactJS [1] app. While interactive, map-based applications can be built using a variety of platforms and frameworks, running in a browser allows us to leverage existing technologies (Table 7.1) for fulfilling many of the defined requirements.

Name	Description
ReactJS	Framework for building state-driven interactive web applications
IndexedDB[2]	Storage API for large amounts of data, implemented in most browsers
Mapbox GL JS[3]	Library providing a highly customizable hardware accelerated map using WebGL
visx[4]	Library providing highly customizable data visualization components
Turf[5]	Library providing a large set of geospatial functions

Table 7.1: Key web technologies used.

[1] "React - A JavaScript library for building user interfaces" https://reactjs.org/ (accessed 16 June 2022)
[2] "Indexed Database API 3.0" https://www.w3.org/TR/IndexedDB/ (accessed 16 June 2022)

7.1.1 Evaluation process

As defined in the evaluation requirements and concept, the evaluation application facilitates the evaluation process at every stage. This includes viewing inspections (Figure A.9), downloading and storing all necessary inspection data using IndexedDB (Figure A.8), creating ground truth (Figure 7.1), prototype runs (Figure A.10) and result visualization (Figure A.11).

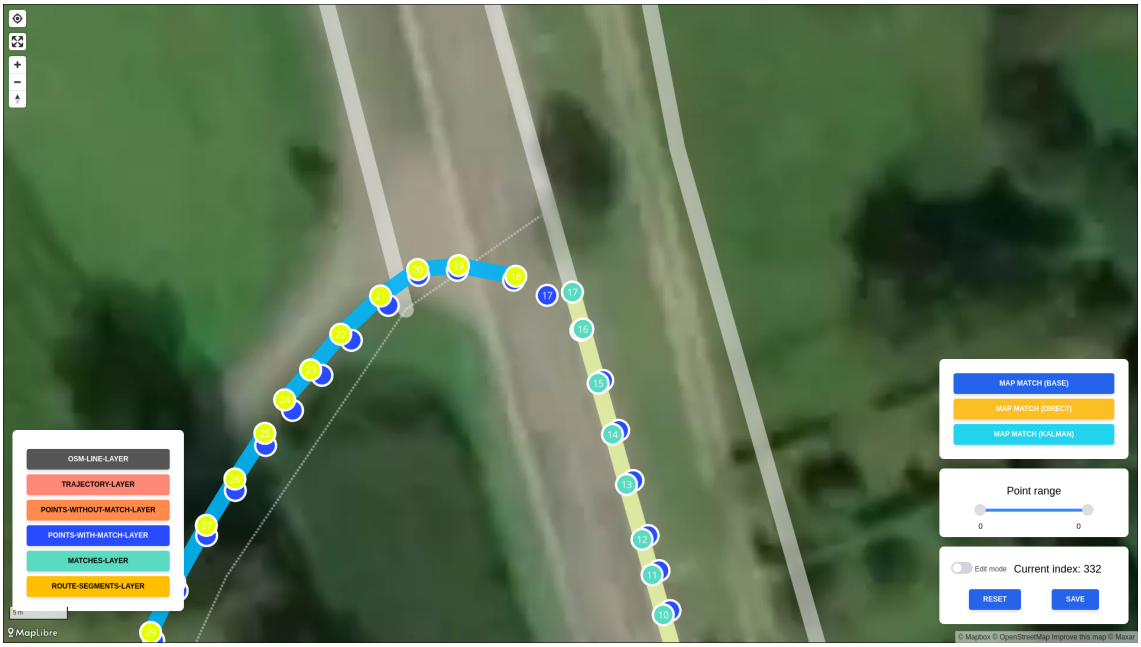


Figure 7.1: *Ground truth view* for visual and interactive manual map matching. Blue points represent the measured locations, green points are on-road matches and yellow points are off-road matches. Overlaid on top of the satellite base map are the known OSM street in light gray and street segments with randomized unique colors. The bottom left shows controls to hide and show map layers.

The in E.F.11 required functionality to snap ground truth matches to nearby OSM streets was able to be implemented by querying street features around the match in motion within a certain bounding box and identifying the closest. The nearest point on this street to the moving match is then identified and used to override the match position in addition to saving the corresponding OSM ID in the match

[3] "Interactive, thoroughly customizable maps in the browser, powered by vector tiles and WebGL" https://github.com/mapbox/mapbox-gl-js (accessed 16 June 2022)

[4] "A collection of reusable low-level visualization components" https://github.com/airbnb/visx (accessed 16 June 2022)

[5] "A modular geospatial engine written in JavaScript" https://github.com/Turfjs/turf (accessed 16 June 2022)

object. This process is similar to the nearest-neighbor approach used for identifying match candidate for map matching. Is a ground truth match moved too far away from nearby street, it is considered off-road and marked as such.

In addition to the required *ground truth view*, a separate map-based view was implemented to aid in the development and evaluation process. As match candidate selection is a crucial process to analyze in HMM map matching, especially with off-road candidates, the *visualization view* was created to present in-depth information of not just the MM result, but all candidates that were considered as well.
Both map views use Mapbox GL JS for their background maps and to display geographic data. In order to retain visual clarity and give the tester choice in what aspects of that data to inspect, information is split into semantic map layers, such as the OSM streets or matches, which can then be individually toggled to be visible or hidden. However, even within a single layer map elements can overlap in situations, such as when a vehicle doubles back. To remedy this, a range slider was implemented, allowing the tester to select any continuous range of points in the inspections and hide the remainder.

After collecting ground truth for an inspection, the tester can view pertinent characteristics such as the measured speed and accuracy of each point (Figure A.10), as defined in requirements E.F.4 and E.F.5.

7.1.2 Discrete Fréchet distance

The discrete Fréchet distance implementation, seen in Listing A.1, uses the *DiscreteFrechet* class to encapsulate the initialization, entrypoint and recursive parts of the algorithm and makes use of dynamic programming principles by storing and reusing intermediate results (*self.mem*).

As the points of the compared polylines are geographical points with latitude and longitude coordinates, the *distance(pointA, pointB)* function calculates the great circle distance between the two points.

7.2 Prototypes

The concept for the MM service is implemented as a Go (Golang[6]) server with a REST API.

[6]"The Go Programming Language" https://go.dev/ (accessed 21 June 2022)

Section 7.2.1 will discuss the pseudocode for the core map matching functions, illustrating the basis on which the off-road implementations of the following sections build.

7.2.1 Core map matching implementation

When implementing the HMM scoring logic and Viterbi algorithm, multiple iterations were made in order to determine whether a more functional or object-oriented approach would be a better fit for the defined concept and what data structures to use to represent the entities defined in Figure 6.1. The pseudocode found in Section A.6 of the appendix, represents a concise version of the final Go code. Instead of creating emissions as concrete objects, they are reflected using a map structure, which efficiently links the candidate object reference to its probability value. Similarly, as transitions are relations between two candidate objects, they can be represented using a two-dimensional map structure.

Firstly, Listing A.2 shows the core map matching function, taking in an array of observations, sorted in ascending order by time, and a map of candidates, relating an observation reference to an array of candidates. The *sigma* and *beta* parameters refer to the HMM parameters σ_z and β as defined by Newson and Krumm, influencing how emission and transition probabilities are calculated. For the purposes of this work, they are experimentally set constants, having the following values:

$$sigma = 2, \quad beta = 0.12 \tag{7.1}$$

The core of the function is a loop, which iterates over all observations and calculates emission and transition probabilities for each and feeds them into the Viterbi algorithm implementation. As transitions are possible between any of the previous and current candidates, they are calculated in a nested loop. The modularity of the *transition probability handling* is left out for clarity and depicted is instead only the base logic. Several functions are not defined in the pseudocode, as their concrete implementation is not immediately relevant to this work Their functionality will instead be outlined here:

- *gc_distance* calculates the great circle distance between two geographic points.
- *e_prob* is an implementation of Equation (1) as defined in [12].
- *t_prob* is an implementation of Equations (2) and (3) as defined in [12].

- *path_distance* calculates the distance between two geographic points along the street network, if possible. If there is no path, the distance is set to the maximum.

The *Viterbi* class is defined in Listing A.3 and implements all functionality to iteratively take in candidates as well as their emission and transitions to finally retrieve the most likely candidate sequence. The algorithm is initialized with each candidate scored using only their emission probability, as there are no initial transitions. With each iteration, the most probable previous candidate is calculated for each new candidate based on the available transitions and this knowledge saved as a *state*, containing both the new candidate and a back pointer to the previous state. As the probabilities are multiplied along the candidate sequence, the most likely path can then be retrieved at the end by finding the last state with the highest final probability and iteratively following its back pointers.

7.2.2 Off-road candidate scoring

As outlined in the concept, Section 6.2.2, the scoring has to be adjusted in the off-road prototypes. This is done through modifying the transition probability calculation of Listing A.2 to include a conditional factor. Listing 7.1 shows how the proposal by Haunert and Budig is implemented using the *isOffroad* attribute of candidates, the number of on-road candidates and the constants *psi* (ψ) and *phi* (ϕ).
For the purposes of the evaluation, these two parameters were set to the following values, found to give the best results during the testing of Haunert and Budig:

$$phi = 10, \quad psi = 1.5 \tag{7.2}$$

```
    factor = 1
    if fromCandidate.isOffroad {
        if toCandidate.isOffroad {
            factor = 1 / (psi*onRoadCandidateCount + 1.0)
        } else {
            factor = psi / (psi*onRoadCandidateCount + 1.0)
        }
    } else {
        if toCandidate.isOffroad {
            factor = 1.0 / (phi*onRoadCandidateCount + 1.0)
        } else {
            factor = phi / (phi*onRoadCandidateCount + 1.0)
        }
    }

    transitions[fromCandidate][toCandidate] = t_prob(
        beta,
        path_distance(fromCandidate, toCandidate),
        gps_distance
    ) * factor
```

Listing 7.1: Off-road candidate scoring pseudocode.

7.2.3 Kalman filter

Kalman filters have many uses and can be applied and tuned in numerous ways. Due to the constraints of the scope of this work, the implementation used for the *kalman off-road* prototype will be limited in the inputs and internal model it uses.

Pseudocode for the Kalman filter implementation, based on [4], can be found in Listing A.4. Once again, an object-oriented approach is taken, which can encapsulate not only the relevant initialization, prediction, and update steps, but also the internal state of the filter. As the only available input indicating covariance or process noise is the reported location accuracy, the covariance matrix and process noise matrix are simplified to a single value each. Additionally, the used dynamic system model is simplified and reflects basic Newtonian movement with constant speed. This is implemented in the *calc_destination* function, which simply calculates the resulting geographic position, given a starting position, distance and course.
Due to these simplifications, the remaining matrix operations can be calculated as individual terms to increase code readability and facilitate debugging.

Summarizing the functionality of this implementation, the filter is initialized with an initial state, containing current position, speed, and course of the vehicle, as

well as the reported accuracy. The *predict* and *update* functions can then be iteratively called with observations:

- *predict* uses the previous state information and the dynamic system model to predict the state and covariance *deltaTime* seconds in the future.

- *update* takes in the current measured state and accuracy and compares it to the predicted state. The new internal *state* is then set to a combination of the two compared states, the nature of which is determined by the calculated *gain*.

After one *predict* and *update* iteration, the internal *state* represents the filters' prediction for the current time.
How this implementation is used to create off-road candidates for the *kalman off-road* prototype can be seen in Listing A.5.

8 Results

The results presented in this chapter were calculated using collected ground truth data and MM results from ten inspections selected from the vialytics system.

Firstly, key findings of the metrics concerning how the measured trajectory and ground truth relate, are presented in Section 8.1. Having established the context for the evaluation, Section 8.2 then describes the prototype results.

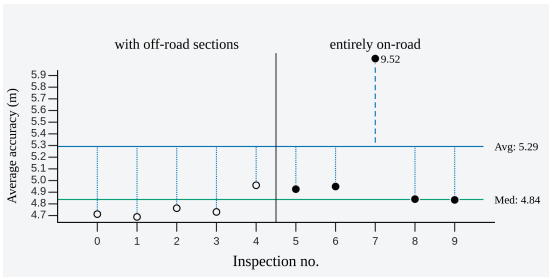
8.1 Test set characteristics

As described in Chapter 5, the inspections in the test set were collected to represent relevant MM issues with the existing MM system. They were not collected in a specific order, however for the purposes of the result visualizations, each inspection is given a number. Additionally, the set is separated into inspections that contain sections that were classified as off-road in the ground truth (nos. 0 to 4), meaning they appear to be outside the known street network, and inspections that are entirely on-road (nos. 5 to 9). Figure 8.1 depicts this classification clearly and shows how four out of the five inspections with off-road segments in fact contain a more off-road points than on-road.

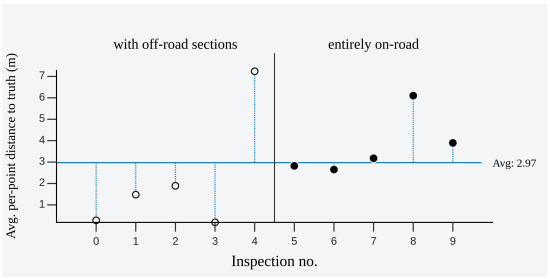
Figure 8.1: Percentage of points with an off-road match out of the total number of points per inspection ($\frac{n_o}{n_p}$).

To aid in visualizing the inspection distributions, the average is depicted as a line and a dotted line is drawn from each datum to the average, visualizing the deviation from it. In cases where outlying data would extend the scale too much to see differences in the other datums, outliers are drawn at the maximum scale height with a dashed line and their true value written next to them. Additionally, the median is drawn in these cases, as the average is sensitive to outliers.

While the exact position of ground truth matches does not reflect the true position of the vehicle, but rather only aims to be a reference for the corresponding OSM line, it is still of interest to view the average distance between ground truth and measured points compared to the horizontal accuracy reported by the measurement device. Figure 8.2 shows this comparison, where the accuracy value can be understood as the radius of a measurement error circle and the per-point distance to the corresponding ground truth is the great circle distance between the two positions. Aside from the outlying measurement in inspection seven, the average accuracy appears to be closely clustered around the median of 4.84 meters and is thus within the typical range of GPS devices. The average per-point distances to the ground truth largely match these results, being within the accuracy range. However, the results of inspections with off-road segments are strongly skewed towards less distance and should be treated with the least confidence, as the manually placed matches have no OSM lines to reference and placement using satellite imagery can be unreliable (cf. Section 2.3).



(a) Average reported accuracy per inspection ($\overline{\sigma}$)



(b) Average per-point distance from measured position to ground truth match ($\overline{d_{gt}}$)

Figure 8.2: Per-inspection comparison of reported accuracy and calculated per-point distance to ground truth.

Looking at the measured trajectory and ground truth route trajectory as wholes, Figure 8.3 depicts the discrete Fréchet distance between the two. This shows that, despite the similar average per-point accuracies, the more maximum-oriented Fréchet distance reflects much stronger variance in the test set.

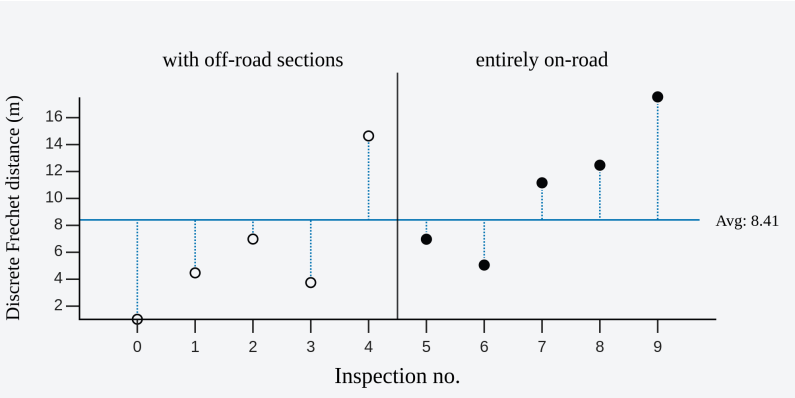


Figure 8.3: Discrete Fréchet distance between measured trajectory and ground truth route per inspection (δ_{gt}).

Further figures illustrating the nature of the test set can be found in the appendix, Section A.3.

8.2 Prototype results

For visualizing the prototype results, the inspection numbering is consistent with the charts in Section 8.1. To facilitate comparison, the results of all three map matching approach implementations, the *base* implementation with no off-road specific logic, *direct off-road* and *kalman off-road* prototypes, will be shown in the same charts. Implementations are identified using different colors, line patterns and line markers.

Firstly, we compare the line matching accuracy in Figure 8.4. Since both off-road prototypes are based on the base implementation and the *kalman* prototype is based on the same scoring extensions of the *direct* prototype, similar results in this area were to be expected. It is nevertheless noteworthy that the *base* implementation accuracy suffers greatly from off-road sections, despite points with no matches being filtered out and therefore not counting against the metric. In both off-road implementations, the on-road candidates can be unlikely enough for the off-road candidates to be chosen, leading to a correct classification of off-road segments. As can be seen in the right set of inspections, this did not significantly negatively influence their accuracy in purely on-road situations.

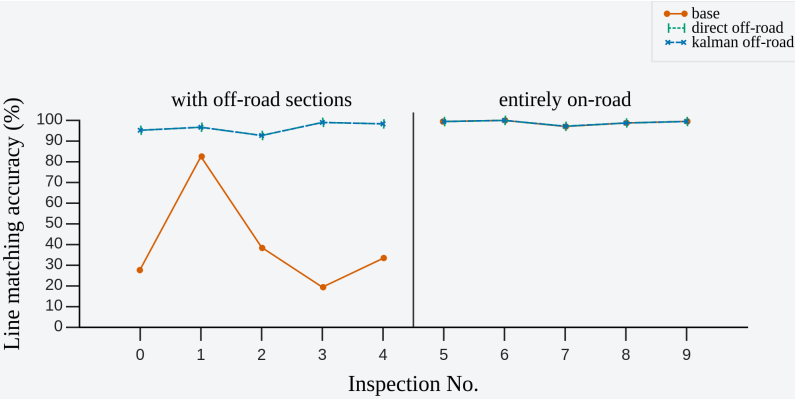


Figure 8.4: Line matching accuracy per inspection (a_l).

Both off-road candidates have the same number of points wrongly classified as off-road, with one false positive each in inspections one, three, six and eight. On the other hand, false negatives, shown in Figure 8.5, were much more common. This behavior is a trade-off and can be adjusted using the psi and phi parameters.

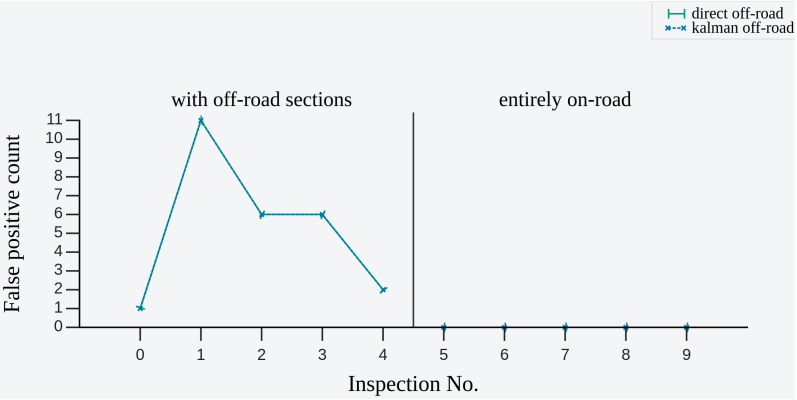
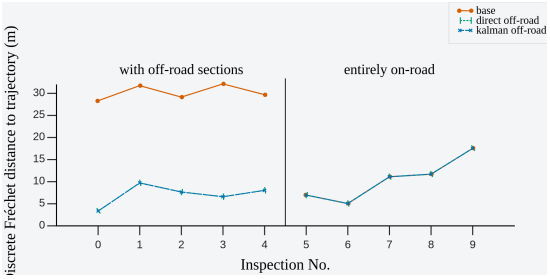


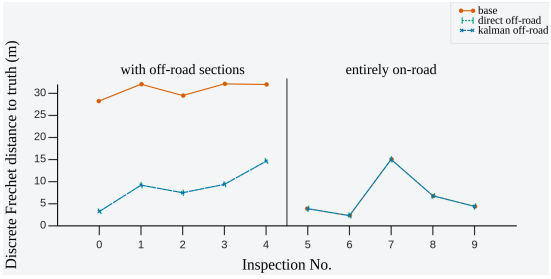
Figure 8.5: False positives per inspection (FP).

Precision and recall curves can be found in the appendix, Section A.4. More than 90 % of locations outside of the known street network are correctly classified as off-road, while less than 3 % of points are falsely classified as off-road.

The pattern in Figure 8.4 is repeated when looking at discrete Fréchet distances, depicted in Figure 8.6, where the off-road prototypes have significantly lower distances both to the measured trajectory and the ground truth route than the *base* implementation.



(a) Distance between matched route and measured trajectory (δ_{mt})



(b) Distance between matched route and ground truth route (δ_{mg})

Figure 8.6: Prototype result discrete Fréchet distance comparisons.

Figure 8.7 depicts the computational time per inspection point needed for the map matching process by each of the implementations. All MM operations were run on an AMD Ryzen 5700u CPU in a largely uncontrolled environment. While the difference between implementations is small, the *base* application consistently takes the least time to execute. However, there is no clear pattern between the off-road prototypes, indicating that the Kalman filter calculations may be negligible compared to the time spent on calculating HMM probabilities and running the Viterbi algorithm for the additional off-road candidates.
The outlier at inspection three is most likely due to a lack of optimization of distance calculations in the *base application*, specifically regarding operations on atypically long OSM lines, and is therefore reflected in all three datums.

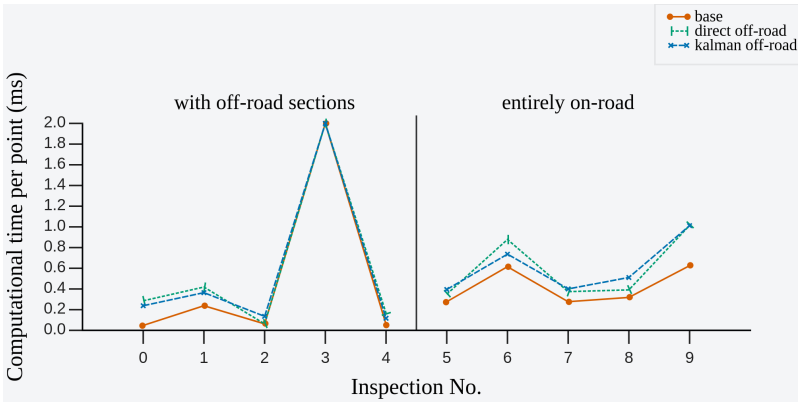


Figure 8.7: Computational time per point for each implementation ($\frac{\Delta t_c}{n_p}$).

9 Discussion

Based on the results from evaluating the prototypes on the test set of ten inspections, we believe extending existing HMM map matching methods with approaches to detect off-road sections can improve MM performance in situations where the given street network data is incomplete or incorrect. More specifically, the increased line matching accuracy on inspections with off-road sections without a significant number of false positives shows this approach could have an almost purely positive impact when implemented. The drop in accuracy of the *base* implementation can be attributed to the start and end of off-road segments, where match candidates can still be found within the search radius and are wrongly selected as matches. The off-road prototypes avoid this issue by explicitly identifying missing streets instead of purely relying on a distance cutoff.

However, this evaluation was specific to the use case and dataset of vialytics. The inspection dataset is small, and the ground truth collection method could be inconsistent. The set of inspections with off-road sections also contains situations where the underlying cause for the incomplete street network information is not a lack of information in OSM, but rather in the filtered copy of OSM data in the vialytics system. Thus, while the result supports further investigation towards making use of the evaluated approaches at vialytics, it can only be generalized to a limited extend for other MM use cases and cannot be used to draw conclusions about the completeness or correctness of OSM data.

The lack of difference between the *direct* and *kalman* prototypes in the compared metrics may indicate further shortcomings of this work. One possible explanation could be the lack of truly raw location measurement data. As previously described, the geographical locations of points in the vialytics system are the result of unknown sensor processing in the operating system of the smartphone, as well as linear interpolation in the vialytics app due to irregular sampling rates. These processes may already modify the measurement data too much for the implemented Kalman filter to have a meaningful impact. Additionally, the used implementation uses simplifications in many aspects, such as the modeling of process noise, covariance, and the dynamic system. Kalman filter implementations more specifically suited and tuned to this use case may yield better results. The paper by Murphy et al. proposing the use of a Kalman filter

for map matching used an extended Kalman filter, integrated it with a different HMM implementation and ultimately used an entirely different base model instead [1].

Another area which was only explored to a limited extent are the uses of detected off-road segments beyond preventing mismatches of on-road points. Map matching and map building, or map inference, are closely related fields and such knowledge of off-road segments could be used to improve the incomplete street network data. In such a use case, the use of a Kalman filter may also have additional benefits, as the predicted locations could prove better for inferring streets and intersections.

Finally, several parameters such as $sigma$, $beta$, phi and psi, were assigned a constant value for all testing. Experimenting with different values could result in a better understanding of how they influence the HMM or improve results.

[1] "A New Real-Time Map-Matching Algorithm at Lyft | by Marie Douriez | Lyft Engineering" https://eng.lyft.com/a-new-real-time-map-matching-algorithm-at-lyft-da593ab7b006 (accessed 21 June 2022)

10 Conclusion

This work has presented a use case for an off-line vehicle map matching algorithm which is aware of incomplete street network data and can benefit from this knowledge. Using this use case, research questions were formulated to define the core objectives of this work alongside requirements set by the context. Several fitting approaches from state-of-the-art literature were then presented and partially integrated into two prototypes, both based on a shared HMM map matching implementation. The prototypes, alongside the base implementation were then evaluated on a test set of measured vehicle trajectories. Ground truth data for the test set was collected through tool-assisted manual matching, allowing for not just a reference for which street a given measurement should be matched to, but also whether it is considered off-road or on-road.

The evaluation results showed that the prototypes classified more than 90 % of locations outside of the known street network correctly as off-road, without negatively impacting the matching accuracy for on-road trajectories in a significant way. This off-road detection inherently resulted in accuracy benefits, due to off-road locations no longer being incorrectly matched to a street. However, the prototypes add computational complexity to the HMM map matching algorithm and therefore increase the processing time needed.
Using a Kalman filter to pre-process the measured locations before using them as off-road match candidate did not meaningfully affect any of the compared metrics.

Future work should investigate finding the optimal parameters for the HMM and off-road scoring calculations on a larger test set, as well as whether dynamically or conditionally changing them leads to better results.
Furthermore, more possibilities of using known off-road trajectory segments should be explored, such as in the context of map building.

Acronyms

CNN convoluted neural network

CS customer success

GIS geographic information system

GPS global positioning system

HMM Hidden Markov Model

ML machine learning

MM map matching

OSM OpenStreetMap

OSM ID OpenStreetMap ID

QA quality assurance

REST representational state transfer

sIMM semi-interacting multiple model filter

VGI volunteered geographic information

Glossary

Core API Central API for data access to vialytics data.

Fréchet distance Measure of similarity, or distance, between curves that takes the continuity of points on the curves into account.

geographic information system System, through which users can view, analyze, and interact with geographic data.

great circle distance Shortest distance between two points on the surface of a sphere, along the surface of the sphere.

Hidden Markov Model Statistical model, in which a series of unobservable hidden states are related to a series of observations.

inspection A batch of recorded data, consisting of an image, positional data, a timestamp and more.

Kalman filter Algorithm, that can model measurement error on a series of measurements over time and can make predictions for future measurements which can be more accurate than a single measurement.

machine learning Algorithms, which use input data to improve their results over time.

map matching Process, which matches the measured location data with the known street geometry in order to find the most likely corresponding route on the street network.

match candidate Street segment or a point on a street segment, which is a possible part of the true vehicle route.

OpenStreetMap Map-based and explicit VGI project.

OpenStreetMap ID Integer number, assigned to each segment of an OSM way or OSM line. They are not stable, in the sense that they may be reassigned to a different segment, but they do uniquely identify an OSM line in a given relation.

Viterbi algorithm Dynamic programming algorithm for obtaining the most likely sequence of hidden states in a Hidden Markov Model.

weak Fréchet distance Version of the Fréchet distance that allows non-monotone continuous reparameterizations while calculating the distance between two curves.

Bibliography

[1] Geoff Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, sep 2017. ISSN 01989715. doi: 10.1016/j.compenvurbsys.2017.05.004.

[2] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *VLDB 2005 - Proceedings of 31st International Conference on Very Large Data Bases*, volume 2, pages 853–864, 2005. ISBN 1595931546.

[3] Pingfu Chao, Wen Hua, and Xiaofang Zhou. Trajectories know where map is wrong: an iterative framework for map-trajectory co-optimisation. *World Wide Web*, 23(1):47–73, jan 2020. ISSN 15731413. doi: 10.1007/s11280-019-00721-w. URL https://link.springer.com/article/10.1007/s11280-019-00721-w.

[4] Akash Deep, Monika Mittal, and Vikas Mittal. Application of Kalman Filter in GPS Position Estimation. *8th IEEE Power India International Conference, PIICON 2018*, jul 2018. doi: 10.1109/POWERI.2018.8704368.

[5] Thomas Devogele, Laurent Etienne, Maxence Esnault, and Florian Lardy. Optimized Discrete Fréchet Distance between trajectories. *Proceedings of the 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*, 9, 2017. doi: 10.1145/3150919. URL https://doi.org/10.1145/3150919.3150924.

[6] Adam Van Etten. City-scale road extraction from satellite imagery v2: Road speeds and travel times. *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pages 1775–1784, mar 2020. doi: 10.1109/WACV45572.2020.9093593.

[7] Mohammad Ali Goudarzi and René Jr Landry. Assessing horizontal positional accuracy of Google Earth imagery in the city of Montreal, Canada. *Geodesy and Cartography*, 43(2):56–65, apr 2017. ISSN 20297009. doi: 10.3846/20296991.2017.1330767. URL https://www.tandfonline.com/doi/abs/10.3846/20296991.2017.1330767.

[8] Jan-Henrik Haunert and Benedikt Budig. An algorithm for map matching given incomplete road data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems - SIGSPATIAL '12*, page 510, New York, New York, USA, 2012. ACM Press. ISBN 9781450316910. doi: 10.1145/2424321.2424402. URL http://dl.acm.org/citation.cfm?doid=2424321.2424402.

[9] Jasmeet Kaur, Jaiteg Singh, Sukhjit Singh Sehra, and Hardeep Singh Rai. Systematic literature review of data quality within openstreetmap. In *Proceedings - 2017 International Conference on Next Generation Computing and Information Systems, ICNGCIS 2017*, pages 159–163. Institute of Electrical and Electronics Engineers Inc., nov 2018. ISBN 9781538642054. doi: 10.1109/ICNGCIS.2017.35.

[10] Huajian Mao, Wuman Luo, Haoyu Tan, Lionel M Ni, and Nong Xiao. Exploration of ground truth from raw GPS data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 12, pages 118–125, New York, New York, USA, 2012. ACM Press. ISBN 9781450315425. doi: 10.1145/2346496.2346515.

[11] James Murphy, Yuanyuan Pao, and Albert Yuen. Map matching when the map is wrong: Efficient on/off road vehicle tracking and map learning. *IWCTS 2019 - Proceedings of the 12th International Workshop on Computational Transportation Science*, 2019. doi: 10.1145/3357000.3366143.

[12] Paul Newson and John Krumm. Hidden Markov map matching through noise and sparseness. *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 336–343, 2009. doi: 10.1145/1653771.1653818.

[13] Efstratios Rappos, Stephan Robert, Philippe Cudré-Mauroux, and Philippe Cudré. A Force-Directed Approach for Offline GPS Trajectory Map Matching. *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 18, 2018. doi: 10.1145/3274895. URL https://doi.org/10.1145/3274895.3274919.

[14] Hansi Senaratne, Amin Mobasheri, Ahmed Loai Ali, Cristina Capineri, and Mordechai (Muki) Haklay. A review of volunteered geographic information quality assessment methods, jan 2017. ISSN 13623087.

[15] Mudhakar Srivatsa, Raghu Ganti, Jingjing Wang, and Vinay Kolar. Map matching: Facts and myths. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 474–477, 2013. ISBN 9781450325219. doi: 10.1145/2525314.2525466. URL https://dl.acm.org/doi/10.1145/2525314.2525466.

[16] Fernando Torre, David Pitchford, Phil Brown, and Loren Terveen. Matching GPS traces to (possibly) incomplete map data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems - SIGSPATIAL '12*, page 546, New York, New York, USA, 2012. ACM Press. ISBN 9781450316910. doi: 10.1145/2424321.2424411. URL http://dl.acm.org/citation.cfm?doid=2424321.2424411.

[17] John E. Vargas-Munoz, Shivangi Srivastava, Devis Tuia, and Alexandre X. Falcao. OpenStreetMap: Challenges and Opportunities in Machine Learning and Remote Sensing. *IEEE Geoscience and Remote Sensing Magazine*, 9(1): 184–199, mar 2021. ISSN 21686831. doi: 10.1109/MGRS.2020.2994107.

[18] Hong Wei, Yin Wang, George Forman, Yanmin Zhu, and Haibing Guan. Fast Viterbi map matching with tunable weight functions. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 613–616, 2012. ISBN 9781450316910. doi: 10.1145/2424321.2424430.

[19] Hong Wei, Yin Wang, George Forman, and Yanmin Zhu. Map matching: Comparison of approaches using sparse and noisy data. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 434–437, 2013. ISBN 9781450325219. doi: 10.1145/2525314.2525456. URL http://dx.doi.org/10.1145/2525314.2525456.

[20] Xiaofang Zheng, Yu; Zhou. *Computing with Spatial Trajectories*. Springer New York, 2011. doi: 10.1007/978-1-4614-1629-6.

Attachments

A.1 Evaluation process diagram

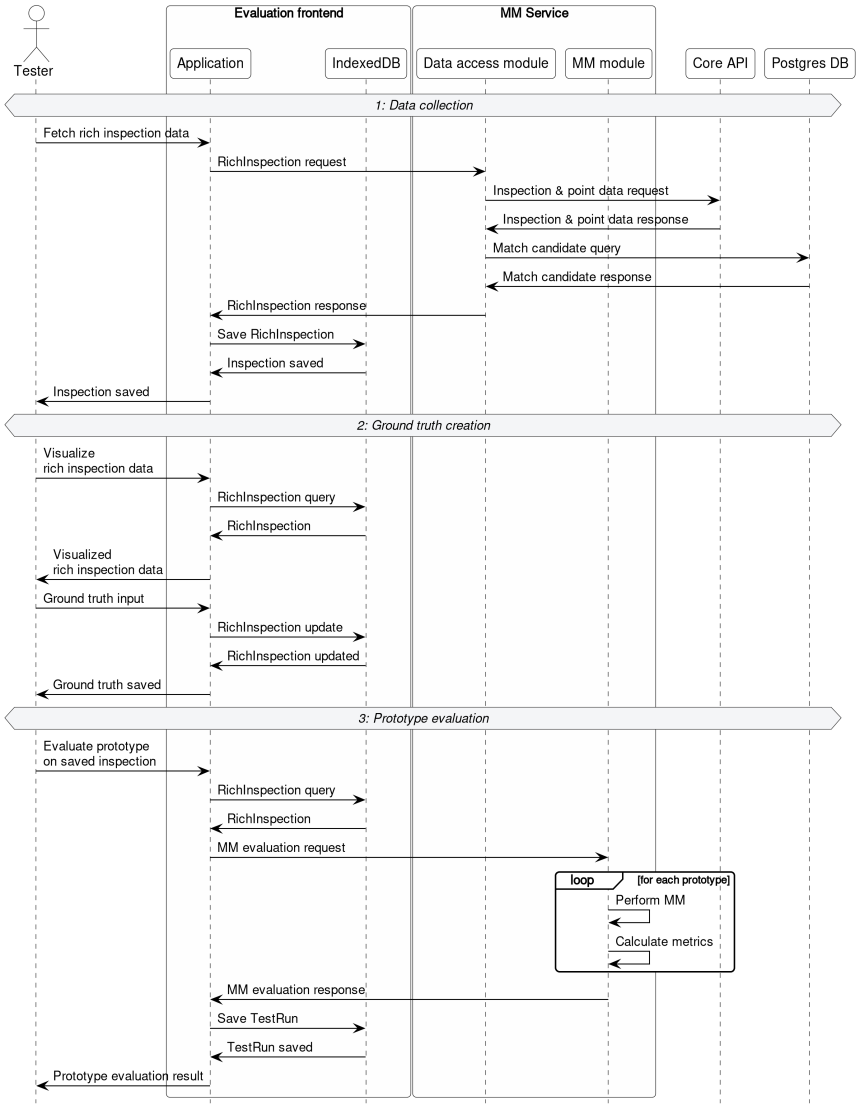


Figure A.1: Evaluation process concept for a single inspection and prototype.

A.2 Discrete Fréchet distance pseudocode

```
class DiscreteFrechet:
    def __init__(self, p, q):
        self.mem = []
        # Initialize 2D array to -1
        for i in p:
            subMem = []
            for j in q:
                subMem.append(-1)
            self.mem.append(subMem)

    def computeRecursive(self, i, j):
        if self.mem[i][j] > -1:
            return self.mem[i][j]
        elif i == 0 and j == 0:
            self.mem[i][j] = distance(self.p[i], self.q[j])
        elif i > 0 and j == 0:
            self.mem[i][j] = max(
                self.computeRecursive(i-1, 0),
                distance(self.p[i], self.q[j])
            )
        elif i == 0 and j > 0:
            self.mem[i][j] = max(
                self.computeRecursive(0, j-1),
                distance(self.p[i], self.q[j])
            )
        elif i > 0 and j > 0:
            self.mem[i][j] = max(
                min(
                    self.computeRecursive(i-1, j),
                    self.computeRecursive(i-1, j-1),
                    self.computeRecursive(i, j-1)
                ),
                distance(self.p[i], self.q[j])
            )
        else:
            self.mem[i][j] = Inf

        return self.mem[i][j]

    def distance(self):
                return self.computeRecursive(len(self.p)-1, len(self.q)-1)

```

Listing A.1: Discrete Fréchet distance based on [5].

A.3 Additional test set visualizations

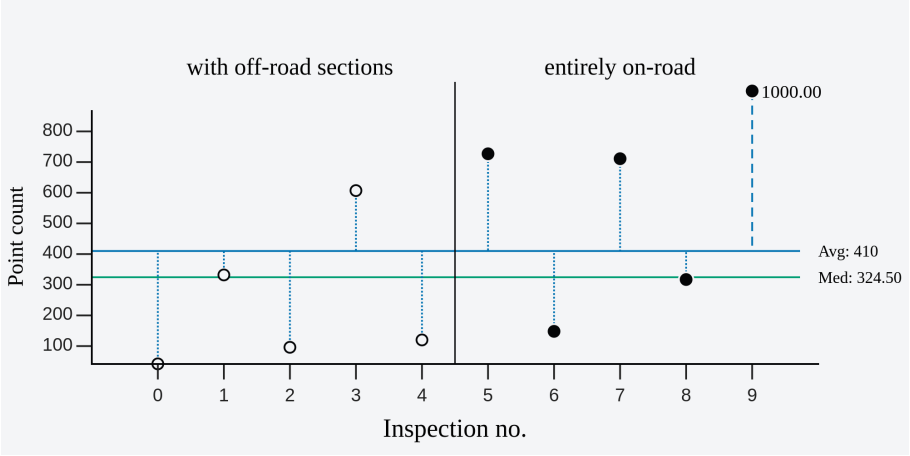


Figure A.2: Number of points per inspection.

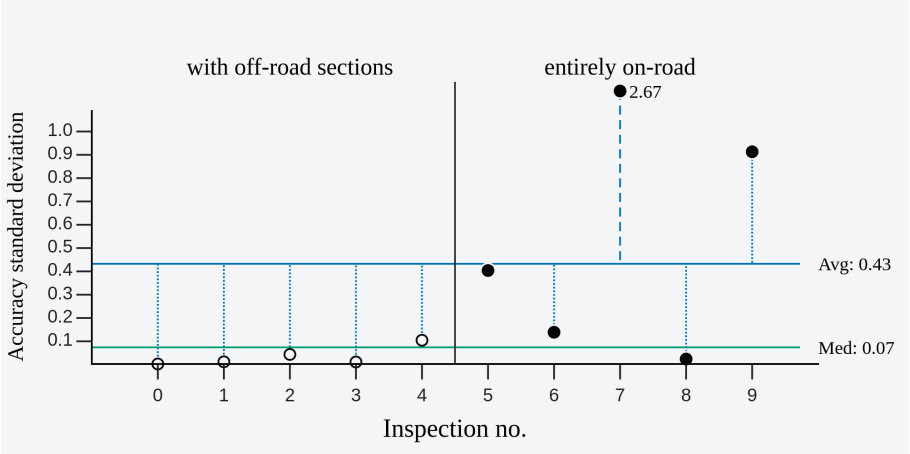


Figure A.3: Standard deviation of reported accuracy per inspection.

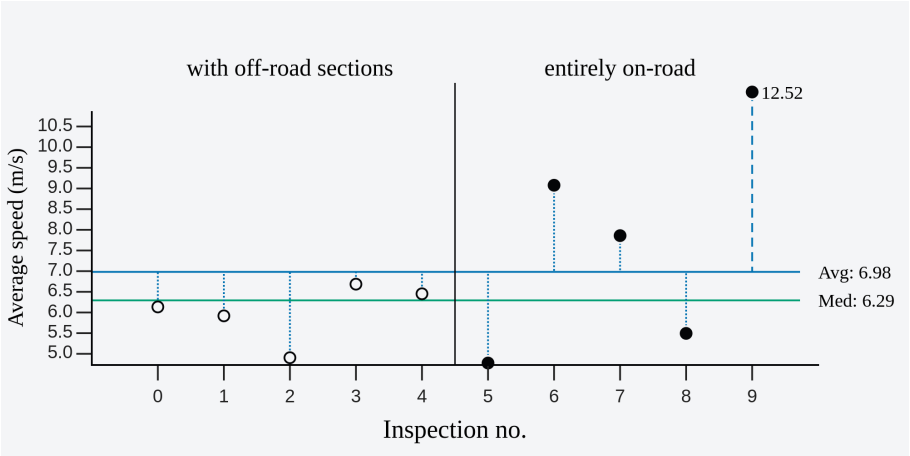


Figure A.4: Average measured speed per inspection.

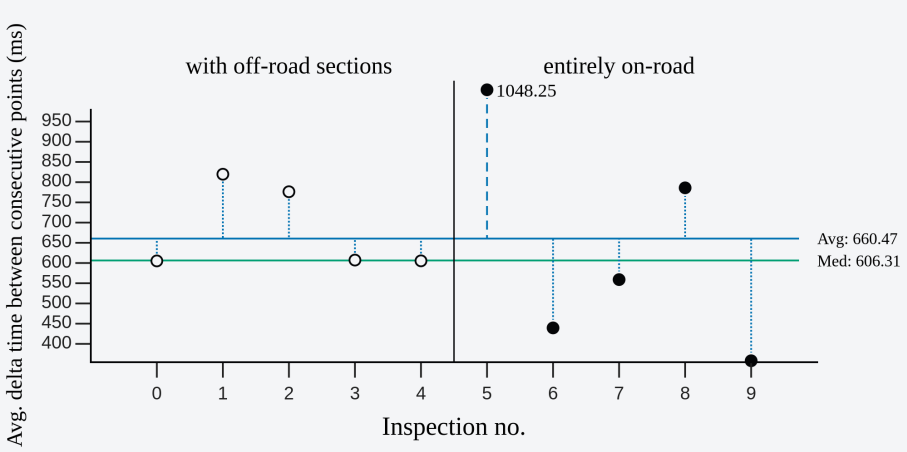


Figure A.5: Average time difference between consecutive points per inspection.

A.4 Additional prototype result visualizations

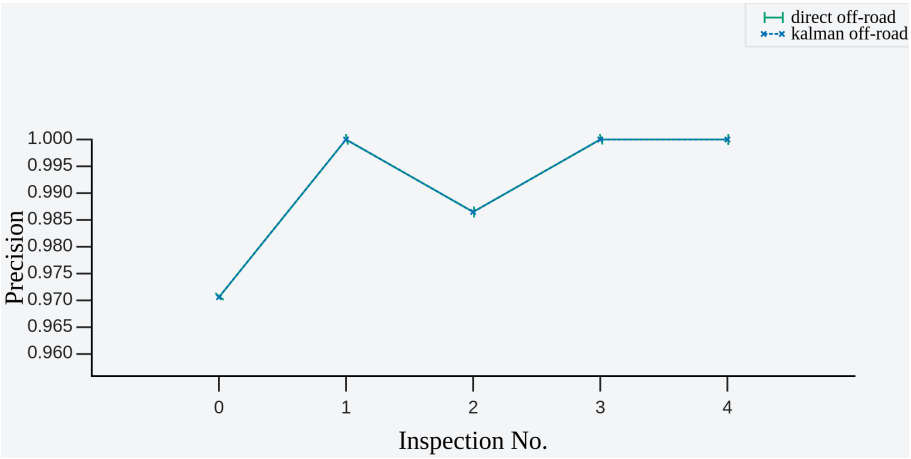


Figure A.6: Off-road precision on inspections with off-road segments.

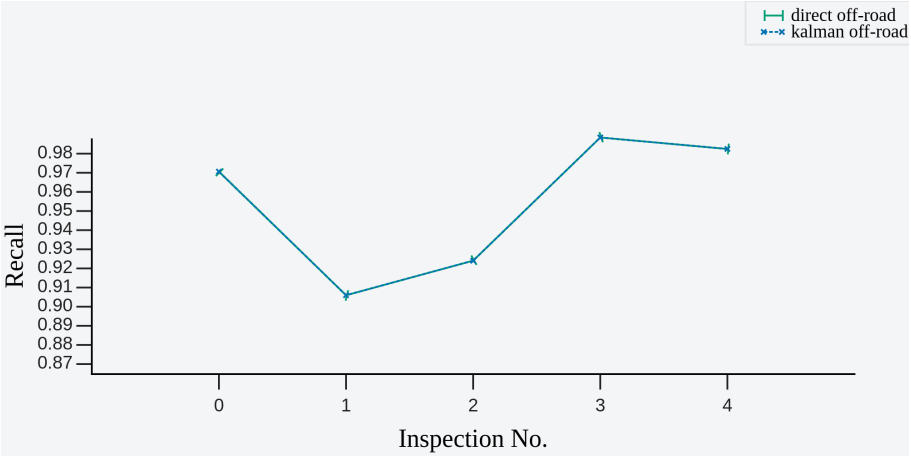


Figure A.7: Off-road recall on inspections with off-road segments.

A.5 Evaluation application screenshots

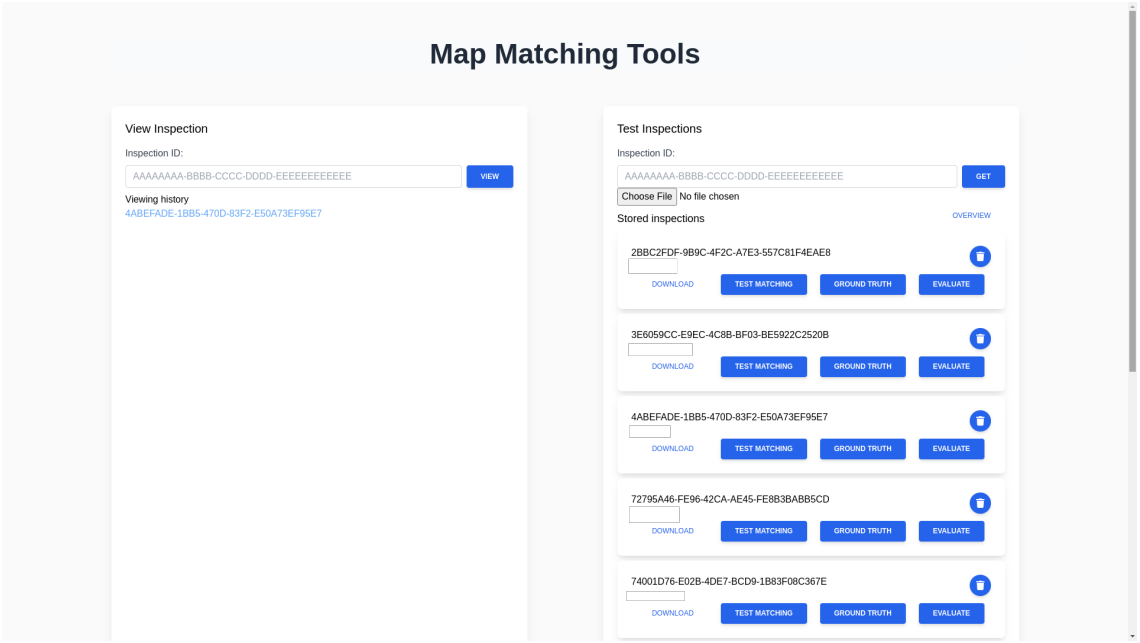


Figure A.8: Index page of evaluation application. On the left: Controls for viewing the MM results of an inspection in the *Visualization view*. On the right: Inspections in the test set, controls for adding inspections, and going through the evaluation process steps.

Figure A.9: *Visualization view* for detailed visual analysis of MM results. Blue points represent the measured locations, green points are on-road matches and gray points are all candidate matches for the currently selected point (purple outline). Detailed information is shown for the MM result in the bottom right as well as for the selected marker in the top. The bottom left shows controls to hide and show map layers.

Figure A.10: *Inspection test run view* for displaying a given inspection with ground truth data, visualizing key metrics as well as triggering MM prototype runs.

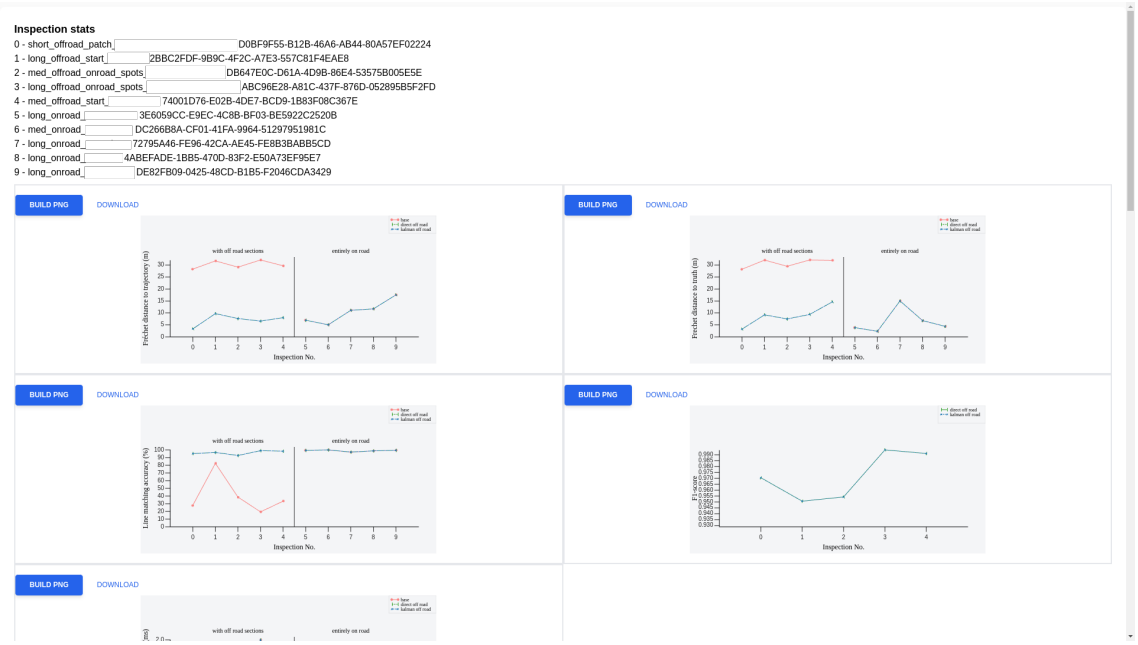


Figure A.11: *Evaluation view* for visualizing metrics for the entire inspection test set.

A.6 Core map matching pseudocode

```
def map_match(observations, candidates, sigma, beta):
    viterbi = Viterbi()
    for i, obs in observations:
        emissions = map[Candidate]float
        transitions = map[Candidate]map[Candidate]float
        for _, candidate in candidates[obs]:
            emissions[candidate] = e_prob(sigma, gc_distance(candidate, obs))
        if i == 0:
            viterbi.init(obs, candidates[obs], emissions)
        else:
            prev_obs = observations[i-1]
            gps_distance = gc_distance(prev_obs, obs)
            for j, fromCandidate in candidates[prev_obs]:
                for k, toCandidate in candidates[obs]:
                    transitions[fromCandidate][toCandidate] = t_prob(
                        beta,
                        path_distance(fromCandidate, toCandidate),
                        gps_distance
                    )
        viterbi.next(obs, candidates[obs], emissions, transitions)
    return viterbi.retrieveSequence()

```

Listing A.2: HMM & Viterbi map matching pseudocode.

```
class Viterbi:
    def __init__(self, candidates, emissions):
        for _, candidate in candidates:
            self.last_scores[candidate] = emissions[candidate]
            self.last_states[candidate] = {
                state: candidate,
                back_pointer: None
            }

    def next(self, candidates, emissions, transitions):
        states = map[Candidate]{state: Candidate, back_pointer: Candidate}
        scores = map[Candidate]float
        for _, candidate in candidates:
            max_t_prob = -Inf
            max_prev_candidate = None
            for prevCandidate, prevScore in self.last_scores:
                prob = prevScore * transitions[prevCandidate][candidate]
                if prob > max_t_prob:
                    max_t_prob = prob
                    max_prev_candidate = prevCandidate

            e_prob = emissions[candidate]
            scores[candidate] = max_t_prob * e_prob
            states[candidate] = {
                state: candidate,
                back_pointer: self.last_states[max_prev_candidate]
            }
        self.last_scores = scores
        self.last_states = states

    def retrieveSequence(self):
        max_last_candidate = None
        max_p = -Inf
        for candidate, score in self.last_scores:
            if score > max_p:
                max_last_candidate = candidate
                max_p = score
        max_last_state = self.last_states[max_last_candidate]
        sequence = [max_last_state.state]

        bp = max_last_state.back_pointer
        while bp is not None:
            sequence.append(bp.state)
            bp = bp.back_pointer

        return sequence

```

Listing A.3: Viterbi pseudocode.

A.7 Kalman filter pseudocode

```
class Kalman:
    def __init__(self, state, accuracy):
        self.state = state
        self.accuracy = accuracy
        self.covariance = accuracy * accuracy

    def predict(self, deltaTime):
        predictedPos = calc_destination(
            self.state.position,
            self.state.speed * deltaTime,
            self.state.course
        )
        self.predictedState = {
            position: predictedPos,
            speed: self.state.speed,
            course: self.state.course
        }
        noise = self.accuracy * self.accuracy * deltaTime
        self.predictedCovariance = self.covariance * deltaTime + noise

    def update(self, state, accuracy):
        noise = accuracy
        gain = self.predictedCovariance * (1/(self.predictedCovariance + noise))
        self.covariance = (1-gain) * self.predictedCovariance

        deltaLat = state.position.lat - self.predictedState.position.lat
        deltaLng = state.position.lng - self.predictedState.position.lng
        deltaSpeed = state.speed - self.predictedState.speed
        deltaCourse = state.course - self.predictedState.course

        self.state = {
            position: {
                lat: self.predictedState.position.lat + gain*deltaLat,
                lng: self.predictedState.position.lng + gain*deltaLng,
            },
            speed: self.predictedState.position.speed + gain*deltaSpeed,
            course: self.predictedState.position.course + gain*deltaCourse
        }

```

Listing A.4: Kalman filter pseudocode.

```
def add_kalman_candidates(observations, candidates):
    kf = new Kalman()
    kf.init({
            position: {
                lat: observations[0].lat,
                lon: observations[0].lon,
            },
            speed: observations[0].speed,
            course: observations[0].course,
        },
        observations[0].Accuracy,
    )
    for i, obs in observations:
        if i == 0:
            predictedCandidate = {
                lat: obs.lat,
                lon: obs.lon,
            }
            candidates.append(predictedCandidate)
            continue
        kf.predict(obs.time-observations[i-1].time) # Time difference is in seconds
        kf.update({
                position: {
                    lat: obs.lat,
                    lon: obs.lon,
                },
                speed: obs.speed,
                course: obs.course,
            },
            obs.Accuracy,
        )
        predictedState = kf.state
        predictedCandidate = {
            lat: predictedState.position.lat,
            lon: predictedState.position.lon,
        }
        candidates.append(predictedCandidate)

```

Listing A.5: Kalman candidate prediction pseudocode.

Hochschule Reutlingen
Reutlingen University

Erklärung zur Abgabe einer Bachelor- / Master-Thesis

Ich versichere ehrenwörtlich, dass ich

- die abgegebene Thesis selbständig verfasst habe,
- alle benutzten Quellen und Hilfsmittel - dazu zählen auch sinngemäß übernommene Inhalte, leicht veränderte Inhalte sowie übersetzte Inhalte - in Quellenverzeichnissen, Fußnoten oder direkt bei Zitaten angegeben habe,
- alle wörtlichen und sinngemäßen Zitate von Textstücken, Tabellen, Grafiken, Fotos, Quellcode usw. aus fremden Quellen als solche gekennzeichnet und mit seitengenauen Quellenverweisen versehen habe,
- die von mir eingereichten Dokumente und Artefakte noch nicht in dieser oder ähnlicher Form einer anderen Kommission zur Prüfung vorgelegt wurden,
- alle nicht als Zitat gekennzeichneten Inhalte selbst erstellt habe und dass ich
- den „Leitfaden für gute wissenschaftliche Praxis im Studiengang MKI“[1] kenne und achte.

Mir ist bekannt, dass unmarkierte und unbelegte Zitate und Paraphrasen Plagiate sind und nicht als handwerkliche Fehler, sondern als eine Form vorsätzlicher Täuschung der Prüfer gelten, da fremde Gedanken als eigene Gedanken vorgetäuscht werden mit dem Ziel der Erschleichung einer besseren Leistungsbewertung.
Mir ist bekannt, dass Plagiarismus die Standards guter wissenschaftlicher Praxis, die Regeln des Studiengangs Medien- und Kommunikationsinformatik, die Studien- und Prüfungsordnung der Hochschule Reutlingen (§ 10 Täuschung und Ordnungsverstoß) und das Landeshochschulgesetz von Baden-Württemberg (§ 3 Wissenschaftliche Redlichkeit Abs. 5, § 62 Exmatrikulation Abs. 3) missachtet und seine studienrechtlichen Folgen vom Nichtbestehen bis zur Exmatrikulation reichen.
Mir ist auch bekannt, dass Plagiate sogar das Urheberrechtsgesetz (§ 51 Zitate, § 63 Quellenangabe, § 106 Unerlaubte Verwertung urheberrechtlich geschützter Werke) verletzen und zivil- und strafrechtliche Folgen nach sich ziehen können.

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Datum: _____

Unterschrift: _____

[1] https://bscwserv.reutlingen-university.de/bscw/bscw.cgi/d2871027/GWP.pdf