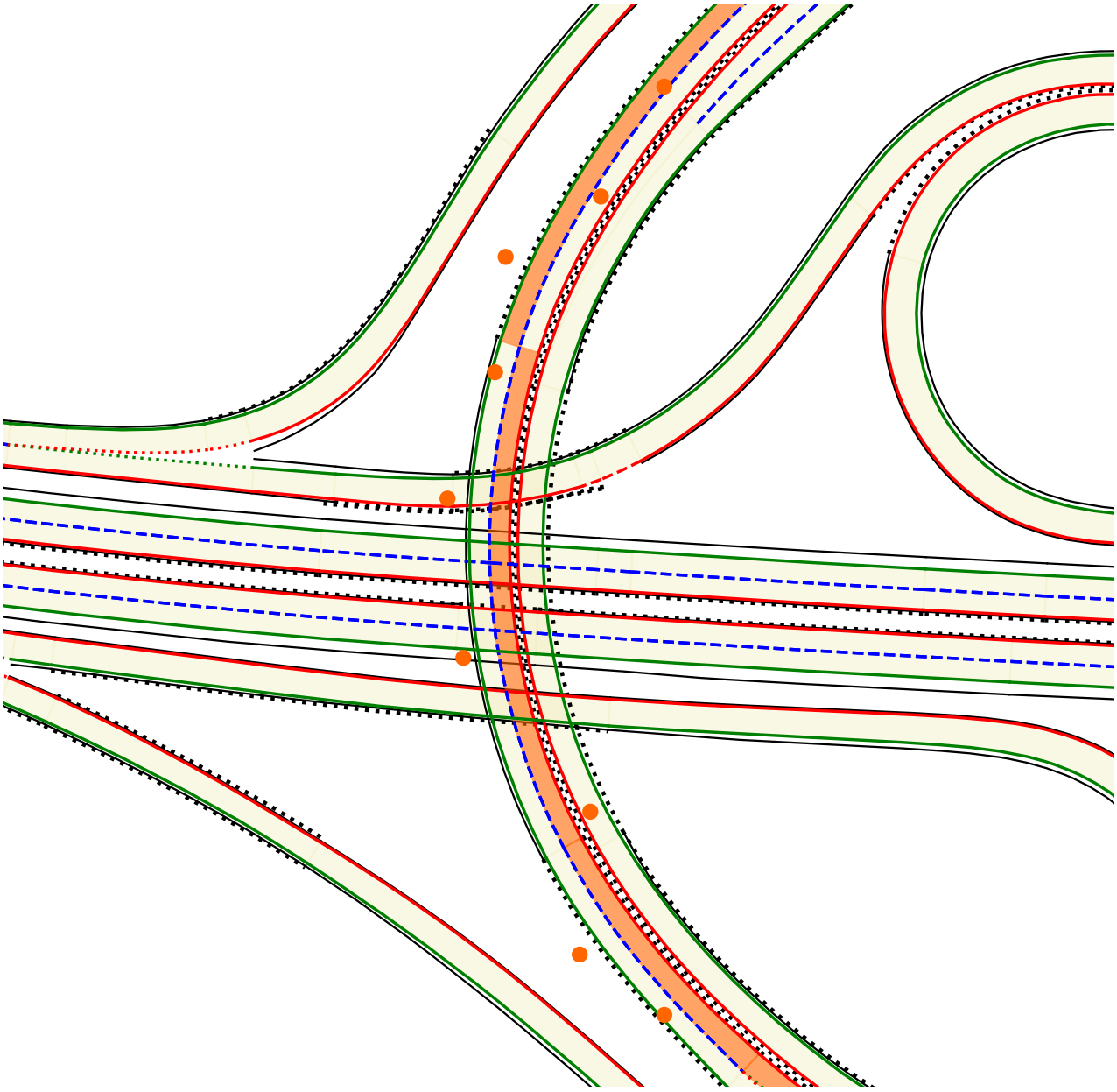
CHALMERS



Lane-Level Map Matching using Hidden Markov Models

Master's thesis in Computer Science and Complex Adaptive Systems

ELLEN KORSBERG
ELIZA NORDÉN

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2019

MASTER'S THESIS IN COMPUTER SCIENCE AND COMPLEX ADAPTIVE SYSTEMS

Lane-Level Map Matching using Hidden Markov Models

ELLEN KORSBERG
ELIZA NORDÉN

Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2019

Lane-Level Map Matching using Hidden Markov Models
ELLEN KORSBERG
ELIZA NORDÉN

Master's thesis 2019:52
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Illustration of vehicle locations being matched to lanes in a map.

Chalmers Reproservice
Göteborg, Sweden 2019

PREFACE

This master's thesis has been completed as a conclusion of our M.Sc. in Engineering at Chalmers University of Technology. The thesis was conducted at Zenuity, Gothenburg.

We would like to start by thanking our supervisors at Zenuity, Anders Hansson and Roza Maghsood. Your dedication and support throughout the whole work process have been instrumental to the outcome of this thesis. To the whole Magellan team at Zenuity — we appreciate all the input we received from you which yielded many valuable insights. We hope that this thesis outcome will be of value to your future work. Thanks also to Selpi, our supervisor and examiner at Chalmers, for guiding us through the master thesis process and adding structure to our work flow. Lastly, thank you to Lisa Sjöblom, Lovisa Hagström and Christian Klinteberg for all the energizing coffee breaks and reviving lunch strolls.

ABSTRACT

Map matching is the procedure of matching vehicle location and sensor data to a digital map. New high-definition maps, designed for autonomous vehicles, open up for the possibility of matching to lanes rather than roads. Inferring the lane-level positions of vehicles will be useful for updating and building probe-sourced maps, and thereby arguably essential for autonomous driving.

This thesis seeks to solve the lane-level map matching problem using a Hidden Markov Model. The Viterbi algorithm is used to decode it. The model is tested on a data set yielded through the Volvo Drive Me project and collected by commercial vehicle sensors, including a GPS receiver, an inertial navigation system and a forward-looking camera. For the sake of simplicity, the RADAR and LiDAR sensors are excluded. Among the sensor data used, lane changes and the type of road lane markings as detected by the vehicle proves to be particularly important.

Two metrics for evaluating model performance are proposed. The first metric is the recall, i.e. the fraction of correct matches. However, the lanes to which the observations are matched vary widely in length. Therefore, we introduce the path length error (PLE) as a complementary metric. As the name indicates, it considers the length of the incorrect routes.

A naive matcher, that simply matches GPS coordinates to the closest lane, is used for benchmarking. Attaining 95% median recall and 3% median PLE, we conclude that our model is high-performing and robust to errors. For comparison, the naive matcher scores 77% median recall and 26% median PLE. Our model is however shown to struggle without reliable vision detections. It would therefore be meaningful to investigate the inclusion of additional vehicle sensors.

Keywords: Autonomous Driving, Hidden Markov Model, High-Definition Map, Lane-Level, Map Matching, Viterbi Algorithm

DEFINITIONS

GLOSSARY

Bounded Variable Sliding Window Modification of Variable Sliding Window that sets an upper bound on the time before a solution is returned.

Centerline Polyline in the center of a lane, with unique ID and metainformation such as speed, direction, lane border marking types.

Connectivity-based Methods based on exploiting the graph structure of the map.

Extrapolation-based Methods based on sampling and extrapolating coordinates without regards to connectivity.

Fixed Sliding Window Online algorithm that returns solutions within fixed time lags without optimality guarantees.

Ground truth The actual location.

HD map High-definition map built for autonomous vehicles.

Hidden Markov Model Statistical model of a system that can be modeled as a Markov process.

- **Emission probability** The probability of a certain observation given a certain state.
- **Hidden state** Underlying location not directly visible to the observer, i.e. hidden.
- **Initial state probability** The state probability distribution at the first time step.
- **Observation** Measurement data of the ground truth.
- **Transition probability** The likelihood of moving from a certain state to a certain other state.

Lane Part of a road, intended for use by a single line of vehicles.

Lane change Maneuvering the vehicle into a different lane.

Lane group Group of lane segments that share borders with each other and have the same start and end.

Lane marker Synonym to Lane marking.

Lane marking Painted markings used to divide lanes.

Lane segment Segment of a lane, represented as a 2D-polygon and containing metainformation given by the centerlines.

Lane stitching The procedure of cropping and stitching together matched lane segments into a continuous path.

Map matching The procedure of assigning geographical objects to locations on a digital map.

- **Lane level** Matching to lane segments.
- **Road level** Matching to road segments.

Markov process Stochastic process where the current state depends only on the previous state.

Overpass Road that crosses over another road.

Path length error Algorithm performance metric, based on the fraction of incorrect route.
Penalty Function that decreases the matching probability of a candidate lane according to some parameter constraint.
Real-time kinematics Technique used to improve the GPS coordinates.
Recall Algorithm performance metric, based on the fraction of correctly matched centerline IDs.
Underpass Road that crosses under another road.
Variable Sliding Window Online algorithm that returns solutions given variable input increments and guarantees optimality.
Viterbi Algorithm HMM decoder of choice, that solves the map matching problem by finding the most probable state sequence.
  Offline Implementation of the algorithm that is fed the complete drive log and returns the full path.
  Online Implementation of the algorithm that is fed driving data in small batches and returns the path in increments.
Volvo Scalable Product Architecture Vehicle platform developed and manufactured by Volvo Cars.

ACRONYMS

AD Autonomous driving.
ADAS Advanced driver assistance system.
AV Autonomous vehicle.
BVSW Bounded Variable Sliding Window.
FSW Fixed Sliding Window.
GPS Global Positioning System.
HD High Definition.
HMM Hidden Markov Model.
IMU Inertial Measurement Units.
INS Inertial Navigation System.
PLE Path length error.
RTK Real-time kinematics.
SPA Scalable Product Architecture.
VA Viterbi algorithm.
VSW Variable Sliding Window.

CONTENTS

Definitions

1 INTRODUCTION 1
- 1.1 Background 1
- 1.2 Previous Works 2
- 1.3 Purpose and research questions 2
- 1.4 Thesis Scope 3
- 1.5 Report layout 3

2 THEORY 5
- 2.1 Map matching 5
- 2.2 Hidden Markov Models 6
  - 2.2.1 Markov property 7
  - 2.2.2 Emission probabilities 7
  - 2.2.3 Transition probabilities 8
  - 2.2.4 Initial state distribution 8
  - 2.2.5 Trellis diagram 8
- 2.3 Viterbi Algorithm 9
  - 2.3.1 Derivation and algorithm 9
  - 2.3.2 Online Viterbi algorithm 11

3 MAP REPRESENTATION 17
- 3.1 Centerline assignment 17
- 3.2 Graph representation 20
- 3.3 Lane types 20
- 3.4 Lane width analysis 21

4 OBSERVATION DATA 24
- 4.1 Sensors 24
  - 4.1.1 GPS 24
  - 4.1.2 Inertial navigation system 25
  - 4.1.3 Forward looking camera 25
- 4.2 Variables 26
  - 4.2.1 Latitude and longitude 26
  - 4.2.2 Altitude 27
  - 4.2.3 Number of satellites 27
  - 4.2.4 Heading 27
  - 4.2.5 Speed 27
  - 4.2.6 Yaw rate 28
  - 4.2.7 Distance to lane markers 28
  - 4.2.8 Lane marker types 28
  - 4.2.9 Confidence of vision system 28
  - 4.2.10 Lane change indicator 28
- 4.3 Data analysis 28

4.3.1 Lane marker type identifications 29
4.3.2 Lane change indicator 30
4.3.3 Yaw rate during lane change 32
4.3.4 Lane width 34

5 IMPLEMENTATION 36
5.1 Hidden states in HMM 36
5.2 Observations in HMM 36
5.3 Emission probability 37
5.3.1 GPS observation probability 37
5.3.2 Penalties 37
5.3.3 Final emission probability 41
5.4 Initial probability 41
5.5 Transition probability 42
5.5.1 Connectivity-based probabilities 42
5.5.2 Extrapolation-based probabilities 45
5.5.3 Filtering illegal transitions 48
5.5.4 Accounting for lane changes 48
5.6 Implementation of the Viterbi algorithm 49
5.6.1 Modification of the Viterbi algorithm 49
5.6.2 Online Viterbi algorithm 50

6 EVALUATION METHOD 52
6.1 Ground truth 52
6.2 Classification of result 53
6.3 Metrics 54
6.3.1 Recall 54
6.3.2 Stitching lanes 55
6.3.3 Path length error 56

7 RESULTS AND DISCUSSION 57
7.1 First evaluation 57
7.2 Tuning of neighborhood depth 58
7.3 Performance on test data 61
7.4 Importance of probabilistic components 61
7.5 Result analysis 64
7.6 Privacy aspects 72

8 CONCLUSION 73
8.1 Future work 73

BIBLIOGRAPHY 75

LIST OF FIGURES

Figure 1 Map matching on road level and lane level 6
Figure 2 Discrete HMM 7
Figure 3 Trellis diagram of HMM 8
Figure 4 Offline VA decoder 12
Figure 5 VSW decoder 15
Figure 6 FSW decoder 16
Figure 7 TomTom map 18
Figure 8 Zoomed in TomTom map 19
Figure 9 ID assignment to lane centerlines 20
Figure 10 Lane merges and lane splits 21
Figure 11 Lane width frequency 22
Figure 12 Width of current and neighboring lanes 23
Figure 13 Sampling local lane widths 23
Figure 14 Inertial Navigation System 26
Figure 15 Vehicle heading and distance to lane markers 27
Figure 16 Noisy GPS on a two-lane road 29
Figure 17 Frequency of marker type classifications 30
Figure 18 Incorrect lane change indications 31
Figure 19 Accuracy of lane change signals 32
Figure 20 Yaw rate during lane changes 33
Figure 21 Lane widths by map and observations 35
Figure 22 Probabilistic penalties 39
Figure 23 Limiting candidate lanes on detected marker types 40
Figure 24 Valid lane transitions 42
Figure 25 Lane-connectivity on different depths 43
Figure 26 Constant connectivity-based transitions 43
Figure 27 Refined connectivity-based transitions 44
Figure 28 Extrapolation of vehicle position 46
Figure 29 Extrapolation-based transitions 46
Figure 30 Speed distribution used in extrapolations 47
Figure 31 Heading deviation 47
Figure 32 Traffic law constraints 48
Figure 33 Lane change signal added to transitions 49
Figure 34 Holes in TomTom map 50
Figure 35 Overpasses 53
Figure 36 Ground truth and Viterbi path comparison 54
Figure 37 Lane stitching 55
Figure 38 North-east bounding box 59
Figure 39 Performance for different depths 60
Figure 40 Performance of down-scaled models 63
Figure 41 Resulting path of a typical drive 64
Figure 42 Good case 1: correct lane with bad GPS 65

Figure 43 Good case 2: compatibility with missing lanes 66
Figure 44 Good case 3: handling lane changes 66
Figure 45 Bad case 1: no detected lane markings 67
Figure 46 Bad case 2: large GPS error 67
Figure 47 Bad case 3: lane change indicator enforces lanes 68
Figure 48 Bad case 4: ambiguous lane markings 68
Figure 49 Bad case 5: holes and interference from other roads 69
Figure 50 Recall for different GPS standard deviation 70

LIST OF TABLES

Table 1 Data from TomTom map 17
Table 2 Lane width errors for different lane types 22
Table 3 Data from vehicle sensors 26
Table 4 Results when using model with constant connectivity-based transitions 57
Table 5 Results when using model with refined connectivity-based transitions 58
Table 6 Results when using model with extrapolation-based transitions 58
Table 7 Results when using the final model on the test data 61
Table 8 Results when using a naive matcher on the test data 61

1 INTRODUCTION

Today's maps are designed for human use. More specifically, they are intended to be used for turn-by-turn navigation purposes [1, 2]. Other environmental information, such as the type and location of lane markings, any debris lying on the road and road maintenance obstructions, is visually observed by the map user as she travels along the roads. Autonomous vehicles (AVs), however, require very different maps. These need to be in high-definition (HD), providing the robots with very precise localization and the possibility to perceive their environment [2, 3, 4]. HD maps for AVs also need to be updated continuously, to track events such as road accidents or traffic congestion [1].

Goldman Sachs has predicted that the market for advanced driver assistance systems (ADAS) and autonomous driving (AD) will grow from US$3 billion in 2015 to US$96 billion by 2025 [4]. With HD maps arguably being essential for the industry, the competition for being first at mapping the world has many contestants [5, 6].

Just as the AV needs minute information about its environment, it needs to know its position on the road. This problem is called map matching, and can more formally be described as the procedure of matching geographical objects to digital map locations [7]. Today, existing map matching algorithms, built on e.g. probability theory, fuzzy logic theory and belief theory [8], can successfully map GPS coordinates to a certain road segment in order to give information about the surroundings of a vehicle. However, specific features of the road such as the current lane of the car are difficult to obtain via map matching, mostly due to the noisy nature of GPS signals.

Adding sensor data such as speed, yaw rate and detected lane markings and lane changes to the GPS data, the objective of this thesis is to develop a lane-level map matcher based on robust Hidden Markov Models (HMMs), which has yet only been used for road-level map matching [9]. The HD map is provided by TomTom [10]. The data used has been recorded by Volvo test vehicles via the Drive Me project [11].

1.1 BACKGROUND

This thesis is carried out at the ADAS and AD software company Zenuity [12], in close collaboration with team Magellan. Magellan, and related teams at Zenuity, work with building maps from data collected by sensors, i.e. probe-sourced data.

For the team and the company as a whole, information about the current lane is valuable for a multitude of applications. In a short term perspective, the result of this thesis will help to infer the position of a vehicle in the road network. The solution can then be used to align noisy data and validate the

HD map. It can also be used to assist localization algorithms, that need the current lane information to find the exact position of the vehicle. Additionally, the solution is an integral part for AVs to navigate an HD map, and long term, could be incorporated in some ADAS or AD software.

1.2 PREVIOUS WORKS

Hidden Markov Models are probabilistic models useful for modeling time series [13]. HMMs have mostly been used in speech recognition applications, first implemented by IBM as early as the 1980's [14]. In 2009, Microsoft's Newson and Krumm [9] were the first to propose HMMs to solve the map matching problem. With great success, they managed to map sparse and noisy GPS to a road network. Goh, et al. implemented an online version of Newson and Krumm's HMM in 2012 [15], tackling the trade-off between accuracy and output delay. They outperformed traditional online methods with a fixed sliding window through the use of a variable sliding window and support vector machines to learn the HMM parameters. Unlike the fixed sliding windows used for benchmarking, the variable sliding window guaranteed the global optimum solution.

In 2017, Luo et al. [16] evaluated an implementation of the HMM-based map matching on both GPS data and cellular network data. Their finding indicated that HMM-based algorithms have both higher efficiency and accuracy than other map matching algorithms. In 2018, Murphy and Pao at Lyft, Inc. [17] combined HMMs and standard free-space tracking methods in such a way that their approach was robust to incorrect or incomplete maps, allowing tracking of vehicles outside the known road networks.

While this research in HMM-based map matching methods has shown great promise for the field, no known HMM approach for lane-level map matching exists today. Since HD maps are a novelty, not much research in the general area of lane-level map matching has been conducted. In fact, thus far only the following two research papers about the topic have been published. Rabe et al. [18] proposed a method for ego-lane estimation based on least square optimization and hypothesis testing in 2016, using sensor data such as GPS, odometry, visual lane marking detectors and radar. Evaluating their model in an urban setting, Rabe et al. got very low error rates. The second article on lane-level map matching by Li et al. [19] focuses on the so called integrity problem. Acknowledging that an algorithm in an autonomous navigation system should not make a decision if there is an ambiguity in the matching of lanes, they presented a formalization of this integrity problem and offered a solution through particle filtering.

1.3 PURPOSE AND RESEARCH QUESTIONS

The vehicle industry is putting a lot of effort into developing ADAS, equipping new car models with a multitude of sensors that log a vast amount of data. This thesis project takes advantage of this new data when implementing a map matching algorithm that is capable of matching noisy consumer

GPS and probe data to a high definition road map at the lane-level. The output of the algorithm is a lane matched to each GPS measurement.

As previously mentioned, the main objective of the project is to develop a functioning map matching algorithm based on HMMs. From this, the following research questions are defined:

- How, and based on which parameters, should the Hidden Markov Model be implemented?
- How can the performance of the map matching algorithm be measured?

1.4 THESIS SCOPE

This thesis project corresponds to a magnitude of 30 academic credits for each person. In order to fit the scope of thesis to the time available, the following limitations were defined:

- RADAR and LiDAR sensor data is not used, since these need a significant amount of pre-processing in order to be deemed useful. In contrary, the vision data, containing clearly labeled objects such as lane markings and land marks, is already processed and does not need much further extraction. Together with GPS and odometry data, i.e. data from motion sensors, the vision data is decidedly in scope.
- Model fitting is performed offline, i.e. with a static data set. Although actual probe-sourced solutions will require continuous online parameter estimation, this thesis can be said to be a proof of concept, rather than a complete product.
- The map matching algorithm is primarily implemented as an offline solution, in the sense that all the required information is given before the algorithm proceeds to find a solution. In contrast, an online algorithm handles its input piece-by-piece in a serial fashion, meaning that the whole input is not given beforehand but rather processed continuously. The latter is necessary for real-time applications, and has higher demand on the time efficiency, primarily. A simple online solution is also offered, to illustrate how it can be achieved. However, focus lies on the offline map matching .
- The GPS coordinates and probe data will be mapped to road lane segments which have median length 23 meters, meaning that the vehicle can be anywhere within that segment. Depending on time, the segments might be split into smaller sub-segments in order to refine the accuracy of the vehicle's location on the road lane.

1.5 REPORT LAYOUT

The report is structured as follows. First, necessary theory is introduced in Chapter 2. The used map and sensor data are thoroughly explained in

Chapters 3 and 4. Some variable analyzes are also performed. Based on those findings, a model is proposed in Chapter 5. The implementation and adaptation of the Viterbi algorithm is described in Section 5.6. Suggestions for performance metrics are given in Chapter 6. Finally, the algorithm is evaluated in Chapter 7, and the thesis work is concluded in Chapter 8.

2 THEORY

This chapter states the necessary theory. First, the problem of map matching is described, in the context of standard, or road-level, as well as lane-level applications. As stated before, the sensor and map data are incorporated into a Hidden Markov Model, which is given a mathematical formulation below. This model is then decoded by the Viterbi algorithm to solve the map matching problem, and as such, the chapter is concluded with a description of the algorithm.

2.1 MAP MATCHING

As previously mentioned, map matching is the technique of matching location data to a digital map in order to identify where a vehicle is in a road network [7, 20], as visualized in Figure 1a. It is a vital component of navigation and route guidance systems [21].

Typically, the location of a vehicle is provided through a global positioning system, GPS, either by itself or in combination with an inertial navigation system or dead reckoning. These sensors are not perfect, however, and using them introduces an error in the location data [20].

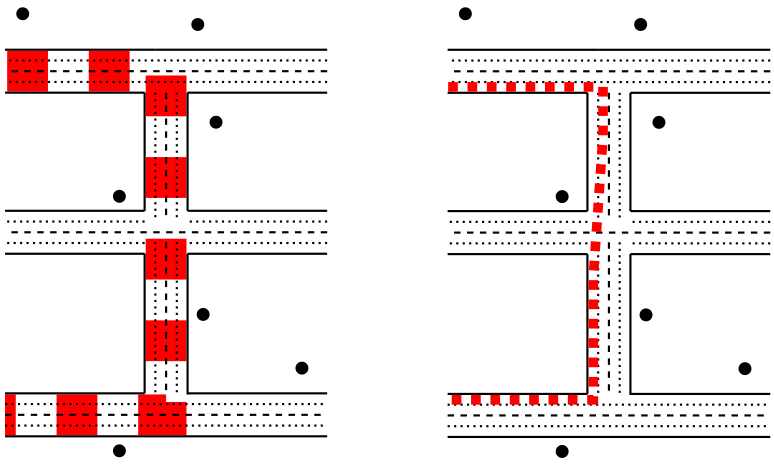
A digital map contains geographical information, providing the location of a user with a spatial reference. Similarly to the location systems, maps are not always perfect. It may have positional errors, caused by e.g. measurement errors when creating the map. It may also be incomplete, or not up-to-date with the true road network [21].

If both data and map were perfect, the process of locating a user on a road network would be trivial — it would simply be possible to take the physical location data and output the corresponding road from the map. Now, since this is not the case, map matching is necessary to reconcile these inaccurate location and map data and yield an accurate digital area wherein the vehicle is located [20, 21]. A formal definition of the map matching problem is given in Definition 1.

Definition 1. A vehicle is moving along a finite set of roads, N. These are estimated as N_{est} on a digital map. A location system provides estimates P^k_{est} of the true vehicle position P^k at discrete times $\{t_0, ..., t_k, ..., t_K\}$. *Map matching* aims at matching P^k_{est} to a road $E \in N_{est}$, and then determine the true road in N.

The process of creating a map matching algorithm includes three steps:

1. Extract interesting features to be used for map matching. These include position and shape features from the available data, as well as the map.
2. Calculate the matching similarity, defined by the matching algorithm, between the location data and all road sequence candidates.



(a) Road-level. **(b)** Lane-level.

Figure 1: GPS location measurements (black dots) on road network. The red dashed line shows the traversed path obtained by solving the map matching problem.

3. Select the path with the highest similarity.

These key concepts of map matching can also be applied to lane-level inference. That is, lane-level map matchers match location data to lanes, rather than roads. See Figure 1b for a visualization of the difference. The lane-level matching can be used to aid localization algorithms in advanced, future navigation systems. These would, for example, identify whether the vehicle is in the recommended lane. If not, the system could aid the driver by proposing a lane change [20]. Extrapolating to the AD case, the vehicle would need to make these decisions on its own.

Given that road-level map matching is non-trivial, the task of a higher resolution map matching becomes complex. Merely location data and road networks are no longer sufficient. Instead, the problem puts higher demands on the accessible information.

Location data needs to be complemented with other sensor data. Odometry data, such as vehicle speed and heading are examples of such. The digital map needs to contain lane-level information, at the very least. Metadata, such as speed limits and lane headings may also be useful.

2.2 HIDDEN MARKOV MODELS

A Hidden Markov Model (HMM) is a statistical model in which the system being modeled is assumed to be a stochastic process with unobserved, i.e. hidden, states. The states are contained in a set $\mathbf{S}$, while $\mathbf{X} \subseteq \mathbf{S}$ describes states in a sequence. More explicitly, the set of states might be $\mathbf{S} = \{s_1, s_2\}$ and a state sequence $\mathbf{x}$ could look like $\mathbf{x} = [x_1, x_2, x_3] = [s_2, s_2, s_1]$, where $x_1, x_2, x_3 \in \mathbf{X}$ and $s_1, s_2 \in \mathbf{S}$.

The states of an HMM are not directly visible to the observer, which is why they are called hidden. Rather, there exist observations $\mathbf{Y}$ that stem from these hidden states. The sequence of observations are generated by

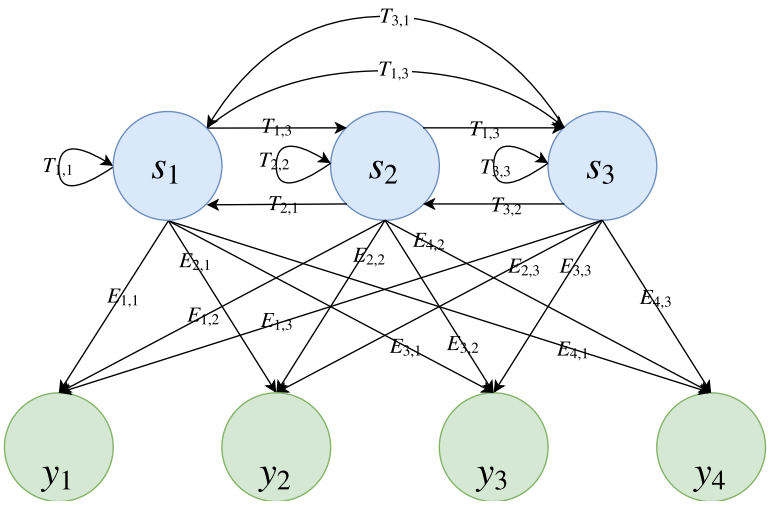


Figure 2: A graphical representation of a discrete HMM with three states s_1, s_2, s_3 and four observations y_1, y_2, y_3, y_4. The corresponding transition probabilities are $T_{i,j}$, $i, j \in \{1, 2, 3\}$, and the emission probabilities are $E_{k,i}$, $k \in \{1, 2, 3, 4\}, i \in \{1, 2, 3\}$.

a second stochastic process. In that sense, an HMM is a doubly stochastic process [22].

The HMM is constituted of the three main parameters: transition probabilities, emission probabilities and initial state distribution. As a sub-category of Markov Models, the HMM also satisfies the Markov property. This property and the main components are described in the following subsections. The architecture of the HMM is illustrated in Figure 2. For HMM decoding purposes, however, it is more useful to have another representation. As such, the trellis diagram in Figure 3 is presented and described.

2.2.1 Markov property

A stochastic process $X = \{X_k\}_{k=0}^{K}$ is a Markov process if it fulfills

$$P(X_K = x_k | X_{k-1} = x_{k-1}, X_{k-2} = x_{k-2}, ..., X_0 = x_0) = P(X_k = x_k | X_{k-1} = x_{k-1})$$

for all times $k = \{1, ..., K\}$. That is, the current state x_k only depends probabilistically on the previous state. This is known as the Markov property.

2.2.2 Emission probabilities

The emission probability is associated with the second stochastic process, which models the distribution of observations. Each observation has an emission probability

$$E_k = p(y_k | x_k) \quad x_k \in \mathbf{X}, \tag{1}$$

which is a probability distribution function (pdf) that reflects the probability of observing y_k at time t_k when being in state x_k [16, 23]. When considering explicit states, the emission probability is more appropriately given as

$$E_{k,i} = p(y_k | s_i) \quad s_i \in \mathbf{S}. \tag{2}$$

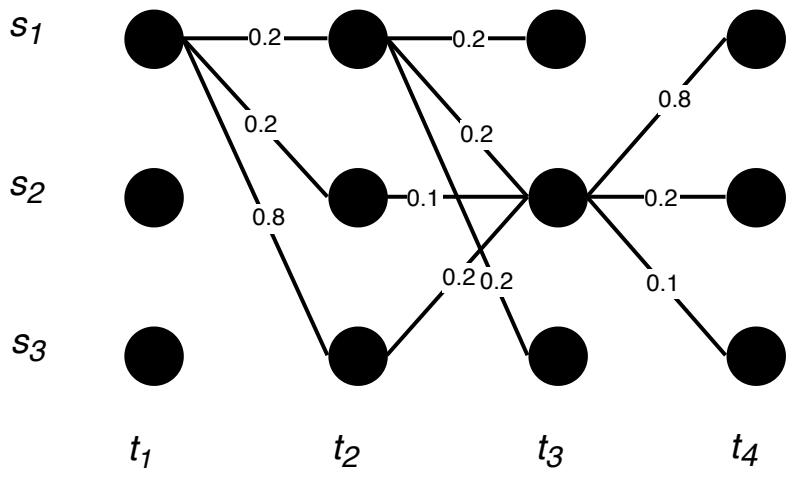


Figure 3: Trellis diagram representation of the HMM depicted in Figure 2 for four time steps. Edge weights correspond to the product of transition and emission probabilities. In this instance, some of the products are zero. Corresponding edges have effectively been removed.

2.2.3 Transition probabilities

The transition probability is the likelihood of moving from a state x_k to a state x_{k+1} at time t_k. It can be written as

$$T_k = p(x_{k+1}|x_k) \quad x_k, x_{k+1} \in \mathbf{X}. \tag{3}$$

When considering explicit states, the transition probability is more appropriately given as

$$T_{i,j} = p(x_{k+1} = s_j|x_k = s_i) \quad s_i, s_j \in \mathbf{S}. \tag{4}$$

2.2.4 Initial state distribution

The distribution of initial state probabilities describes the likelihood of starting in each state,

$$\Pi_i = p(x_0 = s_i) \quad s_i \in \mathbf{S}. \tag{5}$$

2.2.5 Trellis diagram

A discrete HMM can be represented as a trellis diagram, where time steps are incorporated. Each node in the diagram corresponds to a distinct state at a given time, and the edges represent possible transitions to states at the next time step. The discrete HMM given in Figure 2 has a trellis representation as shown in Figure 3 for three time steps.

A useful property of this particular representation is that every possible state sequence in the model corresponds to a unique path through the trellis, and the other way around. Because of this, it is a useful representation when applying dynamic programming algorithms to an HMM, for example when finding the most probable path through the model by using the Viterbi algorithm [24].

2.3 VITERBI ALGORITHM

Inferring which sequence of states caused a specific sequence of observations is called decoding [25]. The Viterbi algorithm, first proposed in 1967 by Andrew Viterbi, is one such decoder [26, 24, 27, 28]. It is, in fact, the most commonly used algorithm for decoding HMMs [25]. It solves the problem of estimating the maximum likelihood of state sequences, i.e. finds the most probable state sequence.

The set of transition sequences can be defined as $\boldsymbol{\xi} = \{\xi_1, ..., \xi_{K-1}\}$, with the transitions $\xi_k = (x_{k+1}, x_k)$ at the given time k. These map one-to-one to the state sequence $\mathbf{x} = (x_1, ..., x_K)$. Using this notation, the observations mentioned in Section 2.2, $\mathbf{y} = [y_1, \ldots, y_K]$, $y_k \in \mathbf{Y}$, can be described as the output of a channel, whose input is the transition sequences. The channel is memory-less in the sense that $p(\mathbf{y}|\boldsymbol{\xi}) = \prod_{k=0}^{K} p(y_k|\xi_k)$, i.e. each observation y_k only depends probabilistically on the transition ξ_k [24].

With all necessary notations introduced, it is now possible to formally define the maximum likelihood estimation problem.

2.3.1 Derivation and algorithm

The Viterbi algorithm seeks to find

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) \iff \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \iff \arg\max_{\mathbf{x}} p(\mathbf{x}, \mathbf{y}) \quad \forall\, \mathbf{y} \in \mathbf{Y},$$

where $\mathbf{Y}$ is the set of possible observation sequences. The first equivalence is derived from Bayes' rule, $p(\mathbf{x}|\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})/p(\mathbf{y})$, where the denominator has been omitted. This is allowed since $p(\mathbf{y})$ is independent of $\mathbf{x}$ and as such, can be treated as a constant. The second equivalence follows from the chain rule. The joint probability can be rewritten as a recursion according to

$$\begin{aligned}
&p(x_1, ..., x_k, y_1, ..., y_k) = \\
&\{\text{chain rule}\} = \\
&p(x_k, y_k|x_1, ..., x_{k-1}, y_1, ..., y_{k-1})p(x_1, ..., x_{k-1}, y_1, ..., y_{k-1}) = \\
&\{\text{Markov property}\} = \\
&p(x_k, y_k|x_{k-1})P(x_1, ..., x_{k-1}, y_1, ..., y_{k-1}) = \\
&\{\text{Bayes' rule}\} = \\
&p(y_k|x_k, x_{k-1})p(x_k|x_{k-1})p(x_1, ..., x_{k-1}, y_1, ..., y_{k-1}) = \\
&\{\text{memoryless observation noise}\} = \\
&p(y_k|x_k)p(x_k|x_{k-1})p(x_1, ..., x_{k-1}, y_1, ..., y_{k-1}),
\end{aligned}$$

for all $k = 2, ..., K$. $p(y_k|x_k)$ is recognized as an emission probability, and $p(x_k|x_{k-1})$ a transition probability. Using the shorthand notation,

$\mathbf{x} = (x_1, ..., x_K) \equiv \mathbf{x}_{1:K}$ and $\mathbf{y} = (y_1, ..., y_K) \equiv \mathbf{y}_{1:K}$, the maximization problem becomes

$$\begin{aligned}
&\mathbf{x}^*_{1:K} = \\
&\arg\max_{\mathbf{x}_{1:K}} p(x_1, ..., x_K, y_1, ..., y_K) = \\
&\arg\max_{x_K} p(y_K|x_K) p(x_K|x^*_{K-1}) \arg\max_{\mathbf{x}_{1:K-1}} p(x_1, ..., x_{K-1}, y_1, ..., y_{K-1}) = \\
&\arg\max_{x_K} p(y_K|x_K) p(x_K|x^*_{K-1}) \mathbf{x}^*_{1:K-1},
\end{aligned}$$

where x^*_{K-1} is the last element of $\mathbf{x}^*_{1:K-1}$, and $\mathbf{x}^*_{1:K-1}$ is defined analogously to $\mathbf{x}^*_{1:K}$. The maximum likelihood estimation problem is thus recursive, with the last recursion $\mathbf{x}^*_1 = (x^*_1) = p(y_1|x_1)p(x_1|x_0)$. x_0 is a global initial state, with $p(x_1|x_0)$ describing the probability of starting in some state x_1. $p(x_1|x_0)$ is therefore the initial distribution Π. Using these results, the Viterbi algorithm is defined in Algorithm 1.

Algorithm 1 The Viterbi algorithm. $\mathbf{y}$: $K \times 1$ is the sequence of observations, $\mathbf{S}$: $N \times 1$ is the state space, T : $N \times N$ the transition matrix, E : $K \times N$ the emission matrix and the Π : $N \times 1$ the initial distribution.

```
1:  function VITERBI(y, S, T, E, Π)
2:      for n = 1, ..., N do
3:          probtable[n, 1] = Π_n
4:          pointer[n, 1] = 0
5:      end for
6:      for k = 2, ..., K do
7:          for n = 1, ..., N do
8:              probtable[n, k] = max_i [probtable[i, k − 1] × T_{i,n} × E_{k,n}]
9:              pointer[n, k] = arg max_i [probtable[i, k − 1] × T_{i,n} × E_{k,n}]
10:         end for
11:     end for
12:     return BACKTRACE(probtable, pointer)
13: end function
14:
15: function BACKTRACE(probtable, pointer)
16:     beststateindex[K] = arg max_i probtable[i, K]
17:     bestpath[K] = s_{beststateindex[K]}
18:     for k = K, ..., 2 do
19:         beststateindex[k − 1] = pointer[beststateindex[k], k]
20:         bestpath[k − 1] = s_{beststateindex[k−1]}
21:     end for
22:     return bestpath
23: end function
```

The VA does not have to exhaustively examine all paths in the trellis in order to find the optimal one. Instead, when encountering a situation where it is possible to reach a certain state through several paths, only the most likely path is retained. The inferior paths are redundant and thus pruned, implying that the search space is truncated. By this, it is reasonable to let the aggregated non-eliminated edges be referred to as surviving paths.

When the VA has processed the whole sequence, the effect of the path pruning is that every node in the trellis only has one incoming edge. More specifically, this means that every state has a unique former state at every

time step. The VA proceeds to obtain the optimal, unique path originating from the most likely last state by back-tracing the singular incoming edges.

An explicit instance where VA is used to find the most probable path through the trellis in Figure 3 is visualized in Figure 4.

Rows $2-5$ in Algorithm 1 have time complexity $\mathcal{O}(N)$ since it loops over N states and performs a simple multiplication, taking constant time. Rows $6-11$ iterate over $K-1$ observations and loop over all N states in each iteration. For every state, two maximum operations are done which involves comparison against all other $N-1$ states. Thus, the time complexity for this part becomes $\mathcal{O}((K-1) \times N \times (N-1)) = \mathcal{O}(KN^2)$. Rows $18-21$ are evaluated in $\mathcal{O}(KN)$ time since it loops over $K-1$ observations and involves a look-up in an array, taking linear time in the size of the array. Hence, the total time complexity of Algorithm 1 is $\mathcal{O}(N + KN^2 + KN) = \mathcal{O}(KN^2)$.

2.3.2 Online Viterbi algorithm

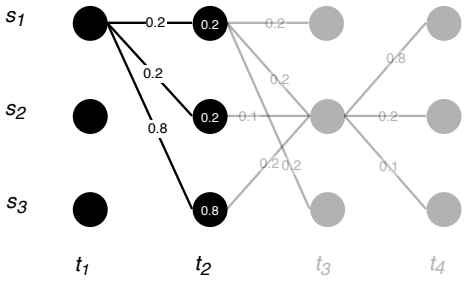
As described in Section 2.3, the Viterbi algorithm finds the optimal state sequence given a finite number of observations. Having considered all observations, it identifies the most probable last state and traces back-pointers until the start of the sequence, and thereby gathers the ultimate path. An apparent deficiency of this approach is its inability to handle infinite observation sequences. Additionally, it is not able to yield incremental results in the form of a sequence of sub-paths, even with a finite number of observations [29].

There are certainly cases where an incremental output is desired. For example, an application delivering real-time information about a vehicle's traversed route relies on such output. In turn, observations would be fed to the algorithm in small patches. One approach to yield optimal sub-paths given gradually inputted observations is to consider so called convergence points, explained further down. This method is often referred to as Variable Sliding Window (VSW) and is further described in the following section. While VSW guarantees optimality, the sequence does not necessarily contain any convergence point. If there is no convergence point in an infinite sequence of observations, the VSW algorithm will never return any path. A modified approach solving this particular issue is also presented later in this section.

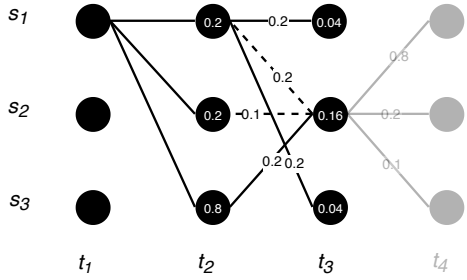
An alternative online method, Fixed Sliding Window (FSW), uses fixed time lags within which the algorithm is required to return a solution. Although it is certain to yield sub-solutions at predefined time steps, it comes with the drawback of no optimality guarantees. Such a fixed-lag method is described further below.

Variable Sliding Window

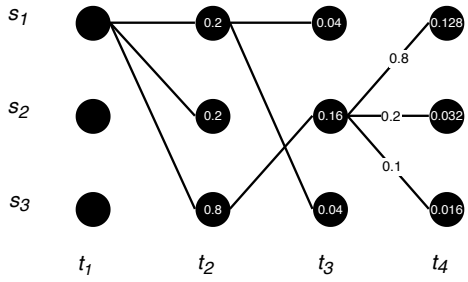
An online Viterbi algorithm can be achieved by using VSW. The key concept of VSW is to delay the output until a convergence point is found. A convergence point has been found whenever all back-pointers point to the same state. This means that the sub-path derived before the convergence point is sure to be part of the final optimal path. The observation window size expands forward for every new observation and shrinks from behind



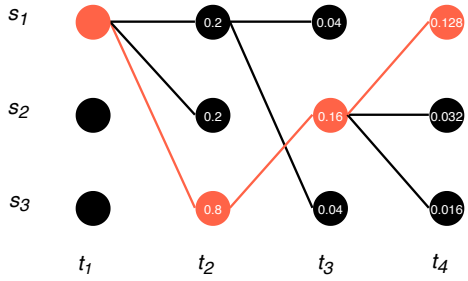
(a) At time t_2, each state only has one incoming edge from states in t_1. As such, no elimination is performed. The scores are stored in the nodes.



(b) At time t_3, accumulated scores are calculated as the product of the scores stored in t_2 and the edge weights. The highest scores are stored in each node. The edges corresponding to a lower score do not survive, as indicated by the dashed lines.



(c) In t_4, each node has only one incoming edge. Therefore, all edges survive.



(d) Lastly, back-tracing is performed from the most probable last state. The final path is found to be $[s_1, s_3, s_2, s_1]$.

Figure 4: Process of decoding the trellis in Figure 3 with the offline VA. The algorithm takes the full sequence and dynamically updates the scores of the surviving subpaths, storing the accumulated scores in the nodes. After the last time step, back tracing is performed and the found path, marked in red, is returned.

when a convergence point is found in the sequence covered by the window. The name Variable Sliding Window refers to the fact that the window sizes vary according to the topology of the graph representation [15]. So, by regarding variable window sizes of observations based on the occurrence of convergence points, a VSW algorithm is capable of yielding optimal subpaths without requiring the entire sequence of observations [30].

However, since the existence of convergence points is not guaranteed, the VSW algorithm risks to never return any solution. One way to prevent this is to set an upper bound on the window size. That way, when the bound is reached, the most likely solution up until the current stage is returned. This modified VSW algorithm can be referred to as Bounded Variable Sliding Window (BVSW) [15], and can also be used to achieve an online Viterbi algorithm. The VSW algorithm is given in provided in pseudocode Algorithm 2. Rows $11-13$ provide the modification of the original Viterbi algorithm which achieves the desired behaviour of VSW.

Algorithm 2 Online Viterbi algorithm using VSW. $\mathbf{y}: K \times 1$ is the sequence of observations, $\mathbf{S}: N \times 1$ is the state space, $T: N \times N$ the transition matrix, $E: K \times N$ the emission matrix and the $\Pi: N \times 1$ the initial distribution.

```
1:  function Viterbi(y, S, T, E, Π)
2:      for n = 1, ..., N do
3:          probtable[n, 1] = Π_n
4:          pointer[n, 1] = 0
5:      end for
6:      for k = 2, ..., K do
7:          for n = 1, ..., N do
8:              probtable[n, k] = max_i [probtable[i, k − 1] × T_{i,n} × E_{k,n}]
9:              pointer[n, k] = arg max_i [probtable[i, k − 1] × T_{i,n} × E_{k,n}]
10:         end for
11:         if pointer[1, k] = ... = pointer[N, k] then
12:             return Backtrace(probtable, pointer)
13:         end if
14:     end for
15:     return Backtrace(probtable, pointer)
16: end function
17:
18: function Backtrace(probtable, pointer)
19:     beststateindex[K] = arg max_i probtable[i, K]
20:     bestpath[K] = s_{beststateindex[K]}
21:     for k = K, ..., 2 do
22:         beststateindex[k − 1] = pointer[beststateindex[k], k]
23:         bestpath[k − 1] = s_{beststateindex[k−1]}
24:     end for
25:     return bestpath
26: end function
```

An explicit example of how the VSW algorithm processes an observation sequence and yields the optimal solution in increments can be seen in Figure 5. Since the last observation were registered at time t_4, no new observations are processed by the algorithm. Having all sub-sequences returned by the algorithm, $[s_1, s_3]$ and $[s_2, s_1]$, their concatenation form the optimal path

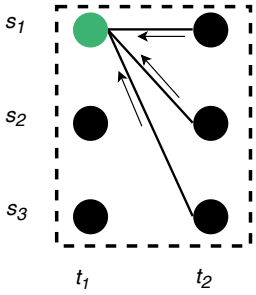
of the considered problem instance, $[s_1, s_3, s_2, s_1]$, which is seen in the last stage in Figure 5.

Fixed sliding window

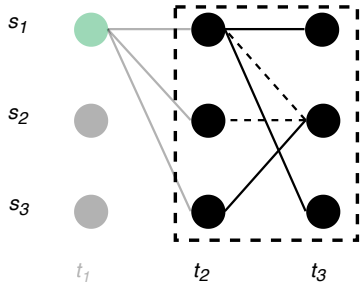
The FSW algorithm divides the observation sequence into smaller chunks of a predefined fixed length. The fixed length can also be referred to as a fixed time lag. Given that the fixed time lag is set to ω, the algorithm waits for ω number of observations until it starts to decode the sequence and returns a solution. Then it waits until ω more observations are given as input and processes that sequence. In this way, the sequence chunks are handled independently at every stage in the decoding process [15].

An illustration describing the process of the FSW algorithm, when the time lag is set to 2, is given in Figure 6. Concatenating all the solutions returned for the sequence chunks gives the final solution, which can be seen in the last stage in Figure 6. Even though the final path is the same as the one returned when applying the VSW algorithm, as seen in Figure 5, it is not generally the case. This is because the FSW approach does not guarantee optimality, which VSW does.

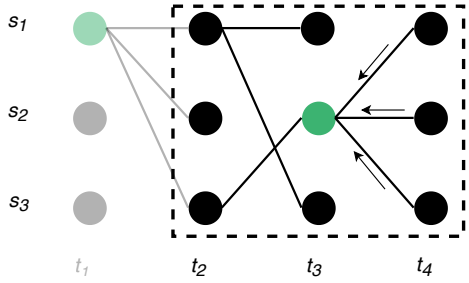
Longer time lag, or equivalently bigger window size, yields more accurate results at the cost of longer output delays, and vice versa. As such, it is evident that there is a trade-off between accuracy and time delay. The desired performance of the FSW algorithm is dependent on the application, which is why no general guideline of viable window sizes can be stated [30].



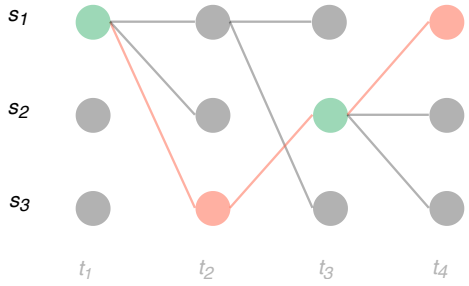
(a) At time t_2, all incoming edges come from s_1, which is identified as a convergence point. The subpath $[s_1]$ is returned.



(b) There are no convergence points found at time t_3, implying that nothing is outputted. Just as for offline VA, lower-scoring edges are eliminated.

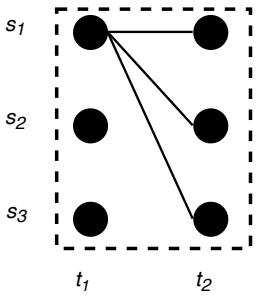


(c) In t_4, the convergence point s_2 is established, and the algorithm outputs the locally optimal subpath $[s_3, s_2]$.

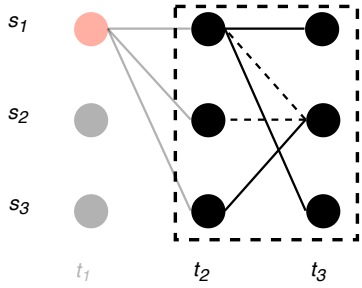


(d) Lastly, backtracing is performed from t_4 to the previous convergence point. The subpath $[s_1]$ is yielded.

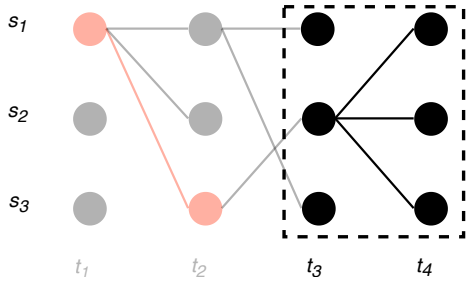
Figure 5: VSW decoding of the trellis in Figure 3. Edge weights and node scores have been omitted for the sake of simplicity. The VSW algorithm handles observations from times t_1, t_2, t_3, t_4 successively. Whenever a convergence point is encountered, the Viterbi solution up until that point is returned. The convergence points are recognized as green nodes, which together with the red nodes and edges form the optimal solution. The surrounding black dashed line indicates the window size.



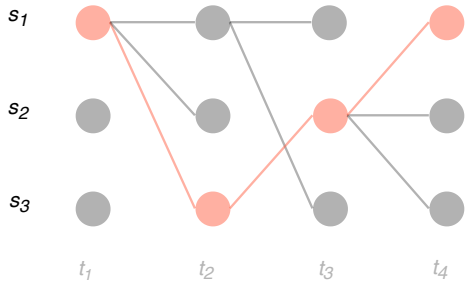
(a) At time t_2, backtracing is performed to t_1 and the most probable path thus far is yielded. According to the edge weights in Figure 3, this path is $[s_1]$.



(b) At time t_3, sub-optimal edges are eliminated. The locally optimal subpath $[s_3]$ is outputted.



(c) In t_4, the algorithm outputs the sequence $[s_2]$.



(d) The last backtracing step yields the subpath $[s_1]$.

Figure 6: FSW decoding of the trellis in Figure 3. Edge weights and node scores have been omitted for the sake of simplicity. The FSW algorithm handles observations from times $t_1, \dots t_4$ in chunks $[t_1, t_2]$, $[t_2, t_3]$, $[t_3, t_4]$ and $[t_4]$, when the time lag is set to 2. Once FSW receives two observations, it decodes the sequence and returns the most likely solution for that particular instance. Then it waits for another observation, slides the window forward and solve that sequence independently of the last one. The red nodes and edges form the obtained solution. The surrounding black dashed line indicates the window size.

3 MAP REPRESENTATION

The HD map is provided by the map company TomTom [10] and is accessed through an API built in-house. The mapped area is restricted to larger roads and their immediate surroundings in Gothenburg. These roads are Lundbyleden, Västerleden, Oscarsleden, Söderleden, Kungsbackaleden, Dag Hammarsköldsleden and parts of Hisingsleden, as seen in Figure 7.

Roads consist of one or multiple parallel lanes, each of which has a *centerline*, which in turn is represented as a polyline. A polyline is specified by a sequence of points which are connected by smaller line segments. Each line segment is defined by the longitude and latitude of its start and end point. Thus, the polylines are two-dimensional. The altitudes of the polylines are not known, implying that it is not possible to separate an overpass from an underpass merely by using the map API.

Also the left and right lane markers, signifying the lane border, are polylines. They are also associated with their marker type. These can be solid or dashed, for instance. The map also captures road delimiters such as guard rails and shaded area markings, as is seen in Figure 8. The different types of data available from the map are given in Table 1.

3.1 CENTERLINE ASSIGNMENT

The lane centerlines contain meta data such as speed limits, lane widths and lane markings, and range from 1 to 1000 meters in length. A centerline is given a unique ID and is defined as the stretch of lane for which the metadata remains constant. The longitudinal position of e.g. a traffic sign, or the change in lane marker type, is thus the delimiter between two different centerlines. An example of this is shown in Figure 9.

Lanes that share borders with each other and have the same start and end points are grouped together into so called *lane groups*. These share the

Table 1: Different type of data available from the map.

Type	**Variable**	**Description**	**Unit**
Orientation	*latitude*	Latitude	[degree]
	longitude	Longitude	[degree]
Lane specifics	*id*	Lane ID	
	speed limit	Speed limit	[m/s]
	width	Width of lane	[m]
	heading	Heading	[degree]
	left marker type	Type of left marker	
	right marker type	Type of right marker	

Figure 7: Google map image of Gothenburg, courtesy of Google, Inc., layered with the TomTom map. Blue lines represent roads and lanes. Red, black and yellow markers indicate traffic signs, stop lights, junctions and other road-related objects.

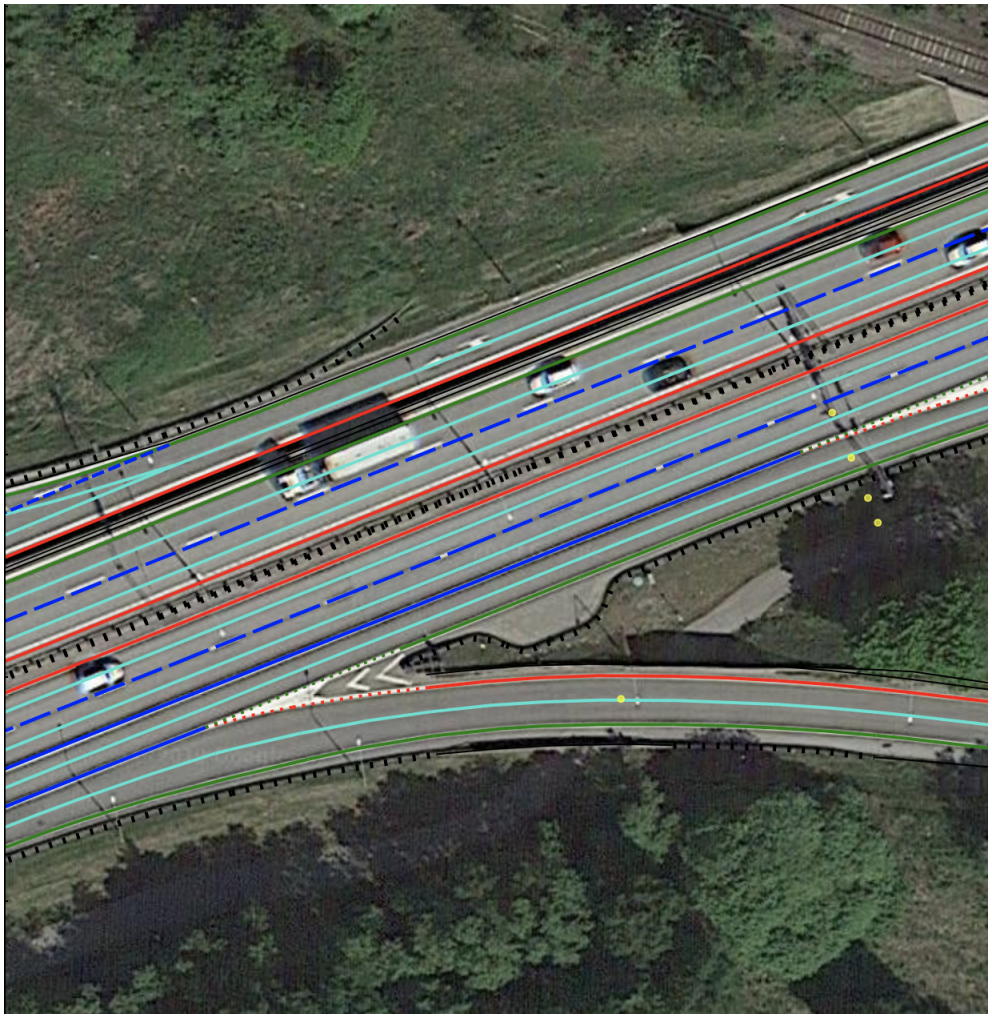


Figure 8: Zoomed in image of the map in Figure 7, better showing the various information provided by TomTom. Red and green lines are right and left road borders, blue lines are lane markers. Guard rails are represented by the black short-dashed polylines, and red and green dots enclose shaded areas, color-coded analogously to the road borders. Cyan-colored lines are lane centers.

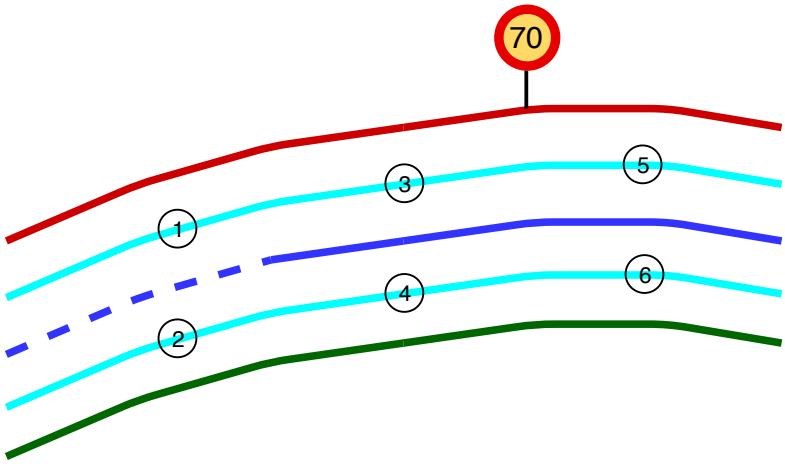


Figure 9: Example of how lane centerlines are assigned IDs. An eastbound road with two parallel lanes is shown. The cyan-colored lines are centerlines, located in the middle of each lane. The red and green lines are the left and right road markings, respectively, and the blue solid and dashed line is the lane marking separating the two lanes. The encircled numbers represent centerline IDs. The two lanes are split into separate centerlines where the lane marker changes from dashed to solid, and where the speed sign appears, yielding a total of six different centerlines, with IDs 1 to 6.

same metadata, such as speed limit, width and heading. In Figure 9, lane centerlines $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6\}$ compose three lane groups.

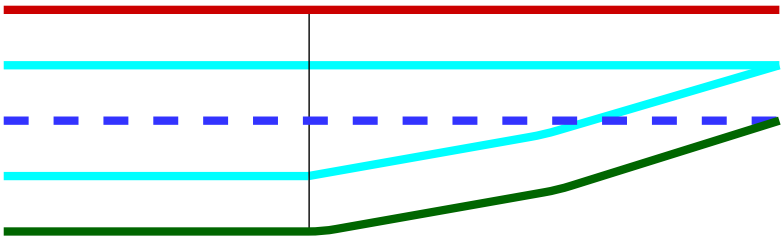
3.2 GRAPH REPRESENTATION

Additional meta data that is contained within a centerline is its immediate topology. Specifically, a centerline knows its lateral neighbors. It also knows its predecessor and successor. In Figure 9, centerline 4 is a lateral neighbor to centerline 3. Centerline 1 is the predecessor to centerline 3, and 5 its successor. This connectivity information allows the road network to be constructed as a graph, with centerlines as edges and connections as nodes.

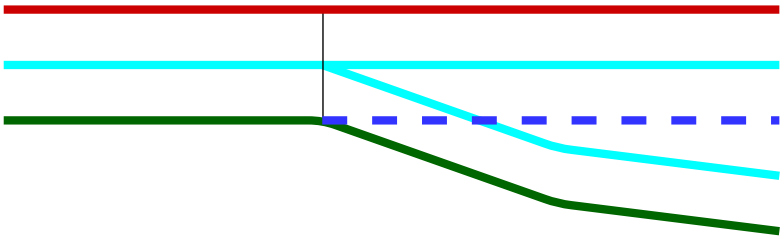
Unfortunately, the TomTom map is not complete. Certain lane centerlines are missing, leaving holes in the map. For map matching purposes, this is problematic since an observation on that lane cannot be properly matched against the map. A bigger misfortune is, however, that the topology is lost. Put differently, nodes and edges are missing. Using the example in Figure 9 again, imagine that centerline 3 is missing. This means that centerline 1 only has an edge to 2. The only possible graph path from centerline 1 to centerline 5 is thus $[1, 2, 4, 6, 5]$. If also centerline 4 is missing in the map, there is no valid path from centerlines 1 and 2 to 5 and 6.

3.3 LANE TYPES

There are three types, or categories, of lanes, dependent of the graph structure. Typically, one centerline has one predecessor and one successor, as in



(a) Two lanes of type continuation merging into one, with the triangular-shaped lane being of type merge.



(b) A lane of type continuation splitting into two lane of type split.

Figure 10: Visualization of Figure 10a lane merges and Figure 10b lane splits.

Figure 9. It is simply a *continuation* of the previous lane. However, there are occurrences where it instead has two predecessors or two successors, as in the cases of two lanes merging or splitting. Hence, these lanes can be categorized as *merges* and *splits*.

Merges and splits have different geometries than continuations. Where continuations have near-parallel left and right lane markings, merges and splits have more irregular shapes. A lane merge tends to have a pointy end, as shown in Figure 10a. Splits come in tuples, as shown in Figure 10b. One of these typically has a pointy start, while the other is more regularly shaped.

3.4 LANE WIDTH ANALYSIS

As seen in Table 1, the map contains information about the widths of the lanes. In order to establish whether information about lane widths could potentially provide useful information for the task of map matching, an investigation of the variable is carried out.

The first property studied is the variation in lane widths. For example, if they all are very similar to each other, the feature is useless for the purpose of map matching. If, however, the widths do vary in such a way that they aid map matching, they might be interesting to incorporate. The more unique properties a lane has, the more refined the distinction between lanes are. By this, it is favorable to have lanes with neighboring lanes having distinct widths. The lane width frequencies are visualized in a histogram in Figure 11. From the histogram, it can be established that 50% of the lanes maintain a width between 3.49 and 3.90 meters.

When studying the correlation between the width of a lane and the median lane width of its neighbors, it is evident that it has a linear tendency,

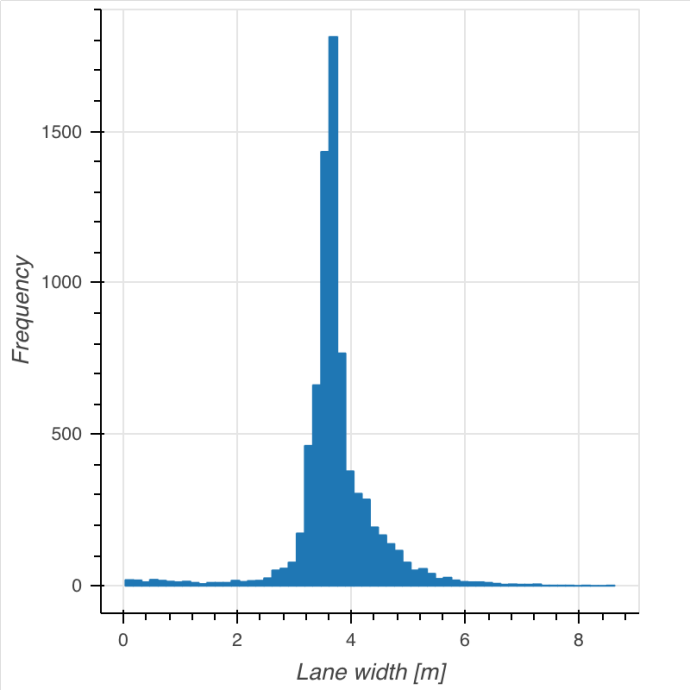


Figure 11: Histogram of lane widths in the map.

Table 2: Lane width errors for different lane types.

Type	Mean [m]	Standard error [m]
Continuation	0.0188	0.0057
Merge	0.1382	0.0175
Split	0.1762	0.0185

see Figure 12. This implies that lanes located close to each other tend to be of similar widths. By the horizontal string of points in Figure 12, it is apparent that lanes of various widths seem to be surrounded by lanes having a width around 3.7 meters. The cluster in the middle indicates that lanes which are around $3.5 - 4.5$ meters wide tend to have neighbors maintaining a similar width.

Secondly, a study of widths is performed based on lane types. This is motivated by the fact that the widths are given as scalar values, while the actual widths vary across the lanes. Since merges and splits have irregular geometries, it can be hypothesized that the reported widths may be misleading for these types. If the hypothesis holds, it may indicate that the lane width variable is unfit to use for map matching.

To get estimates of the true widths, points are sampled from the lanes. Local lane widths are then calculated at these points. A schematic view of the sampled widths of a lane is seen in Figure 13. The width error is then taken as the mean width difference between the reported lane width and the sampled widths. The errors for different lane types are seen in Table 2. Based on the results summarized in the table, it can be concluded that the lane width is very accurate for lane continuations, but more erroneous for lane merges and splits.

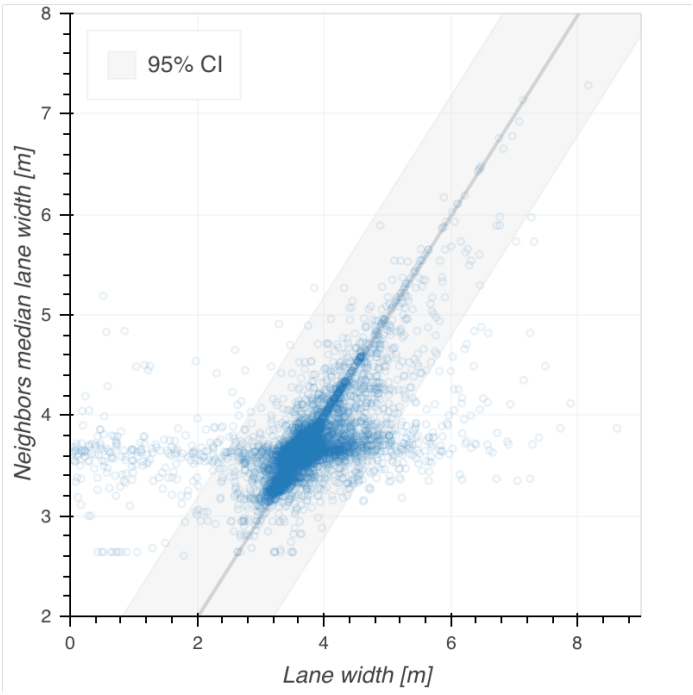


Figure 12: Scatter plot of lane widths and median of their neighbors lane widths.

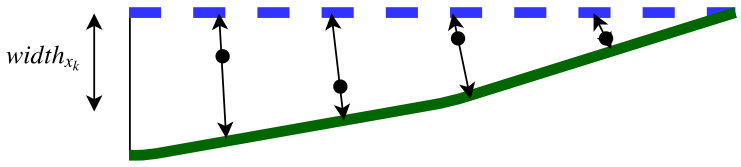


Figure 13: A lane merge with sampled points (black dots) and local lane widths, given by the lengths of the two-headed arrows. The reported lane width, $width_{x_k}$ is also shown.

4 OBSERVATION DATA

The data available at each time step is yielded by the sensors described in Section 4.1. This data can be divided into three categories based on the sensors from which the data originates. Positional data is retrieved by GPS, described in Section 4.1.1, data reflecting the movement of the vehicle is given by the Inertial Navigation System (INS), as described in Section 4.1.2, and information considering lane markers is gathered by a vision system, described in Section 4.1.3.

The Drive Me data contains a number of variables listed and described in Section 4.2. This data is split into a training and test set with the ratio $30-70$, corresponding to data set sizes of 475 and 1107 drives, respectively. The drives last from 1 to 12 minutes, and cover distances between 10 and $10,000$ meters.

Data analysis of the training set is performed in Section 4.3, and the model is then adapted accordingly to the findings. These analyses rely on the ground truth, i.e. having knowledge of the hidden states. The process of determining these is detailed in Section 6.1. The test set is later used to evaluate the model performance.

4.1 SENSORS

Sensors are devices that detect and measure changes in the surrounding environment. Events registered by the sensor can then be sent to some electronic component that can process the information. This project will consider data retrieved from sensors collecting and identifying information such as GPS positions, speed, acceleration, angular velocity and lane markings. In the following subsections, the sensors used to collect these data are described.

4.1.1 GPS

The Global Positioning System (GPS) is a satellite-based navigation system that provides continuous positioning and timing information worldwide under any weather conditions [31]. GPS consists of 24 evenly spaced satellites placed in circular 12-hour orbits inclined 55° to the equatorial plane. This constellation scheme provides the desired coverage all over the world in addition to being cost-effective. In order to determine the position of a GPS receiver, it must have clear sight (i.e. no blocking mountains or buildings) to at least four satellites. With the mentioned setup, it is guaranteed that a minimum of four satellites are in a good geometric position 24 hours of the day anywhere on the earth, which thus constitutes the foundation for determining a position [32].

The position of a GPS receiver is determined by the resection process, a method for determining an unknown location given known positions, using the measured distances to the satellites. There are no public restrictions regarding receiving the signals emitted from the GPS satellites. Hence, anyone can connect a GPS receiver to an antenna, which can receive signals from the satellites, and thus determine positions in the world [31].

Consumer-grade GPS

Consumer-grade GPS offers inexpensive and manageable technology for collecting positions. Its accuracy depends highly on the manufacturer, but it is generally stated that the measurement accuracy is within $15 - 20$ meters of the actual position [33]. Most modern cars have some consumer-grade GPS incorporated. Positions obtained from that system can be used as a signal when inferring the path of a vehicle on lane-level. However, as these signals correspond to noisy estimated locations, it is not sufficient for the task of identifying traversed lanes. The actual path traversed by the vehicle is preferably derived by positions generated from a real-time kinematic GPS, described below, as they correspond to more accurate measurements of the actual locations.

Real-time kinematic GPS

Real-time kinematics (RTK) is a technique used to enhance the GPS accuracy. It involves a method of carrier-phase differential GPS positioning with centimeter-level position accuracy in real time. Conceptually, it utilizes one or more fixed base stations that send corrections to a moving receiver which in turn are used to improve the position accuracy [34]. Receivers adapted for RTK GPS are costly, which is why it is used only for a limited area of applications. For that reason, it is not readily incorporated into products intended for the general mass [35].

4.1.2 Inertial navigation system

Inertial measurement units (IMUs) are electronic devices that measure the inertial movement of a vehicle [36]. An IMU usually has six degrees of freedom, yielded by three axis accelerometers and three axis gyroscopes. An inertial navigation system (INS) inputs the IMU measurements into navigation equations. The outputted odometric data is then smoothed by an estimation filter before being transmitted to the vehicle [37]. An illustration of the INS is shown in Figure 14.

4.1.3 Forward looking camera

A forward looking camera, placed on a traversing vehicle, is able to detect lane-markings on the road. Cameras applicable for standard vehicles are capable of detecting markers lying inside a 25 meter disc [38]. The detections give information about distances to the closest markings on the left and right

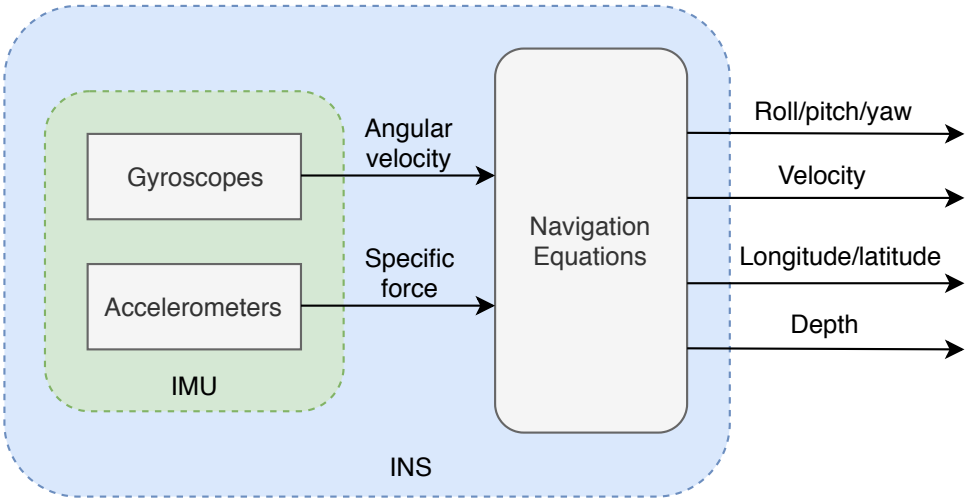


Figure 14: Illustration of the INS.

side of the vehicle and their corresponding type. The marker types that can be recognized by the camera include e.g. solid and dashed.

4.2 VARIABLES

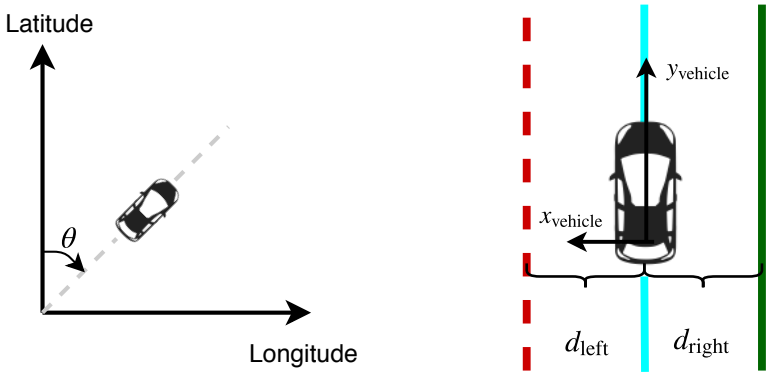
A summary of the available data given by an observation y_k at time t_k can be seen in Table 3. Explanations of these variables follow below.

Table 3: Different type of data from the vehicle sensors categorized by its sources.

Sensor	Variable	Description	Unit
GPS	*latitude*	Latitude	[degree]
	longitude	Longitude	[degree]
	altitude	Altitude	[m]
	satellites	Number of satellites	
INS	*heading*	Heading	[degree]
	speed	Speed	[m/s]
	yaw rate	Yaw rate	[degree/s]
Vision	*left marker type*	Type of left marker	
	right marker type	Type of right marker	
	left confidence	Confidence left marker	
	right confidence	Confidence right marker	
	lane change	Lane change indicator	
	width	Distance between left and right marker	[m]

4.2.1 Latitude and longitude

As the vehicles are equipped with both a SPA and an RTK receiver, two separate versions of the latitude and longitude are collected. SPA, or Volvo Scalable Product Architecture, GPS is the specific consumer grade GPS re-



(a) Measurement of the vehicle heading θ.

(b) Measurement of the distances to the left and right lane markers, d_{left} and d_{right}, as captured by the vision system.

Figure 15: Schematic representation of the variables heading, distance to left lane marker and distance to right lane marker.

ceiver used by the Volvo vehicles in the Drive Me project. The SPA receiver returns the coordinates with a rate of 1 Hz, while the much more accurate RTK GPS has a sampling rate of 40 Hz.

4.2.2 Altitude

As with the latitude and longitude, the altitude, measured as meters above the sea, is returned both by the SPA and RTK receivers, at 1 Hz and 40 Hz respectively.

4.2.3 Number of satellites

The SPA receiver also outputs the number of satellites to which it is connected with. This number depends on the environment. Inside a tunnel, for instance, there is no satellite in-view. If there are few obstacles between the receiver and sky, however, the receiver can connect to virtually all satellites located on the current hemisphere.

4.2.4 Heading

The vehicle heading is measured as the clockwise angle from the latitudinal axis, as shown in Figure 15a. The heading is calculated by the inertial navigation system (INS). The INS uses the GPS data, and therefore, the vehicle heading is reported for both GPS systems.

4.2.5 Speed

The vehicular speed is given in the unit meters per second and is calculated by the INS.

4.2.6 Yaw rate

The yaw rate, corresponding to the angular velocity, i.e. steering angle over time, is given in the unit degrees per second and is calculated by the INS.

4.2.7 Distance to lane markers

The lateral distances from the vehicle to the closest left and right marker, d_{left} and d_{right}, measured by the camera according to Figure 15b, are given in two variables. The distances are stated in meters.

4.2.8 Lane marker types

The vision sensor identifies the types of the left and right lane marker, e.g. solid or dashed, which are stored in two variables. There are four explicit marker types: solid, dashed, botts dots and double lane, and three additional types: invalid, undecided and no marking, describing uncertain detections or non existing markings.

4.2.9 Confidence of vision system

There are two variables holding information about how confident observations of the left and right marker types are, respectively. The confidences take values in $\{0, 1, 2\}$ where 0 corresponds to the lowest confidence and 2 the highest.

4.2.10 Lane change indicator

This variable is assigned values in $\{0, 1, 2\}$. When the vehicle switches to a left lane, the indicator shows 1, and when changing to a right lane it is 2. When no lane changes occur, the lane change indicator is 0.

4.3 DATA ANALYSIS

This section describes the data analysis performed on vehicle sensor variables of special interest. This analysis is based on the training data. The ability of the vision system to identify lane marker types is investigated in Section 4.3.1. The performance of the lane change indicator is analyzed in Section 4.3.2. Further, yaw rates potential to identify lane changes is investigated in Section 4.3.3. Lastly, the correspondence between lane widths seen by the vision system compared to the map is looked into in Section 4.3.4.

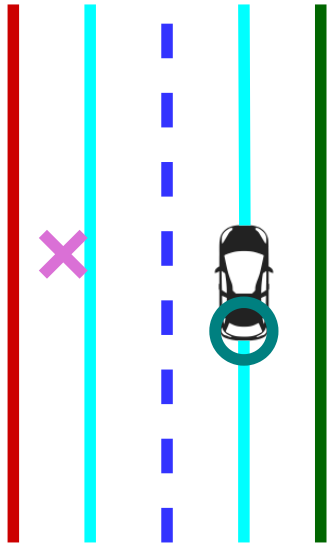


Figure 16: Two-lane northbound road. The vehicle traverses in the right lane, as indicated by the RTK GPS coordinates given as teal-colored circle. The orchid-colored cross is the noisy SPA GPS coordinate.

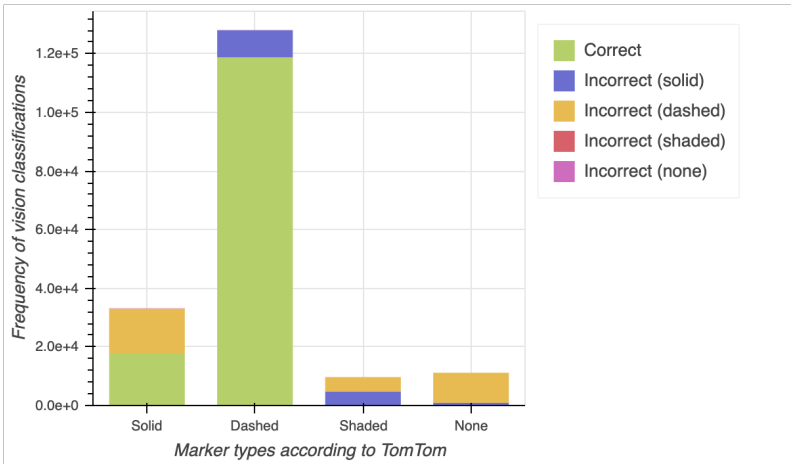
4.3.1 Lane marker type identifications

The vision system that the vehicle is equipped with constitutes the primary qualification to derive in which lane segment the vehicle is when the GPS is erroneous. For example, consider a vehicle traversing a two-lane northbounded road, see Figure 16. As can be seen, the vehicle is traversing the right lane while the noisy GPS places the vehicle in the left lane. Assuming that the vision system performs as expected, it will report that the left lane marker is dashed and the right one solid. This can be used to emphasize that the right lane is the most probable location of the vehicle. When there are more than three adjacent lanes with the same direction, there will be ambiguities between the enclosed lanes since they might have the same left and right marker type.

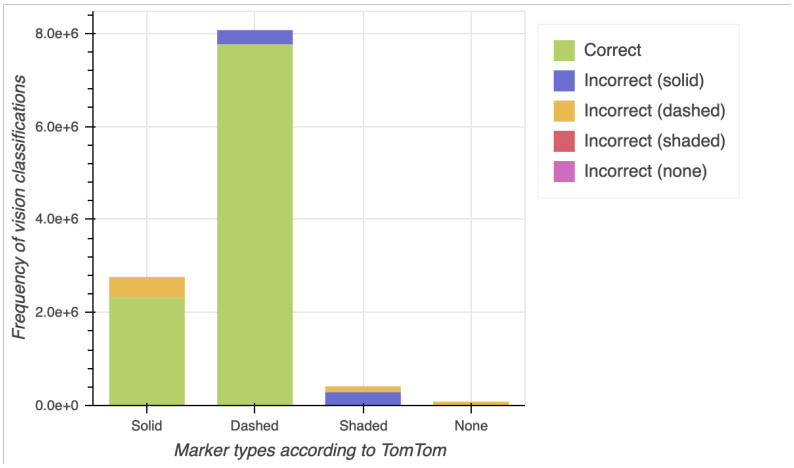
However, if the vision system repeatedly reports incorrect left and right marker types it is problematic to rely heavily on the vision data. Thus, it is legitimate to investigate the ability of the vision sensors to identify lane marker types correctly. Knowing the path a vehicle has followed while collecting data from the sensors, it is possible to check if the vision system might have identified lane marker types which do not correspond to the types presented at corresponding locations in the TomTom map.

The vision system is capable of registering solid and dashed markers, as well as the absence of any markers. According to the map, the lanes are typically bordered by such solid and dashed markers. Also no markings and shaded areas are prevalent. Other marker types also occur in the map. However, since these markers are not represented in the classes of the vision system, and since the markers are relatively few, they have been omitted from the forthcoming analysis.

Since each marker type classification is assigned a confidence value, it is natural to perform one vision classification error analysis per confidence



(a) Confidence level 1, corresponding to medium confidence.



(b) Confidence level 2, corresponding to the highest confidence.

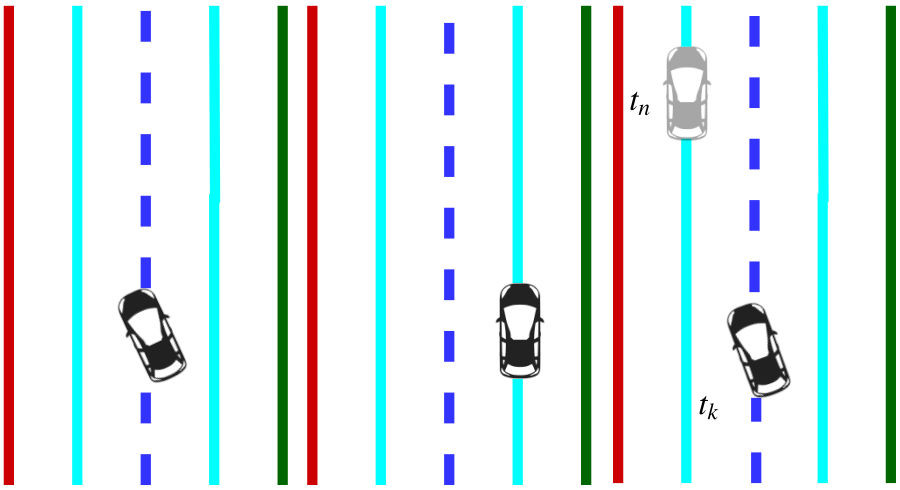
Figure 17: Frequency of marker type classifications by the vision system, sorted by true lane marker types and correctness of said classifications.

level. The vision system does not report any marker type back if the confidence is 0. Therefore, only confidences 1 and 2 are examined.

The accuracy of the classifications is 75% for confidence 1 and 89% for confidence 2. The fraction of correct and incorrect marker types per true marker type is seen in Figure 17. Studying the figure, it is clear that the vision sensor is better at correctly classifying dashed markers than solid ones, for both confidence levels. Solid lane markers are more often mistaken as dashed markers, than the other way around. Shaded areas, which lack representation by the vision system, are reported as either solid or dashed markers. The proportions between these vary over confidence levels. Absences of lane markers are seemingly labeled as dashed. This analysis shows that the vision sensor does not perform ideally, and also that it exhibits over-confidence.

4.3.2 Lane change indicator

The lane change indicator is a variable with big potential for map matching purposes — especially for finding the correct match among parallel lane



(a) A lane change occurs but the indicator fails to report it.

(b) The lane change indicator reports a lane change even though the vehicle is not changing lane.

(c) The vehicle changes lane at time t_k, but the indicator reports the lane change at a different time t_n.

Figure 18: Three cases for which the lane change indicator is incorrect.

segments. A further analysis of the accuracy and robustness of the sensor is therefore of interest. A ground truth lane change parameter is used for evaluation, based on the RTK coordinates and the connectivity of the map, i.e. how lanes in the map are connected to each other. In a similar fashion to the lane marker type analysis, it is then possible to yield information about the accuracy of the lane change classifications.

The lane change indicator can be incorrect in several ways. These are visualized in Figure 18. The cases in Figures 18a and 18b are straightforward. The third case, Figure 18c, where the indicator reports the lane change at a different time step, is more complicated. If $|t_n - t_k|$ is small, the lane change indicator could be considered correct. If the difference is large, it should be considered incorrect.

To take this time difference into consideration, a tolerance is introduced. For Figure 18c, a left or right lane change signal is considered to be correct if the same lane change has occurred within the tolerated time window, i.e. if $|t_n - t_k| < \varepsilon$ for some tolerance level ε. The fraction of correct lane change classifications is seen in Figure 19. A lane change signal output of 0, corresponding to no lane change, is nearly always correct irregardless of the time tolerance. This lane change state is also the most frequent one. Depending on the time tolerance, the accuracy of the left and right output signals vary. Only a small fraction of the lane changes are correctly detected by the vehicle system in the same time step, corresponding to a time tolerance of 0 s. As the tolerance increases, so does the correctness of the indicator signals. After 2 s, the accuracies remain constant.

This analysis shows that very few lane changes are reported immediately as they occur. Instead, it takes the lane change indicator up to 2 seconds to report the change of lanes. A total of 86% of left and right lane changes are

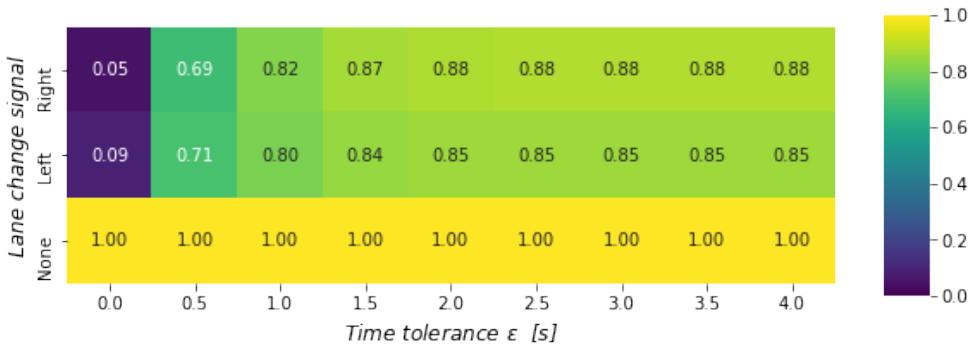


Figure 19: Correctly matched lane change signal frequencies per indicator type and for a range of tolerated time differences between the lane change detection and the true lane change.

reported within this 2 second time window. The remaining 14% are false positives or false negatives as described by Figures 18a and 18b.

4.3.3 Yaw rate during lane change

According to Rabe et al. the steering angle approximately resembles a sine wave over time during a lane change. This wave can be modeled by fitting a parameterized sine function $s(\omega)$ of the yaw rate ω,

$$s(\omega) = a \cdot \sin(f\omega + p) + c \tag{6}$$

using data where lane changes are known to occur [18]. a, f, p and c are parameters meant to be estimated based on data.

Using the lane change indicator, described in Section 4.2.10, it is possible to gather yaw rates maintained during left and right lane changes. Yaw rates corresponding to left lane changes have to switch sign in order to be studied alongside yaw rates originating from right lane changes. It might seem redundant to incorporate an additional metric to identify lane changes, as the lane change indicator is part of the sensor data in every time step. However, if the lane change indicator is solely based on visual data, it might give faulty information when the conditions are not optimal, e.g. when the sight is impaired due to fog. By also studying the yaw rates, it might be possible to identify lane changes which have not been picked up by the visual data sensors.

In order to estimate the parameters in Equation (6), it is necessary to obtain data by extracting yaw rates were lane changes occur. This is not a trivial task, since various driving conditions affect the lane change maneuver. For example, it takes longer time to perform a lane change on wider lanes than on more narrow ones when maintaining the same speed. Furthermore, the yaw rates during a lane change performed in a curve exhibit a rather noisy pattern, implying that it is unlikely to resemble a sine wave. Having mentioned this, the technique to extract yaw rate data was naive in the sense that only time intervals around 9 seconds were considered. It also included manual verification to assure that the lane changes were not performed in a curve.

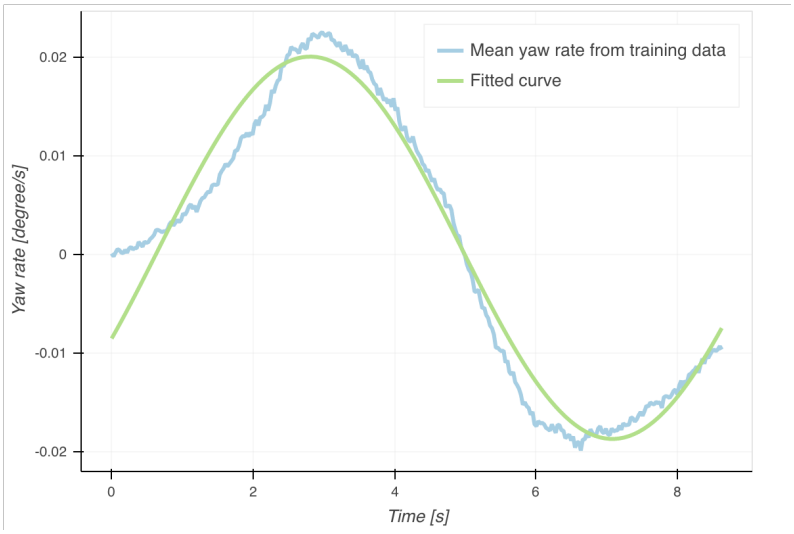


Figure 20: Mean yaw rate from 32 lane changes, shown in blue, has been used to fit a parameterized sine curve, presented in green.

In the training data set, 32 manually verified lane changes were found to have yaw rates resembling a one-period sine wave. The mean of these yaw rates can be seen in blue in Figure 20. Using least-squares optimization to fit this data to Equation (6) yields a sine wave which can be seen in green in Figure 20.

Once Equation (6) has been estimated, it can be used to define a classifier that detects eventual lane changes by solely considering the yaw rates. By checking the mean squared error between $s(\omega)$ and $s(\Omega)$, where Ω is a range of yaw rates originating from a drive which might involve lane changes, one can deduce whether the error is lower than a predefined threshold. If that is the case, the yaw rates Ω are said to follow a sine curve indicating that a lane change likely has occurred.

When using the classifier described above to identify lane changes solely based on yaw rates, it was found to perform worse than when simply looking at the lane change indicator. For the test data set, where the lane change signal reported 404 lane changes, only 65 of them were found by the classifier. The classifier also identified 47 lane changes which the lane change signal did not find. However, upon manual verification it was discovered that the tentative lane changes detected by the classifier were exclusively false detections. That is, the vehicle did not change lane where the classifier predicted it did.

It should be noted that the data considered when testing the classifier might correspond to drive sessions in nice weather and with clear sight, making the lane change indicator highly reliable. Hence, with the available data, it can not be confidently determined if the classifier would identify lane changes not picked up by the vision sensors when weather conditions limit their performance. The classifier thus has to be tested on more varying data in order to establish if it is a good complement to the lane change indicator for the task of identifying lane changes.

4.3.4 Lane width

As shown in Figure 11 in Section 3.4, the lane segments vary in width. It might therefore be interesting for lane-level map matching purposes. The variables are investigated further in order to explore this possibility. In the TomTom map, each lane x_k is associated with a lane width, $width_{x_k}$. The vehicular data of an observation y_k also contains a measurement, $width_{y_k}$, comparable to $width_{x_k}$.

The two variables are plotted against each other in Figure 21. The mean and standard deviation are calculated as $\mu = \text{mean}(width_{x_k} - width_{y_k}) = 0.21$ and $\sigma = \text{std}(width_{x_k} - width_{y_k}) = 0.18$ for the reported confidence level 2. For a normal distribution, 99.7% of the values are contained in the range $\mu \pm 3\sigma$. Hence, the 99.7^{th} percentile can be derived for the estimated width as reported by the camera,

$$width_{y_k}^{est} \approx width_{x_k} - 0.21 \pm 0.54,$$

where the term 0.54 is calculated as $3\sigma \equiv CI_{99.7}$, where CI stands for confidence interval, and can be defined as the camera error.

Based on the standard deviation of the camera error, σ, the camera ought to be capable of accurately differentiating between two lane segments of different widths, if this difference is larger than $2CI_{99.7}$ meters. The standard deviation of the lane widths in Figure 11 is 1.83 meters, which is large compared to the camera error. If this was not the case, it would not be meaningful to penalize lanes with widths differing from the lane widths captured by the vehicle camera. By the results in this investigation, however, it is still of interest to implement such a penalty.

However, it is necessary to take into consideration that the lane width reported by the map is inaccurate for lane merges and splits, as concluded in Section 3.4. A lane width-based penalty should thus depend also on the lane types of the candidates. Such measures can be taken, but it is also legitimate to question the expected benefit of a lane width penalty altogether, given the complexity of deriving it.

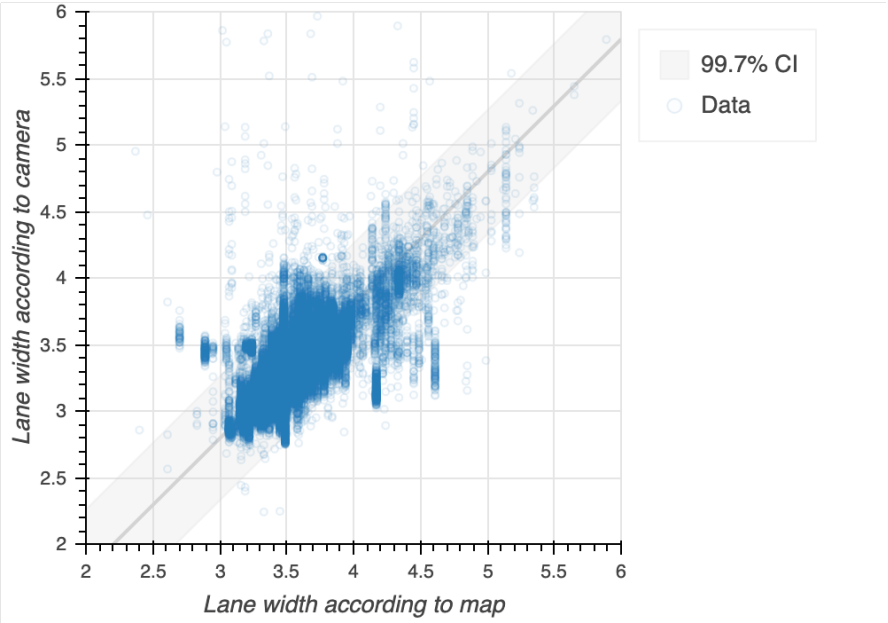


Figure 21: Scatter plot of lane widths, filtered on confidence level 2. Also shown is the 99.7% confidence interval of a linear fit.

5 IMPLEMENTATION

The map matching problem is represented using an HMM. The model is constituted by some essential parameters; states, observations, emission probabilities, transition probabilities and initial probabilities. Proposals for these components are given in Sections 5.1 to 5.5. Lastly, the implementation of the Viterbi algorithm is described in Section 5.6.

5.1 HIDDEN STATES IN HMM

The state space of the HMM is taken as the set of lane segments provided by TomTom, seen as centerlines with IDs in Figure 9. Formal description of a lane segment is given in Definition 2. It is based on the definition of a lane border, which is thus provided in Definition 3.

Definition 2. A *lane segment* is an $M_l + M_r$-point polygon whose M_l, M_r points are given by the left and right lane borders (Definition 3). It is also defined by its lane specifics (Table 1), derived from the corresponding centerline.

Definition 3. A *lane border*, $r = \{p_m | m = 1, \ldots, M\}$, is a polyline spanned by M longitudinal and lateral coordinates p_m, defining the left or right border of a lane segment.

Matching observations to the hidden states implies that the position of the vehicle is contained within the area of the lane section, given by its matched centerline. The lanes can be $1 - 1050$ meters long and there are $7,714$ lane segments. For a typical lane of width 3.7 meters, this area ranges from around 4 m^2 to 3885 m^2. In terms of localization, the actual vehicle location can be either strongly narrowed down, or very vague.

To get more comparable states, the centerlines could be split into multiple parts so that the area of each state polygon is similar in size. According to the time complexity analysis performed in Section 2.3.1, the solution time of the VA increases as N^2. Therefore, this state implementation would take significantly longer time and as such, is not used. However, it is worth noting that the precision of the actual vehicle location would improve, implying there is a trade-off between time efficiency and position accuracy.

5.2 OBSERVATIONS IN HMM

Not all vehicle sensor data variables have been found useful for the purpose of map matching, which is why the set of observation variables can be compressed. For a single vehicle drive, a so called trajectory is created. It is defined in Definition 4. As can be seen, some of the variables in Table 3 have

been omitted. This is because they were found to be unfit for the problem of lane-level map matching, as described in Section 4.3.

Definition 4. A *trajectory* is a sequence of data points, $Y = (y_k | k = 1, ..., K)$, collected by a vehicle during a drive session. The point y_k is defined by the current $latitude_{y_k}$, $longitude_{y_k}$, $heading_{y_k}$, $speed_{y_k}$, $left\ marker\ type_{y_k}$, $right\ marker\ type_{y_k}$, $left\ confidence_{y_k}$, $right\ confidence_{y_k}$ and $lane\ change_{y_k}$ from Table 3.

The trajectory points are sampled at 1 Hz frequency, which is the frequencies of the SPA latitudes and longitudes. This means that a data point is generated every 25 meter for a vehicle driving at 90 km/h.

5.3 EMISSION PROBABILITY

The emission probability $E_k = p(y_k|x_k)$ reflects how likely it is to observe y_k given that the vehicle was driving in candidate lane $x_k \in \mathbf{X} \subseteq \mathbf{S}$ at time t_k. It is intuitive to let the likelihood of making an observation in a lane decrease with increasing distance between the GPS location of the observation and the lane. This is explicitly defined in Section 5.3.1. In addition to this, the probability is more refined by incurring penalty factors based on speed, heading and lane marker type. These are introduced in Section 5.3.2.

5.3.1 GPS observation probability

The probability of observing y_k when being in lane x_k can be modeled according to a normal distribution as

$$P(y_k) = \frac{1}{width_{x_k}} \int_{-0.5 width_{x_k}}^{0.5 width_{x_k}} \frac{1}{\sqrt{2\pi\sigma_{GPS}^2}} e^{\frac{-(l-d)^2}{2\sigma_{GPS}^2}} \, dl, \tag{7}$$

where $width_{x_k}$ is width of the lane, d the perpendicular distance between the GPS coordinate of the observation and the centerline of the lane and σ_{GPS} the estimated standard deviation of the GPS error. The lane width is accounted in the hope of contributing to finer distinction between closely located lanes having different widths [15].

5.3.2 Penalties

By the noisy nature of the location measurements, it is not sufficient to solely consider the GPS location in order to infer which lanes a vehicle has traversed. Additional vehicle sensor and map data, such as speed, heading and lane marker types, can be used to deduce penalty factors. These factors range between 0 and 1, with the later defined lane marker type penalty being the exception. It attains values between 0 and 1 if lane markers do not match, and values between 1 and 2 if they do. As such, it is not a true penalty.

These factors can be incorporated in the emission probabilities, so that the latter have a higher dependency on the unique characteristics of each

lane. If a penalty is lower than 1, the matching probability of a specific lane is decreased. For the special case of lane marker type penalty, a value greater than 1 enhances the lane likelihood. This should make it easier for the algorithm to distinguish between the candidate lanes. Descriptions of the different penalties implemented are given below. They can also be seen in Figure 22.

Speed

Hypothetically, drivers tend to not exceed speed limits. Although maybe not a universal truth, the data used in this work reinforces this assumption. Thus, a factor penalizing speeding can be introduced. Parallel roads that are located close to each other and have different speed limits can then be differentiated on the basis of speed differences. The implication of the penalty is thus to reduce the number of possible lanes, as well as to adjust their matching fitness.

The speed penalty is defined as

$$S(x_k, y_k) = \frac{speed\ limit_{x_k}}{\max\left(0, speed_{y_k} - speed\ limit_{x_k}\right) + speed\ limit_{x_k}}. \tag{8}$$

Here, $speed_{y_k}$ is the vehicle speed and $speed\ limit_{x_k}$ the speed limit on the lane [15]. A plot showing the speed penalty for different road speed limits can be seen in Figure 22a.

Heading

A vehicle will generally maintain about the same heading as the road it traverses. Thus, in cases where there are clusters of closely located roads, e.g. urban areas or overpasses, candidate road lanes could be filtered by considering the vehicle heading compared to the road headings. The heading differs slightly when changing lanes, but generally not above 20°. Thus, no penalty is applied when the heading difference is smaller than 20°. When the headings differ more than 90°, the corresponding lane is assigned a zero probability.

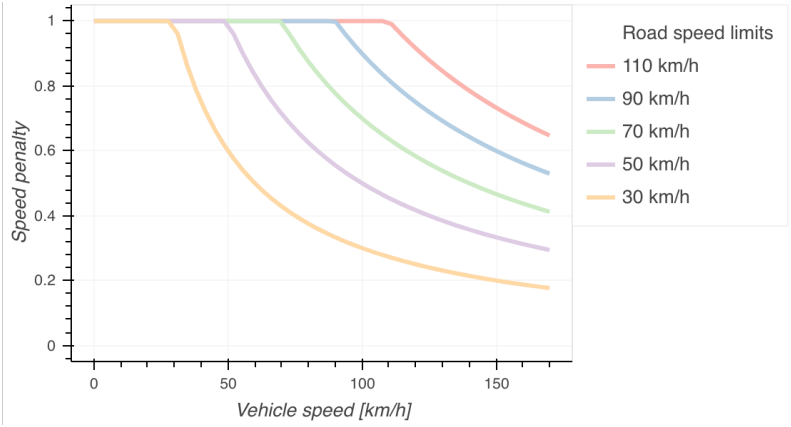
The heading penalty is thus defined as follows

$$H(x_k, y_k) = \begin{cases} 0 & \text{if } 90° \leqslant \Delta_{heading}(x_k, y_k) \\ \frac{heading_{x_k}}{heading_{x_k} + \Delta_{heading}(x_k, y_k)} & \text{if } 20° \leqslant \Delta_{heading}(x_k, y_k) < 90° \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

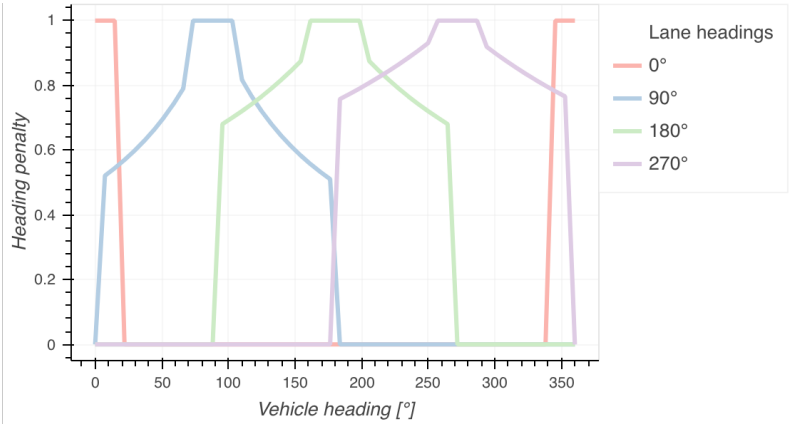
where $heading_{y_k}$ is the vehicle heading, $heading_{x_k}$ the heading of the lane and $\Delta_{heading}(x_k, y_k)$ the difference between the headings. A plot showing the heading penalty for different lane headings can be seen in Figure 22b.

Lane marker type

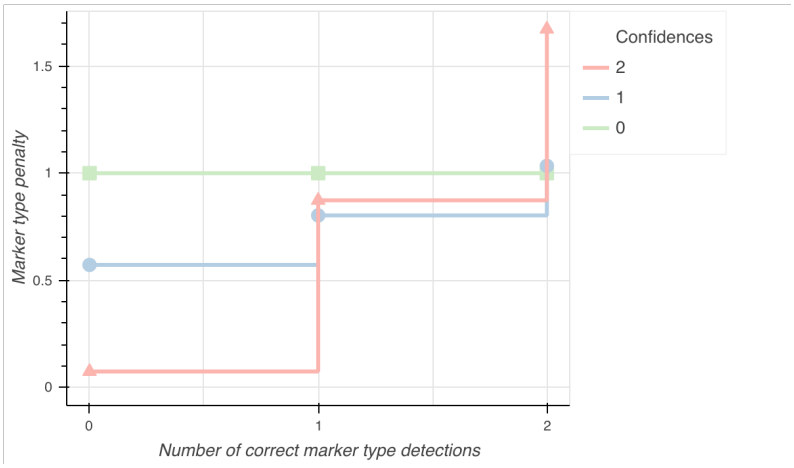
The forward looking camera in the vehicle, described in Section 4.1.3, detects lane-markings on the road. For the task of lane-level map matching, this is one of the sensors that are capable of describing which lane the vehicle traverses. For example, marker types for parallel lanes on two-lane



(a) Speed penalty inclined for a vehicle maintaining a speed between $0 - 170$ km/h on roads with speed limits 30, 50, 70, 90 and 110 km/h.

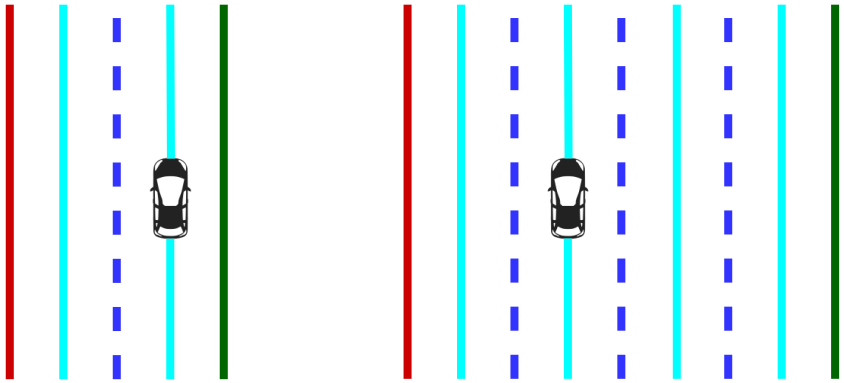


(b) Heading penalty for a vehicle traversing with heading ranging between 0° and 360° on lanes with headings 0°, 90°, 180° and 270°.



(c) Marker type penalty, with scale parameter $c = 1$, for a vehicle whose vision system detects 0 to 2 lane marker types correctly (with confidences 0, 1, 2) according to the map.

Figure 22: Plot of penalties incorporated in the emission probability.



(a) The vehicle, traversing in the right lane, will see a dashed left marker and solid right marker. This excludes the possibility for the vehicle to be in the left lane.

(b) When there are four lanes heading in the same direction, there is an ambiguity for the middle lanes which have the same left and right marker type. Therefore, only the leftmost and rightmost lanes can be matched unambiguously.

Figure 23: The lane marker types identified by the forward looking camera can be used to narrow down the candidate lanes that the vehicle might be traversing.

roads are not the same. This information combined with information from the camera can be used to deduce the most likely lane. In Figure 23a the vehicle's forward looking camera will report a dashed marker to the left and solid to the right. As this correspond to the marker types of the right lane, one can conclude that the vehicle probably traverses that lane.

Still, ambiguities arise on roads containing more than three parallel lanes. As can be seen in Figure 23b, the vehicle is in the second to last left lane and the camera reports dashed markers on both sides. This only limits the two outer lanes which have either left or right solid markers respectively. Thus, according to the vision data, the vehicle could be in either of the two middle lanes.

A mathematical expression has to be derived in order to incorporate a penalty considering the marker types registered by the camera on the vehicle compared against types given by the map. As there are two identified marker types at each observation, the left and right one, two variables are sufficient for maintaining the probabilities of detecting the same marker types as given by the candidate lane marker types. They are defined as

$$\begin{aligned} &p_{\textit{left equal}}(x_k, y_k) = \\ &= P\left(\textit{left marker type}_{x_k} = \textit{left marker type}_{y_k} \,\middle|\, \textit{left marker type}_{y_k}\right) \end{aligned} \tag{10}$$

and

$$\begin{aligned} &p_{\textit{right equal}}(x_k, y_k) = \\ &= P\left(\textit{right marker type}_{x_k} = \textit{right marker type}_{y_k} \,\middle|\, \textit{right marker type}_{y_k}\right), \end{aligned} \tag{11}$$

where $\textit{left marker type}_{x_k}$ and $\textit{right marker type}_{x_k}$ are the left and right marker types on the lane and $\textit{left marker type}_{y_k}$ and $\textit{right marker type}_{y_k}$ the left and

right types registered by the vehicle. The probabilities in Equations (10) and (11) are estimated for the different marker types using the training data set.

Using Equations (10) and (11) and incorporating the confidences described in Section 4.2.9 yields

$$\begin{aligned} &\textit{left penalty}(x_k, y_k, c) = \\ &= (1 - p_{\textit{left equal}}(x_k, y_k)) \times \left(1 - c \times \frac{\textit{left confidence}_{y_k}}{\textit{max confidence}}\right) + \\ &\quad p_{\textit{left equal}}(x_k, y_k) \times \left(1 + c \times \frac{\textit{left confidence}_{y_k}}{\textit{max confidence}}\right) \end{aligned} \tag{12}$$

and

$$\begin{aligned} &\textit{right penalty}(x_k, y_k, c) = \\ &= (1 - p_{\textit{right equal}}(x_k, y_k)) \times \left(1 - c \times \frac{\textit{right confidence}_{y_k}}{\textit{max confidence}}\right) + \\ &\quad p_{\textit{right equal}}(x_k, y_k) \times \left(1 + c \times \frac{\textit{right confidence}_{y_k}}{\textit{max confidence}}\right), \end{aligned} \tag{13}$$

where c is a scale parameter. $\textit{left confidence}_{y_k}$ and $\textit{right confidence}_{y_k}$ are the confidence levels reported by the vehicle and *max confidence* the maximum confidence that can be registered by the vision system. These penalties take values in the interval $(1 - c, 1 + c)$, but are truncated at 0. When the camera confidently reports that the left or right lane marker type is not the same as the types for some lane segment, the penalties will maintain a low value. Correspondingly, when it confidently reports that the marker types are equal, the penalty values are higher. Remember that a low penalty means that a lane is less likely, while high penalty means more likely.

The total lane marker type penalty is then obtained as the average of Equations (12) and (13),

$$M(x_k, y_k, c) = \frac{\textit{left penalty}(x_k, y_k, c) + \textit{right penalty}(x_k, y_k, c)}{2}. \tag{14}$$

After some tuning, the scale parameter is set to $c = 1$. A plot showing the marker type penalty for different confidence levels can be seen in Figure 22c.

5.3.3 Final emission probability

Having the GPS observation probability defined according to Equation (7) and the penalties given in Equations (8), (9) and (14), the emission probability is given by the product of all these factors. Thus, the probability of observing y_k on lane x_k is calculated as

$$E_k = P(y_k) \times S(x_k, y_k) \times H(x_k, y_k) \times M(x_k, y_k, c). \tag{15}$$

5.4 INITIAL PROBABILITY

The initial probability Π reflects which lane the vehicle is likely to have started on. For the sake of simplicity, it is common to let the initial probability distribution be uniform. However, since all observations from the drive

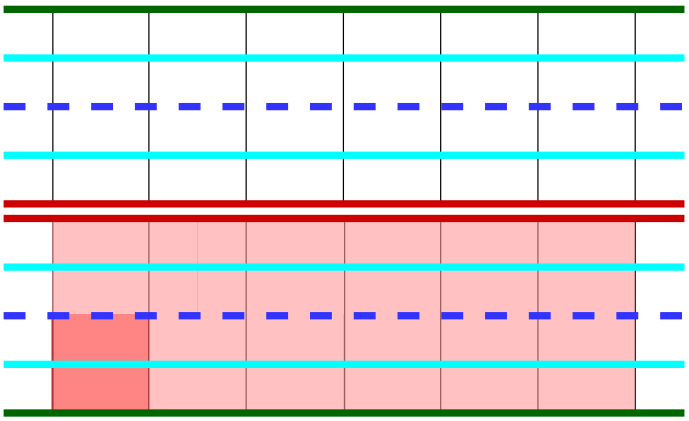


Figure 24: The figure depicts a road with two lanes in both east- and west-bound direction. For a vehicle traversing the lower lane segment to the left (dark red), it can transition to lanes which it is directly or indirectly connected to (bright red). The thin black lines indicate different lane segments.

are known initially, it is justified to let the initial distribution consider this information. This allows for a more sophisticated guess of which lane the vehicle likely started the drive at. Re-using the emission probability introduced in Section 5.3, the initial probability is defined as

$$\Pi_i = E_{0,i}, \quad i = 1, \ldots, N. \tag{16}$$

5.5 TRANSITION PROBABILITY

The transition probability $T_{i,j} = P(s_j|s_i)$ describes the likelihood of moving to lane $s_j \in \mathbf{S}$ when being in lane $s_i \in \mathbf{S}$. Two ways to derive transition probabilities will be introduced. One which uses the connectivity of the road network provided by the TomTom map, see Section 5.5.1, and one data-driven approach which does not require knowledge about lane connections, see Section 5.5.2. These are later evaluated on a small bounding box of the map, and the best-performing transition probability model is used for final evaluation.

5.5.1 Connectivity-based probabilities

The transition probabilities can be modeled by considering the connectivity governed by the TomTom map. That way it will be possible to move from a lane to lanes in its local neighborhood, as visualized in Figure 24.

In order to get a reasonable distribution of the probabilities, the connectivity-level should be considered. The connectivity-level concerns how closely connected the lanes are. For a specific lane, all lanes located laterally next to it are denoted $\mathbf{S}^0$, which are represented in purple in Figure 25. Lanes located next to the lanes following directly after lanes in $\mathbf{S}^0$ are denoted $\mathbf{S}^1$, colored green in Figure 25. Continuing in the same manner, lanes located at depth d from a lane are denoted $\mathbf{S}^d$, yellow in Figure 25,

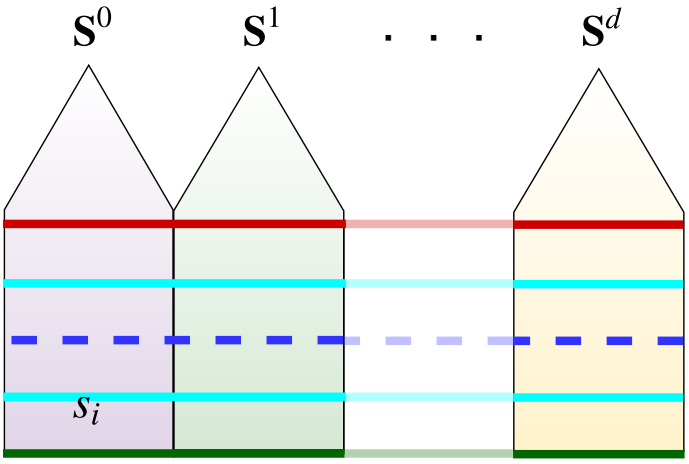


Figure 25: The lanes that lane $s_i \in \mathbf{S}$ are directly or indirectly connected to can be grouped into sets $\mathbf{S}^0, \mathbf{S}^1, \ldots, \mathbf{S}^d$, where $0, 1, \ldots, d$ reflect on which depth the lanes are connected.

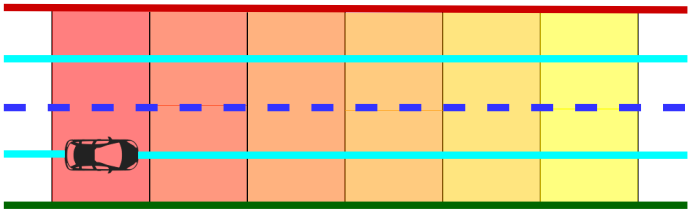


Figure 26: According to the transitioning rules in Equation (17), the vehicle is most likely to remain within the two first lane groups until the next time step, and least likely to transition to the lanes furthest away.

where depth indicates the number of intermediate lanes between the lane and lanes in $\mathbf{S}^d$.

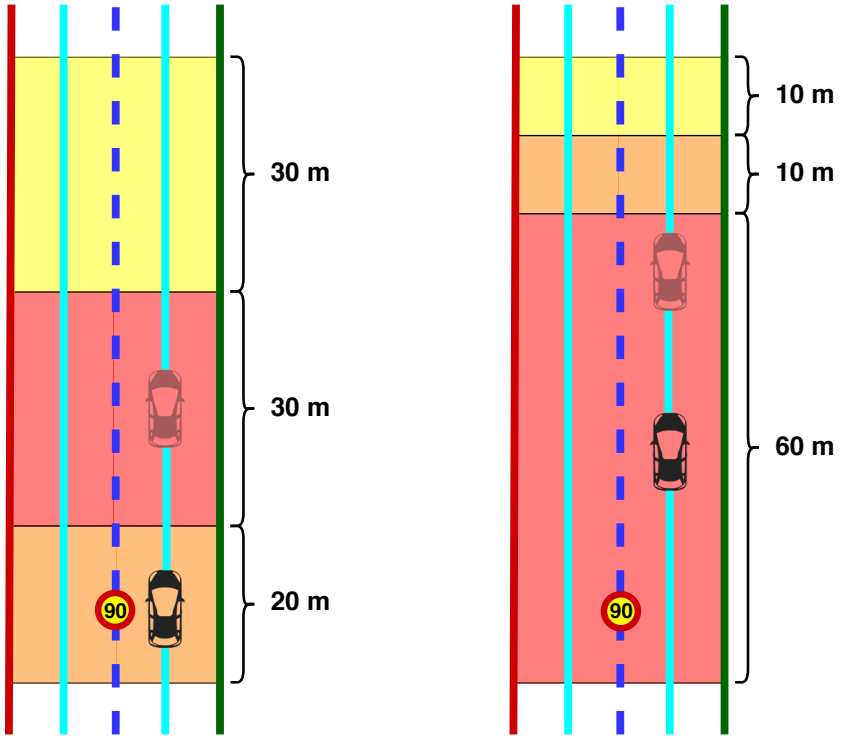
Constant probabilities

To accomplish simple and viable connectivity-based transitions, it is possible to assign probabilities that are constant for each neighborhood depth. That is, there is a constant probability p_0 of moving from a lane in $\mathbf{S}^0$ to other lanes in $\mathbf{S}^0$, and another constant probability p_1 of moving from the lane in $\mathbf{S}^0$ to each of the lanes in $\mathbf{S}^1$, and so on. Thus, a lane can transition to lanes in $\mathbf{S}^0, \ldots, \mathbf{S}^d$ with probabilities $p_0 \ldots, p_d$, respectively. By the assumption that it is decreasingly likely to transition to lanes at increasing depth, the probabilities are assigned such that $p_0 > p_1 > \ldots > p_d$. By this reasoning, the probabilities of transitioning from $s_i \in \mathbf{S}$ to a lane $s_j \in \mathbf{S}^d$ can be modeled as

$$T_{i,j} = \frac{D - d}{D}, \tag{17}$$

where D is the maximum depth considered.

Consider the case shown in Figure 26, where the vehicle is traversing the lane in the lower left corner. The transition probabilities are calculated according to Equation (17) using maximum depth 5. The colors of the lanes in Figure 26 reflect the likelihood of transitioning to them. Red indicates high transition probabilities while yellow reflects lower probabilities.



(a) Given that the vehicle traverses the lower 20 meter lane maintaining a speed around 90 km/h, it is most likely to transition into the 30 meter lane until the next time step.

(b) When the vehicle is in a lane stretching above 50 meters having speed limit 90 km/h, it is most likely that it will remain in the same lane at the next time step.

Figure 27: Transition probabilities calculated according to Equations (18) and (19) for two different cases, where red reflects higher probabilities and yellow lower.

Refined probabilities

The probabilities proposed in Section 5.5.1 are not very refined. For example, it is possible to incorporate information concerning the speed limit and lane length to deduce more accurate probabilities.

Consider a vehicle transmitting observations every second. It traverses lane $s_i \in \mathbf{S}$ at time t_k, which has speed limit 90 km/h and is 10 meters long. Given that the vehicle maintains about the same speed as indicated by the speed limit on the lane, it has moved around 25 meters until the next time step $k+1$. Since the length of the lane is 10 meters, the vehicle has likely passed the lane it traversed at time step k.

By this reasoning, the probabilities of transitioning from s_i to a lane $s_j \in \mathbf{S}^0$ can be modeled as

$$T_{i,j} = \frac{1}{|\mathbf{S}^0|} \times \frac{1}{|\mathit{length}_{s_i}/2 - \Delta l| + 1} \tag{18}$$

where length_{s_i} is the length of lane s_i and $\Delta l = \mathit{speed\ limit}_{s_i} \times (t_{k+1} - t_k)$ the distance travelled between times t_k and t_{k+1} given the speed limit at lane s_i. Using Equation (18), transitions from s_i to $s_j \in \mathbf{S}^d$ can be derived as

$$T_{i,j} = \frac{1}{|\mathbf{S}^d|} \times \frac{1}{|d_{s_i,s_j} - \Delta l| + 1} \tag{19}$$

where d_{s_i,s_j} is the distance between the longitudinal middle of lanes s_i and s_j. Consider the case shown in Figure 27a where the vehicle is traversing a lane which is 20 meters long having speed limit 90 km/h. Calculating transitions according to Equations (18) and (19) reveal that the vehicle is most likely to be in a lane following directly after the former one at the next time step.

When the vehicle is in a lane which is 60 meters long where the speed limit is 90 km/h, it is more likely that it will remain within the same lane until the next time step, as visualized in Figure 27b.

It should be stressed that the vehicle's location within a lane is not taken into consideration when deriving these transition probabilities. By obvious means, a vehicle traversing practically at the end of a very long-stretched lane is certain to have left that lane at the next time step. For the opposite case, when the vehicle is at the beginning of a long lane, it is more likely that it is still in the same lane at the next time step. However, since the consumer-grade GPS produces noisy location measurements, it is not possible to accurately deduce where in the lane the vehicle is. By this, there is no sophisticated way to incorporate such information when calculating transition probabilities. Therefore, the vehicle is always assumed to be in the middle of the lane (length-wise), which is reflected in Equation (18) by the term $\mathit{length}_{s_i}/2$ in the denominator.

5.5.2 Extrapolation-based probabilities

Another method for estimating the transition probabilities of the model is the so-called extrapolation based transition method. For a given position in a state x_k at time step k, it is possible to predict the state at time step $k+1$ by means of extrapolation, as shown in Figure 28. The estimated distance traveled between the two time steps, Δl, is estimated as the product of the sampling frequency, $\Delta t = t_{k+1} - t_k$, and the vehicle speed s. The direction traveled is given by the vehicular heading. The extrapolated point is calculated as

$$x_{ext} = x + \Delta t \times v \times \cos h, \tag{20}$$

$$y_{ext} = y + \Delta t \times v \times \sin h, \tag{21}$$

with (x, y) being the vehicular position at time t_k.

The transition probability $T_{i,j} = P(x_{k+1} = s_j | x_k = s_i)$ is calculated as the likelihood of extrapolating any point in s_i to s_j. For fixed speed v and heading h, the probability is described by

$$\iint\limits_{(x,y)\in \mathit{poly}(s_i)} \mathbb{1}_{(x_{ext},y_{ext})\in \mathit{poly}(s_j)}\, dx\, dy,$$

where $\mathbb{1}_{(x_{ext},y_{ext})\in \mathit{poly}(s_j)}$ is the indicator function

$$\mathbb{1}_{(x_{ext},y_{ext})\in \mathit{poly}(s_j)} = \begin{cases} 1 & \text{if } (x_{ext}, y_{ext}) \in \mathit{poly}(s_j), \\ 0 & \text{otherwise.} \end{cases}$$

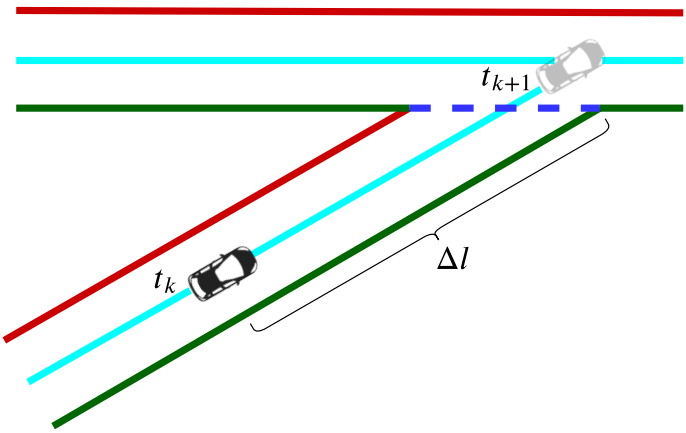


Figure 28: Extrapolation of vehicle position at time t_k to time t_{k+1}.

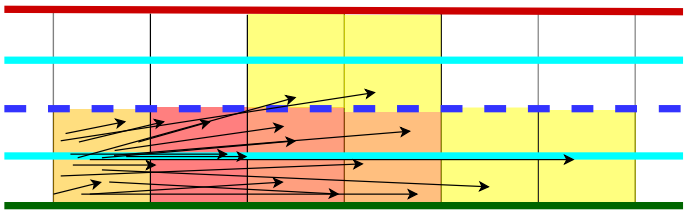


Figure 29: Visualization of the extrapolation-based probabilities. Arrows represent the extrapolations from one lane into surrounding lanes.

v and h could be modeled as the current speed limit and road lane heading. However, this approach does not capture e.g. a vehicle driving slower than the limit, or a vehicle in the process of changing lanes, whose heading would differ from the one of the road. Instead, the parameters are random variables generated by some underlying probability distributions, $f_v(v)$ and $f_h(h)$:

$$T_{i,j} = \iint_{poly(s_i)} \int_h \int_v \mathbb{1}_{(x+v\Delta t \cos h, y+v\Delta t \sin h) \in poly(s_j)} f_v(v) f_h(h) dv dh dx dy. \quad (22)$$

Calculating transition probabilities according to Equation (22) for a specific lane could yield transitions as visualized in Figure 29.

Probability distributions

Different probability distributions $f_v(v)$ and $f_h(h)$ are tried. A simple pdf of the speed is a normal distribution with estimated parameters $\mu_s = \textit{speed limit}_{s_i} - 0.93$ and $\sigma_s = 2.20$. Alternatively, the speed can be modeled as a skew-normal distribution [39, 40],

$$f_v(v) = \frac{2}{\omega} \phi\left(\frac{v-\xi}{\omega}\right) \Phi\left(\alpha \frac{v-\xi}{\omega}\right), \quad (23)$$

with ϕ and Φ being the probability density function (pdf) and cumulative distribution function (cdf) of the standard normal distribution. α is a skewness parameter. ξ and ω are location and scale parameters. These parame-

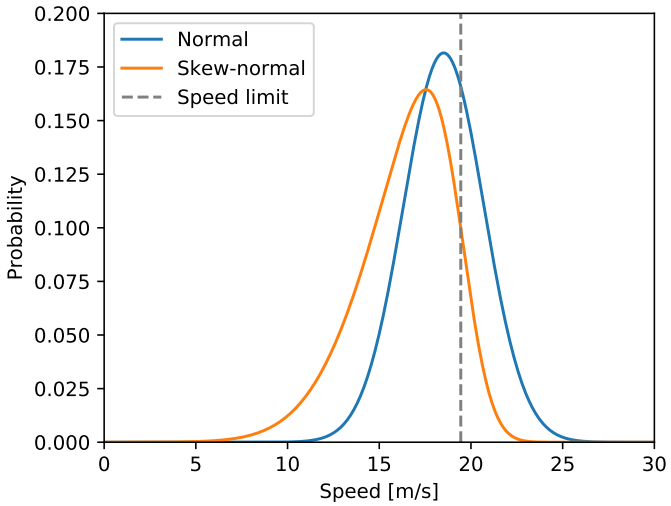


Figure 30: The normal and skew-normal speed distribution with the speed limit set to 19.44 m/s.

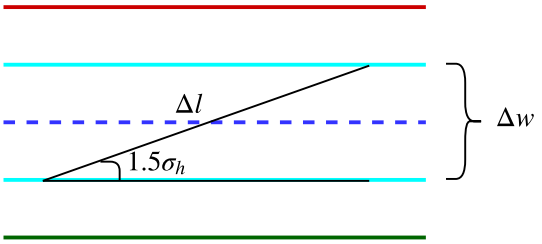


Figure 31: Calculation of the heading deviation, σ_h.

ters are decided by evaluating the VA performance for different values, on a small data set. Based on this analysis, they are set to

$$\alpha = -3,$$
$$\xi = \textit{speed limit}_{s_i},$$
$$\omega = 4.$$

The shapes of the normal and skew-normal distributions can be seen in Figure 30.

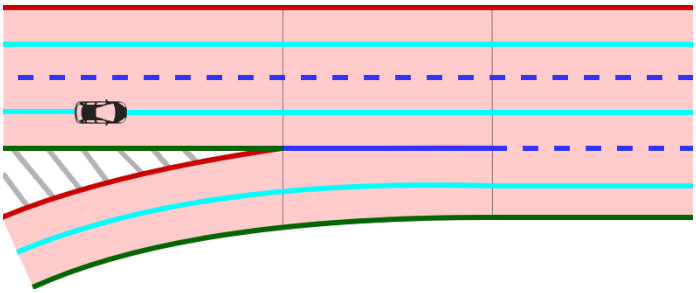
The distribution of the heading is assumed to be normal. μ_h is set to the heading of the closest sub-centerline of s_i. From a primitive data analysis of lane changes, it is approximated that it takes 10 seconds to complete a lane change maneuver. The standard deviation σ_h is chosen such that the probability that the vehicle moves at most one tenth lane-widths sideways is 0.8660, corresponding to 1.5σ. In Figure 31, $\Delta l = 10 \times v\Delta t$. The lateral translation $\Delta w = \textit{lane width}$. $1.5\sigma_h$ is the heading deviation, calculated as

$$\sigma_h = \frac{2}{3} \arcsin \frac{\textit{lane width}}{10v\Delta t}.$$

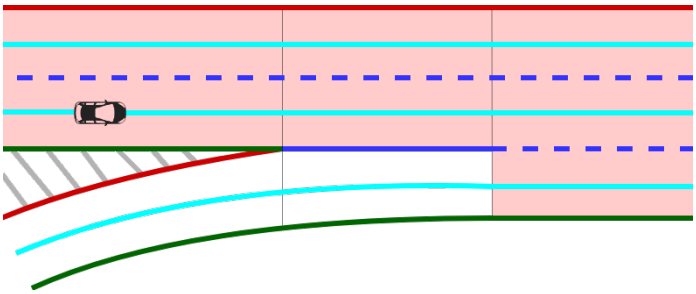
Thus, the heading is distributed according to

$$f_h(h) = \phi\left(\frac{h - \textit{sub-centerline heading}}{\frac{2}{3}\arcsin\frac{\textit{lane width}}{10v\Delta t}}\right),$$

where, once again, ϕ is the density of the standard normal distribution.



(a) Candidate space before adding traffic law constraints.



(b) Candidate space after adding traffic law constraints.

Figure 32: Three lane groups, for which the first and second lane segments are being separated by a shaded area marking, a solid line and a dashed line, respectively. The vehicle is driving in the second lane segment of the first lane group, and the red areas represent possible transitions for that segment.

5.5.3 Filtering illegal transitions

Transitioning from one lane to another may in some cases be forbidden by law. Assuming that the drivers of the vehicles follow the traffic laws, adding these as constraints to the transitions ought to lead to a more accurate Viterbi solution. It also has the benefit of truncating the candidate space, which decreases the computation time.

The two known cases represented in the data set are when two lane segments in the same lane group are separated by a solid line or a shaded area marking. These types of lane borders are forbidden to cross, and thus, there is no possible way of reaching a lane on the opposite side of such a border.

An example road is shown in Figure 32. Without the constraint, the candidate space (i.e. lanes with transition probability > 0) may look like Figure 32a. After adding the constraint, it instead looks like Figure 32b.

5.5.4 Accounting for lane changes

The forward looking camera is able to detect when the vehicle switches lane. Among the sensor data, this is reflected by the variable $\mathit{lane\ change}_{y_k}$ taking values $0, 1$ or 2, describing no lane change, left and right lane change, respectively. This provides useful information which can be incorporated into the transitions probabilities to favour certain lane transitions. For example, if the vehicle sensor data reports a left lane change when being in

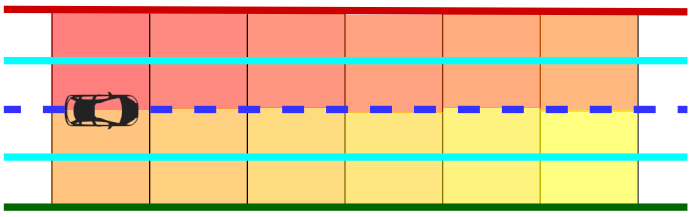


Figure 33: If the sensor data from the vehicle report a left lane change, the vehicle is likely to transition to a lane left of the current one. Red reflects higher transition probabilities and yellow lower.

lane s_i, transitions to lanes left of s_i should be enhanced as visualized in Figure 33.

By obvious means, it is not desired to update transitions probabilities to all lanes straight forward, left or right of s_i since that would involve considering all lanes in the road network. Using the connectivity-level d introduced in Section 5.5.1, it is possible to describe how many succeeding lanes the transition probabilities should be updated for. In Figure 33, six transitions are updated which indicates that the considered depth is 5.

Having transition probabilities $T_{i,j} = P(x_{k+1} = s_j | x_k = s_i)$, these are updated according to the lane change information as follows.

$$T_{i,j} = 0.5 + T_{i,j} \quad \forall s_j \in \{s_j : T_{i,j} > 0\}, \tag{24}$$

and s_j is to the left, right or straight in front of s_i when a left, right or no lane change is detected by the sensor. They are then normalized according to $\sum_{s_j \in S} T_{i,j} = 1$.

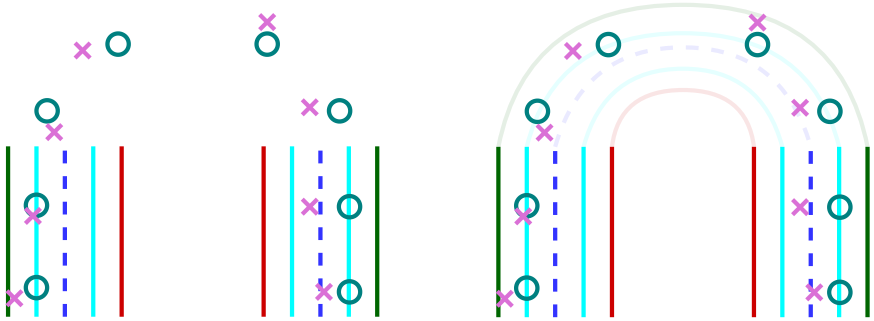
5.6 IMPLEMENTATION OF THE VITERBI ALGORITHM

The Hidden Markov Model is decoded using the Viterbi algorithm. The algorithm has had some alterations made to the offline Viterbi algorithm, so that it can handle data points lying outside the known road network. An in-detail description of the implementation is found in Section 5.6.1. An online version of the algorithm is also implemented, as described in Section 5.6.2

5.6.1 Modification of the Viterbi algorithm

The TomTom map is not complete in the sense that it contains holes. More specifically, there are roads in the network that appear to be dead ends when they are actually not. An example can be seen in Figure 34a, where half of the coordinates are positioned outside the road network. Having this visual overview, it is apparent that the vehicle has likely followed a road similar to the one outlined in Figure 34b.

The map does not provide any information about these missing roads, which is why an observation might not have any candidate lanes since there are no lanes present in the disc around the observed location. If an observation sequence contains observations which do not have any candidate states,



(a) The vehicle appears to be following a road which is not represented in the road network.

(b) The transparent road represent the imagined road traversed by the vehicle.

Figure 34: RTK (teal-colored) and SPA (orchid-colored) coordinates positioned outside road network.

there are no valid state transitions at the corresponding time steps. This implies that the algorithm is unable to find an optimal path, and as such, no path is outputted.

It is certainly desirable to have a stable algorithm which is capable of handling known cases such as the one mentioned previously, where it is at risk of terminating unexpectedly. In order to achieve this, it is convenient to let the Viterbi algorithm match observations not only to lane IDs, but also to an indecisive, or placeholder, variable NaN. For the case visualized in Figure 34a, it would be desirable to match the four middle observations to NaN. The basic idea is that, whenever there are no valid transitions at time t_k, observation y_k is matched to NaN and added to the path back-traced up to time t_{k-1}. The heretofore derived path is concatenated with the result of applying Viterbi recursively to the observation sequence $[y_{k+1}, ..., y_K]$, where t_K is the last time step. If further dead ends would be encountered when processing the sequence $[y_{k+1}, y_{k_K}]$, NaN is again appended to the back-traced path and a recursive call is made to the remainder of the sequence, and so forth. A pseudocode for this modified Viterbi Algorithm is provided in Algorithm 3. Rows $11-13$ contain the added piece of code which make up the modification.

5.6.2 Online Viterbi algorithm

The choice of online algorithm approach depends on the intended application and its purpose, which is why no approach can be considered to be generally preferred. By its implementation simplicity, the VSW approach (described in Section 2.3.2), relying on convergence points, is used to achieve an online Viterbi algorithm. Since these online solutions are guaranteed to be optimal and the same as the offline solutions, the online VA performance is not commented on further.

Algorithm 3 Modified Viterbi algorithm. $\mathbf{y}$: $K \times 1$ is the sequence of observations, $\mathbf{S}$: $N \times 1$ the state space, T : $N \times N$ the transition matrix and E : $K \times N$ the emission matrix.

```
1: function VITERBI(y, S, T, E)
2:     for n = 1, ..., N do
3:         probtable[n, 1] = E_{1,n}
4:         pointer[n, 1] = 0
5:     end for
6:     for k = 2, ..., K do
7:         for n = 1, ..., N do
8:             probtable[n, k] = max_i [probtable[i, k − 1] × T_{i,n} × E_{k,n}]
9:             pointer[n, k] = arg max_i [probtable[i, k − 1] × T_{i,n} × E_{k,n}]
10:        end for
11:        if max probtable[: , k] = 0 then
12:            return BACKTRACE(probtable, pointer)+NaN+VITERBI(y[k: ], S, T, E)
13:        end if
14:    end for
15:    return BACKTRACE(probtable, pointer)
16: end function
17:
18: function BACKTRACE(probtable, pointer)
19:    beststateindex[K] = arg max_i probtable[i, K]
20:    bestpath[K] = s_{beststateindex[K]}
21:    for k = K, ..., 2 do
22:        beststateindex[k − 1] = pointer[beststateindex[k], k]
23:        bestpath[k − 1] = s_{beststateindex[k−1]}
24:    end for
25:    return bestpath
26: end function
```

6 EVALUATION METHOD

The performance of the algorithm is evaluated by metrics which are fitted to describe the fitness of a lane-level map matcher. It is a prerequisite to have the ground truth when defining performance metrics. The process of retrieving ground truth is described in Section 6.1. Before applying evaluation metrics to the result, it is necessary to retrieve information regarding the classifications of the algorithm. Matches are classified according to the methodology presented in Section 6.2. Using this information, the performance metrics are finally introduced in Section 6.3,

6.1 GROUND TRUTH

In terms of map matching, ground truth refers to the true path traversed by the vehicle. For this work, the ground truth is not actually known, but estimated from the RTK. The resulting performance of the algorithm is thus limited by the quality of these RTK coordinates. This is because various parameters in the implementation are tuned in light of results from performance metrics, which are dependent of the ground truth. By this, it is crucial to use ground truth data that is likely to exhibit the actual path the vehicle has followed.

The sensor data retrieved by vehicles contains position measurements obtained from an RTK GPS, described in Section 4.1.1. Since these position measurements provide centimeter-level accuracy, they are suited to compose the data which derives the ground truth. It is noteworthy that the algorithm does not use information from the RTK GPS since vehicles usually are not equipped with them. Instead, it uses location measurements from a SPA GPS which most vehicles are equipped with. Considering the accuracy of the RTK coordinates, the distance between a coordinate and the middle line of a lane is key. However, the distance is not conclusive when gathering ground truth since the map is two-dimensional, meaning that e.g. overpasses are simply laid upon possible underlying lanes. Looking at Figure 35, it is obvious by visual inspection that the vehicle followed the rightmost lane traversing north. But if the coordinates were simply matched to the closest lane middle line, coordinates at times t_3 and t_4 would potentially be matched to the lower and upper horizontal lanes (situated on an overpass), respectively.

To prevent such faulty matches when gathering ground truth, the lane headings are also considered. By comparing the heading registered by the vehicle against the lane heading, it is possible to deduce that the two middle coordinates in Figure 35 probably should not be matched to the horizontal

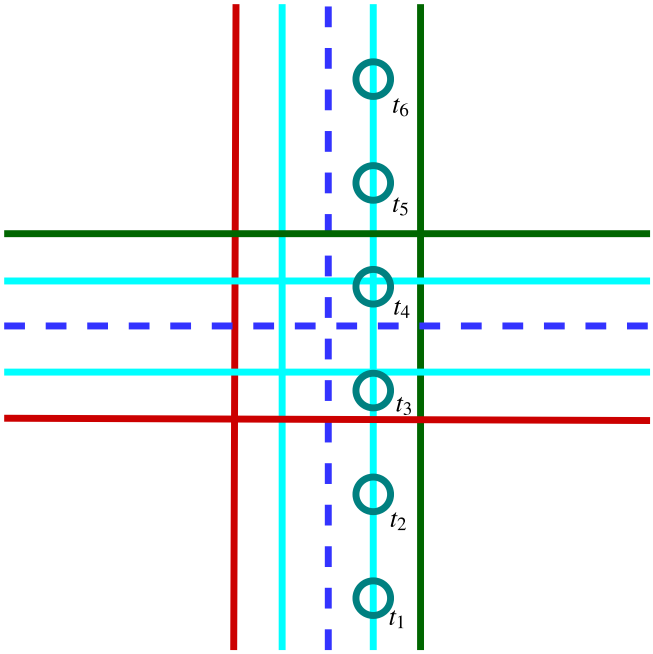


Figure 35: Two-lane roads where the horizontal road is an overpass. The teal-colored rings are RTK GPS coordinates observed at times $t_1, \ldots, t_6$. By only looking at individual observations, it is unclear whether the vehicle has been on the overpass or the underlying road at times t_3 and t_4.

lanes since the heading differences are around 90°. The estimated true lane traversed at time t_k, $GT(y_k)$, is thus defined as

$$GT(y_k) = s_i, \quad \text{where} \begin{cases} \Delta distance_{y_k,s_i} \leqslant \Delta distance_{y_k,s_j} & \forall s_j \neq s_i, \\ \Delta heading_{y_k,s_i} \leqslant \Delta heading_{y_k,s_j} & \forall s_j \neq s_i. \end{cases} \tag{25}$$

y_k is the observation at time t_k and s_i is the ground truth lane. $\Delta distance_{y_k,s_{(\cdot)}}$ is the distance between the RTK coordinate in y_k and middle line in lane $s_{(\cdot)}$ and $\Delta heading_{y_k,s_{(\cdot)}}$ is the heading difference between the observation and lane heading. Having Equation (25), the estimated true lanes traversed by a vehicle at times $t_1, \ldots, t_K$ are collected as

$$GT(y_1), \ldots, GT(y_K).$$

This is referred to as the ground truth path.

6.2 CLASSIFICATION OF RESULT

When having both the true and estimated path, given by ground truth and the Viterbi algorithm, respectively, it is possible to compare them and conclude where the algorithm matched lanes correctly and incorrectly. The correctness and incorrectness can be expressed in terms of different classes. Lanes which the algorithm matched correctly, according to the ground truth, are classified as *true positives*. Lanes part of the path returned by the Viterbi algorithm which are not present in ground truth are referred to as *false positives*, while lanes in ground truth which are not in the estimated path are *false*

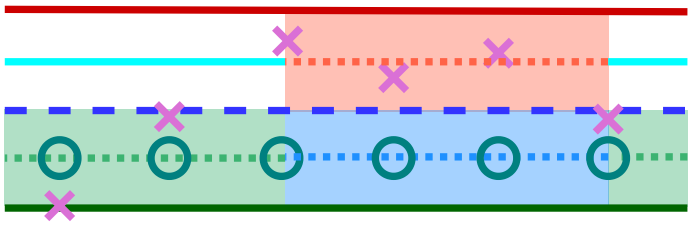


Figure 36: Comparing the ground truth against the path returned by the Viterbi algorithm. Green lane segments are true positives, red is for false positives and blue represents false negatives. The orchid-colored crossed and teal-colored rings represent SPA and RTK coordinates, respectively.

negatives. Given an estimated path $[x_1, \ldots, x_K]$ and the true path $[\hat{x}_1, \ldots, \hat{x}_K]$, the classifications can be defined as follows.

$$\mathcal{TP} := \{x_i : \quad x_i = \hat{x}_i\} \tag{26}$$

$$\mathcal{FP} := \{x_i : \quad x_i \neq \hat{x}_i\} \tag{27}$$

$$\mathcal{FN} := \{\hat{x}_i : \quad x_i \neq \hat{x}_i\} \tag{28}$$

In Figure 36, the vehicle has traversed the lower lane. However, the noisy SPA coordinates, visualized as orchid-colored crosses, resulted in the algorithm returning a faulty path. Given that the vehicle traversed in an eastbound direction, the estimated path is correct for the two first and the last time step, as indicated by the green color (true positives). The third, forth and fifth time steps are matched to the upper lane, colored red (false positives), while the true lane is the lower one, colored blue (false negatives).

6.3 METRICS

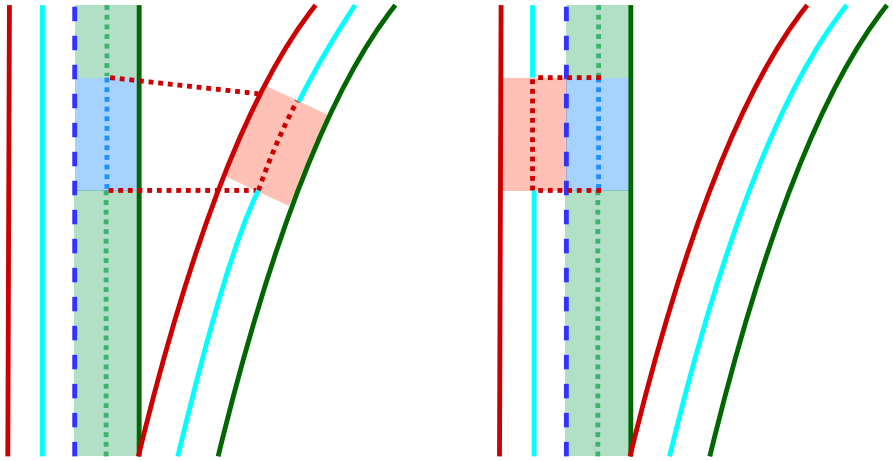
A metric showing the relative amount of correctly matched lanes, i.e. recall, is introduced in Section 6.3.1. The recall only considers correct and incorrect matches for every observation, which does not reveal the level of incorrectness. More specifically, it does not consider for how long distances the algorithm is correct or incorrect. Therefore, an additional metric is derived. The metric needs the lengths of the correctly and incorrectly traversed paths. As such, a principled approach to stitch lane segments together is derived and presented in Section 6.3.2. When matches and mismatches have been stitched together into continuous paths, the so called path length error metric can finally be calculated according to Section 6.3.3.

6.3.1 Recall

Having classification sets according to Equations (26) to (28), the recall of an estimated path is given by

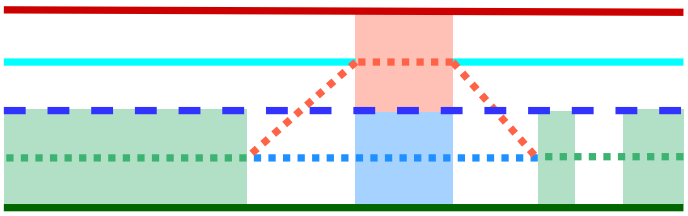
$$\text{recall} := \frac{|\mathcal{TP}|}{|\mathcal{TP}| + |\mathcal{FN}|} \tag{29}$$

where $|\mathcal{TP}|$ and $|\mathcal{FN}|$ are the number of true positive and false negative classifications, respectively.



(a) When incorrectly matching to a lane which is not directly connected to the previous and succeeding matched lanes, lines connecting the lanes have to be incorporated.

(b) When an incorrectly matched lane is located right next to the previous and succeeding matched lanes, it is considered less erroneous than incorrectly matching to a lane lying further away, as in Figure 37a. This is implied by the shorter length of the red colored dotted line.



(c) The result of stitching lanes in between an estimated path and its corresponding ground truth path, with stitches added also between non-consecutive lane segments.

Figure 37: Three cases of lane stitching. Dotted lines represent the imagined driving path of the vehicle. Colors follow the scheme presented in Figure 36.

6.3.2 Stitching lanes

Incorrectly matching an observation to a lane which is far away should be considered a bigger mistake than matching to a closely located one. For example, incorrectly matching an observation to a lane on a diverging road as shown in Figure 37a, is considered to be worse than incorrectly matching to a neighboring lane as shown in Figure 37b. If only the lengths of the incorrect lane segments were considered in an error metric, i.e. not the imagined driving path between the lanes, the reported errors would be equal for the cases in Figures 37a and 37b. As this is not desired, lines connecting lanes to previous and succeeding lanes is needed in order to estimate the length of the incorrect path. The result of stitching the lanes is clearly visible in Figure 37a, where the red dotted lines stitch the lanes together. In comparison, the stitching in Figure 37b shows that the length of the faulty path is shorter when the incorrectly matched lane is closer to the previous and succeeding lane.

While this section has only considered stitching lanes where the estimated path has an incorrectly matched lane which is located laterally apart from the surrounding matched lanes, the stitching is performed similarly for every lane in the path. Regardless of the classification of a lane, it is stitched to the next lane. This permits the creation of a continuous path also when both the ground truth and the map matched path skip a lane segment (caused by the low sampling frequency). Adding stitches between the ground truth and estimated path of a drive could yield a result similar to Figure 37c.

6.3.3 Path length error

Inspired by the loss function proposed by Krumm and Newson in [9], the so called path length error (PLE) is derived. Similarly to their metric, the PLE is based on the fraction of the length of the incorrect route, but at lane-level resolution. Its formal expression is given by

$$\text{PLE} := \frac{\sum_{x_i \in \mathcal{FP}} length_{x_i} + \sum_{x_i \in \mathcal{FN}} length_{x_i}}{\sum_{x_i \in \mathcal{TP}} length_{x_i} + \sum_{x_i \in \mathcal{FN}} length_{x_i}}, \tag{30}$$

where $length_{x_i}$ is the length of lane x_i.

7 RESULTS AND DISCUSSION

With all necessary components defined, it is finally possible to evaluate the map matcher. First, a preliminary test is performed on a subset of the data, as described in Section 7.1. This preliminary test shows which transition model performs best. Parameter tuning in the best model is described in Section 7.2. It is followed by an extensive evaluation on the test data in Section 7.3, covering the whole mapped road network. An analysis of the importance of the various emission penalties is covered in Section 7.4. Strengths and weaknesses of the algorithm are also identified and discussed in Section 7.5. Lastly, some privacy aspects of the user is discussed in Section 7.6.

7.1 FIRST EVALUATION

A first evaluation of the model is performed on a subset of the whole data set, whose coordinates are restricted to the small bounding box seen in Figure 38. The size of this bounding box data is 420 drive logs. Both the constant and refined connectivity-based (using default depth 7) and extrapolation-based transition probabilities are evaluated. The results are summarized in Tables 4 to 6.

It can be concluded that among the transition models described in Section 5.5, the model with the constant connectivity-based transition probabilities performs better than the models with refined connectivity-based transitions and extrapolation-based probabilities. It also has the advantage of being simplistic in design and relatively computationally cheap. Therefore, this will be the choice of transition model for the more extensive evaluation following in this chapter.

Although the extrapolation-based transition model has been shown to be defeated in terms of performance, it has a hypothetical use case which the connectivity-based transition model lacks, namely for trajectories that for some reason lie outside the known road network. This can happen if the map is incomplete or out of date, leading to missing or incorrect lane segment connections. As described in a Section 3.2, the incompleteness of the TomTom map is known. It would therefore be possible to combine the two

Table 4: Results when using model with constant connectivity-based transitions and data contained in the bounding box.

Metric	Recall	Path length error
Mean	0.9103	0.1035
Median	0.9504	0.0368
SD	0.1346	0.2200

Table 5: Results when using model with refined connectivity-based transitions and data contained in the bounding box.

Metric	Recall	Path length error
Mean	0.9023	0.0917
Median	0.9344	0.0454
SD	0.1217	0.1932

Table 6: Results when using model with extrapolation-based transitions and data contained in the bounding box.

Metric	Recall	Path length error
Mean	0.9068	0.1066
Median	0.9344	0.0446
SD	0.1459	0.2664

transition models in order to increase the model performance. To facilitate the analysis of the results, however, only the connectivity-based transitions will be used. Instead, the combined-transitions model is left as a proposed investigation for future work.

7.2 TUNING OF NEIGHBORHOOD DEPTH

As described in Section 5.5.1, the connectivity-based transition probabilities vary depending on the maximum depth considered. Generally, a smaller depth means that fewer transitions are considered, while bigger depth means more transitions. It is not intuitively obvious which depth yields optimal result in terms of the performance metrics presented in Chapter 6. A small depth truncates the candidate space and may lead to cases where a transition is entirely missed. On the other hand, a large depth may amplify the errors in the underlying assumptions, leading to highly unlikely transitions.

The performance is reported in terms of recall and PLE of the training data. Its neighborhood depth-dependence can be seen in Figure 39. The mean recall reaches 0.9119 at best. This corresponds to the peak of the blue curve in Figure 39a at depth 11. The median improves with increasing depth from 1 to 3, whereon it remains at 0.95 for all succeeding depths. The standard deviation remains within a tight interval for all depths and only has local fluctuations.

A neighborhood depth of 11 is optimal also when considering the PLE derived in Figure 39b. For that depth, the mean PLE attains the value 0.0909. Similar to the median recall, the median PLE decreases for the initial depths and then remains quite stable around 0.035. The standard deviation clearly has an increasing trend with bigger depth.

From these results, it can be argued that the algorithm yields the best possible result when using constant connectivity-based transition probabilities with depth 11.

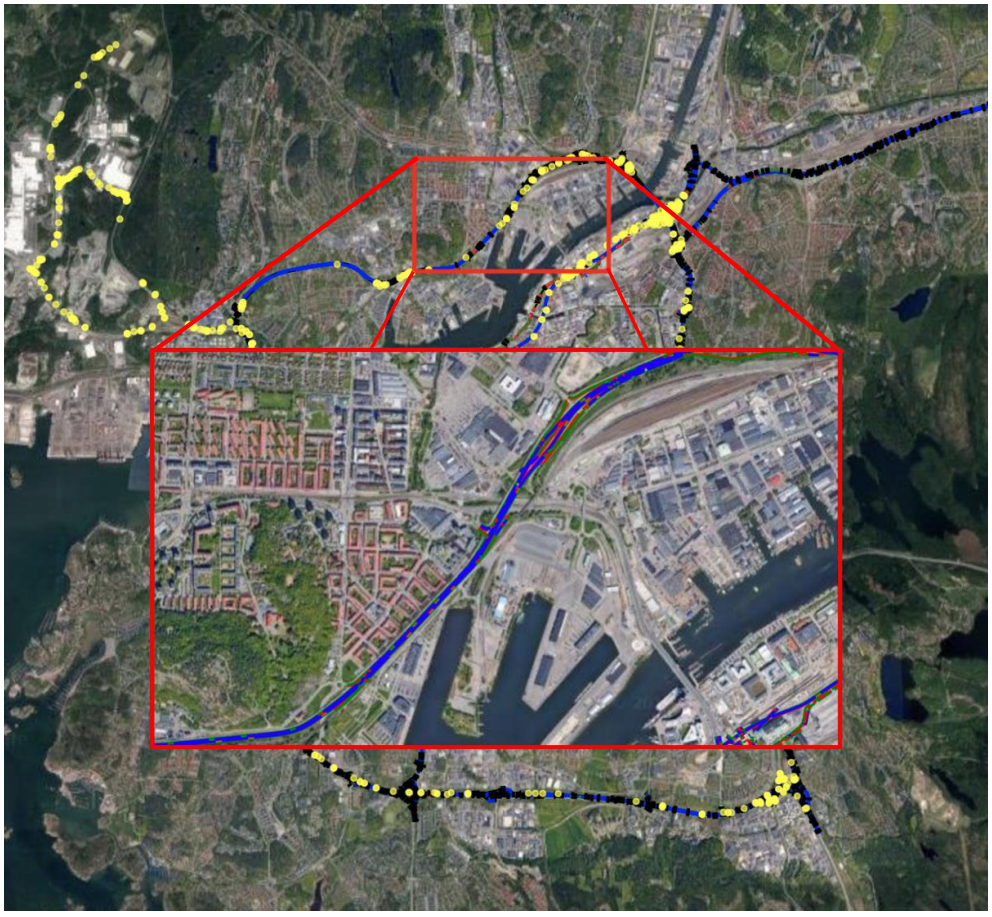
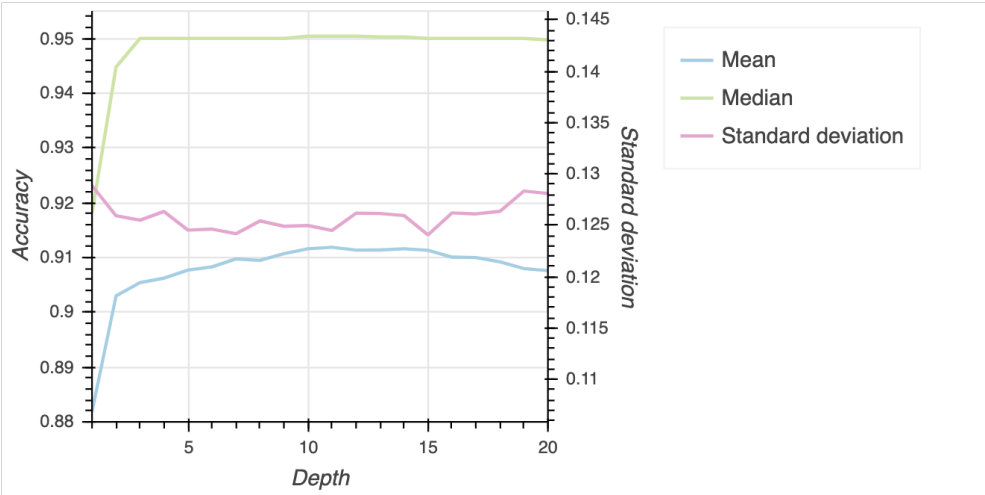
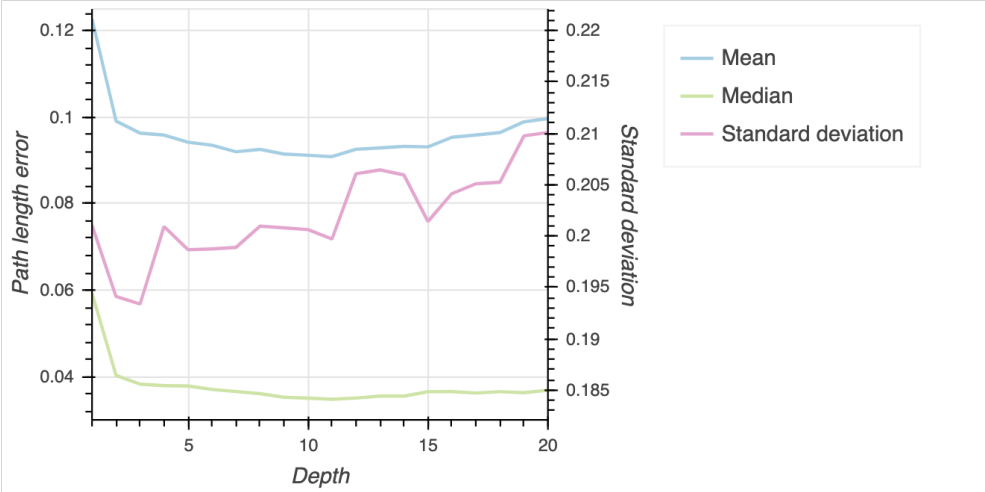


Figure 38: Enhancement of the north-east bounding box, covering a stretch of Lundbyleden, seen together with Google map. Just as in Figure 7, the blue lines represent roads and lanes and red markers traffic signs. Stop lights and junctions are marked by yellow and black objects.



(a) Recall.



(b) Path length error.

Figure 39: Performance of the model using constant connectivity-based transition probabilities with different depths.

Table 7: Results when using the final model on the test data.

Metric	Recall	Path length error
Mean	0.9140	0.0983
Median	0.9508	0.0331
SD	0.1337	0.2196

Table 8: Results from using a naive matcher on the test data.

Metric	Recall	Path length error
Mean	0.7207	0.3827
Median	0.7667	0.2576
SD	0.2225	0.3857

7.3 PERFORMANCE ON TEST DATA

After fixating the depth, the final model can be defined and run on the test data. The results are summarized in Table 7 and are the final results of the map matching algorithm. As seen, the reported numbers are very similar to the ones in the above section. A naive matcher, which matches GPS locations to the nearest lane, is used as benchmark. Its performance is summarized in Table 8.

7.4 IMPORTANCE OF PROBABILISTIC COMPONENTS

Based on the performed analyses on the digital map and the observation data, it was concluded that some variables were important to the map matching problem. These were consequently incorporated into the model in various ways. The proposed transition probabilities not only use the current and next lanes as inputs, but also the value of the lane change variable. As such, the transitions are dynamic. Other variables, i.e. speed, heading and left or right lane marker types, affect the map matching through penalties, based on assumptions about the driving characteristics.

To measure the individual importance of these variable inclusions, they will be omitted from the complete model, one at a time. The performance differences will then be analyzed and, hopefully, yield new insights. The results are not only of interest from a thesis perspective, but also for the future applications of the implemented map matching algorithm. Sometimes, all variables may not be available, or deemed untrustworthy. In these instances, the algorithm can run without the corresponding probabilities and/or penalties. If the missing variable is of low significance, the algorithm will be expected to still be successful. If a crucial variable is excluded, however, the outcome of the map matching algorithm becomes unreliable.

As can be seen in Figure 40, the model without the penalties introduced in Section 5.3 and without accounting for lane changes, as presented in Section 5.5.4, yields a median recall below 0.8. The complete model, which incorporates all penalties and accounts for lane changes, reaches a median recall around 0.95. From the drop in median recall when omitting the marker

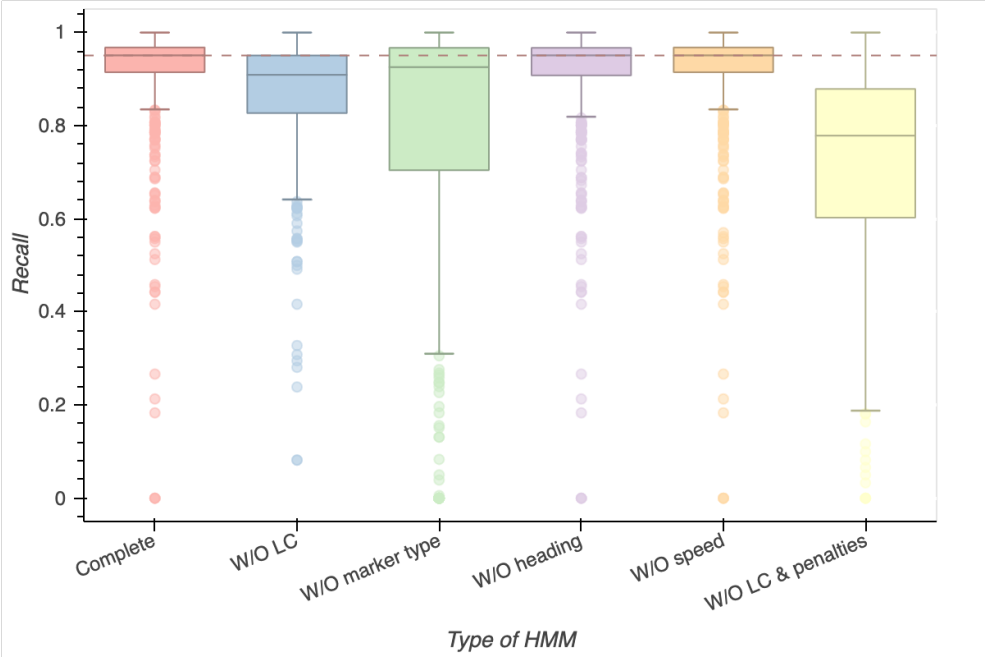
type penalty and not accounting for lane changes, it is apparent that those are crucial components of the algorithm.

Discarding the lane change consideration from the complete model yields the biggest drop in recall among all components. When omitting this component, the transition probabilities are independent of the lane change signal provided by the vehicle. Thus, even though the vehicle signals e.g. a left lane change, it is equally likely to drive left, straight and right according to the transition probabilities. By the noisy nature of the GPS measurements, they are not capable of detecting lateral vehicle movements. Among all data available from the sensors given in Chapter 4, the lane change indicator is one of the primary sensors fitted to accurately detect lateral movements. Since a map matching lane-level algorithm is sought, it is crucial to have data that captures movements between parallel lanes. By this, it is evident that the lane change indicator provides a good foundation for lane-level map matching and accounting for lane changes enhance the performance of the algorithm.

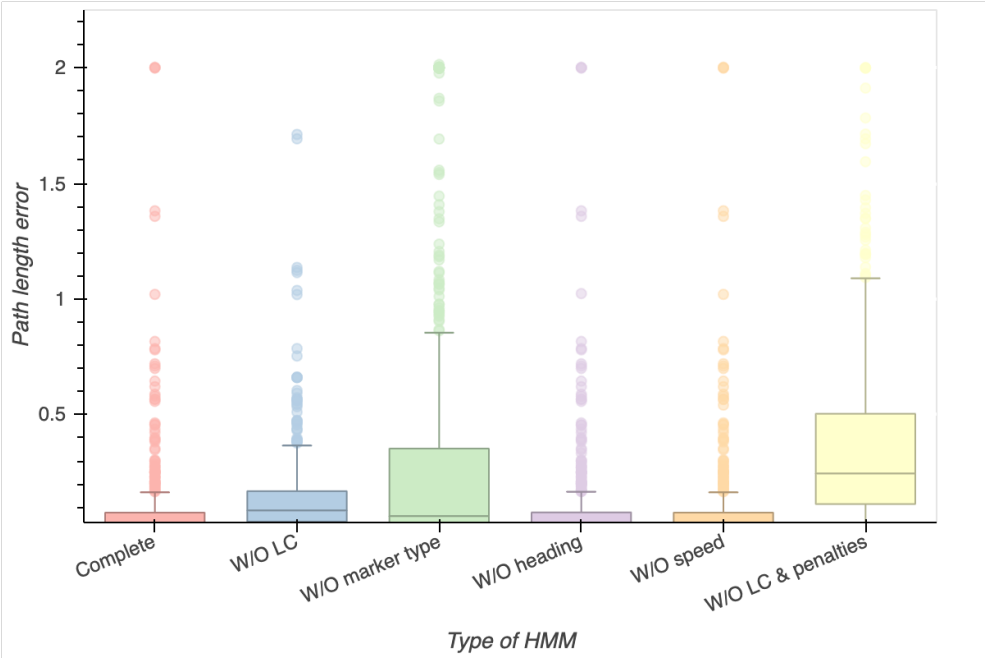
The lane marker types help the algorithm to distinguish between two or more parallel lanes, if these have different lane markings. At the exclusion of the marker type penalty, Figure 40a shows how the median recall drops with a few points. However, the drop is not as significant as when omitting the lane change consideration. A possible explanation for this is that, although they might help narrow down the set of candidate lanes, the marker types are not always unique for parallel lanes. Hence, it is not always useful. As Figure 17 showed, the reported lane markings cannot always be trusted, even when the sensor is very confident. This probably also factors into the importance of the marker type penalty. It can still be concluded that, besides the lane change signal, the marker type detections are key to achieve the required resolution for lane-level map matching. Hence, it is certainly desired to incorporate information about lane marker types when performing map matching on lane-level.

As can be seen in the box plot in Figure 40a, omitting the heading and speed penalties do not have any notable effect on the performance of the algorithm. Considering the detail of lane-level map matching, it is not too surprising to find that the speed and heading do not provide sufficient information to separate lanes. In all likelihood, the speed limits and lane headings are almost exclusively the same for parallel lanes. This implies that the speed and heading registered by a vehicle cannot be unambiguously matched to a certain lane.

It is, however, noteworthy that a different transition model may make use of the speed and heading information. For instance, the extrapolation-based transition model proposed in Section 5.5.2 and other spatial transition models, may allow overpasses as candidate lanes. The heading and speed penalty will then filter those out in a manner analogous to the ground truth setup. Albeit not of importance for the final model, speed and heading penalties should not be underestimated.



(a) Recall.



(b) Path length error.

Figure 40: Box plots of the performance on the test data for different models. From left to right, these are; the complete model, a model without added lane change transitions, without added lane marker type penalty, without heading penalty, without speed penalty, and a model without any added penalties or lane change transitions. The dashed line shows the median score of the complete model.

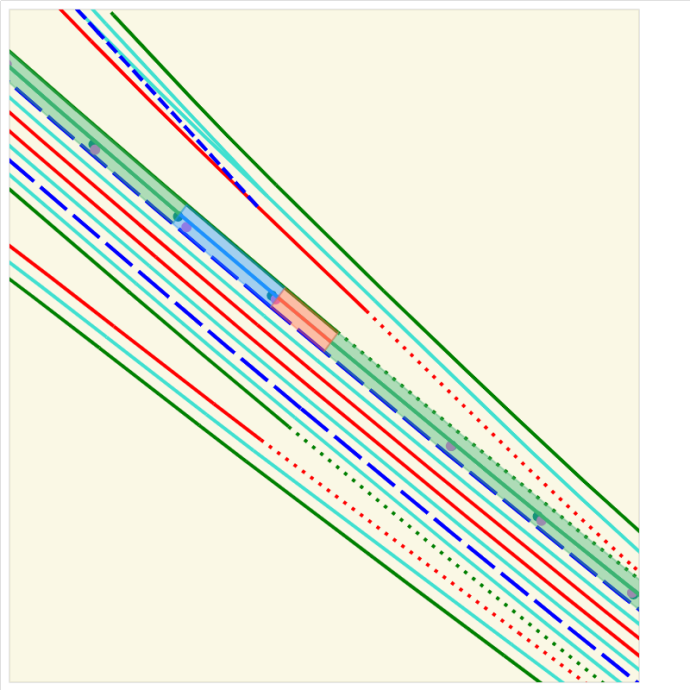


Figure 41: A typical, high-scoring drive. The small but nevertheless prevalent errors are caused by the reported GPS locations being either behind or in front of the actual vehicle location. The lane segments have been colored according to the scheme in Figure 36.

7.5 RESULT ANALYSIS

The map matcher proposed by Rabe et al. in [18], based on least square optimization, achieves error rate 0.2%. Comparison of results is however not possible due to different error metrics and dissimilar map and sensor data. Especially, they utilize RADAR data, which has been excluded in this work. Furthermore, they evaluate their model in urban areas containing complex road structures such as intersections. The evaluation in this work is rather performed almost exclusively on highways and the data does not contain any drives in intersections.

Instead, the achieved result of 95.1% median recall and 3.3% median path length error are compared to the benchmark map matcher reported in Section 7.3, for which all data and map conditions remain the same. From the performance gain, it can be concluded that our map matcher is multiple levels of sophistication above the benchmark.

For most of the evaluated drive logs, the algorithm performs very well. A typical drive may look like Figure 41, where the errors are caused by a lag in the GPS data. Such situations have small effect on the path length error, which is overall low.

The variation in performance is, however, quite large. This holds the implication that there are situations where the map matching results are non-ideal. Through an investigation of some good and error-prone drives, the overall strengths and weaknesses of the algorithm can be identified.

The Viterbi algorithm is successful at finding the correct path even for high-complexity cases. Examples of such cases are seen in Figures 42 to 44.

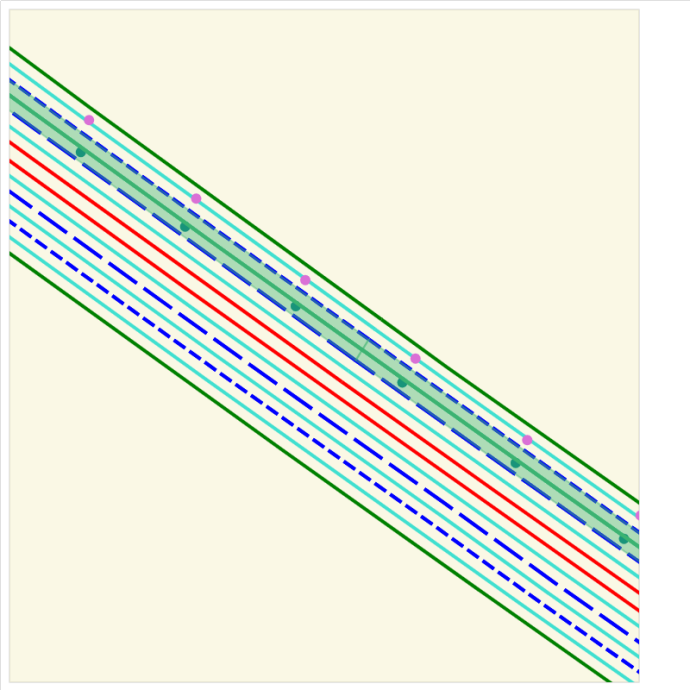


Figure 42: Good case 1: with help from the lane marker type penalty, the correct path is found even though the location measurements fed to the algorithm (orchid-colored) are located in a different lane.

There also exist some cases for which the VA performs sub-optimally. These are shown and explained in Figures 45 to 49. From these case studies, it is possible to conclude some general strengths and weaknesses of the model. The strengths can be summarized as:

- robust for noisy GPS data when the vision system is reliable
- enforces lane change detections
- handles holes in the map.

The weaknesses are summarized as:

- inaccurate for noisy GPS data together with unreliable vision system
- cannot separate between parallel lanes with the same marker types
- inflexible transition model when having unreliable vision system
- insufficient handling of holes in the map at complex road network geometries.

The following sections describe these properties more thoroughly.

GPS error

The GPS observation probability presented in Section 5.3.1 gives the probabilistic distribution of the distance between a candidate lane and the perceived vehicle (i.e. GPS) location. The larger the GPS error, the smaller observation probability of the true lane. For data where the vision system reports

Figure 43: Good case 2: the VA competently deals with missing lanes.

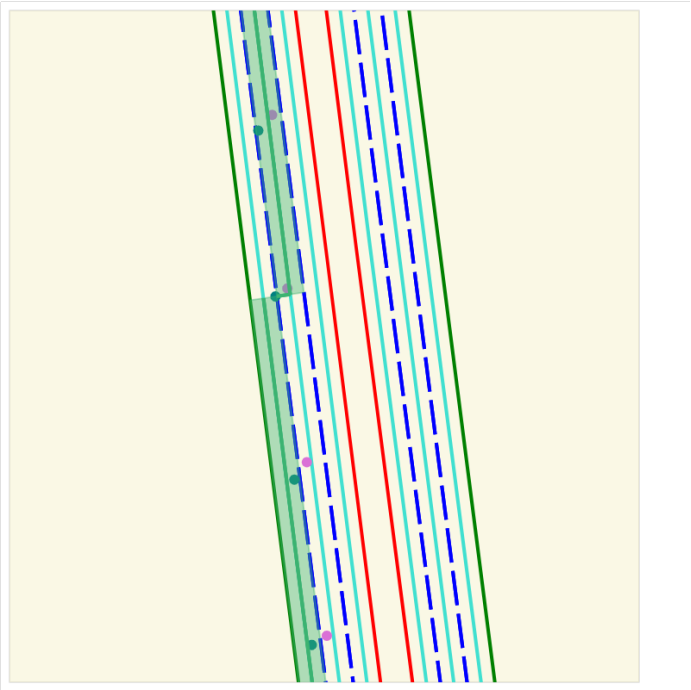


Figure 44: Good case 3: the VA correctly follows the vehicle through a lane change.

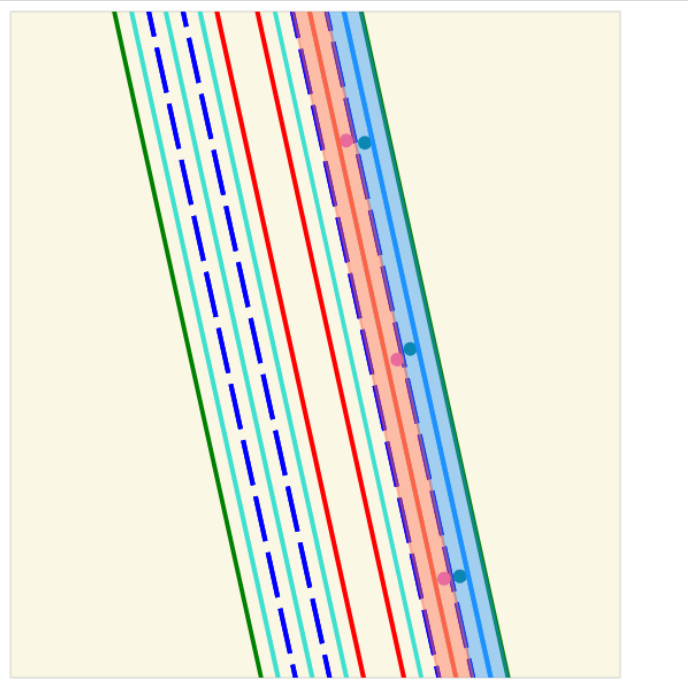


Figure 45: Bad case 1: when lane marker type confidences are always 0, only the GPS location (orchid-colored) is used to distinguish between parallel lanes.

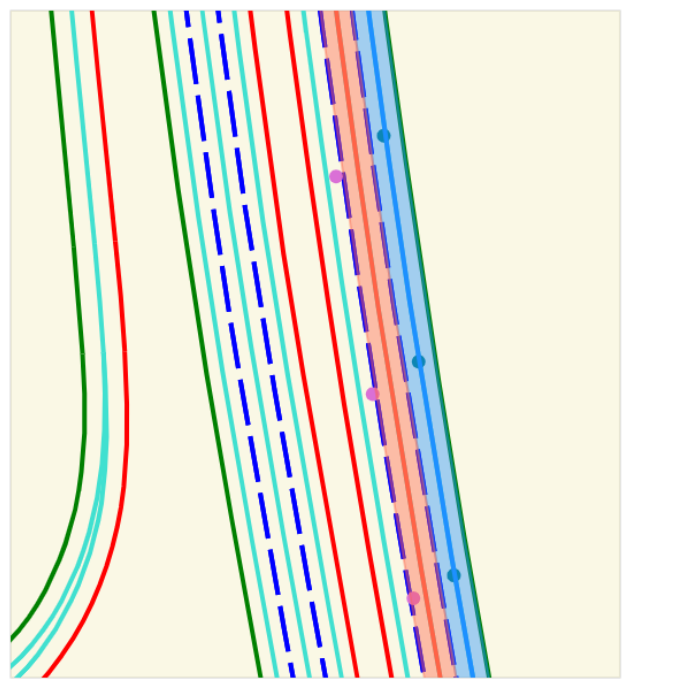


Figure 46: Bad case 2: left and right lane marker types are correctly reported as (dashed, solid), as for the true states, but the GPS error is too big. This leads to the middle lane being identified as the most probable one, even though it has incorrect lane markers.

Figure 47: Bad case 3: the GPS data is straightened up over time, but the VA fails at changing the path into the correct one. This is caused by the added lane change transitions being too influential - since there is no lane change, a straight path is enforced. There are also no lane markers to get help from, as the detector again reports 0 confidence.

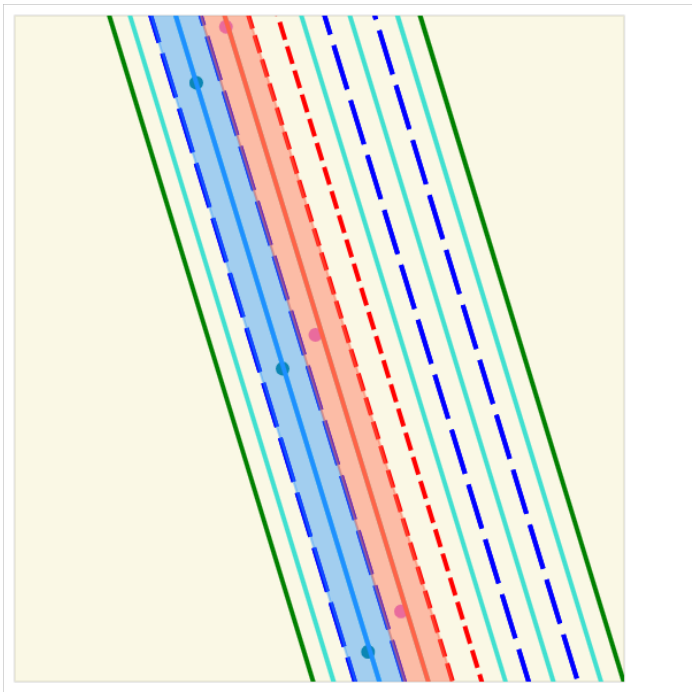


Figure 48: Bad case 4: the leftmost and middle lanes both have dashed left and right marker types, making it impossible to differentiate between the two.

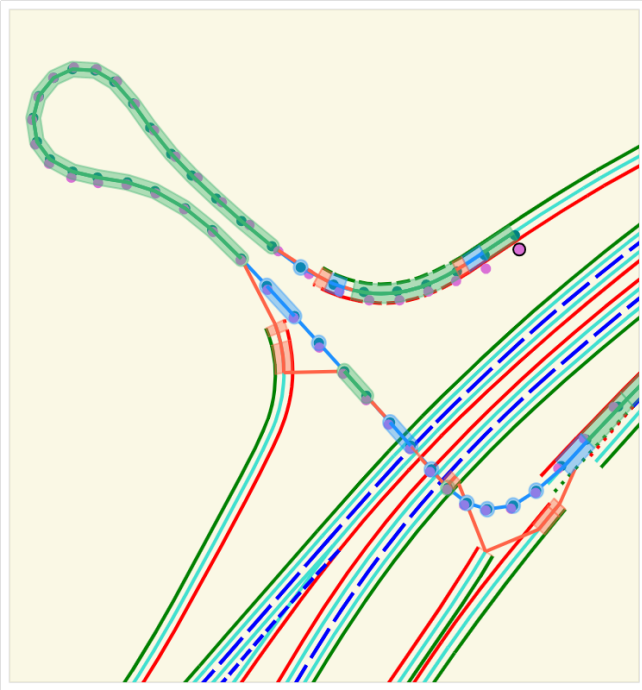


Figure 49: Bad case 5: the algorithm successfully handles holes in the map when no mapped lanes are nearby, as in the upper loop. When other lanes interfere, however, the points get matched to them instead of the placeholder lane, NaN. This can be seen at the (missing) overpass.

left and right confidence 2, i.e. where we can assume that the reported lane marker types are largely correct, the model has recall above 80% when the GPS position is less than 8 meters off. With larger GPS errors, the performance quickly drops.

As such, when the GPS noise is within a few meters while all other vehicle sensors perform ideally, the algorithm generally has no difficulties to accurately identify the traversed lanes. However, when the GPS error is large, the true lane becomes a highly unlikely candidate, even if the lane markers are correctly identified. In other words, the observation probability is more influential than the lane marker type penalty. The result is an incorrectly matched path. A representative case of this can be seen in Figure 46, where the GPS error is approximately 5 meters. Although this error is lower than the tolerance specified above, the Viterbi algorithm still fails to find the correct path. The probable cause of this is that the GPS error here is perpendicular to the road heading. On the other hand, when the lateral GPS error is only moderately wrong ($\leqslant 4$ meters), as in Figure 42, the algorithm succeeds at mapping to the correct lane.

As a test, the influence of the GPS observation probability was changed by trying different values of σ_{GPS} in Equation (7). The mean recall on the test set was still highest when using $\sigma_{\text{GPS}} = 3$, as can be seen in Figure 50. Although a less strict GPS observation probability resolved some problematic cases, the overall performance on the test data was decreased.

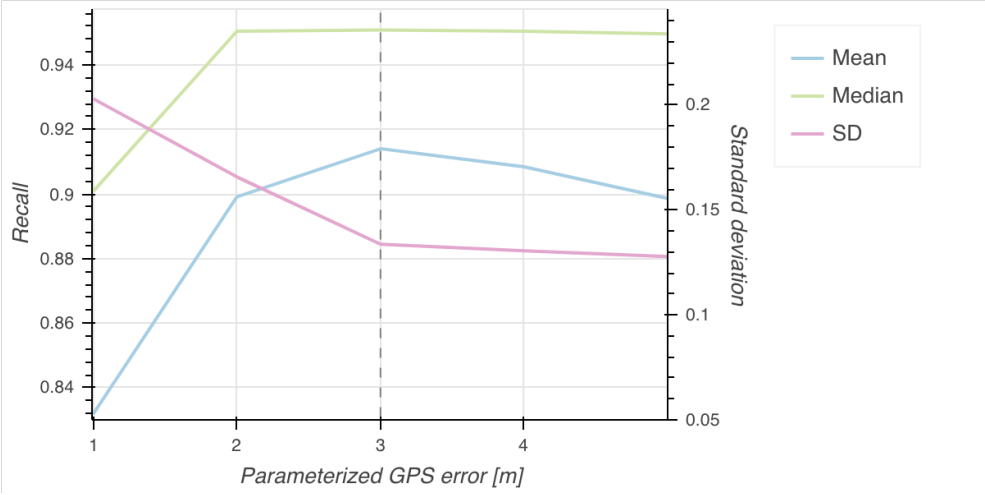


Figure 50: Recall on test set for different values of the parameterized GPS standard deviation, σ_{GPS}. The dashed line indicates which value of σ_{GPS} yields the highest mean recall.

Uncertain vision detection

No lane marker type penalty is applied for drives with uncertain vision detection. When a vehicle produces unreliable vision detection, only the GPS measurements are used to identify the traversed lanes. In combination with erroneous GPS locations measurements, the path returned by the Viterbi algorithm is likely incorrect. A drive affected by this is shown in Figure 45.

Rabe et al. achieved very low error rates utilizing RADAR data in their algorithm. Adding redundant RADAR data also to our model would arguably make it less dependent on the vision system and make the algorithm more competent at handling failing lane marker type detections.

Lane change transitions

When actual lane changes are detected, the algorithm is helped by the enhanced lane change transition probabilities. This is what happens in Figure 44. Normally, it is also helped when no lane changes are detected, since this helps it to follow a straight path rather than change lanes according to noisy GPS locations. Nonetheless, the same lane change transitions can in some situations cause undesired output. This may happen in situations where the GPS measurements are faulty, in combination with missing or incorrectly reported lane marker types.

In Figure 47, the GPS measurements are initially noisy but converge towards the actual locations over time. The inaccurate location measurements, together with unreliable marker detections, contribute to faulty lane matching in the beginning. When the location measurements become more accurate, the algorithm still matches to lanes following a straight path from the previous incorrectly matched lanes. This behavior originates from the added lane change transition probabilities, which disfavors lateral movements unless the lane change detector signals one. Combined with the lack of lane marker detections, as in Figure 47, this transition model becomes too inflexible. Knowing this, it seems justified to let the transition probabilities

be non-dynamic when having observations reporting low confidences. That way, no direction is favored for a vehicle with unreliable vision detections. In turn, an initially incorrect path has an improved chance of getting back to the correct lanes. When evaluating the algorithm with this supplement, the overall performance was however not enhanced compared to the complete model.

Ambiguous lane marker types

When there are more than one lane on a road where at least two lanes have the same marker types, information picked up by the vision system are not sufficient to uniquely determine which lane the vehicle is in. For example, Figure 48 shows a case where the vehicle correctly identifies dashed lane markers on both sides. However, the road contains three lanes, whereof two of them have dashed markers on both sides. According to the vision system, the vehicle could be in any of the two lanes with the same markers. If the GPS measurements would have low error and place the vehicle in the actual lane, ambiguous lane marker types would not provide any problems. Now, the whole problem of map matching is motivated by the inability to derive paths solely from inaccurate GPS measurements. Thus, it is primarily interesting to consider cases where the location measurements are not perfect.

In Figure 48, the GPS places the vehicle in the leftmost lane while it actually traverses the middle one. Combined with the ambiguous marker types, this leads to the algorithm incorrectly identifying the leftmost lane as the most probable one. In lack of additional sensors capable of distinguishing between lanes with the same marker types, there is no obvious way to resolve this.

Li et al. address ambiguity issues by letting the map matching output contain multiple lane hypotheses. While they use Particle Filtering, the proposed map matcher could certainly be modified to output more than a unique path. More specifically, the VA could return not only the most likely path, but also the second to most probable and so on. Amongst these hypotheses, the correct path would need to be verified externally.

Holes in the map

Through the Viterbi modification in Section 5.6.1, the map matching algorithm is still capable of yielding a path, using a NaN-lane as placeholder for the missing lane segment(s). Figure 43 shows the resulting path from one such case in which the VA was successful. A less ideal solution is found in Figure 49. As opposed to this second case, no candidate lanes are found for the GPS locations in the first case. Hence, those points are immediately matched to NaN. In the second case, however, there are other lanes in close proximity to the data points. These are subsequently added to the candidate spaces. The lane heading penalty, Equation (9), then removes candidates for which the heading difference is greater than 90°. In this case, the lanes below the underpass appear to have a heading difference contained within this tolerance. In conclusion, the VA is competent at dealing with holes in the map if no nearby lanes have too similar heading to the observation.

The simple off road inclusion in this report builds the route through interpolation of GPS points. Murphy and Pao propose a more sophisticated approach to handle insufficient map data in their HMM map matcher in [17]. Adding a Kalman smoother to enhance GPS-estimated off-road positions yields a less noisy path than using the GPS positions directly, and is therefore more suitable e.g. for updating unmapped areas.

7.6 PRIVACY ASPECTS

Questions about privacy naturally arise when developing systems capable of tracking people's location and behavior. In practice, there is no guarantee against map matching being used with malicious intent. In that case, lane-level map matching could for instance be used to profile risk-prone drivers.

8 CONCLUSION

As ADAS and AD features are becoming increasingly prevalent in vehicle technology, so grows the importance of map matching. The purpose of this thesis was to propose and implement a novel lane-level map matching algorithm based on HMMs. The research questions can be reiterated as; finding suitable parameters for the model, and suitable metrics to measure the algorithm performance. The states were taken directly as the lane segments given by the TomTom map. Transition and emission probabilities where chosen such as to capture important variables. The recall and PLE provide metrics well-adapted for measuring the performance of a lane-level map matching algorithm.

In contrast to the HMM map matcher proposed by Newson and Krumm [9], the algorithm proposed in this work is capable of matching to lanes rather than roads. Beyond GPS measurements, additional vehicle sensor data is incorporated into the model. An HD map is used instead of a low resolution map, providing information about e.g. lane markings on centimeter-level.

An online decoding functionality is also implemented, based on the work presented by Goh et al. [15]. Using convergence points, the algorithm is optimal in map matching performance but sub-optimal in output delay.

The proposed algorithm, in both offline and online states, has shown to perform significantly better than a naive matcher. In addition, it is capable of handling holes in the road network in a principled way. Through studying some representative cases, it has been established that the algorithm is robust to noisy GPS measurements with a confident vision system. However, unreliable vision data risks to reduce the performance of the algorithm.

This simplistic model does not require any RADAR or LiDAR data. Even so, its performance can be considered to be competitive to other map matching algorithms which are not as light-weight. As vehicles owned by the general mass are typically not equipped with expensive sensors such as LiDAR, an algorithm only using data from ordinary sensors is certainly desired.

Of course, the model is not without flaws and can benefit from further refinement. We suggest some areas of future work below.

8.1 FUTURE WORK

The proposed algorithm has been evaluated on data and a road network consisting of highways. As such, it is not known how it performs in other traffic situations, e.g. at junctions and in urban traffic. It would certainly be interesting to try this, and also study how the model can be extended to do so successfully. High-raise buildings, causing the already noisy GPS signal to deteriorate further, and the close vicinity to other roads typically

pose problems for map matching algorithms. Therefore, an urban-scenario extension may call for the integration of additional variables.

RADAR data, for instance, may be a useful complement to the vision data in urban scenarios but also in the evaluated highway cases. Seeing that the map matcher performance suffers when the GPS is very noisy and the lane marker type detector unreliable, and when it is reliable but candidate lane segments have the same lane markers, it is desired to have another sensor capable of mapping its surroundings. RADAR data can be used to find objects on and outside the road. The position of these objects are given relatively to the vehicle of interest. For example, if a vehicle makes detections of cars traversing beside it, it might give insight into whether the vehicle is in an outer or inner lane. Additionally, detections of traffic signs would provide information about distances to the road side. This could help deduce if the vehicle is in a far left or right lane. A possible extension is thus to investigate a way to incorporate RADAR data into the model.

In the presented model, the states were chosen as the lane segments given by the TomTom map. As lane segments are represented by polygons spanned by the lane borders, observations are matched to an area wherein the vehicle is assumed to be. Thus, if an observation is matched to a lane stretching above 1 kilometers, the localization is not very precise after all. This localization could be refined by splitting the lane segments into even smaller pieces and letting them represent the states in the model. For the map used, it was convenient to use the the lane segments given by TomTom directly. However, having states correspond to a smaller area would undoubtedly yield more narrow localization.

The connectivity-based transitions were found to perform best for the data used to evaluate the algorithm. This does not mean that transitions derived through spatial approaches, e.g. extrapolation-based, are useless. In fact, at places where there are holes in the road network, the connectivity-based transitions are unable to detect anything other than a dead end. Spatial-based approaches would however be capable of inferring a transition between lanes separated by a hole in the map.

In general, the choice of transition model does not need to be fixed to start with. Hypothetically, the algorithm would benefit of being able to switch between different transitions models based on the road network situation. That would make it possible to apply extrapolation-based transitions when encountering dead ends in the road network, while transitions relying on the topology would be used otherwise. An investigation of such flexible transition models remains to be done.

A Variable Sliding Window approach was chosen to achieve an online version of the Viterbi algorithm. The technique was favorable by its optimality guarantee and ease of implementation. There are a range of other online approaches that could have been selected if it better fitted the application purpose. As the end use of an online lane-level map matcher has not been thoroughly investigated, it remains to figure out a valuable purpose and choose online extension accordingly.

BIBLIOGRAPHY

[1] G. Miller. "Autonomous Cars Will Require a Totally New Kind of Map". In: *Wired* (Dec. 15, 2014). URL: https://www.wired.com/2014/12/nokia-here-autonomous-car-maps/.

[2] K. Chellapilla. *Rethinking Maps for Self-Driving*. Oct. 15, 2018. URL: https://medium.com/@LyftLevel5/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6.

[3] H. Vardhan. *HD Maps: New age maps powering autonomous vehicles*. Sept. 22, 2017. URL: https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles.

[4] Synced. *The Golden Age of HD Mapping for Autonomous Driving*. Aug. 11, 2018. URL: https://medium.com/syncedreview/the-golden-age-of-hd-mapping-for-autonomous-driving-b2a2ec4c11d.

[5] M. Bergen. "Nobody Wants to Let Google Win the War for Maps All Over Again". In: *Bloomberg* (Feb. 21, 2018). URL: https://www.bloomberg.com/news/features/2018-02-21/nobody-wants-to-let-google-win-the-war-for-maps-all-over-again.

[6] S. Levy. *Your Next Gig: Map the Streets For Self-Driving Cars*. Oct. 11, 2017. URL: https://www.wired.com/story/your-next-gig-map-the-streets-for-self-driving-cars/.

[7] C. S. Jensen and N. Tradišauskas. "Map Matching". In: *Encyclopedia of Database Systems*. Ed. by L. Liu and M. T. Özsu. Boston, MA: Springer US, 2009, pp. 1692–1696. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_215. URL: https://doi.org/10.1007/978-0-387-39940-9_215.

[8] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. "Current map-matching algorithms for transport applications: State-of-the art and future research directions". In: *Transportation research part c: Emerging technologies* 15.5 (2007), pp. 312–328.

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

[10] *TomTom*. URL: https://www.tomtom.com (visited on 01/08/2019).

[11] *Drive me*. URL: https://www.volvocars.com/intl/buy/explore/intellisafe/autonomous-driving/drive-me (visited on 01/08/2019).

[12] *Zenuity*. URL: https://www.zenuity.com (visited on 02/11/2019).

[13] Z. Ghahramani. "An introduction to Hidden Markov Models and Bayesian networks". In: *International Journal of Pattern Recognition and Artificial Intelligence* 15.01 (2001), pp. 9–42. DOI: 10.1142/S0218001401000836.

[14] L. R. Bahl et al. "Acoustic Markov models used in the Tangora speech recognition system". In: *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*. Apr. 1988, 497–500 vol.1. DOI: 10.1109/ICASSP.1988.196628.

[15] C. Y. Goh et al. "Online map-matching based on Hidden Markov model for real-time traffic sensing applications". In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. Sept. 2012, pp. 776–781. DOI: 10.1109/ITSC.2012.6338627.

[16] A. Luo, S. Chen, and B. Xv. "Enhanced Map-Matching Algorithm with a Hidden Markov Model for Mobile Phone Positioning". In: *ISPRS International Journal of Geo-Information* 6.11 (2017), p. 327.

[17] J. Murphy and Y. Pao. "Map matching when the map is wrong: Efficient vehicle tracking on- and off-road for map learning". In: *Computing Research Repository* (Sept. 2018).

[18] J. Rabe et al. "Lane-level map-matching based on optimization". In: *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE. 2016, pp. 1155–1160.

[19] F. Li et al. "Lane-level map-matching with integrity on high-definition maps". In: *2017 IEEE Intelligent Vehicles Symposium (IV)* (2017), pp. 1176–1181.

[20] L. Xi et al. "Map Matching Algorithm and Its Application". In: *International Conference on Intelligent Systems and Knowledge Engineering 2007*. Oct. 2007. DOI: 10.2991/iske.2007.127.

[21] J. Greenfeld. "Matching GPS Observations to Locations on a Digital Map". In: *Proceedings of the 81st Annual Meeting of the Transportation Research Board*. Jan. 2002, p. 13.

[22] O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2009. Chap. 1. Main Definitions and Notations.

[23] D. Ramage. "Hidden Markov models fundamentals". In: *Lecture Notes. http://cs229. stanford. edu/section/cs229-hmm. pdf* (2007).

[24] G. D. Forney. "The viterbi algorithm". In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[25] D. Jurafsky and J. H. Martin. "Hidden Markov Models". In: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2018). URL: https://web.stanford.edu/~jurafsky/slp3/A.pdf.

[26] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2 (Apr. 1967), pp. 260–269. ISSN: 0018-9448. DOI: 10.1109/TIT.1967.1054010.

[27] J. Omura. "On the Viterbi decoding algorithm". In: *IEEE Transactions on Information Theory* 15.1 (Jan. 1969), pp. 177–179. ISSN: 0018-9448. DOI: 10.1109/TIT.1969.1054239.

[28] G. Slade. *The Viterbi algorithm demystified*. Mar. 2013.

[29] J. Bloit and X. Rodet. "Short-time Viterbi for online HMM decoding: Evaluation on a real-time phone recognition task". In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2008, pp. 2121–2124.

[30] G. Wang and R. Zimmermann. "Eddy: an error-bounded delay-bounded real-time map matching algorithm using HMM and online Viterbi decoder". In: *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2014, pp. 33–42.

[31] A. El-Rabbany. *Introduction to GPS: the global positioning system*. Artech house, 2002.

[32] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global positioning system: theory and practice*. Springer Science & Business Media, 2012.

[33] M. G. Wing, A. Eklund, and L. D. Kellogg. "Consumer-grade global positioning system (GPS) accuracy and reliability". In: *Journal of forestry* 103.4 (2005), p. 169.

[34] R. B. Langley. "Rtk gps". In: *GPS World* 9.9 (1998), pp. 70–76.

[35] T. Takasu and A. Yasuda. "Evaluation of RTK-GPS performance with low-cost single-frequency GPS receivers". In: *Proceedings of international symposium on GPS/GNSS*. 2008, pp. 852–861.

[36] T. Sobh and X. Xiong. *Prototyping of Robotic Systems - Applications of Design and Implementation*. IGI Global, 2012. Chap. 4.4.11 Inertial Measurement Unit, p. 97. ISBN: 978-1-4666-0176-5.

[37] L. Joseph. *Learning Robotics Using Python - Design, Simulate, Program, and Prototype an Interactive Autonomous Mobile Robot from Scratch with the Help of Python, ROS, and Open-CV!* Packt Publishing, 2015. Chap. 6.3.1 Inertial Navigation. ISBN: 978-1-78328-753-6.

[38] R. P. D. Vivacqua et al. "Self-Localization Based on Visual Lane Marking Maps: An Accurate Low-Cost Approach for Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 19.2 (2018), pp. 582–597.

[39] A. Azzalini. "A Class of Distributions Which Includes the Normal Ones". In: *Scandinavian Journal of Statistics* 12.2 (1985), pp. 171–178. ISSN: 03036898, 14679469. URL: http://www.jstor.org/stable/4615982.

[40] A. Azzalini. "Skew-Normal Distribution". In: *International Encyclopedia of Statistical Science*. Ed. by M. Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1342–1344. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_523. URL: https://doi.org/10.1007/978-3-642-04898-2_523.