Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Øyvind Klåpbakken

Map matching using hidden Markov models

Master's thesis in Applied Physics and Mathematics
Supervisor: Jo Eidsvik
January 2020

NTNU
Norwegian University of Science and Technology

Øyvind Klåpbakken

Map matching using hidden Markov models

Master's thesis in Applied Physics and Mathematics
Supervisor: Jo Eidsvik
January 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

Map matching refers to the process of using a sequence of position measurements to estimate a connected route on a road network. Position measurements used for map matching are typically obtained from a GPS device located in a vehicle moving through the road network. The purpose of this thesis is to explore and evaluate several different hidden Markov model (HMM) formulations that enable map matching. These methods all require knowledge about the underlying road network. This road network is used to create a state space where each state represents a road segment.

It is ensured that the HMM-based map matching methods are consistent with the assumptions of an HMM. The state estimate obtained from the Viterbi algorithm can, therefore, be regarded as the "most probable road segment sequence." One of the contributions of this thesis is to combine ideas from earlier works in a way that is consistent with the assumptions of an HMM. Another contribution comes through the introduction of a state space that, to the author's knowledge, has not been used before. This augmented state space formulation extends the state space by incorporating additional elements that, among other things, include information about the direction of travel. The use of the Baum-Welch algorithm is also believed to be a significant contribution, seeing as it has not been applied to this specific problem before.

The experiments presented in this thesis are conducted using data from simulated traversal on real road networks. The primary goal is to assess the performance of four distinct HMM-based methods. A simple benchmark method is also included to provide context. The experiments are divided into three parts: a parameter search used to identify good parameter choices for the transition probability and emission probability, a part dedicated to the estimation of transition probabilities, and a part dedicated to performance evaluation. The quality of the route estimates is assessed using the Hausdorff distance because of its ability to quantify the degree of correctness for the route estimates. The results of the experiments show that one achieves substantial performance gains by moving from the simple to the augmented state space. This is especially evident in the case with high sampling frequency and low variance, where we observe a 90.7% decrease in Hausdorff distance for the new augmented state space HMM approach when compared to the approach with a simple state space.

Sammendrag

Kartmatching refererer til prosessen der en sekvens med posisjonsmålinger brukes til å estimere en sammenhengende rute på et veinettverk. Posisjonsmålinger som brukes til kartmatching er typisk innhentet ved hjelp av en GPS-enhet som befinner seg i et kjøretøy som beveger seg gjennom veinettverket. Formålet med denne avhandlingen er å utforske og evaluere flere forskjellige "skjult Markov modell"-formuleringer (SMM-formuleringer) som muliggjør kartmatching. Disse metodene krever kjennskap til det underliggende veinettverket. Dette veinettverket blir brukt til å konstruere et tilstandsrom hvor hver tilstand representerer et veisegment.

Det sørges for at de SMM-baserte kartmatching-metodene overholder antagelsene i en SMM. Tilstandsestimatet man får fra Viterbi-algoritmen kan derfor anses som "den mest sannsynlige sekvensen av veisegmenter". Ett av bidragene i denne avhandlingen er kombineringen av idéer fra tidligere verker på en måte som tilfredsstiller antagelsene i en SMM. Et annet bidrag kommer gjennom introduksjonen av et tilstandsrom som, så vidt forfatteren vet, ikke har blitt brukt før. Denne utvidede tilstandsrom-formuleringen forstørrer tilstandsrommet ved å inkorporere tilleggselementer som blant annet inneholder informasjon om bevegelsesretningen. Bruken av Baum-Welch-algoritmen er også trolig et betydelig bidrag, siden den ikke har blitt brukt til dette spesifikke problemet tidligere.

Eksperimentene som presenteres i denne avhandlingen er utført ved bruk av data fra simulert bevegelse på reelle veinettverk. Hovedformålet er å evaluere ytelsen til fire distinkte SMM-baserte metoder. En naiv sammenligningsmetode er også inkludert for å tilføre kontekst. Eksperimentene er delt inn i tre deler: et parametersøk som brukes til å identifisere gode parametervalg for overgangs- og emisjonssannsynlighetene, en del dedikert til estimering av overgangssannsynligheter og en del dedikert til prestasjonsevaluering. Kvaliteten på ruteestimatene blir vurdert ved hjelp av Hausdorff-avstanden på grunn av dens evne til å kvantifisere ruteestimatets grad av korrekthet. Resultatene fra eksperimentene viser at man oppnår en betydelig økning i prestasjon ved å gå fra det enkle til det utvidede tilstandsrommet. Dette er spesielt tydelig i situasjonen med høy samplingsfrekvens og lav varians, hvor man oppnår 90.7% reduksjon i Hausdorff-avstand når SMM-tilnærmelsen med utvidet tilstandsrom sammenlignes med tilnærmelsen hvor det enkle tilstandsrommet brukes.

Acknowledgements

I want to direct many thanks to my supervisor for this master's thesis, Professor Jo Eidsvik at the Norwegian University of Science and Technology (NTNU). His insights and advice have been instrumental in shaping this thesis. The lecturers and teaching assistants that have contributed to my academic development also deserve thanks. Finally, I want to thank my parents for their continuous support throughout my education.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

Map matching can provide value to people and organizations by enabling them to estimate travel routes using nothing but a GPS receiver. An estimated route can be valuable by itself, as when used for tracking individual vehicles, or in a group where the group is used to gain insights about the overall behaviour of some population. One of the most compelling use cases relates to automatic taxation of vehicular travel based on, for instance, distance travelled, areas visited, and roads that are taken. Another application could be analysis of traffic flow in some road network. Conducting an analysis of the behaviour of vehicles moving through a road network is complicated without translating the measured positions into actual routes. By using route estimates instead of measured positions, one makes it easier to obtain information about traversal on the underlying network. An overall view of the behaviour of vehicles on a specific road network could, for instance, be valuable to decision-makers when planning for the future of the road network.

The author's initial exposure to the problem of map matching was through a representative of the Norwegian company *DEFA* [1]. They had an interest in improving the tracking and route estimation capabilities of their smartphone app. This resulted in a project that explored the benefits of simple methods based on hidden Markov models (HMMs) when compared to more naive methods. This project was finalized prior to the summer of 2019. In late summer 2019 we came into contact with *The Institute of Transport Economics* (TØI) [2], which is a "national institution for transport research and development" in Norway. Even though there are several data privacy issues that hampers the use of TØI's position data in a MSc thesis, they expressed an interest in learning more about map matching approaches and how they could be applied to data in the possession of TØI. This served as an affirmation that further exploration was of interest to researchers within TØI, and likely other researchers at similar institutions as well, and that there was a genuine use case for the capabilities provided by HMM-based map matching methods.

[1] https://www.defa.com/

[2] https://www.toi.no/

1.2 Previous work

The work presented in Newson and Krumm [1] is an influential and frequently cited paper on map matching using HMMs. This paper presents how one can create a map matching algorithm that enables integration of noisy measurements and road network restrictions by formulating the problem as an HMM. The state space is comprised of road segments that form connections between intersections. Information about the layout and restrictions of the road network is introduced into the model by defining transition probabilities that incorporate information about the distance between various road segments. The transition probability is somewhat involved, but the idea is that the distance travelled on the road network between two measurements is close to the distance between the two measurements. The noisy measurements are accounted for by assuming that the great circle distance between a position measurement and road segment is normally distributed around zero. This enables map matching using the *Viterbi algorithm,* with the output being a sequence of road segments. The authors of the paper collected GPS data and ground truth by driving a "known, planned route" with a GPS device located inside the vehicle. The algorithm is shown to perform very well on this specific route and was able to perfectly estimate the route travelled. It is worth noting that, although the authors refer to the method as "using HMMs", the assumptions of an HMM is violated by letting the transition probability depend on future observations. This does in no way discredit their work, but it would be more precise to describe the method as "based on the ideas of an HMM", or "inspired by HMMs". It also has the effect of making the theoretical properties of an HMM invalid for this method, which is an important note, seeing as the authors claim that the "goal of the algorithm is to find the most probable path". This outcome can only be guaranteed if the assumptions of the HMM hold.

The violation of the assumptions of an HMM by the method presented in Newson and Krumm [1] is noted in Raymond *et al.* [2]. The paper proceeds by introducing an HMM-based map matching algorithm that satisfies the assumptions of an HMM. This method uses the set of nodes in the graph representing the road network as its state space. The measurements are assumed to be normally distributed around the "true" node, while the distance between subsequent nodes is assumed to be exponentially distributed. The performance is shown to be comparable to what is achieved in Newson and Krumm [1].

The earliest paper found on map matching using HMMs is Hummel [3]. The HMM in this paper is, as in later work, constructed by using the road segments as the state space. The emission probabilities are defined by assuming a zero-mean normal distribution for the distance between an observation and the road segment one is on. The paper also assumes that information about the heading of the vehicle is present. This is incorporated into the model by assuming a zero-mean normal distribution for the difference between the orientation of the road segment and the heading of the vehicle. The transition probability is constructed so that, when leaving a state, all directly connected states are equally likely to follow. All

other states are assigned probability zero. A few additional extensions are also suggested. This includes changing the state space to include a flag denoting the travel direction, as well as a change to the transition probabilities that gives U-turns a low probability. It is reported that an "error-free route estimate" is obtained for real-world data, but there is no information about the exact methodology used for evaluation.

Several other papers dedicated to this topic are focused on adaptations that lower the computational cost of the methods. A notable example is Yang and Gidófalvi [4], where candidate search is done using a combination of the k-nearest neighbour algorithm and R-trees. The possible candidates are the individual segments of the road network. This is combined with the use of a hash table that gives quick access to shortest paths. This, in addition to other changes and adaptions such as penalization of backtracking, results in a method that is both efficient and well-performing. We do, however, again note that the resulting method does not satisfy the assumptions of an HMM since the transition probabilities depend on future observations.

1.3 Overview

This section is an overview of what will be presented in the various chapters of the thesis. Chapter 2 is a detailed presentation of the theory behind HMMs. This includes material on the methods and techniques that make it possible to use HMMs for map matching. The chapter starts by introducing the concept of a Markov chain. We proceed by providing a general definition of an HMM. This includes the introduction of an observation sequence and a hidden state sequence, along with assumptions about the sequences and how they relate to each other. This is followed by the introduction of the *forward algorithm*, the *backward algorithm*, the *Viterbi algorithm*, and finally the expectation-maximization (EM) algorithm for HMMs (also known as the *Baum-Welch algorithm*). The *forward algorithm* enables us to compute the probability of a certain observation sequence (under the assumptions of the HMM). The *backward algorithm* is used for computing quantities required for the parameter estimation enabled by EM-algorithm. The *Viterbi algorithm* enables us to obtain the maximum a posteriori (MAP) estimate of the state sequence of the HMM. The last section of the chapter is devoted to issues related to scaling. The issue of scaling becomes important during the implementation of the various algorithms.

In Chapter 3, we will move from the general formulation of an HMM to the specific formulation that allows us to do map matching. Defining the components of the HMM requires that we have established the context the problem exists in. This is done in the first part of the chapter, where we formally define the problem. This does, among other things, include representing the road network as a graph and defining the observation sequence as a sequence of points in some coordinate system. Constructing a suitable HMM requires a state space that can somehow represent the position of the object moving through the road network. One must

also define transition probabilities, emission probabilities, and initial probabilities that are consistent with and sensible for the specific state space. We will explore two distinct state spaces. The first one has been used in previous work but does not incorporate information about the travel direction. We, therefore, propose a new state space formulation that incorporates travel direction by defining the states such that information about a preceding location is included. This is followed by an explanation of how the *Baum-Welch algorithm* can be applied to our problem and how the resulting parameter estimates can be interpreted. We then present how route estimates can be obtained from the state estimates found using the *Viterbi algorithm*. The chapter is closed off with a discussion of how we evaluate the performance of our method when the ground truth is known.

In Chapter 4, we introduce the set-up of our experiments. This includes a discussion of the data sources and how we can use these to create a road network. We proceed by giving a presentation of the route- and measurement simulation procedure. This is followed by a relatively concise overview of the experiments that will be conducted. The next sections present the experiments in detail. The first section introduces the details of the parameter search. The purpose of the parameter search is to find parameter values that maximize the performance of the map matching methods. We move on to presenting how the *Baum-Welch algorithm* is used for learning the transition probabilities of the model. The section is concluded with a presentation of how the final performance evaluation is performed. The chapter ends with a discussion of the software developed for this thesis. The code presented here has been open-sourced and can be found on the author's personal GitHub profile [3]. Parts of it has also been published as a package, and is available under the name `hmmpy` on the Python Package Index (PyPI) [4].

Chapter 5 contains the results of the experiments presented in Chapter 4. The results are discussed in Chapter 6 before the thesis is concluded with a final summary of the thesis and a discussion of potential further work.

[3] https://github.com/klaapbakken
[4] https://pypi.org/project/hmmpy/

Chapter 2

Theory

2.1 Markov processes

Let $\mathbf{X} = \{X_i\}_{i=1}^N$ be a finite sequence of random variables taking on values in the space $\mathcal{X}$. Such a sequence is often referred to as a stochastic process on $\mathcal{X}$, with $\mathcal{X}$ being known as the state space of the stochastic process. If the state space is discrete we will refer to the stochastic process as a Markov chain [5]. A Markov chain is a k-th order discrete-time Markov chain if the time between transitions is fixed and the Markov property,

$$\begin{aligned} &\mathrm{P}\left(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\right) \\ &= \mathrm{P}\left(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_{n-k} = x_{n-k}\right), \end{aligned} \tag{2.1}$$

holds for all $n \in \mathcal{N} = \{1, \ldots, N\}$.

In the following sections we will exclusively be working with first-order Markov chains. A Markov chain is said to time-homogeneous if, for all $(x, y) \in \mathcal{X}^2$, we have that $\mathrm{P}(X_n = x \mid X_{n-1} = y)$ is independent of n. In this case the transition probabilities will be denoted by $a(y, x) = \mathrm{P}(X_n = x \mid X_{n-1} = y)$. We will assemble these transition probabilities in a transition matrix $\mathbf{A}$ by associating each state with an index from the index set $\mathcal{I} = \{1, \ldots, M\}$, where $M = |\mathcal{X}|$ is the cardinality of $\mathcal{X}$. We will let $x^{(i)}$ denote the state identified by index $i \in \mathcal{I}$. The element in position (i, j) of the transition matrix $\mathbf{A}$ is given by $a_{ij} = \mathrm{P}\left(X_n = x^{(j)} \mid X_{n-1} = x^{(i)}\right)$. In order to fully describe the Markov chain we also require a probability distribution for the initial state. This is known as the initial probability and will be denoted by either $\pi(x) = \mathrm{P}(X_1 = x)$ or $\pi_i = \mathrm{P}\left(X_1 = x^{(i)}\right)$, depending on the situation.

The probability of a realization of a Markov chain, denoted by $\mathbf{x} = \{x_i\}_{i=1}^N$, is given by

$$\mathrm{P}(\mathbf{X} = \mathbf{x}) = \pi(x_1) \prod_{i=2}^{N} a(x_{i-1}, x_i). \tag{2.2}$$

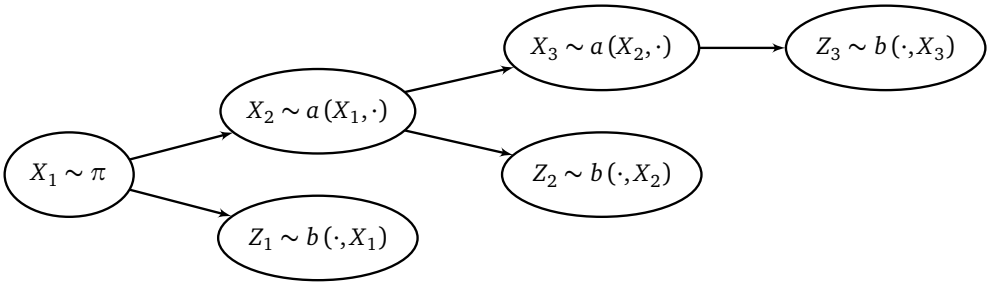


Figure 2.1: An illustration of an HMM. The graph shows both the latent variables and the observations, along with information about the probability distribution that they arise from.

2.2 Hidden Markov models

Let $\mathbf{X} = \{X_i\}_{i=1}^N$ be a time-homogeneous Markov chain, as defined in Section 2.1. Let $\mathbf{Z} = \{Z_i\}_{i=1}^N$ be another sequence of random variables. In a hidden Markov model (HMM) one assumes that, for each $n \in \mathcal{N}$, the random variable Z_n has a distribution that depends on the unobserved random variable X_n. These unobserved random variables are often referred to as the hidden states or the latent variables. The sequence $\mathbf{Z}$ will be referred to as the observations. We will for the remainder of this chapter treat each $Z_n, n \in \mathcal{N}$, as a discrete random variable. This does not affect the forthcoming derivations in any significant way, seeing as the approach is similar for a continuous state space. It is done purely to avoid notational inconvenience. The random variable Z_n is, given X_n, $n \in \mathcal{N}$, assumed to be conditionally independent of all other variables, both hidden and observed [6] [7]. We have

$$\begin{aligned} \mathrm{P}(Z_n = z \mid \mathbf{X} = \mathbf{X}, \mathbf{Z}_{1:n-1} = \mathbf{z}_{1:n-1}, \mathbf{Z}_{n+1:N} = \mathbf{z}_{n+1:N}) = \\ \mathrm{P}(Z_n = z \mid X_n = x_n), \end{aligned} \tag{2.3}$$

where $\mathbf{Z}_{i:j}, i \leq j, i, j \in \mathcal{N}$ denotes the subsequence $\{Z_i, \ldots, Z_j\} \subseteq \mathbf{Z}$. The relation between the observed variables $\mathbf{Z}$ and the hidden states $\mathbf{X}$ is, for each $n \in \mathcal{N}$, given by some conditional probability distribution

$$\mathrm{P}(Z_n = z \mid X_n = x) = b(z, x). \tag{2.4}$$

This conditional probability distribution is known as the emission probability of the HMM. In the case of a continuous $Z_n, n \in \mathcal{N}$, one would swap the probability mass function defining $b(z, x)$ for the appropriate probability density function. In Figure 2.1 we show the various components of the HMM and how they fit together. The probability of a realized observation sequence $\mathbf{z}$ when given the state sequence $\mathbf{x}$ is the product of the conditional marginal distributions.

$$\mathrm{P}(\mathbf{Z} = \mathbf{z} \mid \mathbf{X} = \mathbf{x}) = \prod_{i=1}^{N} \mathrm{P}(Z_i = z_i \mid X_i = x_i) = \prod_{i=1}^{N} b(z_i, x_i). \tag{2.5}$$

2.3 Forward algorithm

Consider an HMM as defined in Section 2.2. The probability of an arbitrary observation sequence $\mathbf{z}$ is given by

$$\mathrm{P}(\mathbf{Z}=\mathbf{z}) = \sum_{\mathbf{x}\in\mathcal{X}^N} \mathrm{P}(\mathbf{Z}=\mathbf{z} \mid \mathbf{X}=\mathbf{x})\,\mathrm{P}(\mathbf{X}=\mathbf{x}). \tag{2.6}$$

In general, evaluating this requires $2N \cdot M^N$ calculations, which is computationally infeasible for large values of M and N. It is therefore desirable to use an alternative method for evaluating this probability, one that leverages the conditional independence assumptions of the HMM. The preferred method is known as the *forward algorithm* and allows us to evaluate the probability using only M^2N calculations [8]. This is done by computing only the required quantities for every time-step using a recursive relation. We will now turn our attention to derivation of the mentioned recursive relation.

By rewriting (2.6) using (2.5) and (2.2) we get

$$\begin{aligned}\mathrm{P}(\mathbf{Z}=\mathbf{z}) &= \sum_{\mathbf{x}\in\mathcal{X}^N} \prod_{i=1}^{N} b(z_i, x_i)\,\mathrm{P}(\mathbf{X}=\mathbf{x}) \\ &= \sum_{\mathbf{x}\in\mathcal{X}^N} \left[\pi(x_1)\, b(z_1, x_1) \prod_{i=2}^{N} b(z_i, x_i)\, a(x_{i-1}, x_i) \right].\end{aligned} \tag{2.7}$$

The probability of being in state x at time n while having observed the sequence $\mathbf{z}_{1:n}$ up until that point is denoted by

$$\alpha_n(x) = \mathrm{P}(X_n = x, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}). \tag{2.8}$$

The definition of conditional probability [5] states that, for two random variables X and Y,

$$\mathrm{P}(X=x, Y=y) = \mathrm{P}(X=x \mid Y=y)\,\mathrm{P}(Y=y). \tag{2.9}$$

Using this we can write $\alpha_1(x)$ as

$$\alpha_1(x) = \mathrm{P}(Z_1 = z_1 \mid X_1 = x)\,\mathrm{P}(X_1 = x) = b(z_1, x)\,\pi(x). \tag{2.10}$$

By conditioning on the event $X_n = y$, and subsequently applying the definition of conditional probability, we get, for $1 \leq n < N$,

$$\begin{aligned}\alpha_{n+1}(x) &= \sum_{y\in\mathcal{X}} \mathrm{P}(\mathbf{Z}_{1:n+1} = \mathbf{z}_{1:n+1}, X_n = y, X_{n+1} = x) \\ &= \sum_{y\in\mathcal{X}} \mathrm{P}(Z_{n+1} = z_{n+1} \mid \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}, X_n = y, X_{n+1} = x) \\ &\quad \cdot \mathrm{P}(\mathbf{Z}_{1:n} = \mathbf{z}_{1:n}, X_n = y, X_{n+1} = x).\end{aligned} \tag{2.11}$$

By utilizing the conditional independence of Z_{n+1} given X_{n+1}, the Markov property (2.1), and the definition of conditional probability (2.9) we arrive at the recursive relation between $\alpha_n(x)$ and $\alpha_{n+1}(x)$, given by

$$\begin{aligned}\alpha_{n+1}(x) = &\sum_{y\in\mathcal{X}} \mathrm{P}\left(Z_{n+1} = z_{n+1} \mid X_{n+1} = x\right) \\ &\cdot \mathrm{P}\left(X_{n+1} = x \mid X_n = y, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \\ &\cdot \mathrm{P}\left(X_n = y, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \\ &= b\left(z_{n+1}, x\right) \sum_{y\in\mathcal{X}} a\left(y, x\right) \alpha_n(y).\end{aligned} \tag{2.12}$$

This results in Algorithm 1, which outlines the various steps that enable the computation of $\alpha_n(x^{(i)}) \,\forall\, (n,i) \in \mathcal{N}\times\mathcal{I}$. The probability of the event $\mathbf{Z} = \mathbf{z}$ can

Algorithm 1: Forward algorithm for calculating $\alpha_n(x^{(i)})$ for $(n,i) \in \mathcal{N}\times\mathcal{I}$.

for $i = 1{:}M$ **do**
  $\alpha_1(x^{(i)}) = b\left(z_1, x^{(i)}\right)\pi\left(x^{(i)}\right)$
end for
for $n = 1{:}N\text{-}1$ **do**
  for $i = 1{:}M$ **do**
    $\alpha_{n+1}(x^{(i)}) = b\left(z_n, x^{(i)}\right)\sum_{j=1}^{M} a\left(x^{(j)}, x^{(i)}\right)\alpha_n(x^{(j)})$
  end for
end for

now be computed by realizing that

$$\begin{aligned}\sum_{x\in\mathcal{X}} \alpha_N(x) &= \sum_{x\in\mathcal{X}} \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_N = x\right) \\ &= \sum_{x\in\mathcal{X}} \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N} \mid X_N = x\right)\mathrm{P}\left(X_N = x\right) \\ &= \mathrm{P}\left(\mathbf{Z} = \mathbf{z}\right)\end{aligned} \tag{2.13}$$

2.4 Backward algorithm

We will now introduce another method, known as the *backward algorithm* [8], that enables computation of the quantity

$$\beta_n(x) = \mathrm{P}\left(\mathbf{Z}_{n+1:N} = \mathbf{z}_{n+1:N} \mid X_n = x\right). \tag{2.14}$$

The motivation for computing $\beta_n(x)$ will become clear in Section 2.6, where we apply the expectation-maximization (EM) algorithm in order to learn the parameters of the HMM. The *backward algorithm* is, like the *forward algorithm*, based on a recursive relation for computing the desired quantity.

In the following derivation we rely on two important results from basic probability theory. The first result is Bayes' theorem [5],

$$\mathrm{P}(X = x \mid Y = y) = \frac{\mathrm{P}(Y = y \mid X = x)\mathrm{P}(X = x)}{\mathrm{P}(Y = y)}. \tag{2.15}$$

The following result can be derived from the definition of conditional probability (2.9).

$$\mathrm{P}(X = x, Y = y \mid Z = z) = \mathrm{P}(X = x \mid Y = y, Z = z)\mathrm{P}(Y = y \mid Z = z) \tag{2.16}$$

We now proceed to the derivation of the recursive relation.

$$\begin{aligned}
\beta_n(x) &= \mathrm{P}(\mathbf{Z}_{n+1:N} = \mathbf{z}_{n+1:N} \mid X_n = x) \\
&= \frac{\mathrm{P}(\mathbf{Z}_{n+1:N} = \mathbf{z}_{n+1:N}, X_n = x_n)}{\mathrm{P}(X_n = x)} \\
&= \frac{\sum_{y\in\mathcal{X}} \mathrm{P}(Z_{n+1} = z_{n+1}, \mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N}, X_n = x \mid X_{n+1} = y)\mathrm{P}(X_{n+1} = y)}{\mathrm{P}(X_n = x)} \\
&= \sum_{y\in\mathcal{X}} \Bigg[\frac{\mathrm{P}(Z_{n+1} = z_{n+1} \mid \mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N}, X_n = x, X_{n+1} = y)}{\mathrm{P}(X_n = x)} \\
&\quad \cdot \mathrm{P}(\mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N}, X_n = x \mid X_{n+1} = y)\mathrm{P}(X_{n+1} = y) \Bigg] \\
&= \sum_{y\in\mathcal{X}} \Bigg[b(z_{n+1}, y)\mathrm{P}(\mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N} \mid X_n = x, X_{n+1} = y) \\
&\quad \cdot \frac{\mathrm{P}(X_n = x \mid X_{n+1} = y)\mathrm{P}(X_{n+1} = y)}{\mathrm{P}(X_n = x)} \Bigg] \\
&= \sum_{y\in\mathcal{X}} b(z_{n+1}, y)\beta_{n+1}(y)a(x, y)
\end{aligned} \tag{2.17}$$

The path to this result is started by conditioning on $X_{n+1} = y$ and summing over all $y \in \mathcal{X}$. We proceed by repeatedly applying the laws of conditional probability and then recalling that Z_{n+1} is independent of all other variables when X_{n+1} is given. The final step is a direct application of Bayes theorem.

This result gives rise to the *backward algorithm*, see Algorithm 2, which enables us to compute $\beta_n(x^{(i)})$ for $(n, i) \in \mathcal{N} \times \mathcal{I}$.

2.5 Viterbi algorithm

One of the primary objectives when working with HMMs is the estimation of the latent variables when given a sequence of observations. This is typically done by finding the sequence of latent variables that maximizes the posterior probability

Algorithm 2: Backward algorithm for calculating $\beta_n(x^{(i)})$ for $(n,i) \in \{1,\dots,N\} \times \mathcal{I}$

for $i = 1{:}M$ **do**
  $\beta_N(x^{(i)}) = 1$
end for
for $n = N\text{-}1{:}\text{-}1{:}1$ **do**
  for $i = 1{:}M$ **do**
    $\beta_n(x^{(i)}) = \sum_{j=1}^{M} b\left(z_{n+1}, x^{(j)}\right) \beta_{n+1}(x^{(j)}) a\left(x^{(i)}, x^{(j)}\right)$
  end for
end for
.

$\mathrm{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{Z} = \mathbf{z})$. The method of choice for accomplishing this is known as the *Viterbi algorithm* [8] [9] [10]. We will refer to the resulting estimate as the maximum a posteriori (MAP) estimate and denote it by

$$\mathbf{x}^\star = \arg\max_{\mathbf{x} \in \mathcal{X}^N} \mathrm{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{Z} = \mathbf{z}). \tag{2.18}$$

We note that

$$\mathbf{x}^\star = \arg\max_{\mathbf{x} \in \mathcal{X}^N} \frac{\mathrm{P}(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z})}{\mathrm{P}(\mathbf{Z} = \mathbf{z})} = \arg\max_{\mathbf{x} \in \mathcal{X}^N} \mathrm{P}(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}). \tag{2.19}$$

There are also other ways of estimating the latent variables. One possible approach is to, for each $n \in \mathcal{N}$, choose the state x that maximizes $b(z_n, x)$. By doing this one ignores the information about the probability of various state transitions, and the MAP estimate is therefore usually preferred. This method can, however, be useful for benchmarking and baseline estimates.

The *Viterbi algorithm* can be understood intuitively by thinking about movement through a lattice, as seen in Figure 2.2. Let us say that we are in state x at time n. There are now M possible candidates that we can move to for time $n+1$, with all transitions having a specified transition probability. The lattice is constructed by considering all possible transitions for all states at each time step. The *Viterbi algorithm* works by utilizing how the most likely path to certain states evolves. Let $\delta_n(x)$ be the probability of the most likely sequence of states ending in state x at time n. In other words,

$$\delta_n(x) = \max_{\mathbf{x}_{1:n-1}} \mathrm{P}(\mathbf{X}_{1:n-1} = \mathbf{x}_{1:n-1}, X_n = x, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}). \tag{2.20}$$

By using this we can find the recursive relation

$$\begin{aligned}\delta_{n+1}(y) &= \max_{\mathbf{x}_{1:n}} \mathrm{P}\left(\mathbf{X}_{1:n} = \mathbf{x}_{1:n}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}, X_{n+1} = y, Z_{n+1} = z_{n+1}\right) \\ &= \max_{x} [\mathrm{P}\left(X_{n+1} = y, Z_{n+1} = z_{n+1} \mid X_n = x\right) \\ &\cdot \max_{\mathbf{x}_{1:n-1}} \mathrm{P}\left(\mathbf{X}_{1:n-1} = \mathbf{x}_{1:n-1}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}, X_n = x\right)] \\ &= \max_{x} [\mathrm{P}\left(Z_{n+1} = z_{n+1} \mid X_{n+1} = y\right) \mathrm{P}\left(X_{n+1} = y \mid X_n = x\right) \delta_n(x)] \\ &= \max_{x} [\delta_n(x) a\left(x, y\right)] \, b\left(z_{n+1}, y\right).\end{aligned} \tag{2.21}$$

We arrived at this relation by using (2.9) for the first step and (2.16) for the second step.

The procedure can be started by realizing that $\delta_1(x) = \mathrm{P}\left(X_1 = x, Z_1 = z_1\right) = \alpha_1(x)$, as defined in Equation (2.8). To find the most likely state sequence, and not just the probability of it, we introduce the quantity

$$\psi_n(x) = \arg\max_{y} [\delta_{n-1}(y) a\left(y, x\right)]. \tag{2.22}$$

Calculating $\psi_n(x) \, \forall \, x \in \mathcal{X}$ at every time $2 \leq n \leq N$ allows us to find the preceding state when knowing the state that follows it. The last state in the state sequence is the one that maximizes $\delta_N(x)$. One can then find the most likely path in its entirety by evaluating $\psi_n(x)$ at the various time steps. In Figure 2.2 we show the lattice with dotted lines moving from left to right. These dotted lines indicate that, at each time step, $\delta_{n+1}(x)$ is computed by considering all the transitions that lead to state x. Upon reaching the last time-step, seen on the right in Figure 2.2, one identifies the state x that maximizes $\delta_N(x)$. By using the quantities $\psi_n(x)$, $(n, i) \in \mathcal{N} \times \mathcal{I}$, we can now backtrack to find the sequence of states that maximizes $\mathrm{P}\left(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}\right)$. This is, as we recall, the MAP estimate $\mathbf{x}^\star$. The backtracking is indicated by the solid lines going from right to left in Figure 2.2.

This leads us to Algorithm 3, the *Viterbi algorithm*. This algorithm is not numerically stable for large values of N. This is due to the fact that $\delta_n(x) a\left(x, x^{(i)}\right)$ approaches zero as N increases, and the terms can therefore not be accurately represented in computations. It is, however, straightforward to take the logarithm of all the involved quantities and do all computations on a logarithmic scale instead. This does not affect the correctness since $\log(x)$ is strictly increasing. The alternative implementation is provided in Algorithm 4.

2.6 Expectation-maximization for HMMs

A procedure for estimating the unknown parameters of an HMM can be obtained by using the expectation-maximization (EM) algorithm. The general formulation of the EM algorithm involves finding an expression for the expected value of the log-likelihood of the complete data when given the incomplete data and the current parameter estimates. This is known as the E-step. The parameter estimates are

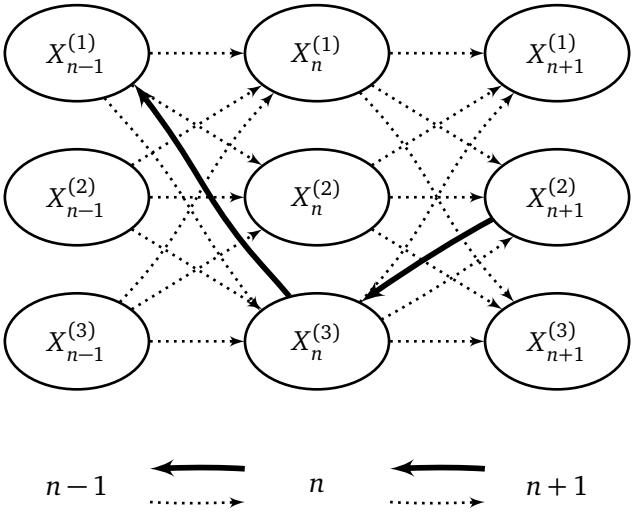


Figure 2.2: An illustration showing how the Viterbi algorithm computes the required quantities by moving through a lattice. Forward movement is indicated by dotted lines, while backtracking to find the MAP estimate is indicated by solid lines.

Algorithm 3: Viterbi algorithm for calculating MAP estimate.

for $i = 1{:}M$ **do**
  $\delta_1(x^{(i)}) = b\left(z_1, x^{(i)}\right)\pi\left(x^{(i)}\right)$
  $\psi_1(x^{(i)}) = 0$
end for
for $n = 1{:}N$ **do**
  for $i = 1{:}M$ **do**
    $\delta_n(x^{(i)}) = \max_x\left[\delta_n(x)a\left(x, x^{(i)}\right)\right] b\left(z_{n+1}, x^{(i)}\right)$
    $\psi_n(x^{(i)}) = \arg\max_x \delta_{n-1}(x)a\left(x, x^{(i)}\right)$
  end for
end for
for $n = N{:}{-}1{:}{-}1{:}1$ **do**
  $x_n^\star = \psi_{n+1}(x_{n+1}^\star)$
end for

Algorithm 4: Numerically stable version of the Viterbi algorithm.

for $i = 1{:}M$ **do**
  $\log\left(\delta_1\left(x^{(i)}\right)\right) = \log\left(b\left(z_1, x^{(i)}\right)\right) + \log\left(\pi\left(x^{(i)}\right)\right)$
  $\psi_1\left(x^{(i)}\right) = 0$
end for
for $n = 1{:}N$ **do**
  for $i = 1{:}M$ **do**
    $\log\left(\delta_n\left(x^{(i)}\right)\right) =$
     $\max_x\left[\log\left(\delta_n(x)\right) + \log\left(a\left(x, x^{(i)}\right)\right)\right] + \log\left(b\left(z_{n+1}, x^{(i)}\right)\right)$
    $\psi_n\left(x^{(i)}\right) = \arg\max_x\left[\log\left(\delta_{n-1}(x)\right) + \log\left(a\left(x, x^{(i)}\right)\right)\right]$
  end for
end for
for $n = N{\text{-}}1{:}{-}1{:}1$ **do**
  $x_n^\star = \psi_{n+1}\left(x_{n+1}^\star\right)$
end for

then updated by finding the parameter values that maximize the expression found in the E-step. This is known as the M-step. It can be shown that by repeatedly doing the E-step and the M-step one will converge to a local maxima of the incomplete likelihood [11]. The EM algorithm is, when applied to HMMs, often referred to as the *Baum-Welch algorithm* [8] [12] [13]. The unknown parameters can, in the case of an HMM, include the transition probabilities between the different states, the initial probabilities, and the emission probabilities connecting the latent variables to the observations. In the following we denote the collection of such unknown parameters by λ.

The E-step involves finding an expression for the expected complete-data log-likelihood with respect to the unobserved data when given the observed data and the current parameter estimates. For a HMM it is very natural to use the observations, $\mathbf{Z} = \{Z_i\}_{i=1}^N$, as the observed data, while the unobserved data is the state sequence, $\mathbf{X} = \{X_i\}_{i=1}^N$. The resulting expression that has to be maximized with respect to λ is

$$Q(\lambda, \lambda^{(i-1)}) = \mathrm{E}\left[\log p(\mathbf{z}, \mathbf{X}; \lambda) \mid \mathbf{z}, \lambda^{(i-1)}\right]. \tag{2.23}$$

This expression can be rewritten as

$$Q(\lambda, \lambda^{(i-1)}) = \sum_{\mathbf{x} \in \mathcal{X}^N} \log p(\mathbf{z}, \mathbf{x}; \lambda) \cdot p(\mathbf{x} \mid \mathbf{z}; \lambda^{(i-1)}), \tag{2.24}$$

where $p(\mathbf{z}, \mathbf{x})$ and $p(\mathbf{x} \mid \mathbf{z})$ is shorthand for $\mathrm{P}(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z})$ and $\mathrm{P}(\mathbf{X} = x \mid \mathbf{Z} = z)$ respectively. We specify the parameters in use whenever clarity regarding this is required.

The M-step involves finding the value of λ that maximizes $Q(\lambda, \lambda^{(i-1)})$ and

updating the parameter estimates to this value. In other words,

$$\lambda^{(i)} = \arg\max_{\lambda} Q(\lambda, \lambda^{(i-1)}). \tag{2.25}$$

We start off by finding an expression for $p(\mathbf{z}, \mathbf{x})$ in terms of already known quantities. This can be done by remembering that the observations, at time n, only depend on X_n. We also recall that X_n only depends on the previous state X_{n-1}. We combine this with the definition of conditional probability in Equation (2.9) to end up with

$$p(\mathbf{z}, \mathbf{x}) = \mathrm{P}(\mathbf{Z} = \mathbf{z}, \mathbf{X} = \mathbf{x}) = \pi(x_1)\, a(x_1, x_2) \prod_{i=2}^{N} a(x_{i-1}, x_i)\, b(z_i, x_i). \tag{2.26}$$

Taking the logarithm of this leaves us with

$$\log p(\mathbf{z}, \mathbf{x}) = \log \pi(x_1) + \sum_{i=1}^{N} \log b(z_i, x_i) + \sum_{i=2}^{N} \log a(x_{i-1}, x_i). \tag{2.27}$$

The next objective is to find the parameters $\lambda^{(i)}$ that maximize $Q(\lambda, \lambda^{(i-1)})$ with respect to λ. We note that, since

$$p(\mathbf{x} \mid \mathbf{z}; \lambda^{(i-1)}) = \frac{p(\mathbf{x}, \mathbf{z}; \lambda^{(i-1)})}{p(\mathbf{z}; \lambda^{(i-1)})}, \tag{2.28}$$

we get

$$\begin{aligned}
\lambda^{(i)} &= \arg\max_{\lambda} \sum_{\mathbf{x} \in \mathcal{X}^N} \log p(\mathbf{z}, \mathbf{x}; \lambda) p(\mathbf{x} \mid \mathbf{z}, \lambda^{(i-1)}) \\
&= \arg\max_{\lambda} \sum_{\mathbf{x} \in \mathcal{X}^N} \log p(\mathbf{z}, \mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z} \mid \lambda^{(i-1)})}{p(\mathbf{z} \mid \lambda^{(i-1)})} \\
&= \arg\max_{\lambda} \sum_{x \in \mathcal{X}^N} \log p(\mathbf{z}, \mathbf{x}) p(\mathbf{x}, \mathbf{z} \mid \lambda^{(i-1)}) \\
&= \arg\max_{\lambda} \hat{Q}(\lambda, \lambda^{(i-1)}).
\end{aligned} \tag{2.29}$$

This can be done because $p(\mathbf{z} \mid \lambda^{(i-1)})$ is constant and does not depend on λ. In order to maximize $Q(\lambda, \lambda^{(i-1)})$ it is therefore sufficient to maximize $\hat{Q}(\lambda, \lambda^{(i-1)})$. The resulting problem is a constrained optimization problem, with constraints ensuring that the transition, emission and initial probabilities satisfy the requirements of a probability distribution. The constraints related to the initial and transition probabilities are $\sum_{i=1}^{M} \pi\left(x^{(i)}\right) = 1$ and $\sum_{j=1}^{M} a\left(x^{(i)}, x^{(j)}\right) = 1 \, \forall \, i \in \mathcal{I}$. The constraints related to the emission probabilities vary depending on the assumptions of the HMM. In our case we will treat the emission probabilities as fixed, and can therefore avoid the problem entirely.

Before we continue we want to find suitable expressions for two quantities that will be required at a later stage in this section. The first quantity of interest is

$$\mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}\right).$$

This can be rewritten as

$$\begin{aligned}
&\mathrm{P}\left(X_n = x^{(i)}, X_{n+1} = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}\right) \\
&= \mathrm{P}\left(Z_{n+1} = z_{n+1}, \mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N} \mid X_{n+1} = x^{(j)}, X_n = x^{(i)}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \\
&\quad \cdot \mathrm{P}\left(X_{n+1} = x^{(j)}, X_n = x^{(i)}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \\
&= \mathrm{P}\left(Z_{n+1} = z_{n+1} \mid X_{n+1} = x^{(j)}, X_n = x^{(i)}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}, \mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N}\right) \\
&\quad \cdot \mathrm{P}\left(\mathbf{Z}_{n+2:N} = \mathbf{z}_{n+2:N} \mid X_{n+1} = x^{(j)}, X_n = x^{(i)}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \\
&\quad \cdot \mathrm{P}\left(X_{n+1} = x^{(j)} \mid X_n = x^{(i)}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \cdot \mathrm{P}\left(X_n = x^{(i)}, \mathbf{Z}_{1:n} = \mathbf{z}_{1:n}\right) \\
&= b\left(z_{n+1}, x^{(j)}\right) \beta_{n+1}(x^{(j)}) a\left(x^{(i)}, x^{(j)}\right) \alpha_n(x^{(i)}).
\end{aligned} \tag{2.30}$$

We will also develop a more suitable expression for

$$\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_n = x^{(i)}\right).$$

We get

$$\begin{aligned}
&\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_n = x^{(i)}\right) \\
&= \mathrm{P}\left(\mathbf{Z}_{1:n} = \mathbf{Z}_{1:n}, \mathbf{Z}_{n+1:N} = \mathbf{Z}_{n+1:N}, X_n = x^{(i)}\right) \\
&= \mathrm{P}\left(\mathbf{Z}_{n+1:N} = \mathbf{Z}_{n+1:N} \mid \mathbf{Z}_{1:n} = \mathbf{Z}_{1:n}, X_n = x^{(i)}\right) \\
&\quad \cdot \mathrm{P}\left(\mathbf{Z}_{1:n} = \mathbf{Z}_{1:n}, X_n = x^{(i)}\right) \\
&= \alpha_n(x^{(i)}) \beta_n(x^{(i)}).
\end{aligned} \tag{2.31}$$

We now wish to develop expressions for estimating the transition probabilities a_{ij} and initial probabilities π_i while keeping the emission probabilities fixed. Any parameters associated with the emission probabilities are assumed to be fixed in a way that satisfies whatever constraints they must maintain. We have

$$\lambda = (\mathbf{A}, \pi). \tag{2.32}$$

The constraints that need to be satisfied are:

$$\sum_{j=1}^{N} a_{ij} = 1, \ \forall i \in \mathcal{I}, \text{ and} \tag{2.33}$$

$$\sum_{i=1}^{N} \pi_i = 1. \tag{2.34}$$

In order to maximize $\hat{Q}(\lambda, \lambda^{(i-1)})$ while satisfying the constraints we use the method of Lagrange multipliers [14]. A general formulation of the the Lagrangian function $\mathcal{L}(\mathbf{x}, \theta)$ is

$$\mathcal{L}(\mathbf{x}, \theta) = f(\mathbf{x}) - \sum_{i=1}^{m} \theta_i g_i(\mathbf{x}), \tag{2.35}$$

where $f(\mathbf{x})$ is the function that we wish to maximize, m is the number of constraints, and $g_i(\mathbf{x}), i = 1, \ldots, m$ comes from a constraint $g_i(\mathbf{x}) = 0$. By taking the derivative

of $\mathcal{L}(\mathbf{x},\theta)$ with respect to its variables and setting it equal to zero we can find the point $\mathbf{x}^\star$ that maximizes f while respecting the constraints [15].

The Lagrangian for our specific problem becomes

$$\mathcal{L}(\lambda,\theta) = f(\lambda) - \sum_{i=1}^{N} \theta_i g_i(\lambda) - \theta_{N+1} h(\lambda), \tag{2.36}$$

where

$$g_i(\lambda) = \sum_{j=1}^{N} a_{ij} - 1,\ i \in \mathcal{I}, \tag{2.37}$$

$$h(\lambda) = \sum_{i=1}^{N} \pi_i - 1, \tag{2.38}$$

and

$$\begin{aligned} f(\lambda) = \sum_{\mathbf{x}\in\mathcal{X}^N} \Bigg[& \log \pi(x_1)\, p(\mathbf{z},\mathbf{x};\lambda^{(i-1)}) \\ & + \sum_{i=1}^{N} \log b(z_i,x_i)\, p(\mathbf{z},\mathbf{x};\lambda^{(i-1)}) \\ & + \sum_{i=2}^{N} \log a(x_{i-1},x_i)\, p(\mathbf{z},\mathbf{x};\lambda^{(i-1)}) \Bigg]. \end{aligned} \tag{2.39}$$

We proceed by taking the derivative of the Lagrangian with respect to the various parameters of λ and setting them equal to zero.

$$\frac{\partial \mathcal{L}(\lambda,\theta)}{\partial a_{ij}} = \frac{\partial f(\lambda)}{\partial a_{ij}} - \theta_i \frac{\partial g_i(\lambda)}{\partial a_{ij}} = 0 \ \ \forall\,(i,j) \in \mathcal{I}^2 \tag{2.40}$$

$$\frac{\partial \mathcal{L}(\lambda,\theta)}{\partial \theta_i} = \frac{\partial g_i(\lambda)\theta_i}{\partial \theta_i} = 0 \ \ \forall\, i \in \mathcal{I} \tag{2.41}$$

$$\frac{\partial \mathcal{L}(\lambda,\theta)}{\partial \pi_i} = \frac{\partial f(\lambda)}{\partial \pi_i} + \theta_{N+1} \frac{\partial h(\lambda)}{\partial \pi_i} = 0 \ \ \forall\, i \in \mathcal{I} \tag{2.42}$$

$$\frac{\partial \mathcal{L}(\lambda,\theta)}{\partial \theta_{N+1}} = \frac{\partial \theta_{N+1} h(\lambda)}{\partial \theta_{N+1}} = 0 \ \ \forall\, i \in \mathcal{I} \tag{2.43}$$

The system shown in equations (2.40) through (2.43) must now be solved. Our first move is to handle (2.40) and (2.41) in order to find update equations for a_{ij}. The continuation is to deal with (2.42) and (2.43) to find update equations for π_i. To start things off, we find that

$$\frac{\partial f(\lambda)}{\partial a_{ij}} = \frac{\partial}{\partial a_{ij}} \sum_{\mathbf{x}\in\mathcal{X}^N} \sum_{n=2}^{N} \log a(x_{n-1},x_n)\, p(\mathbf{z},\mathbf{x};\lambda^{(i-1)}). \tag{2.44}$$

Here, we take the sum of $\log a\,(x_{n-1}, x_n)$ for $n = 1\ldots, N$ for each specific state sequence and weight it by the probability of the corresponding state sequence. It is in our case more suitable to sum over $\log a_{ij}, (i,j) \in \mathcal{I}^2$ for $n = 1\ldots, N$ and weight it by the probability that such a transition would occur at such a time. This is an equivalent approach, seeing as we are still summing over all possible values of $\mathbf{X}$.

$$\begin{aligned}
\frac{\partial f(\lambda)}{\partial a_{ij}} &= \sum_{k=1}^{M}\sum_{l=1}^{M}\sum_{n=2}^{N}\Bigg[\frac{\partial}{\partial a_{ij}}\log a\left(x^{(l)}, x^{(k)}\right) \\
&\quad \cdot \mathrm{P}\left(X_n = x^{(l)}, X_{n+1} = x^{(k)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)\Bigg] \\
&= \sum_{n=2}^{N}\Bigg[\frac{\partial}{\partial a_{ij}}\log a_{ij} \\
&\quad \cdot \mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)\Bigg] \\
&= \frac{\sum_{n=2}^{N}\mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{a_{ij}}.
\end{aligned} \tag{2.45}$$

In preparation for the next step we note that

$$\theta_i \frac{\partial}{\partial a_{ij}} g_i(\lambda) = \theta_i. \tag{2.46}$$

We now have

$$\begin{aligned}
\frac{\partial \mathcal{L}(\lambda, \theta)}{\partial a_{ij}} &= \frac{\sum_{n=2}^{N}\mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{a_{ij}} + \theta_i = 0 \\
\Rightarrow a_{ij} &= \frac{\sum_{n=2}^{N}\mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{\theta_i}.
\end{aligned} \tag{2.47}$$

Continuing from Equation (2.41) we get

$$\begin{aligned}
\frac{\partial \mathcal{L}(\lambda, \theta)}{\partial \theta_i} &= \frac{\partial g_i(\lambda)\theta_i}{\partial \theta_i} = \sum_{j=1}^{M} a_{ij} - 1 = 0 \\
\Rightarrow 1 &= \sum_{j=1}^{M}\frac{\sum_{n=2}^{N}\mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{\theta_i} \\
\Rightarrow \theta_i &= \sum_{n=2}^{N}\sum_{j=1}^{M}\mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right).
\end{aligned} \tag{2.48}$$

Combining (2.47) and (2.48) yields

$$\begin{aligned} a_{ij} &= \frac{\sum_{n=2}^{N} \mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{\sum_{n=2}^{N} \sum_{j=1}^{M} \mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)} \\ &= \frac{\sum_{n=2}^{N} \mathrm{P}\left(X_{n-1} = x^{(i)}, X_n = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{\sum_{n=2}^{N} \mathrm{P}\left(X_{n-1} = x^{(i)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}. \end{aligned} \tag{2.49}$$

In order to find update equations for the initial probabilities we start where we left off in Equation (2.42) and get

$$\begin{aligned} \frac{\partial f(\lambda)}{\partial \pi_i} &= \frac{\partial}{\partial \pi_i} \sum_{\mathbf{x} \in \mathcal{X}^N} \log \pi(x_1) \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, \mathbf{X} = \mathbf{x}; \lambda^{(i-1)}\right) \\ &= \sum_{j=1}^{N} \frac{\partial}{\partial \pi_i} \log \pi(x^{(j)}) \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(j)}; \lambda^{(i-1)}\right) \\ &= \frac{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(i)}; \lambda^{(i-1)}\right)}{\pi_i}, \end{aligned} \tag{2.50}$$

and

$$\theta_{N+1} \frac{\partial h(\lambda)}{\partial \pi_i} = \theta_{N+1}. \tag{2.51}$$

Moving on from Equation (2.43) we get

$$\frac{\partial \mathcal{L}(\lambda)}{\partial \theta_{N+1}} = \sum_{j=1}^{N} \pi_j - 1. \tag{2.52}$$

The argument used to avoid summing over all sequences in Equation (2.50) is similar to the one made before Equation (2.44). We are still "summing out" every possible value of $\mathbf{X}$, but this is now done by using groups of sequences with identical initial state. By combining (2.50) and (2.51) we get

$$\begin{aligned} \frac{\partial \mathcal{L}(\lambda, \theta)}{\partial \pi_i} &= \frac{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(i)}; \lambda^{(i-1)}\right)}{\pi_i} + \theta_{N+1} = 0 \\ \Rightarrow \pi_i &= \frac{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(i)}; \lambda^{(i-1)}\right)}{\theta_{N+1}}. \end{aligned} \tag{2.53}$$

By moving on from Equation (2.43) we get

$$\begin{aligned} &\sum_{j=1}^{N} \pi_j = 1 \Rightarrow 1 = \frac{1}{\theta_{N+1}} \sum_{j=1}^{N} \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(j)}; \lambda^{(i-1)}\right) \\ &\Rightarrow \theta_{N+1} = \sum_{j=1}^{N} \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(j)}; \lambda^{(i-1)}\right). \end{aligned} \tag{2.54}$$

Finally, by combining Equation (2.53) and (2.54) we arrive at

$$\begin{aligned}\pi_i &= \frac{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(i)}; \lambda^{(i-1)}\right)}{\sum_{j=1}^{N} \mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(j)}; \lambda^{(i-1)}\right)} \\ &= \frac{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_1 = x^{(i)}; \lambda^{(i-1)}\right)}{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}.\end{aligned} \tag{2.55}$$

In the literature, such as [8], it is common to express the update equations in terms of the quantities $\xi_n(i,j)$ and $\gamma_n(i)$ defined as

$$\gamma_n(i) = \frac{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}, X_n = x^{(i)}; \lambda^{(i-1)}\right)}{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)} \tag{2.56}$$

$$= \frac{\alpha_n(x^{(i)})\beta_n(x^{(i)})}{\sum_{j=1}^{M} \alpha_n(x^{(j)})\beta_n(x^{(j)})}, \tag{2.57}$$

$$\xi_n(i,j) = \frac{\mathrm{P}\left(X_n = x^{(i)}, X_{n+1} = x^{(j)}, \mathbf{Z}_{1:N} = \mathbf{z}_{1:N}; \lambda^{(i-1)}\right)}{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}\right)} \tag{2.58}$$

$$= \frac{b\left(z_{n+1}, x^{(j)}\right)\beta_{n+1}(x^{(j)})a\left(x^{(i)}, x^{(j)}\right)\alpha_n(x^{(i)})}{\mathrm{P}\left(\mathbf{Z}_{1:N} = \mathbf{z}_{1:N}\right)}. \tag{2.59}$$

We arrived at this by remembering the expressions found in Equation (2.30) and (2.31). By incorporating $\xi_n(i,j)$ and $\gamma_n(i)$ we get the following update equations:

$$a_{ij} \leftarrow \frac{\sum_{n=1}^{N-1} \xi_n(i,j)}{\sum_{n=1}^{N-1} \gamma_n(i)}, \text{ and} \tag{2.60}$$

$$\pi_i \leftarrow \gamma_1(i). \tag{2.61}$$

This leads us to Algorithm 5, known as the *Baum-Welch algorithm*.

In the case of multiple independent sequences of observations [16], denoted by $\mathscr{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_D)$, we get that the likelihood takes the form

$$\prod_{d=1}^{D} p(\mathbf{z}_d, \mathbf{x}_d) = \prod_{d=1}^{D} \pi\left(x_{1,d}\right) a\left(x_{1,d}, x_{2,d}\right) \prod_{i=2}^{N_d} a\left(x_{i-1,d}, x_{i,d}\right) b\left(z_{i,d}, x_{i,d}\right). \tag{2.62}$$

Here, D is the number of sequences. A single sequence of observations is denoted by $\mathbf{z}_d$, $d \in \{1, \ldots, D\}$. Each sequence $\mathbf{z}_d \in \mathscr{Z}$ is associated with a hidden state sequence $\mathbf{x}_d \in \mathcal{X}^{N_d}$. These can be assembled in manner similar to $\mathscr{Z}$, giving us $\mathscr{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_D)$. N_d denotes the number of observations in $\mathbf{z}_d$. A situation with $\mathscr{Z}$ instead of $\mathbf{z}$ does not change the EM procedure in any significant way, except introducing a sum over the various observation sequences in the log-likelihood. If one were to use this log-likelihood instead of the one used in our derivations, one

Algorithm 5: Baum-Welch algorithm, or EM algorithm for HMMs.

Compute $\alpha_n(x^{(i)})$ using the forward algorithm
Compute $\beta_n(x^{(i)})$ using the backward algorithm
for $n = 1{:}N$ **do**
  for $i = 1{:}M$ **do**
    $\gamma_n(i) = \frac{\alpha_n(x^{(i)})\beta_n(x^{(i)})}{\sum_{k=1}^{M}\alpha_n(x^{(k)})\beta_n(x^{(k)})}$
    for $j = 1{:}M$ **do**
      $\xi_n(i,j) = \frac{b\left(z_{n+1},x^{(j)}\right)\beta_{n+1}(x^{(j)})a\left(x^{(i)},x^{(j)}\right)\alpha_n(x^{(i)})}{\sum_{k=1}^{M}\alpha_n(x^{(k)})\beta_n(x^{(k)})}$
    end for
  end for
end for
for $i = 1{:}M$ **do**
  for $j = 1{:}M$ **do**
    $a_{ij} = \frac{\sum_{n=1}^{N-1}\xi_n(i,j)}{\sum_{n=1}^{N-1}\gamma_n(i)}$
  end for
  $\pi_i = \gamma_1(i)$
end for

would end up with the following update equations:

$$a_{ij} \leftarrow \frac{\sum_{d=1}^{D}\sum_{n=1}^{N_d-1}\xi_{n,d}(i,j)}{\sum_{d=1}^{D}\sum_{n=1}^{N_d-1}\gamma_{n,d}(i)}, \text{ and} \tag{2.63}$$

$$\pi_i \leftarrow \frac{\sum_{d=1}^{D}\gamma_{1,d}(i)}{D}. \tag{2.64}$$

Here, $\xi_{n,d}(i,j)$ and $\gamma_{n,d}(i)$ is used to denote $\xi_n(i,j)$ and $\gamma_n(i)$ calculated using the observation sequence $\mathbf{z}_d$. These update equations are known as Levinson's training equations [17]. Algorithm 5 can easily be adapted to this new situation by adding a for-loop that iterates over all observation sequences in order to calculate $\xi_{n,d}(i,j)$ and $\gamma_{n,d}(i)$. The update equations in Algorithm 5 would also be updated to match Equation (2.63) and (2.64).

2.7 Scaling

If one were to implement the *forward algorithm*, the *backward algorithm* and the EM algorithm exactly as presented in the previous sections one would likely run into computational issues, especially for large N. This happens because the values of $\alpha_n(x)$ and $\beta_n(x)$ quickly become too small to be accurately represented on a computer. It is therefore necessary to apply a scaling procedure [8].

The scaling procedure presented here will ensure that all quantities in the *forward algorithm*, *backward algorithm* and the *Baum-Welch algorithm* will take on values that remain reasonably close to 1. This is achieved by swapping $\alpha_n(x)$ in the *forward algorithm* with the scaled values $\hat{\alpha}_n(x)$. The scaled values are computed using

$$\hat{\alpha}_n(x^{(i)}) = \frac{\sum_{j=1}^{M} \hat{\alpha}_{n-1}(x^{(j)}) a\left(x^{(j)}, x^{(i)}\right) b\left(z_n, x^{(j)}\right)}{\sum_{k=1}^{M}\sum_{l=1}^{M} \hat{\alpha}_{n-1}(x^{(l)}) a\left(x^{(l)}, x^{(k)}\right) b\left(z_n, x^{(l)}\right)}, \tag{2.65}$$

for $n \geq 2$. The initial value $\hat{\alpha}_1(x)$ is

$$\hat{\alpha}_1(x) = \frac{b\left(z_1, x\right)\pi\left(x\right)}{\sum_{i}^{M} b\left(z_1, x^{(i)}\right)\pi\left(x^{(i)}\right)}. \tag{2.66}$$

By induction one finds that

$$\hat{\alpha}_{n-1}(x) = \left(\prod_{t=1}^{n-1} c_t\right)\alpha_{n-1}(x), \tag{2.67}$$

where $c_t = \left(\sum_{i=1}^{M} \alpha_n(x^{(i)})\right)^{-1}$. The scaled quantity can be rewritten as

$$\hat{\alpha}_n(x^{(i)}) = \frac{\sum_{j=1}^{M} \alpha_{n-1}\left(\prod_{t=1}^{n-1} c_t\right)(x^{(j)}) a\left(x^{(j)}, x^{(i)}\right) b\left(z_n, x^{(j)}\right)}{\sum_{k=1}^{M}\sum_{l=1}^{M} \alpha_{n-1}\left(\prod_{t=1}^{n-1} c_t\right)(x^{(l)}) a\left(x^{(l)}, x^{(k)}\right) b\left(z_n, x^{(l)}\right)}. \tag{2.68}$$

The values of $\beta_n(x)$ in the *backward algorithm* are swapped with the scaled values $\hat{\beta}_n(x)$, defined as

$$\hat{\beta}_n(x) = c_n \beta_n(x). \tag{2.69}$$

We now note that

$$\hat{\alpha}_n(x) = \left(\prod_{t=1}^{n} c_t\right)\alpha_n(x) = C_n \alpha_n(x) \tag{2.70}$$

$$\hat{\beta}_{n+1}(x) = \left(\prod_{t=n+1}^{n} c_t\right) = D_{n+1}\beta_{n+1}(x) \tag{2.71}$$

By using the scaled quantities $\hat{\alpha}_n(x)$ and $\hat{\beta}_n(x)$ in the EM algorithm we get

$$\begin{aligned} a_{ij} &= \frac{\sum_{n=1}^{M} b\left(z_{n+1}, x^{(j)}\right)\hat{\beta}_{n+1}(x^{(j)}) a\left(x^{(i)}, x^{(j)}\right)\hat{\alpha}_n(x^{(i)})}{\sum_{n=1}^{N-1}\sum_{k=1}^{M} b\left(z_{n+1}, x^{(k)}\right)\hat{\beta}_{n+1}(x^{(k)}) a\left(x^{(i)}, x^{(k)}\right)\hat{\alpha}_n(x^{(i)})} \\ &= \frac{\sum_{n=1}^{M} b\left(z_{n+1}, x^{(j)}\right) D_{n+1}\beta_{n+1}(x^{(j)}) a\left(x^{(i)}, x^{(j)}\right) C_n \alpha_n(x^{(i)})}{\sum_{n=1}^{N-1}\sum_{k=1}^{M} b\left(z_{n+1}, x^{(k)}\right) D_{n+1}\beta_{n+1}(x^{(k)}) a\left(x^{(i)}, x^{(k)}\right) C_n \alpha_n(x^{(i)})}. \end{aligned} \tag{2.72}$$

Since $C_n D_{n+1} = \left(\prod_{i=1}^{N} c_n\right)$ is independent of n we get that the terms $C_n D_{n+1}$ in the nominator and denominator cancel each other out. The update equation therefore

evaluates to the same value regardless of whether the scaled quantities are used or not.

When working exclusively with the scaled values $\hat{\alpha}_n(x)$ we can not evaluate the probability $\mathrm{P}(\mathbf{Z}=\mathbf{z})$ the same way as before. We can, however, use the fact that

$$\log \mathrm{P}(\mathbf{Z}=\mathbf{z}) = -\sum_{n=1}^{N} \log c_n. \tag{2.73}$$

This is justified by

$$\begin{aligned} \sum_{i=1}^{M} \hat{\alpha}_n(x^{(i)}) = C_N \sum_{i=1}^{M} \alpha_N(x^{(i)}) = C_N \mathrm{P}(\mathbf{Z}=\mathbf{z}) = 1 \\ \Rightarrow \mathrm{P}(\mathbf{Z}=\mathbf{z}) = \frac{1}{C_N} \\ \Rightarrow \log \mathrm{P}(\mathbf{Z}=\mathbf{z}) = -\sum_{n=1}^{N} \log c_n. \end{aligned} \tag{2.74}$$

Chapter 3

Methods

3.1 Problem formulation

The objective of this thesis is to explore and assess the viability of using HMMs for map matching. Map matching refers to the estimation of an objects travel route by using a sequence of measurements of the objects position at various times. These measurements are, in real-world scenarios, typically GPS measurements. A travel route is, at its core, exact information about how the position of the object evolves throughout time. This definition is, however, not suitable for our purpose. We will therefore think of a travel route as a sequence of connected edges in graph, where the graph represents a road network. Each edge in the graph can be interpreted as a straight line between its tail and its head. The edges vary in length and can be combined in different ways to represent different kinds of road segments. A sharp turn will be approximated by several short edges, while other types of road segments, such as a freeway, may be accurately represented by fewer, longer edges. When defining a travel route as a sequence of edges we must assume that the travel does indeed happen on some sort of network. Such behaviour is exhibited by cars and other similar vehicles. We will therefore, for the purpose of this thesis, consider only the movement of such vehicles on an appropriate road network.

We will now provide a detailed presentation of the various components involved in the problem. This is required to both define the problem in rigorous manner and formulate an HMM that enables travel route estimation. We will also discuss the observations and the assumptions we make about them. Following this, we will define the HMM. The presentation of the HMM will be divided into three parts; the state space, the transition and initial probabilities, and the emission probabilities.

3.2 Major components

3.2.1 Road network

It is natural to represent a road network as some variant of a graph. Both a directed graph and an undirected graph can be suitable, depending on the situation. An

undirected graph is the simplest way of representing a road network as a graph, but makes it impossible to represent one-way streets. A more accurate representation of a road network can be achieved by using a directed graph, seeing as this can better represent the real-world situation. The graph is defined as

$$G = (N, E), \tag{3.1}$$

where N is the set of nodes and E is the set of edges. The edge set must consists exclusively of directed edges, or exclusively of undirected edges. An undirected edge $e \in E$ will be denoted by $e = \{n_1, n_2\}$, with $n_1, n_2 \in N$. A directed edge will be denoted by $e = (n_1, n_2)$, with $n_1, n_2 \in N$. The notation G_u and G_d will be used when it is necessary to specify whether a graph is respectively undirected or directed. Similarly, the set of undirected or directed edges will be referred to using the notation E_u and E_d.

We will now discuss the variety of ways in which we can think about the edges and nodes. The first interpretation is to think of them simply as the components of a graph that expresses how certain objects in a road network is related to each other. This is, however, not sufficient for our purpose. It is crucial for the problem we are trying to solve that we know the geographical location of the different nodes. When letting the nodes represent nodes in space it is also natural to let the edge represent some sort of curve. In a Euclidean plane it is sensible to let the edge be described by a linear function $f_e(x) = ax + b$, $x_- \leq x \leq x_+$, with x_- and x_+ being the x-coordinates of the nodes connected by the edge. When the points in space are not located on a Euclidean plane, such as when the position is given by longitude and latitude we think of the edges as the shortest path between the two points. We rarely need to concern ourselves about the latter case though, seeing as we can always project the points in a longitude-latitude coordinate system to a UTM coordinate system [18], which is Euclidean.

3.2.2 Measurements

The position measurements we deal with in this thesis are assumed to come from a GPS device that is somehow embedded or positioned in the travelling vehicle. The position measurements are taken at regular, or close to regular, time intervals and provide fairly accurate information about the position of the vehicle. The position is typically reported using a point in some coordinate system, with a longitude-latitude pair being the most common. We will, however, for the purpose of this thesis, assume that the measured positions exist in a Euclidean plane. A UTM coordinate system is consistent with this assumption, and the measurements will therefore exclusively be represented in such a system. This is not a significant restriction since there exists a variety of software solutions for projecting points and other geometries from one coordinate system to another. Formally, the measured positions are represented as points $\mathbf{z} = (z_x, z_y) \in \mathbb{R}^2$.

3.2.3 Hidden Markov model formulation

State space

The first state space variant is quite straightforward. We recall that the road network can be represented as an undirected graph $G_u = (N, E_u)$. By choosing to let the state space be $\mathcal{X}_s = E_u$ we are able to estimate which edge the vehicle was on when the corresponding measurement was made [3]. We will refer to this state space variant as the simple state space.

The second state space variant is slightly more complex than the first variant. The motivation behind this variant is to incorporate information about which edge the vehicle was on directly before ending up on the edge on which the measurement was made. Consider an edge $e = (n_1, n_2) \in E_d$. Let $N(e)$ denote the set of all edges $\tilde{e} = (\tilde{n}_1, \tilde{n}_2) \in E_d$ where $\tilde{n}_2$ is equal to n_1. $N(e)$ is now the set of all directed edges that has a head that coincides with the tail of e. We proceed by letting the set $C(e)$ be defined as

$$C(e) = \{(e, \hat{e})\}_{\hat{e} \in N(e)}. \tag{3.2}$$

The state space $\mathcal{X}_a$ can now be defined as

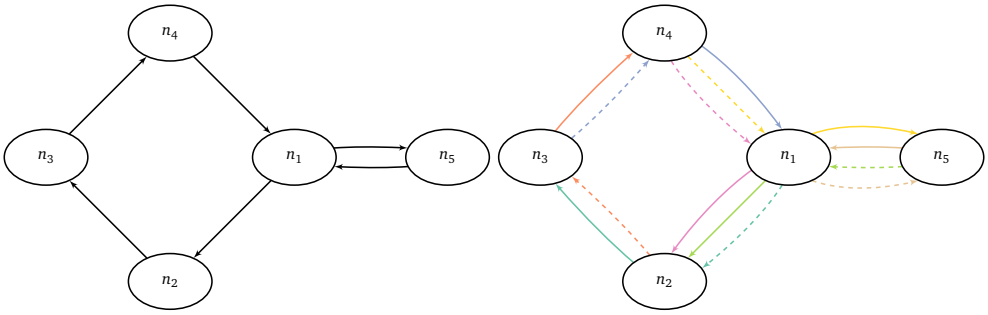
$$\mathcal{X}_a = \bigcup_{e \in E_d} C(e). \tag{3.3}$$

This state space variant will be referred to as the augmented state space.

When using the augmented state space $\mathcal{X}_a$ it is useful to think of the state $(e, \hat{e})$ as two distinct pieces of information. The edge e represents the edge one is currently on, and is, therefore, the "current edge". The edge $\hat{e}$ represents the edge one entered e from, and becomes the "preceding edge". The edges $\hat{e}$ and e share a common node. This will be referred to as the shared node. The node $\hat{n}_1$ in $\hat{e} = (\hat{n}_1, \hat{n}_2)$ will be known as the arrival node. This is done with the intention of making it clear that this the earliest piece of information about the position of the vehicle that is available from the state. The node n_2 in $e = (n_1, n_2)$ will, for similar reasons, be known as the departure node. We will denote the arrival node, shared node and departure node of a state $x \in \mathcal{X}_a$ by A_x, S_x and D_x respectively. The state space, and how it relates to the road network, is illustrated in Figure 3.1. The dotted lines in this figure represent the "preceding edge". The "current edge" is given by a solid line. Each state is assigned a unique color, and consists of the combination of a dotted line and a solid line. Note that there are six edges in the original graph, but seven states in the state space. This happens because the edge (n_1, n_2) can be entered from either (n_4, n_1) or (n_5, n_1).

Transition probabilities and initial probability

When defining the transition probabilities used in the HMM it is important to make sure that transitions that would be unlikely to occur in a realistic scenario are also considered unlikely under the chosen probability model. Our knowledge about movement in the physical worlds tells us that movement over large distances in



(a) An imagined road network represented as a directed graph.

(b) An illustration of the augmented state space $\mathcal{X}_a$ created using the imagined road network.

Figure 3.1: An illustration of how an imagined road network, in the form of a directed graph, can be used to create the augmented state space. Each of the resulting states has a distinct color.

a short amount of time is very unlikely to occur. This preference for movement over short distances can be incorporated into an HMM by using the transition probability

$$a(x, y) = \gamma e^{-\gamma d(x,y)}, \tag{3.4}$$

where $d(x, y)$ is a function that returns the distance between states x and y [2]. The exact definition of this distance function will depend on whether we are using the simple or the augmented state space. In the case of the simple state space the distance function will be defined to be the minimum distance that can be achieved by moving from either node in $x = \left(n_1^{(x)}, n_2^{(x)}\right)$ to either node in $y = \left(n_1^{(y)}, n_2^{(y)}\right)$. The shortest path candidates is computed using Dijkstra's shortest path algorithm [19]. In order to define this properly, let $SP(n_1, n_2)$ be a function that returns the sequence of edges that corresponds to the shortest path between the two nodes $n_1 \in N$ and $n_2 \in N$. When computing the shortest path the edges are weighted according to the length of the corresponding road segment in whatever coordinate system we are working in. The notation $L(e)$, $e \in E$, will be used denote this length. Going forward, when referring to the length of an edge, we will be referring to this length. The length of an edge sequence $\mathbf{e} = (e_1, \ldots, e_{-1})$, given by $|\mathbf{e}|$, can now be defined as

$$|\mathbf{e}| = \sum_{e \in \mathbf{e}} L(e). \tag{3.5}$$

We now get that $|SP(n_1, n_2)|$ denotes the length of the shortest path between the two nodes. The distance $d_s(x, y)$ can be defined as

$$\begin{aligned} d_s(x, y) = \min \Bigg\{ &\left|SP(n_1^{(x)}, n_2^{(y)})\right|, \left|SP(n_2^{(x)}, n_2^{(y)})\right|, \\ &\left|SP(n_1^{(x)}, n_1^{(y)})\right|, \left|SP(n_2^{(x)}, n_1^{(y)})\right| \Bigg\}. \end{aligned} \tag{3.6}$$

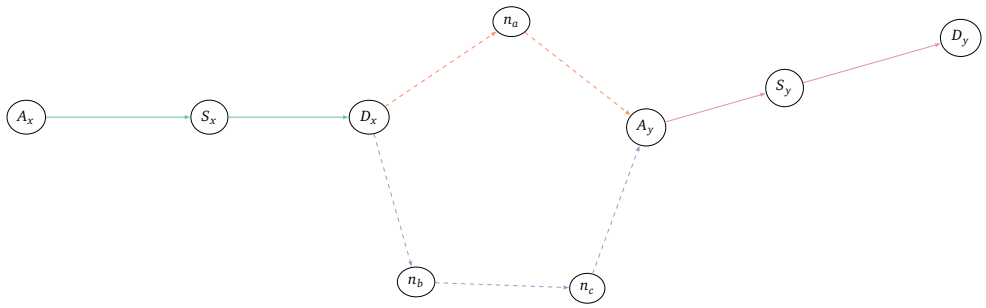


Figure 3.2: Two states, $x = ((S_x, D_x), (A_x, S_x))$ and $y = \big((S_y, D_y), (A_y, S_y)\big)$, shown with solid lines and green and pink colors respectively. Potential paths from x to y shown with dotted lines.

The transition probability for the simple state space becomes

$$a_s(x, y) = \gamma e^{-\gamma d_s(x,y)}. \tag{3.7}$$

In the case of the augmented state space we will incorporate additional information that is available about the transition. When transitioning from a state $x = (e_x, \hat{e}_x) \in \mathcal{X}_a$ to a state $y = \big(e_y, \hat{e}_y\big) \in \mathcal{X}_a$ we know that the path between the states starts at the departure node of x, D_x, moves to the arrival node of y, A_y, and ends at the shared node of y, S_y. The distance between D_x and A_y is taken to be the length of the shortest path between the two nodes. There are two exceptions, though. If $x = y$ there is no distance to travel. If $\hat{e}_y$, the "preceding edge" of y, coincides with e_x, the "current edge" of x, then the only movement that must be made is from the shared node of y, S_y, to the departure node of y, D_y. We get the distance function

$$d_a(x, y) = \begin{cases} 0, & x = y \\ L\big((S_y, D_y)\big), & e_x = \hat{e}_y \\ \big|SP\big(D_x, A_y\big)\big| + L\big((A_y, S_y)\big) + L\big((S_y, D_y)\big), & \text{otherwise} \end{cases}. \tag{3.8}$$

The length of the edge $\big((S_y, D_y)\big)$ is added to the distance because we know that the departure happens from the departure node, and we know that vehicle will cross this edge to get there. A visualization of how two states x and y can be connected is shown in Figure 3.2. Note that the visualization contains more states than just x and y, but we choose to only highlight these two. The transition probability for the augmented state space becomes

$$a_a(x, y) = \gamma e^{-\gamma d_a(x,y)}. \tag{3.9}$$

The initial probability must also be decided. For the simple state space $\mathcal{X}_s$ it seems sensible to use a uniform distribution over all states. This will also be done for the augmented state space, but there is a possibility that another alternative

might be preferable. The reason that we might not want a uniform distribution over all states in $\mathcal{X}_a$ is because it would make starting off at intersections more likely than starting off at an edge with fewer connections. An alternative distribution over the states that makes the sum of the initial probabilities for the states in $C(e)$ equal for all $e \in E_d$ could therefore be considered. This change would, however, be unlikely to produce any discernable difference in performance, and the simpler option is therefore preferred. The initial probabilities become

$$\pi_s(x) = \frac{1}{|\mathcal{X}_s|}, \tag{3.10}$$

and

$$\pi_a(x) = \frac{1}{|\mathcal{X}_a|}. \tag{3.11}$$

We also want to investigate the effects of making a certain transitions illegal by setting the corresponding transition probability to zero. Up until this point we have used transition probabilities that make all transitions possible. The resulting estimate can therefore not be interpreted as a route, and can only be viewed as estimates of which edge the vehicle was at when the corresponding measurement was made. This problem can be avoided by setting the transition probability between unconnected states to zero. This ensures that subsequent states in the MAP estimate are connected. The transition probability in this case becomes

$$a_c(x, y) = \gamma e^{-\gamma d_a(x,y)} \mathbb{1}(e_x = \hat{e}_y \vee x = y). \tag{3.12}$$

The function $\mathbb{1}(e_x = \hat{e}_y \vee x = y)$ evaluates to 1 if the "preceding" part of state y coincides with the "current" part of state x, or if the states are identical. It evaluates to 0 otherwise. When we speak of "a state x connected to a state y", we refer to the ordered pair of states that make the function $\mathbb{1}(e_x = \hat{e}_y \vee x = y)$ evaluate to 1. The ordering is required because the function is not symmetric. When we speak of "connected states" we refer to an ordered pair of states that make the function evaluate to 1.

Emission probability

To complete our formulation of the HMM we must decide on a probability model for the observations. A common assumption in the literature [1] [2] is that the distance between the state, i.e. the edge, and the observation follows a half-normal distribution with some scale parameter σ_p. The emission probability becomes

$$b_s(\mathbf{z} \mid x) = \frac{\sqrt{2}}{\sqrt{\pi\sigma_p^2}} \exp\left(-\frac{||\mathbf{z} - \overline{\mathbf{z}}_x)||_2^2}{2\sigma_p^2}\right), \tag{3.13}$$

with $\overline{\mathbf{z}}_x$ denoting the point on edge x closest to the observation $\mathbf{z}$, and $||\cdot||_2$ denoting the Euclidean norm. An illustration showing the distance between an observation and two different edges is shown in Figure 3.3. The emission probability is in

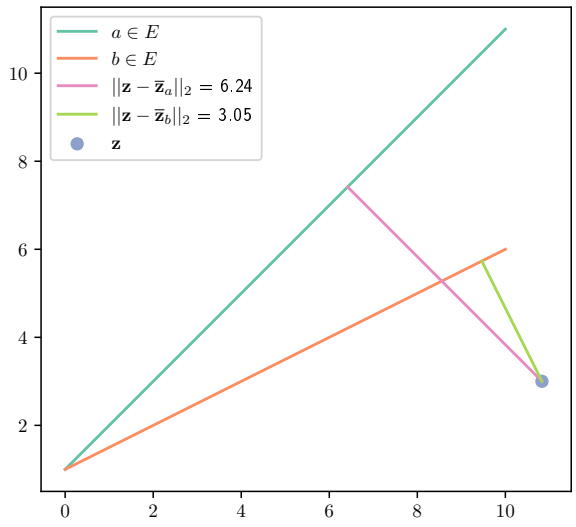


Figure 3.3: A plot showing the distances between an observation $\mathbf{z}$ and two edges a and b.

essence identical when using the augmented state space, but the formulation differs slightly due to the "expanded" state. With states on the form $x = (e, \hat{e})$ the emission probability becomes

$$b_a(\mathbf{z} \mid x) = \frac{\sqrt{2}}{\sqrt{\pi\sigma_p^2}} \exp\left(-\frac{||\mathbf{z} - \bar{\mathbf{z}}_e)||_2^2}{2\sigma_p^2}\right). \tag{3.14}$$

We recall that the assumptions of the HMM, presented in Section 2.2, states the observations should be independent given the state. It is immediately clear that position measurements are not unconditionally independent. We do, however, believe that the states, being road segments, capture enough information about the position of the vehicle to make the assumption about conditional independence hold.

3.3 Learning model parameters

In Section 2.6 we introduced a method, the *Baum-Welch algorithm*, that enables us to update the parameters of the HMM in a way that increases the expected log-likelihood of the observed data. It is not obvious that increasing the expected log-likelihood results in an increase in the performance of the method when measured using suitable performance metrics, but we consider it to be worthwhile to at least explore the possibility. The results from Section 2.6 can be applied directly to our situation and used to learn the transition probabilities $a(x, y)$ and initial probabilities $\pi(x)$. We do, however, make one notable change for the case where only transitions between connected states are allowed. Since we have definite

knowledge about the states that are not connected, we fix the transition probability between these states to zero. This does not change anything about the *Baum-Welch algorithm*, except that we now only update the transition probabilities between states are connected. The update equations is as before, see Equation (2.49) and (2.53), but is only applied for $(i,j) \in \mathcal{I}^2$ where $x^{(i)}$ is connected to $x^{(j)}$ in the sense discussed directly after Equation (3.12).

In the case of multiple independent measurement sequences we adapt the *Baum-Welch algorithm* to work for this scenario. The necessary adaptions are presented in Section 2.6.

3.4 Route estimation

After constructing a hidden Markov model using either of the possible formulations introduced in Section 3.2.3, one can use the *Viterbi algorithm*, see Algorithm 4, to obtain an estimate of the hidden state sequence. The hidden state sequence is, however, only guaranteed to be immediately interpretable as a route estimate if we set the probability for transitions between non-connected states to zero, as discussed in Section 3.2.3. In order to obtain a route estimate from the output of the other methods we need to connect the estimated edges in some way. Seeing as there is no data about where the vehicle has travelled between two measurements, it seems most sensible to assume that the vehicle has travelled the shortest path between the edges [1]. The estimated travel route can then be obtained by finding the shortest path between subsequent non-connected edges. When using the simple state space we find the shortest path between the two edges by applying Dijkstra's shortest path algorithm to find the shortest path that can be obtained by going from either node at the ends of one edge to either node at the ends of the other edge. We recall that the length of a sequence of edges $\mathbf{e}$ is denoted by $|\mathbf{e}|$. We define the set of candidates for the shortest path between state $x = (n_1^{(x)}, n_2^{(x)}) \in \mathcal{X}_s$ and state $y = (n_1^{(y)}, n_2^{(y)}) \in \mathcal{X}_s$ as

$$SP_c(x,y) = \Big\{ SP(n_1^{(x)}, n_1^{(y)}),\, SP(n_1^{(x)}, n_2^{(y)}), \\ SP(n_2^{(x)}, n_1^{(y)}), SP(n_2^{(x)}, n_2^{(y)}) \Big\}. \tag{3.15}$$

The shortest path between state x and state y can now be defined as

$$C_s(x,y) = \underset{\mathbf{e} \in SP_c(x,y)}{\arg\min} \{|\mathbf{e}|\}. \tag{3.16}$$

Before moving on we introduce an alternative way of writing the sequence $\mathbf{s} = (s_1, \ldots, s_{-1})$. We allow for it to be written as

$$\mathbf{s} = (s_1, \diamond(s_2, \ldots, s_{-2}), s_{-1}). \tag{3.17}$$

This enables us to simplify the forthcoming definition of the route estimate. The route estimate in the case of the simple state space becomes

$$\mathbf{x}_c^\star = \left(x_1^\star, \diamond\, C_s(x_1^\star, x_2^\star), \ldots, x_n^\star, \diamond\, C_s(x_n^\star, x_{n+1}^\star), \ldots, x_N^\star\right), \tag{3.18}$$

with $\mathbf{x}^\star = \left(x_1^\star, \ldots, x_N^\star\right)$ denoting the MAP estimate of the hidden states. Here, it is important to remember that any $x \in \mathcal{X}_s$ is an edge and that $\mathbf{x}_c^\star$ therefore is an edge sequence.

Finding the shortest path between two states x and y in the augmented state space $\mathcal{X}_a$ requires a bit more consideration. This problem is closely related to the discussion preceding Equation (3.8). If the states are equal, then there is obviously no need to find a connecting path. This is the case if the states are connected in the sense defined in Section 3.2 as well. If the states are unconnected we create a path between them by finding the shortest path that goes from D_x to A_y, and add the final edge (A_y, S_y). We get

$$C_a(x,y) = \begin{cases} \emptyset, & x = y \vee e_x = \hat{e}_y \\ \left(\diamond\, SP(D_x, A_y), (A_y, S_y)\right), & \text{otherwise} \end{cases}. \tag{3.19}$$

With the individual state estimates written as $x_n^\star = (e_n^\star, \hat{e}_n)$ we get the estimated connected edge sequence

$$\mathbf{e}_c^\star = \left(e_1^\star, \diamond\, C_a(x_1^\star, x_2^\star), \ldots, e_n^\star, \diamond\, C_a(x_n^\star, x_{n+1}^\star), \ldots, e_N^\star\right). \tag{3.20}$$

One of the primary motivations behind using HMMs for map matching is to have a method that is more robust to noise than other alternatives. When given measurements with zero noise it would yield perfect results to simply, for each measurement, pick the edge closest to the measurement as the estimated edge. It therefore makes sense to compare the HMM to such a method. This enables us to see whether the impact of noise is less dramatic for the HMM-based approach than it is for this benchmark method. This benchmark method is easily implemented within the framework developed for the HMM approach. The elements of the benchmark estimate $\tilde{\mathbf{e}} = (\tilde{e}_1, \ldots, \tilde{e_N})$ is defined as

$$\tilde{e}_n = \arg\min_{e \in E} \{||\mathbf{z}^{(n)} - \overline{\mathbf{z}}_e^{(n)}||_2\}, \tag{3.21}$$

with $\overline{\mathbf{z}}_e^{(n)}$ denoting the point on edge e closest to the measurement $\mathbf{z}^{(n)} \in \mathbf{Z}$.

3.5 Performance metrics

To be able to assess the performance of the various methods presented we need to formulate metrics that is able to capture the methods ability to perform in a way that actually solves the underlying problem. The first metric that is natural to consider is the accuracy. If we have access to information about which edge the vehicle was truly on when the measurement was made we can assess whether

the estimated edge matches the true edge. It is, however, worth noting that the accuracy is not the most robust metric in this case, seeing as it will not differentiate between estimates that are "close" to the truth and estimates that are far off. We recall that, depending on our choice of state space, the estimated hidden states are either edges from the set of edges E, or a collection of two edges $(e, \hat{e})$ with $e \in \mathrm{E}$ and $\hat{e} \in N(e)$. We define the accuracy for estimates obtained using the HMM with a simple state space $\mathcal{X}_s$ or the benchmark method as

$$A_s(\mathbf{x}^\star) = \frac{\sum_{i=1}^{N} \mathbb{1}(x_i^\star = x_i)}{N}, \tag{3.22}$$

with $\mathbf{x}^\star = \left(x_1^\star, \ldots, x_N^\star\right)$ denoting the estimated state sequence and $\mathbf{x} = (x_1, \ldots, x_N)$ denoting the sequence of edges the vehicle was truly on when the corresponding measurements were made.

For the estimates obtained using the augmented state space we define the accuracy of the MAP estimate $\mathbf{x}^\star = \left(x_1^\star, \ldots, x_N^\star\right) = \left((e_1^\star, \hat{e}_1), \ldots, (e_N^\star, \hat{e}_N)\right)$ as

$$A_a(\mathbf{x}^\star) = \frac{\sum_{i=1}^{N} \mathbb{1}(e_i^\star = e_i)}{N}, \tag{3.23}$$

with $\mathbf{e} = (e_1, \ldots, e_N)$ being the true edge sequence. The accuracy is defined in this way in order to make it possible compare methods regardless of whether they use $A_a(\mathbf{x}^\star)$ or $A_s(\mathbf{x}^\star)$. It is only in our interest to evaluate the HMMs ability to correctly estimate the hidden states if the hidden states happen to coincide with what we actually want to estimate, namely the edges of the road network. That is why we disregard $\hat{e}_i$, $i = 1, \ldots, N$, when calculating $A_a(\mathbf{x}^\star)$.

The second metric that we will use is the Hausdorff distance, which is a metric that is used for comparing point sets [20]. This is appropriate for our purpose, since the travel route can be interpreted as a sequence of nodes, with the nodes representing points in space. We will denote the position of a node $n \in N$ by $p(n)$, and assume that $p(n) \in \mathbb{R}^2 \; \forall n \in N$. This interpretation makes it possible to translate an estimated travel route to a point set. We define the directed Hausdorff distance between a point set A and point set B as

$$\hat{H}(A, B) = \max_{x \in A} \left\{ \min_{y \in B} \{||x, y||_2\} \right\}, \tag{3.24}$$

where $||\cdot||_2$ denotes the Euclidean distance. The Hausdorff distance is defined as

$$H(A, B) = \max\left\{\hat{H}(A, B), \hat{H}(B, A)\right\}. \tag{3.25}$$

An algorithm for efficient computation of the Hausdorff distance is given in [20].

We will now present a small example to illustrate how the Hausdorff distance works. Consider a point set $A \subset \mathbb{R}^2$ defined as

$$A = \{(1, 2), (2.5, 3), (3, 4), (4, 5)\}, \tag{3.26}$$

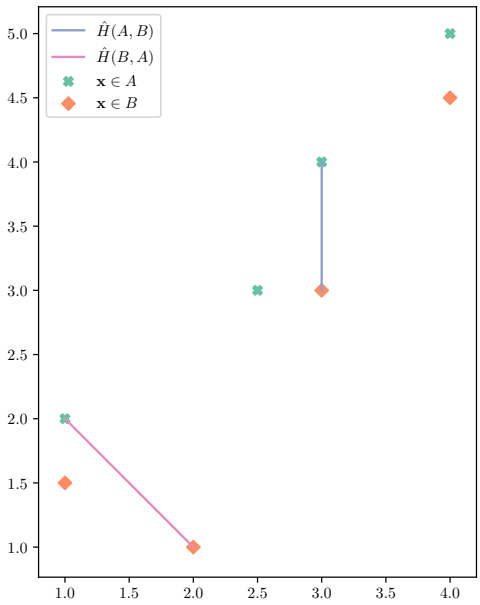


Figure 3.4: Toy example showing two point sets and the directed Hausdorff distance between the two sets.

and a point set $B \subset \mathbb{R}^2$ defined as

$$B = \{(1, 1.5), (2, 1), (3, 3), (4, 4.5)\}. \tag{3.27}$$

The point sets are shown in Figure 3.4 together with two lines connecting the Hausdorff pairs. There is one line corresponding to $\hat{H}(A,B)$ and one corresponding to $\hat{H}(A,B)$. The Hausdorff distance is the length of the longest of the two lines, which in this case is $\hat{H}(B,A)$. We end up with $H(A,B) = \hat{H}(B,A) = \sqrt{2}$.

The estimated edge sequence is $\mathbf{e}^\star$. An edge sequence can also be expressed as a node sequence. The estimated node sequence will be denoted by $\mathbf{n}^\star$. The true node sequence will be denoted by $\mathbf{n}$. We recall that a set of nodes can be interpreted as a set of points in space. The corresponding point set is therefore

$$P^\star = \bigcup_{n^\star \in \mathbf{n}^\star} p(n^\star). \tag{3.28}$$

The true point set is

$$P = \bigcup_{n \in \mathbf{n}} p(n). \tag{3.29}$$

The Hausdorff distance between an estimated edge sequence and a true edge sequence becomes $H(P, P^\star)$, and can be interpreted as the longest possible distance between the nodes in the estimated route and the nodes in the true route. A downside of the Hausdorff distance is that, unlike the accuracy, it considers the estimated route as a whole, and does not treat each estimated edge one-by-one. In other words, the Hausdorff distance does not care about the ordering of estimated

edges. We, therefore, use both metrics. This is done in order to provide information about as many aspects of the methods performance as possible.

Chapter 4

Experiments

This chapter is devoted to giving a detailed presentation of the experiments, their purpose and the framework that must be in place in order to conduct them. The chapter starts by presenting the data and how they are used to create a road network. The next section presents the simulation procedure that enables us to simulate road network traversal and position measurement. Following this, we present the set-up of our experiments. It consists of three main parts: Parameter search, parameter estimation and performance evaluation. The chapter is concluded with a section that presents the software that has been developed for this thesis, and how it enables us to perform the experiments.

4.1 Data

In order to construct a graph representing the road network we need access to a suitable data source. OpenStreetMap (OSM) [1] provides this, both in the form of data dumps that can be loaded into a PostGIS [2] database, or through APIs like Overpass [3]. There are also other potential sources, such as NVDB (Nasjonal vegdatabank) [4], which is a national database for road networks in Norway. The underlying structure of the data available from these sources differ slightly, but we will focus on the data available through OSM since this is what we use for our experiments.

Data in OSM is organized as either *nodes* or *ways*. A node is a point in some coordinate system, along with a unique *node ID*, and is usually associated with a number of tags that tells us which type of real-world object it represents. A way is an ordered collection of such nodes. It also has a unique ID, referred to as the *way ID*. In a geographical information system (GIS), a node can be seen as a geometry of type *Point*, while a way is a *Linestring*. A *Point* is a data structure for representing the position of some point. A *Linestring* is a data structure that

[1] https://www.openstreetmap.org/
[2] https://postgis.net/
[3] https://wiki.openstreetmap.org/wiki/Overpass_API
[4] https://www.vegvesen.no/fag/teknologi/nasjonal+vegdatabank

Node ID	Geometry
0	POINT (0.549 0.715)
1	POINT (0.603 0.545)
2	POINT (0.424 0.646)
3	POINT (0.438 0.892)
4	POINT (0.964 0.383)

Table 4.1: Nodes in the toy example.

Way ID	Nodes	Geometry
0	[0, 1, 2, 3, 4]	LINESTRING (0.549 0.715, 0.603 0.545, 0.424 0....

Table 4.2: Ways in the toy example.

contains an ordered set of points. The ordering of the points in a *Linestring* data structure enables it to be interpreted as a string of straight lines, with each line connecting two neighbouring points. A *Linestring* can therefore be thought of as a piecewise linear curve. A way has, like a node, a set of tags that describe what it represents. This enables us to find the ways that represent roads, and also tells us whether they are one-way streets or not.

The set of nodes N required for the constructing the road network graph $G = (N, E)$ can now be obtained by identifying the ways that represent roads — N is the union of all OSM nodes in the identified ways. The set of edges E is constructed by breaking the identified ways into its individual pieces. If the graph G should be directed we represent the individual pieces as 2-tuples containing the unique nodes, with the first node being the tail of the edge and the second node being the head. If we want an undirected graph it is sufficient to represent the pieces as a set of two unique node IDs. The individual pieces can now be used to create the edge set E of the graph G.

In Table 4.1 and 4.2 we show a simple toy example of the kind of data we can get from OSM. We assume that the way in Table 4.2 represents a two-way street. We are interested combining this with the data in Table 4.1 to construct a road network graph. Table 4.3 shows how the data in Table 4.2 can be broken down into edges. In Figure 4.1 we show the directed graph that represents the road network constructed using data from the toy example. In Figure 4.2 we show the layout of the nodes and edges in the graph.

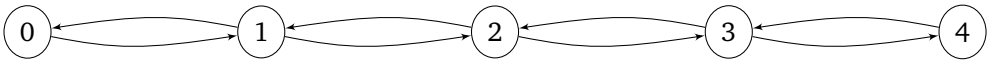


Figure 4.1: Graph constructed using edges and nodes from the toy example.

Edge ID	Nodes	Geometry
0	(0, 1)	LINESTRING (0.549 0.715, 0.603 0.545)
1	(1, 2)	LINESTRING (0.603 0.545, 0.424 0.646)
2	(2, 3)	LINESTRING (0.424 0.646, 0.438 0.892)
3	(3, 4)	LINESTRING (0.438 0.892, 0.964 0.383)
4	(4, 3)	LINESTRING (0.964 0.383, 0.438 0.892)
5	(3, 2)	LINESTRING (0.438 0.892, 0.424 0.646)
6	(2, 1)	LINESTRING (0.424 0.646, 0.603 0.545)
7	(1, 0)	LINESTRING (0.603 0.545, 0.549 0.715)

Table 4.3: Edges derived from nodes and ways in the toy example.

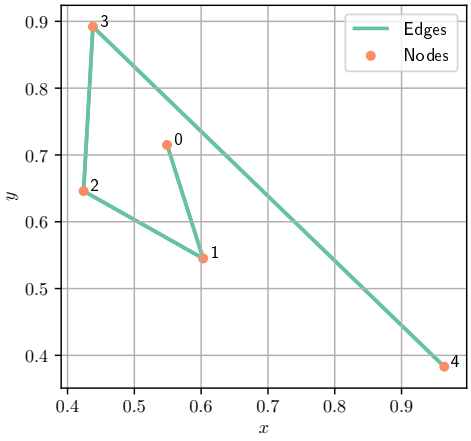


Figure 4.2: Layout of nodes and edges in graph constructed using data from the toy example.

4.2 Simulation

Evaluating the performance of our methods using real-world data would require position measurements that are labeled with the ground truth. One way of doing this would be to take note of the route driven and find the corresponding edges in OSM. This is a very cumbersome process and it is understandable that such data is hard to come by. Perfect, or close to perfect, position measurements obtained with high sampling frequency could have enabled us to create such sufficiently accurate ground truth data, but these kinds of measurements have not been available to the author. In order to enable performance evaluation even without access to such data we simulate routes and the corresponding measurements. This enables us to know the exact route that was travelled and the exact position where the measurements were made. We will also know the conditions the measurements were obtained under. These conditions will in our simulations be dictated by the sampling frequency of the measurements, the speed of the vehicle and the distribution of the measurement noise. This enables us the assess the performance of our methods in a variety of different circumstances, but with the considerable downside of not being able to claim that the results exactly reflect the performance that one would obtain in a real-world situation. Route and measurement simulation requires us to specify the circumstances under which the routes are simulated and measurements are made. We must decide on

- The distribution of the measurements,
- The speed of the object travelling on the road network, v,
- The frequency at which the measurements are made, f, and
- The desired length of the route, d_{min}.

It seems sensible to assume that measurement follows a bivariate normal distribution centered in the true position. This requires us to specify the parameter σ_m associated with the covariance matrix $\Sigma_m = \sigma_m^2 \mathbf{I}_2$. We also assume that, for each $(n_u, n_v) \in E_d$, there exists a categorical distribution over all nodes $n_w \in N$ that follow directly after n_v. This categorical distribution determines how likely it is that a vehicle that is currently on the edge $(n_u, n_v) \in E_d$ continues by moving to a node $n_w \in N$ that follows it. The collection of all these categorical distributions can be thought of as the ground truth for the traffic flow of the road network.

The simulation procedure itself can be split into two parts. The first parts relates to the simulation of the route, i.e. the sequence of nodes that is traversed by the vehicle. The second part relates to simulation of the measurements that are obtained when travelling the route. The route simulation procedure can be seen in Algorithm 6. Before starting the simulation procedure we ensure that a collection of categorical distribution like the one presented in the previous paragraph is in place. This is intended to reflect the real-world situation where certain parts of the road network are more congested than others, with vehicles flowing into one part of the network at greater rate than other parts. At each node one samples the next node from this categorical distribution. Naturally, only directly connected nodes are

possible candidates. The probabilities associated with the various connected nodes is decided ahead of time by sampling from a uniform distribution and dividing each sample by the sum of all samples. We also ensure that a somewhat realistic route is obtained by disallowing U-turns, unless the current node happens to be a dead end. In order to make Algorithm 6 more easily digestible we assume that the function `ChooseNextNode` handles the procedure of sampling from the correct distribution when given the current node n_u and the previous node n_v. The function `EdgeLength` computes the length of an edge $e = (n_u, n_v) \in E_d$.

Algorithm 6: Route simulation algorithm.

Require: Collection of categorical distributions
Require: Graph $G = (N, E)$
$n_{seq} \leftarrow$ empty list
$(n_u, n_v) \leftarrow E$.RandomChoice()
n_{seq}.Insert(n_u)
n_{seq}.Insert(n_v)
$d \leftarrow$ EdgeLength((n_u, n_v))
while $d \leq d_{min}$ **do**
    $n_w \leftarrow$ ChooseNextNode(n_v, n_u)
    $n_u \leftarrow n_v$
    $n_v \leftarrow n_w$
    n_{seq}.Insert(n_v)
    $d \leftarrow d +$ EdgeLength((n_u, n_v))
end while
return n_{seq}

The measurement simulation procedure is shown in Algorithm 7. The algorithm moves along the sequence of linear segments that connect subsequent nodes until a certain distance, given by $\frac{v}{f}$, has been traversed. When this happens one records the current position p on the route and samples a measurement from a bivariate normal distribution centered in p. The function `Interpolate` calculates the point along an edge that is a certain distance away from the tail of the edge. The function `Sample2DNormal` samples from a bivariate normal distribution with a specified mean and covariance. The algorithm is simple in principle, but handling edge cases makes it a bit more involved.

4.3 Experimental set-up

4.3.1 Overview

In this section, Section 4.3, we will present details about the experiments that are conducted. The overall goal of the experiments is to evaluate the performance of the methods presented section Chapter 3. The methods in question will be referred to as

Algorithm 7: Measurement simulation algorithm.

Require: Node sequence n_{seq}
DistanceBetweenMeasurements $\leftarrow \frac{v}{f}$
$m_{seq} \leftarrow$ empty list
$N \leftarrow n_{seq}$.Length()
RemainingSpace $\leftarrow 0$
for i=2:N **do**
  $n_{i-1} \leftarrow n_{seq}[i-1]$
  $n_i \leftarrow n_{seq}[i]$
  SpaceOnEdge $\leftarrow$ EdgeLength((n_{i-1}, n_i))
  AvailableSpace = SpaceOnEdge
  RequiredSpace $\leftarrow$ DistanceBetweenMeasurements $-$ RemainingSpace
  DistanceCovered $\leftarrow 0$
  while *AvailableSpace* $\geq$ *RequiredSpace* **do**
    $\Delta d \leftarrow$ DistanceCovered + RequiredSpace
    $p \leftarrow$ Interpolate($(n_{i-1}, n_i), \Delta d$)
    $m \leftarrow$ Sample2DNormal(p, Σ_m)
    m_{seq}.Insert(m)
    DistanceCovered $\leftarrow$ DistanceCovered + Δd
    AvailableSpace $\leftarrow$ AvailableSpace $-$ RequiredSpace
    RequiredSpace $\leftarrow$ DistanceBetweenMeasurements
    RemainingSpace $\leftarrow 0$
  end while
  RemainingSpace $\leftarrow$ RemainingSpace + AvailableSpace
end for
return m_{seq}

- the benchmark (BM) method,
- the simple (S) method,
- the augmented (A) method,
- the augmented and restricted (AR) method, and
- the augmented, trained and restricted (ATR) method.

The BM method uses the benchmark estimate defined in Equation (3.21). The S, A, and AR methods are HMM-based methods, and is defined using the components introduced in Section 3.2. An overview of the components used for three of these methods is shown in Table 4.4. The ATR method is identical to the AR method, but

Method	State space	Transition probability	Emission probability	Initial probability
S	$\mathcal{X}_s$	$a_s(x, y)$	$b_s(\mathbf{z} \mid x)$	$\pi_s(x)$
A	$\mathcal{X}_a$	$a_a(x, y)$	$b_a(\mathbf{z} \mid x)$	$\pi_a(x)$
AR	$\mathcal{X}_a$	$a_c(x, y$	$b_a(\mathbf{z} \mid x)$	$\pi_a(x)$

Table 4.4: The HMM-based methods with fixed transition probabilities.

uses transition probabilities that are learned using the *Baum-Welch* algorithm. It is therefore not "fixed" in the same sense as the other methods, but depends on the data that is used to estimate the transition probabilities. The reason for including the benchmark method in our experiments is to enable comparison between it and the HMM-based methods.

The experiments presented in this thesis can be divided into three main parts.

- Searching for appropriate values of γ, introduced in Equation (3.4), and σ_p, used in Equation (3.13) and (3.14).
- Estimating the transition probabilities using the *Baum-Welch algorithm*.
- Evaluating the performance of the methods when appropriate values of σ_p and γ are chosen.

The one thing that will remain unchanged throughout all parts of the experiments is the road network. The road network is restricted to only include the largest (weakly, in the case of a directed graph) connected subgraph of the graph contained within the bounding box given by

$$(x_{min}, y_{min}, x_{max}, y_{max}) = (10.415, 63.417, 10.428, 63.423). \tag{4.1}$$

This bounding box is located within a residential area between the NTNU Trondheim campus "Gløshaugen" and the neighbourhood Tyholt, Trondheim. The resulting graph, with nodes shown at their correct position in the UTM zone 32 coordinate system, is seen in Figure 4.3.

The details of the parameter search is presented in Section 4.3.2. The parameter search is conducted for the S, A and AR methods. The parameter search consists of the following steps.

1. Simulate a number of routes on the road network.

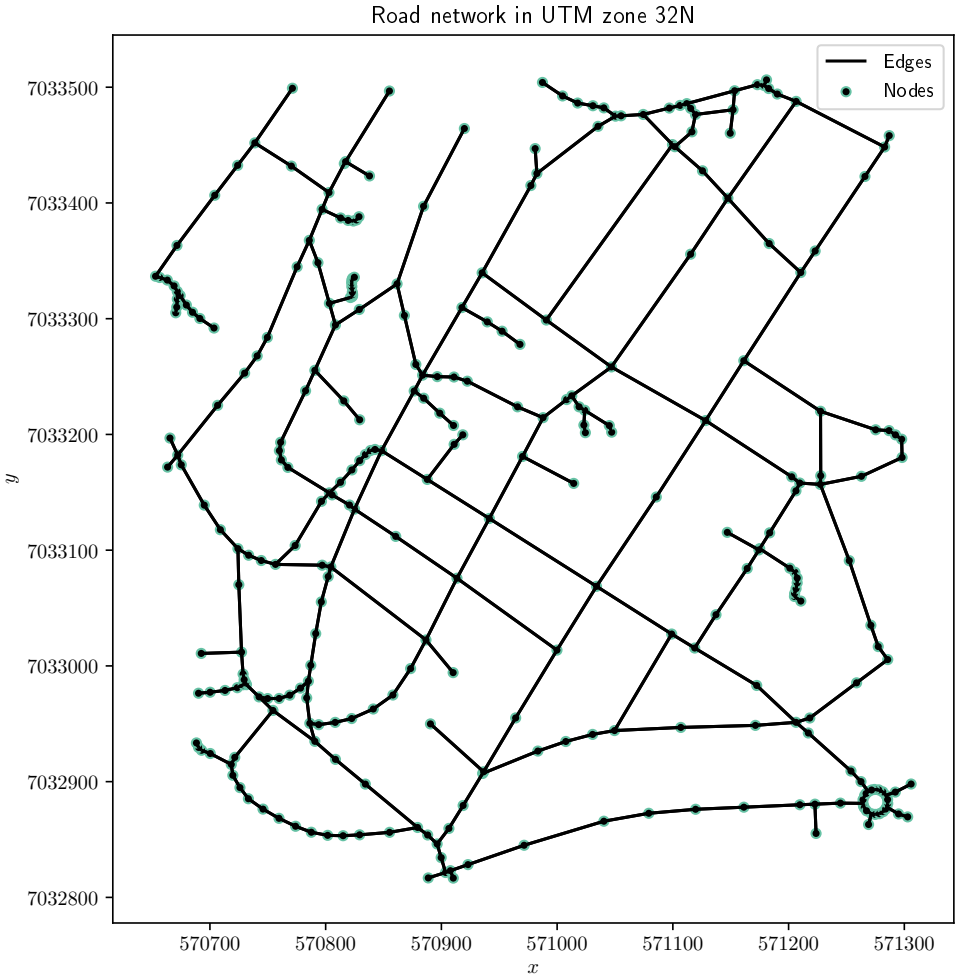


Figure 4.3: The road network used in the experiments. Nodes shown as circles with green edges. Edges shown as straight, black lines connecting the nodes.

2. Simulate a number of measurement sequences for each route. Each measurement belongs to a specific *measurement group*, which is uniquely identified by the simulation parameters used for the group.
3. Compute route estimates for each of the measurement groups, one time for each unique combination of the method parameters γ and σ_p.
4. Compute the Hausdorff distance and accuracy for each of the estimated routes.
5. For each measurement group, identify the method parameters that lead to the best performance.

The exact procedure used for the parameter estimation is presented in Section 4.3.3. The purpose of the parameter estimation is to estimate the transition probabilities used in the ATR method. It is done by first simulating a large number of routes and measurements, and then repeatedly applying the *Baum-Welch algorithm* until the increase in the mean log-likelihood of the measurement sequence tapers off.

The set-up of the final performance evaluation is presented in Section 4.3.4. The idea is similar to that of the parameter search. The difference is that we use a larger number of routes and measurements, and only use methods with optimal parameter choices. The optimal parameters were identified during the parameter search. We now also include the ATR method, using the parameters learned in Section 4.3.3. The objective of the performance evaluation is to quantify how the methods perform under different circumstances, and hopefully enable us to arrive at a conclusion about the viability of the different methods.

We now turn our attention to the details of the route and measurement simulation procedure. When given a simulated route, as produced by Algorithm 6, one can simulate a measurement sequence based on the route using Algorithm 7. The route simulation requires us to specify the length of route, d_{min} and a categorical distribution that dictates how the algorithm behaves when deciding between several possibilities at a junction. The measurement simulation requires us to specify the following parameters:

- The parameter σ_m of the measurement distribution around the true location p, $\mathcal{N}_2(p, \sigma_m^2)$, and
- The distance between measurements, given by $\frac{v}{f}$.

The values shown in Table 4.5 are sensible values of the sampling frequency that one might expect in to see in real-world scenarios. We choose to fix the speed at 30 $\mathrm{km\,h^{-1}}$, giving us $v = 8.33\ \mathrm{m\,s^{-1}}$.

	f
High frequency	1
Low frequency	$\frac{1}{10}$

Table 4.5: The sampling frequencies used in the experiments.

	σ_m
High variance	3
Low variance	8

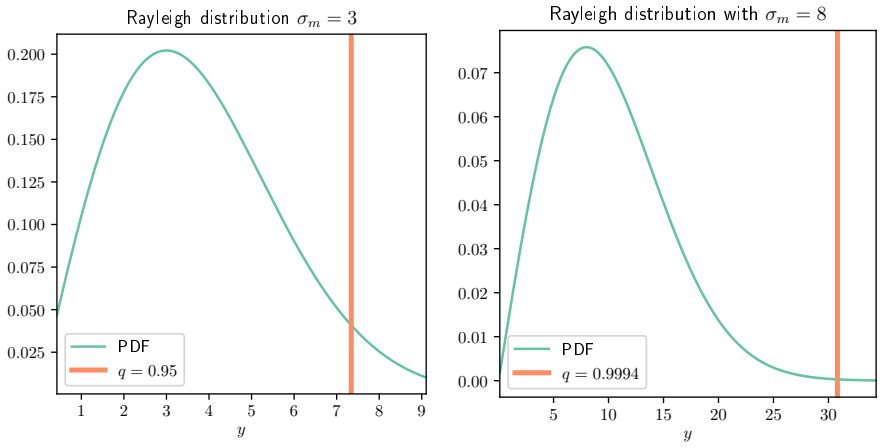
Table 4.6: The standard deviation of the individual components of the simulated measurements used in the experiments.

Sensible values for σ_m are somewhat harder to pin down, seeing as we want them to be at least somewhat similar to what one would observe in the real-world. In order to gain knowledge about how GPS measurements behave in the real world one can consult the *Standard Positioning Service Performance Standard* released in 2008 (SPSPS08) [21]. The SPSPS08 specifies that the 95th percentile of the distribution of observed user range error (URE) for signals in space (SIS) should not exceed 7.8 metres. When assuming that the GPS measurement is normally distributed around the true position we get that the distribution of the Euclidean distance between the true position and the measurement follows a Rayleigh distribution with scale parameter σ_m. This scale parameter is the same as σ_m used in $\Sigma_m = \sigma_m^2 \mathbf{I}_2$. The value of the scale parameter that is required to meet the requirement on the 95th percentile is $\sigma_m \approx 3$, as shown in Figure 4.4a. This requirement is specified for SIS, and we therefore assume that it is ambitious for signals received in urban areas from within a moving vehicle. We will therefore consider the value $\sigma_m = 3$ to be on the lower end of values that we can realistically expect. Another accuracy requirement specified in SPSPS08 is that 99.94% of URE should be less than 30 metres. This corresponds to a scale parameter $\sigma_m \approx 8$, see Figure 4.4b. The conditions this requirement should hold for is more lenient than the previous one. Because of this, we assume that $\sigma_m = 8$ is among the higher values we can realistically observe. The chosen values are shown in Table 4.6.

We will refer to each unique combination of the values in Table 4.5 and Table 4.6 as a "measurement group". The value of d_{max} could take on a wide variety of values, but due to the limited size of our road network it seems sensible to keep the length of each individual route somewhere between a few hundred metres and a couple of kilometers. There is no reason to expect the value d_{max} to affect our results directly, but we do expect the total number of measurements to have an impact on the quality of the estimates obtained using the *Baum-Welch algorithm*.

4.3.2 Parameter search

In this subsection we will present the methodology used to assess how the choice of σ_p and γ impact the performance of our methods under different circumstances. The circumstances are dictated by the choice of simulation parameters σ_m and f. We will use the results to identify which values of the method parameters σ_p and γ are suitable for the different methods in different circumstances. It will in some cases be hard to identify the optimal parameter choice, seeing as



(a) **(b)**

Figure 4.4: Rayleigh distribution with two different choices of scale parameter σ_m.

the two performance metrics only capture certain aspects of the quality of the route estimate. It is also difficult to know the optimal "trade-off" between the two metrics, i.e. knowing how much of an increase in Hausdorff distance one is willing to tolerate if it also increases the accuracy by a certain amount. However, we will in general consider the Hausdorff distance to be the most important of the two metrics, seeing as we believe this metric to better capture the quality of the estimate as a whole. For the unrestricted methods (S, A, and AR) one could obtain a high accuracy, but still observe a route estimate that deviates significantly from the true route. This can happen because the accuracy does not take into account how "far away" the estimated state is from the true state. The Hausdorff distance, on the other hand, does take into account how much the estimated route deviates from the true route.

We take the opportunity to note that, in a real-world scenario, it would not be possible to identify the "best performing" parameter choices without access to the ground truth. We do, however, believe that good initial parameter choices can be arrived at by combining knowledge about the data-generating process with a qualitative inspection of estimates obtained with various parameter choices. One could also combine this with insights obtained from experiments like those presented here, or by performing new simulations where the simulations are more similar to what one believes to be the real-life situation. We, therefore, think that it is possible to identify good parameter choices even without access to ground truth.

The first step in setting up the parameter search is the route simulation. We simulate 15 distinct routes on the road network. Each route is simulated using $d_{min} = 1000$ m. This ensures that each route is at least 1km long, and typically not much longer. This is done using the procedure described in Algorithm 6. Three of

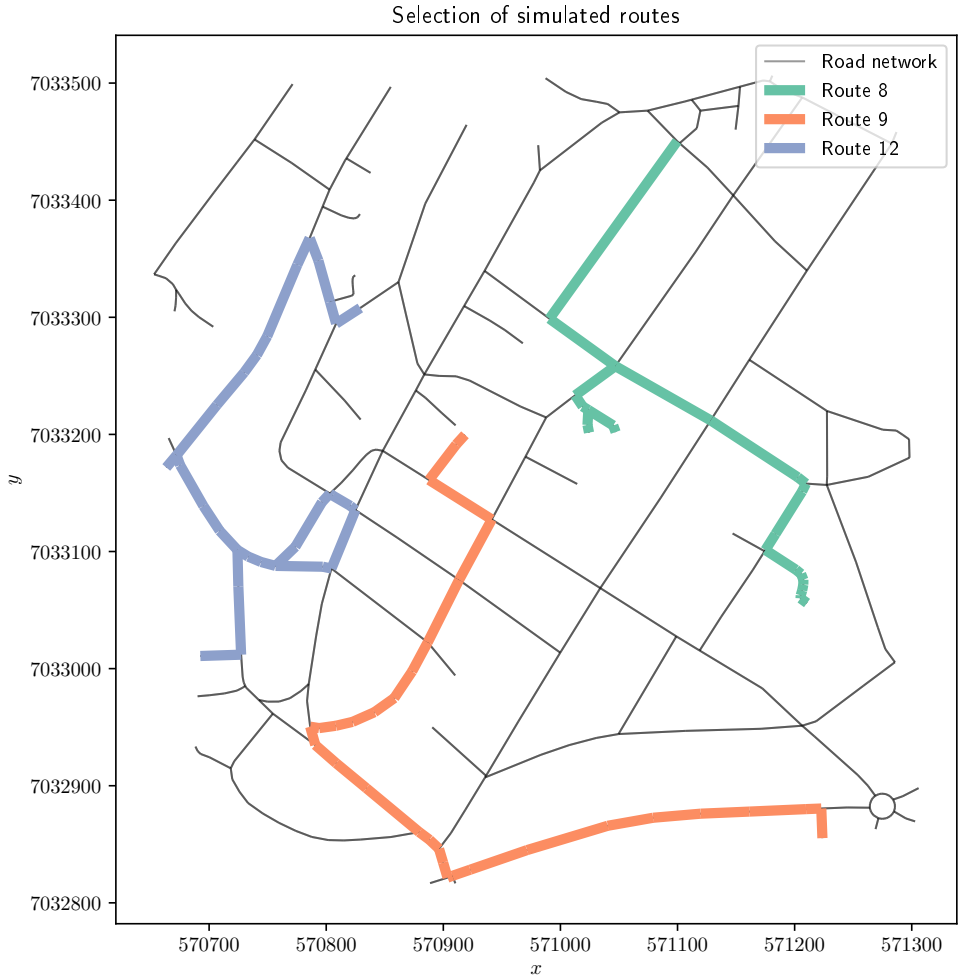


Figure 4.5: A random selection of 3 out of the 15 simulated routes used in the parameter search.

the resulting routes are shown in Figure 4.5. For each of the 15 routes we simulate a number of measurement sequences. The measurements are simulated four times for each route, each time with a unique combination of the simulation parameters listed in Table 4.5 and 4.6. We will refer to a group of measurements that have been simulated with identical simulation parameter choices as a *measurement group*. Each measurement group will consist of 15 measurement sequences, one for each route. A subset of each measurement group is shown together with its corresponding route in Figure 4.6.

For each of the measurement sequences in each measurement group we estimate the route using the methods in Table 4.4, i.e, the S, A and AR methods. There is one notable exception with regards to the AR method. The AR method will not work for the measurement groups where the sampling frequency f is

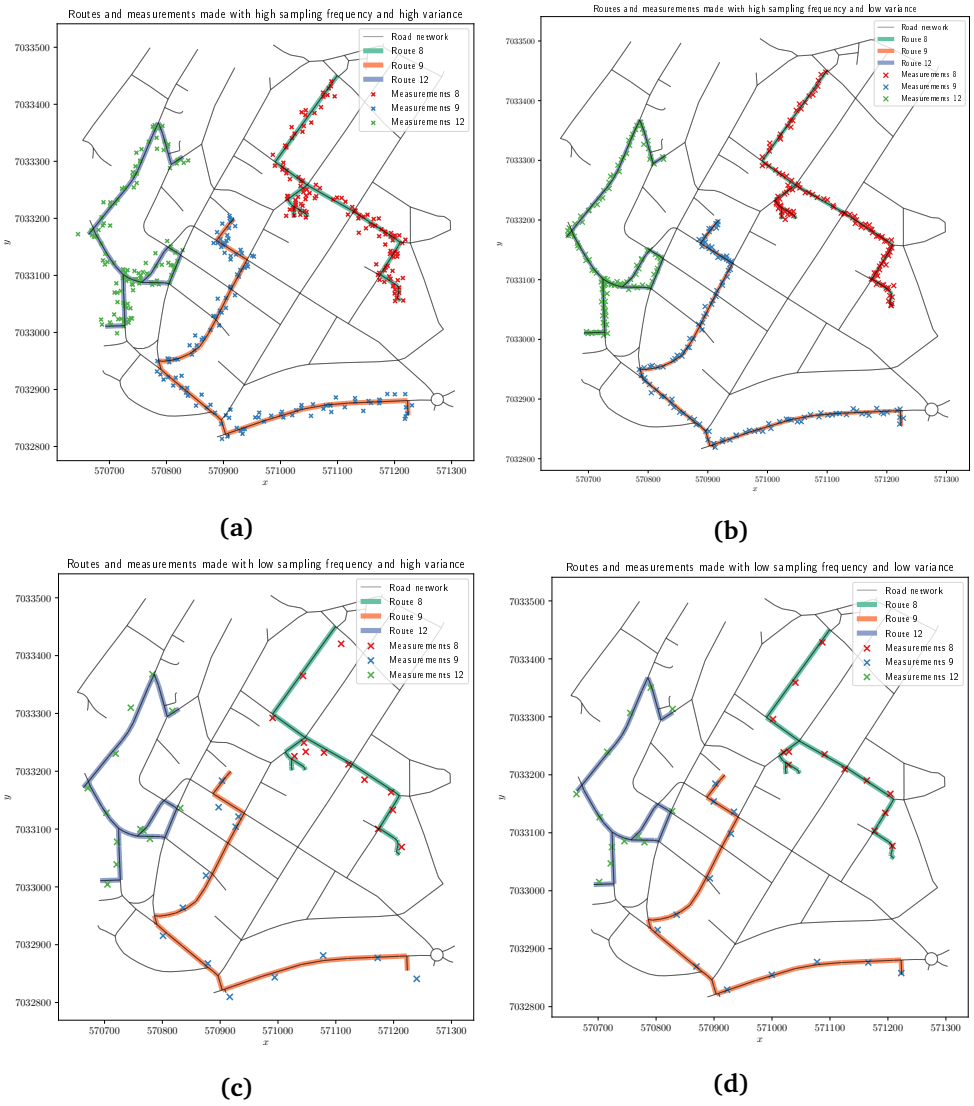


Figure 4.6: Subsets of each measurement group used in the parameter search, shown together with their associated routes.

| σ_p | 2 | 4 | 6 | 8 | 10 |

Table 4.7: The possible values of σ_p used in the emission probability.

| γ | $\frac{1}{100}$ | $\frac{1}{10}$ | $\frac{1}{2}$ |

Table 4.8: The possible values of γ used in the transition probability.

equal to $\frac{1}{10}$. The reason for this is that, because the number of hidden states is equal to the number of observations, the length of the hidden state sequence will typically not be long enough to represent the traversed node sequence when an observation is only added every 10th second. We will therefore only consider the "simple" and the "augmented" HMM variants for measurement groups where $f = \frac{1}{10}$. The benchmark method is also always included. There are no decisions to make in regards to the benchmark method, but it is included to provide context for the performance of the other methods. After obtaining the route estimates we proceed by calculating the Hausdorff distance and accuracy for each of the measurements sequences. Each of the methods require that that we fix the decay parameter γ used in the corresponding transition probability, as well as the σ_p used in the emission probability. It is not computationally feasible to run experiments for a very dense grid of values, but using combinations of values presented in Table 4.7 and 4.8 is expected to give significant insight into how the performance depends on parameter choices. These combinations of values will be tested for all the HMM-based methods that are used in this part of the experiment.

The values in Table 4.7 are chosen so that, for each measurement group, at least one of the choices end up assigning a high probability density to distances (between measurement and true edge) that seem likely for the specified measurement group. There is no exact procedure for determining this, but knowing the parameters of the measurement variances σ_m made it possible to reason about the range of observed distances that would make sense. The half-normal distributions we get for each of the five candidates of σ_p is shown in Figure 4.7. The same line of reasoning is used for deciding on the values γ in the exponential distribution. Since we know the distance between each measurement for the various measurements groups we can identify values of γ that assign high probabilities to distances we expect to observe in at least one of the measurement groups. Plots of the resulting exponential distributions are shown in Figure 4.8.

4.3.3 Parameter estimation

The next part of our experiments is related to the estimation of the transition probabilities using the *Baum-Welch algorithm*. This is a computationally expensive method that requires considerable amounts of data before one can expect to see results that reflect the true situation. It is considered beneficial to start off the *Baum-Welch algorithm* with initial parameters as close to the true situation as possible. There is, naturally, no sure way of determining whether the initial parameters are

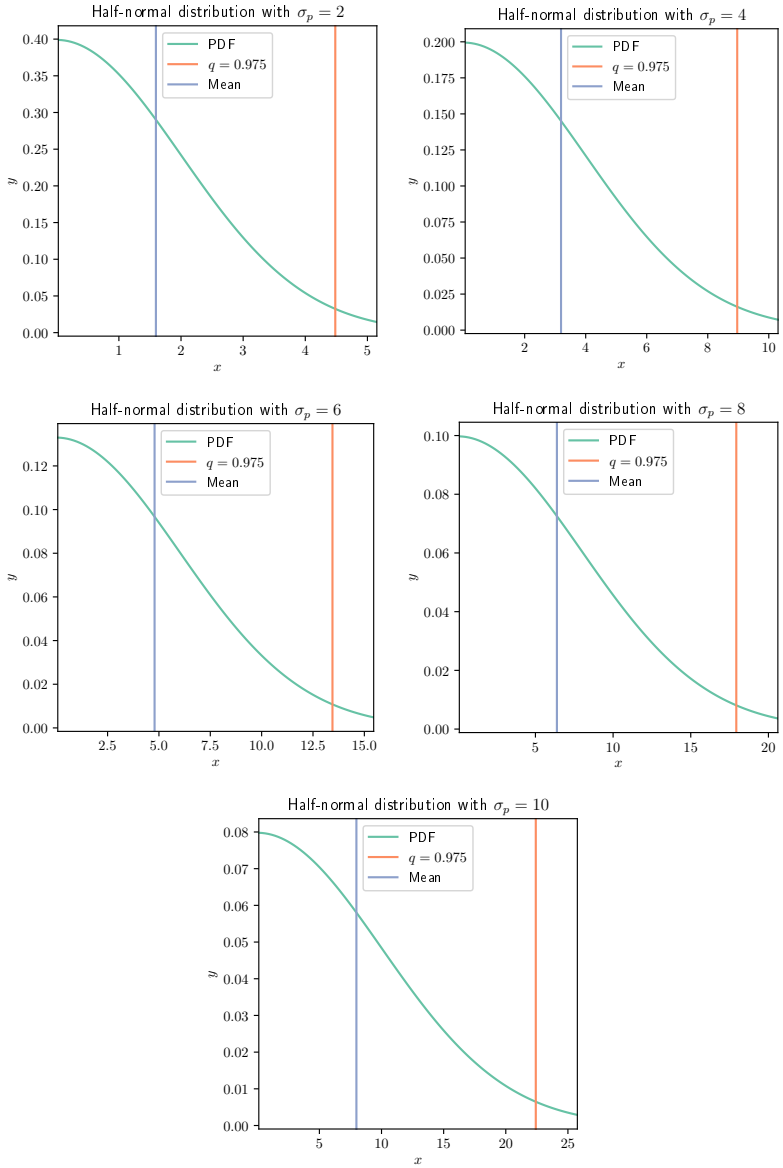


Figure 4.7: Half-normal distributions for the values of σ_p in Table 4.7.

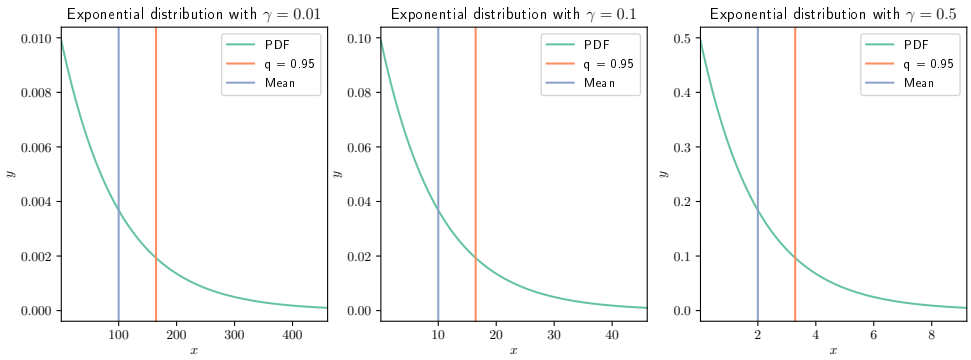


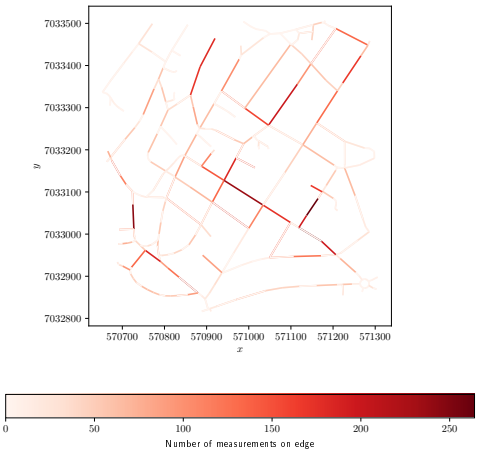
Figure 4.8: Exponential distribution for the values of γ shown in Table 4.8.

close to the truth, but we assume that the parameter values that are arrived at in the parameter search, detailed in Section 4.3.2, are reasonable choices.

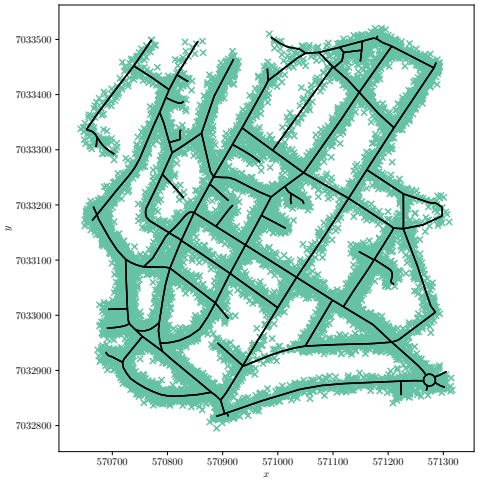
We choose to only apply the *Baum-Welch algorithm* to the "augmented and restricted" HMM formulation. The reason for doing this is twofold; it limits the scope of our experiments and the time required to run our experiments, and we also believe that the resulting transition probabilities has the potential to provide worthwhile insights about the behaviour of underlying system. We recall that the "augmented and restricted" method fixes transition probabilities to zero for transitions between unconnected states. This ensures that non-zero transition probabilities for a given state can be interpreted as the probability that a vehicle will make a specific "choice" when confronted with several options. The estimated transition matrix $\hat{\mathbf{A}}$ can therefore, if the estimates are accurate, be interpreted as estimate of the traffic flow on a road network. Such insights can likely not be obtained through the use of an "unrestricted" formulation.

The amount of data used in Section 4.3.2 is too low for the purpose of parameter estimation using the *Baum-Welch algorithm*. It is also undesirable to learn the transition probabilities using the exact same data as we used when deciding on initial parameters. This is because the initial parameters were chosen based on how well they performed on the data used in the parameter search, but we wish to assess whether the methods, including the method with estimated transition probabilities, perform well on a new data set as well. We therefore simulate 50 new routes, with each route having $d_{min} = 2500$ m. We simulate measurements using the high sampling frequency $f = 1$ combined with either of the values in Table 4.6. We assign the measurements to different measurement groups, as we did in Section 4.3.2. We do not consider measurements made with low sampling frequency because, as mentioned earlier, the "augmented and restricted" variant only works for situations with high sampling frequency. The two measurement groups are visualized in Figure 4.9 and Figure 4.10. In these figures we see how many observations has been collected at the various edges, as well an illustration of all the measurements within the specific measurement group.

It is not entirely straightforward to evaluate the quality of the estimated trans-

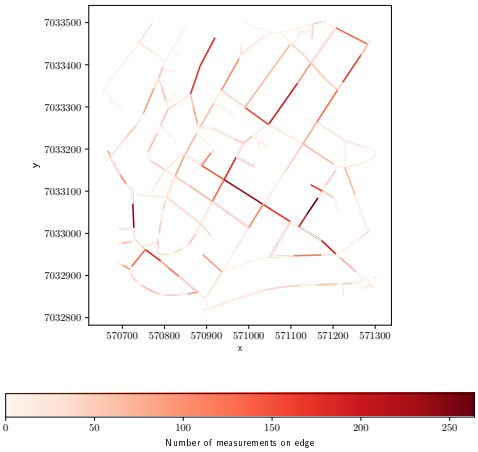


(a) Count of the number of measurements made at each edge.

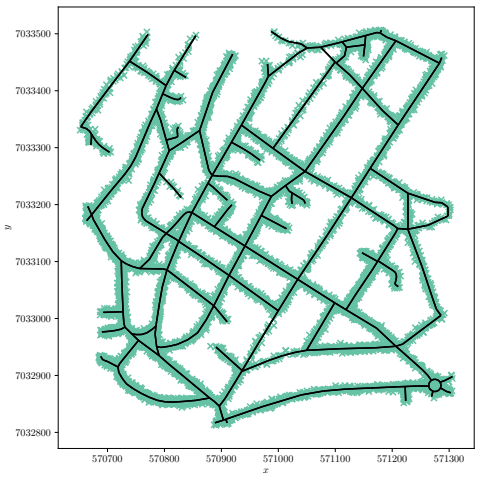


(b) All measurements in the group.

Figure 4.9: Overview of the measurement group associated with high sampling frequency and high variance.



(a) Count of the number of measurements made at each edge.



(b) All measurements in the group.

Figure 4.10: Overview of the measurement group associated with high sampling frequency and low variance.

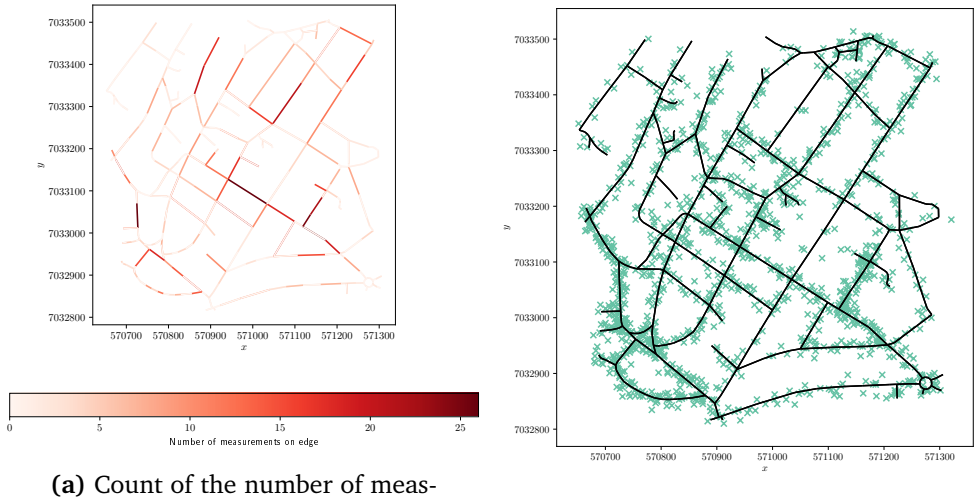
ition probabilities. It would be desirable to somehow assess how the estimated transition matrix compares to the categorical distributions used to determine the traffic flow on the road network. It has, however, been difficult to come up with a method for doing this that we are confident is correct. It also has the downside of being entirely unrealistic in a real-world scenario, seeing as the ground truth about the traffic flow is not known. Instead, we choose to evaluate the estimates by using information that could be available in real-world scenarios. This consists of looking at how the probability of staying in the same state depends on the length of the road segment. We know that the behaviour of both real-world and simulated systems is such that a vehicle will remain on long road segment for a longer time than it would on a short road segment (given that the speed of the vehicle remains constant). It is therefore reasonable to expect higher "self-transition" probabilities for longer road segments.

Another way of getting an idea about the quality of the estimated transition probabilities is to look at the performance of the method that uses these transition probabilities, namely the ATR method. This is done in Section 5.3.

4.3.4 Performance evaluation and comparison

The final part of our experiments concerns itself with performance evaluation. In Section 4.3.2 we presented a procedure for identifying a combination of parameters that yield performance that is, if not best, at least not decidedly worse than any of the alternatives. We are, for simplicity, referring to the parameter combination that we think is optimal as the "optimal parameter combination". In Section 4.3.2 we took these parameters as initial parameters and used the *Baum-Welch algorithm* to update the transition probabilities. Having done this, we now know the transition probabilities that should be used in various scenarios (measurement groups) for the distinct methods presented in this thesis. We now wish to evaluate the performance of these methods when each of the methods is used with the optimal parameter choice for the measurement group in question. Note that not all of these methods can be used for all of the measurement groups. As mentioned in Section 4.3.2, the "restricted" HMMs are not suitable for measurements made with low sampling frequencies. We therefore only consider the BM, S and A methods for these measurements groups. We use all the presented methods for the remaining measurement groups.

The evaluation is done using the measurement groups introduced in Section 4.3.3. Each of these measurement groups consists of 50 measurement sequences simulated on routes with $d_{min} = 2500$ m. In addition to the measurement groups introduced in Section 4.3.3 we simulate two new groups with low sampling frequency combined with either of the values in Table 4.6. The measurement groups are shown in Figure 4.11 and 4.12. The number of routes is still 50, and the value of d_{min} used for the route simulation is still $d_{min} = 2500$ m. We recall that the transition probabilities in the case of the ATR method has been learned using the measurement groups with high sampling frequency mentioned here. This is



(a) Count of the number of measurements made at each edge.

(b) All measurements in the group.

Figure 4.11: Overview of the measurement group associated with low sampling frequency and high variance.

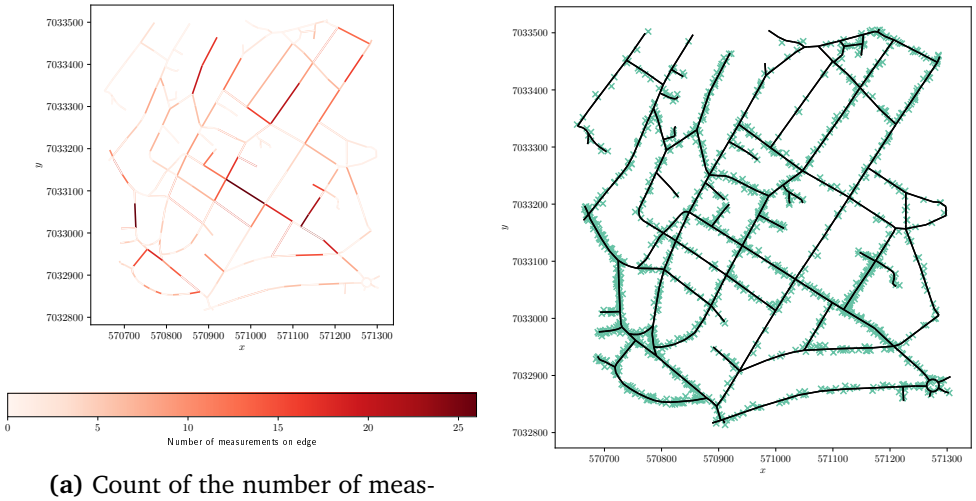
not an issue, since the *Baum-Welch algorithm* only requires observations, and is not exposed to the ground truth in any way.

After estimating the state sequences and routes for each of the measurements in each of the measurement groups we calculate both the accuracy and the Hausdorff distance between the ground truth and the estimate. This allows us to compare the different methods in a quantitative manner.

4.4 Implementation

4.4.1 Overview

Conducting the experiments presented in Section 4.3 requires implementations of the algorithms presented in Chapter 2, the methods and techniques presented in Chapter 3, and the simulation procedures presented in Section 4.2. This section serves as a presentation of the software that has been developed to provide such implementations. The range of problems that must be solved by the software is quite diverse. The software has to be able to interact with OSM in some way. It also has to enable us to use the data from OSM to construct a graph representation of the road network. This must be used create the various state space representations and define the transition-, emission-, and initial probabilities. We also require a simulation procedure that lets us generate routes and measurements sequences. As soon as all this is in place one can apply the techniques associated with the HMM, such as the *Viterbi algorithm* and the *Baum-Welch algorithm*. This requires that all of the functionality presented in Chapter 2 is implemented in a numerically stable way.



(a) Count of the number of measurements made at each edge.

(b) All measurements in the group.

Figure 4.12: Overview of the measurement group associated with low sampling frequency and low variance.

The functionality required to apply the map matching methods has been split into two parts. The first part is the general HMM functionality. This has been assembled into a software library named `hmmpy` (**H**idden **M**arkov **M**odels in **Py**thon). It enables the user to supply an arbitrary state space as well as functions that define the transition-, emission- and initial probability. When an object is constructed with these components as input one has immediate access to methods that implement the *Viterbi algorithm*, the *forward algorithm*, the *backward algorithm* and the *Baum-Welch algorithm*. The library also supports HMMs with discrete observation spaces and Gaussian observations, even though this has not been needed for this thesis. The library is open-sourced with an MIT license, available on GitHub and published as a package to the Python Package Index (PyPI). It should be mentioned that the library is still in its infancy, and it is possible that modifications and extensions will be made after the finalization of this thesis. The library depends on `NumPy`, `SciPy` and `tqdm`.

The second part is related to obtaining and organizing data from relevant sources, keeping track of and ensuring consistency with regards to coordinate systems, constructing the graph, defining the components of the HMM, keeping track of and organizing observations, translating the output of the *Viterbi algorithm* into route estimates, and more. This library is in a more fragile state than `hmmpy`, and is therefore not published to PyPI. It is, however, available on *GitHub* [5] with an open-source license, in case any of the functionality could either be of help or serve as inspiration to someone working on a similar problem. The library will be referred to as `tmmpy` (**T**oolkit for **M**ap **M**atching in **Py**thon) to indicate its relationship to `hmmpy`. The library has multiple dependencies. The most important

[5] https://github.com/klaapbakken/tmmpy

ones are `GeoPandas` (and `Pandas`), `NetworkX`, `Shapely` and `NumPy`.

4.4.2 Hidden Markov models in Python

The Python package `hmmpy` implements three distinct classes that are intended to be exposed to the user. The classes are:

- `HiddenMarkovModel`,
- `DiscreteHiddenMarkovModel`, and
- `GaussianHiddenMarkovModel`.

The classes differ in what they assume about the observations. The first class, `HiddenMarkovModel`, supports any emission probability, but does as a consequence not have any procedure in place for estimating parameters related to the emission probability. This is not a problem for the situation considered in this thesis, but could be an issue in other use cases. The two next classes solve this by assuming either Gaussian distributions or discrete distributions for the observations. It is for these distributions possible to derive update equations within the EM framework presented in Section 2.6 [15]. This enables estimation of either the discrete emission probabilities or the mean and covariance of the various states.

The input to the constructor method of these three classes varies slightly. All three require that the state space is supplied as a list of states. They also require functions that represent the transition probabilities and initial probabilities. The supplied transition probability function should take two objects from the state space and return the probability of transitioning from the first object to the second. The initial probability function should return the initial probability of its only argument, which should be a element from the list of supplied states. The final argument depends on which object is being created. The object `HiddenMarkovModel` requires a function that returns the emission probability of a certain observation when given the observation and the state. The object `GaussianHiddenMarkovModel` requires a `NumPy` array of the initial values of the mean and covariance for the various states. The object `DiscreteHiddenMarkovModel` requires a list of "symbols", which is the observation space, and a function that returns the probability of a supplied symbol when given the state.

After an object has been instantiated one has access to methods such as `decode`, which computes the MAP estimate using the *Viterbi algorithm* and returns the estimated hidden state sequence. Another imporant method is `reestimate`, which runs the *Baum-Welch algorithm* a given number of times in and updates the model parameters at each iteration.

4.4.3 Toolkit for map matching in Python

The main idea behind `tmmpy` is to collect all functionality required for doing map matching in a single library. This ranges from functionality bringing data from different sources into the same format to leveraging the functionality of `hmmpy` for map matching purposes. The classes in the library can be interpreted as parts of a

larger pipeline. The first group of classes is the data source classes. These classes all collect data from some source and converts it to a format expected by the other classes in the library. At the time of publication there exists classes for interfacing with a PostGIS database, the Overpass API of OSM and the NVDB API. The second group of classes consists of the road network classes. The main purpose of these classes is to represent the road network as a graph. There are two classes within this group; one for representing the road network as an undirected graph and one for representing the road network as a directed graph. The third group of classes is the state space classes. These classes uses a road network object to construct a state space. There are two classes in this group; one for the creating an augmented state space, and one for creating the simple state space. The constructor for the augmented state space class requires a directed graph object as input, while the simple state space class requires an undirected graph object as input. Finally, there is a single map matching class. This class accepts a state space object as input, as well as arguments specifying the sort of HMM that should be constructed. The class supports both state spaces, and has logic in place that ensures that suitable HMM components are used. The HMM object from `hmmpy` is automatically constructed when a map matching object is constructed. The map matching class also contains some functionality for various post-processing tasks.

The previous paragraph presented the most essential parts of the library. In addition to these there are a few miscellaneous classes and functions. Most of these are not worth mentioning, but there are a few notable exceptions. The first one is the simulation class. This class contains implementations of the route- and measurement simulation algorithms presented in 4.2, and can be created by passing a road network object to the class's constructor. The final class that deserves a mention is the observation class. It is suitable for representing sequences of GPS measurements, and is used together with the map matching class to ensure that the underlying HMM object receives observation sequences in the format it expects.

Chapter 5

Results

This chapter is dedicated to reviewing the results of the experiments presented in Chapter 4. We will present the results in the order they were introduced there. We start off with the results of the parameter search, where we identify the optimal value of the method parameters. We continue by exploring the effect of the parameter estimation conducted using the *Baum-Welch* algorithm. The last section is a presentation of the the findings from the final performance evaluation, where every method presented in this thesis is evaluated and compared.

5.1 Parameter search

Overview

In this section we will present the results of the experiments detailed in Section 4.3.2. We will consider how the performance of the various methods depend on the parameter choices, and, for each of the methods, determine the most suitable parameter choices for each of the measurement groups. Before we proceed with a quantitative analysis of the estimates we perform a visual inspection of the estimates obtained for the selection of routes shown in Figure 4.5. The estimates are obtained using the AR method with measurements from the measurement group with high sampling frequency and low variance. This is the measurement group in which we expect to observe the best performance. We use the method parameters $\sigma_p = 4$ and $\gamma = \frac{1}{10}$. The measurements are shown in Figure 4.6b. The resulting estimates are shown in Figure 5.1. These estimates seem to be of high quality, but a visual inspection like this does not give us the full picture, seeing as it does not tell us anything about the ordering of estimated states. A thorough understanding of the performance of the methods can only be achieved through a quantitative analysis. The remainder of Section 5.1 is devoted to this.

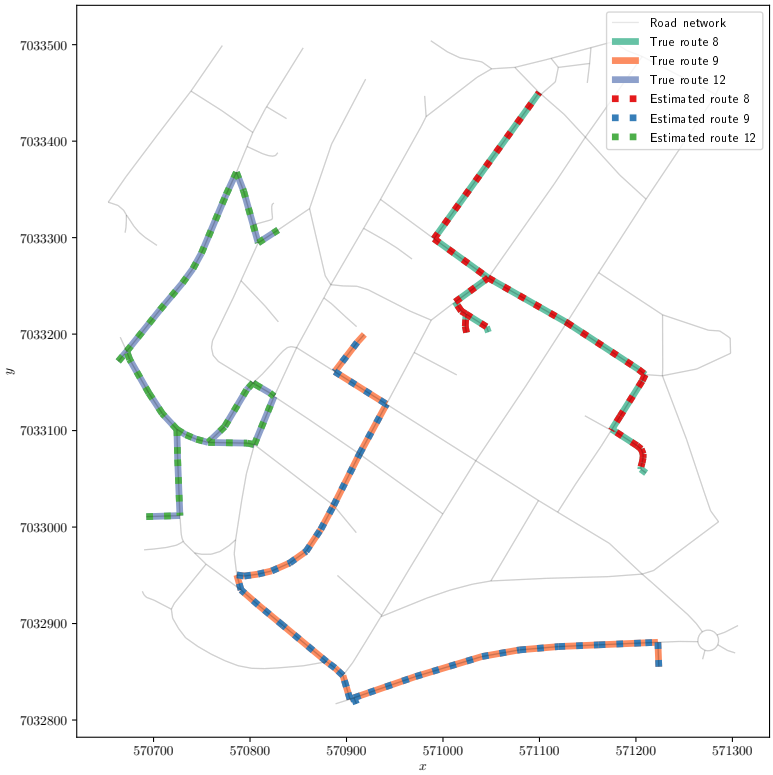
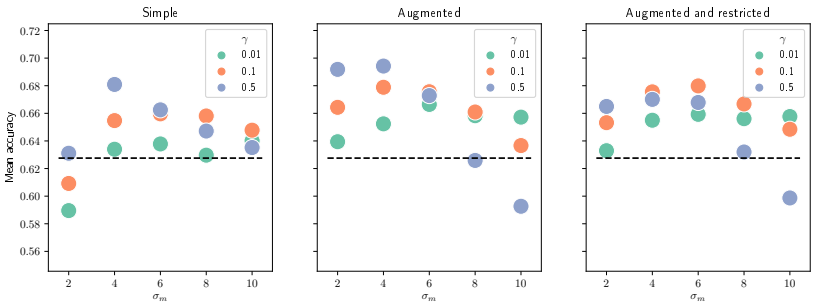
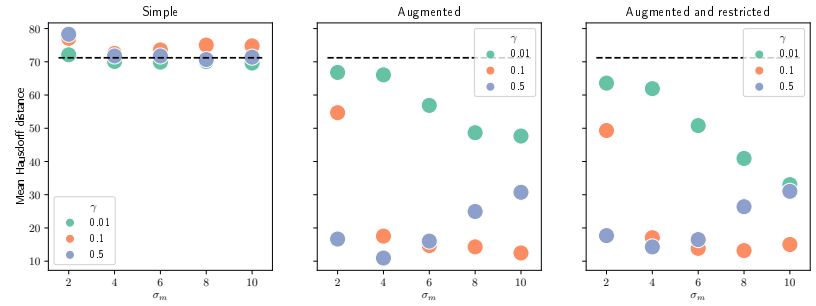


Figure 5.1: Routes estimated using the AR method with $\sigma_p = 4$ and $\gamma = \frac{1}{10}$ for high sampling frequency and low variance measurements.



(a) Mean accuracy for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.



(b) Mean Hausdorff distance for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.

Figure 5.2: Mean accuracy and mean Hausdorff distance for the various parameter choices in the measurement group with high sampling frequency and high variance.

High sampling frequency and high variance

We start off by considering the measurement group with high sampling frequency and high variance. We recall that there are 15 measurements in each measurement group, with each measurement in a group corresponding to a unique route. Our interest lies in determining which combination of the parameter choices in Table 4.7 and 4.8 yield the best performance. A plot showing the mean accuracy for each of the unique parameter combinations is shown in Figure 5.2a. The performance of the benchmark method is shown using a dashed black line. We immediately observe that each of the HMM-based methods outperform the benchmark method when appropriate parameters are chosen. We present a similar plot, but with mean Hausdorff distance instead of mean accuracy, in Figure 5.2b. A summary of the best performing parameter combinations is shown in Table 5.1. Each entry in the table corresponds to a parameter combination that either maximized the mean accuracy or minimized the mean Hausdorff distance for one of the methods.

We now observe that, while the simple method is able to obtain a higher accuracy than the benchmark method, it is not able to outperform the benchmark

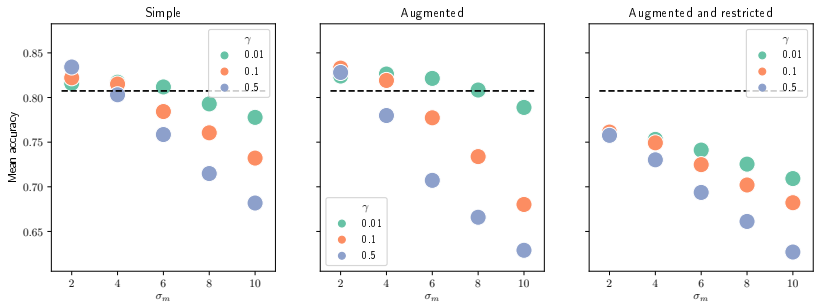
Method	σ_p	γ	Mean accuracy	Mean distance
Benchmark	-	-	0.628	71.208
Simple	4.000	0.500	0.681	71.755
Simple	10.000	0.010	0.640	69.626
Augmented	4.000	0.500	0.694	10.945
Augmented	4.000	0.500	0.694	10.945
Augmented and restricted	6.000	0.100	0.680	13.863
Augmented and restricted	8.000	0.100	0.667	13.185

Table 5.1: The best performing parameter combinations in terms of either mean accuracy or mean Hausdorff distance. The measurement group has high sampling frequency and high variance.

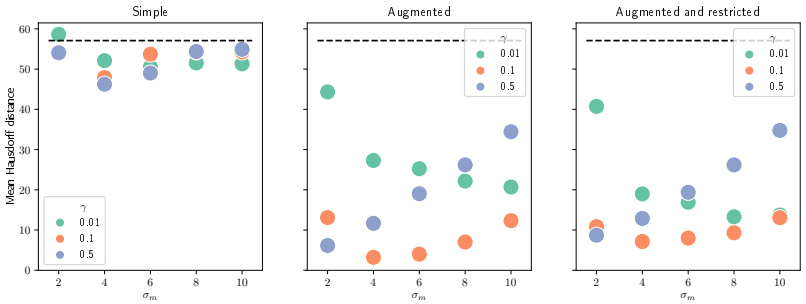
Method	σ_p	γ
S	4	0.5
A	4	0.5
AR	8	0.1

Table 5.2: Optimal parameters for the various methods when applied to measurement sequences obtained using high sampling frequency and high variance.

method (by any substantial margin) when it comes to mean Hausdorff distance. We identify the parameter combination $\sigma_p = 4$ and $\gamma = \frac{1}{2}$ as the best performing parameter combination for the simple method, since it leads to considerably higher accuracy than any other choice. We typically consider mean Hausdorff distance to be the most important of the two metrics, but in this case there is no clear relation between the parameters and the mean Hausdorff distance observed, and we therefore claim that the combination which maximizes the accuracy is the best performing. The parameters that maximize the performance of the augmented method is clearly also $\sigma_p = 4$ and $\gamma = \frac{1}{2}$, since this maximizes both accuracy and Hausdorff distance. Determining which parameters yield best performance for the augmented and restricted method comes to down deciding which metric one considers to be more important, and how much much of a decrease one can tolerate in one metric if there is an increase in another metric. We choose $\sigma_p = 8$ and $\gamma = \frac{1}{10}$ as the best performing parameter combination because of its mean lower Hausdorff distance. The choice between $\sigma_p = 6$ and $\sigma_p = 8$ is close to being arbitrary, seeing as the number of routes is quite low (15) and the difference between the two choices yield results that are hard to differentiate in terms of quality. A full overview of the parameters identified as optimal are shown in Table 5.2.



(a) Mean accuracy for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.



(b) Mean Hausdorff distance for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.

Figure 5.3: Mean accuracy and mean Hausdorff distance for the various parameter choices in the measurement group with high sampling frequency and low variance.

High sampling frequency and low variance

We move on to the measurement group with high sampling frequency and low variance. The mean accuracy for the various methods is given in Figure 5.3a. The mean Hausdorff distance is given in Figure 5.3b. The summary table, with entries selected in the same way as for the previous measurement group, is seen in Table 5.3. We immediately note that the accuracy is, overall, much better than the case with high variance. This is no surprise, but worth mentioning. We also note that mean Hausdorff distance has dropped by a considerable amount.

From Table 5.3 we see that, for the simple method, there is no parameter combination that is better than all others across the board. We prefer a lower mean Hausdorff distance and therefore conclude that the choice of $\sigma_p = 4$ and $\gamma = \frac{1}{2}$ is better. It is somewhat curious that the optimal value of σ_p is 4 for both this case, where the measurement variance is low, and for the previous case, where the measurement variance was high. It does, however, make sense that γ remains the same, seeing as the sampling frequency, and thus the distance between measurements, remains the same. For the augmented method the mean Hausdorff

Method	σ_p	γ	Mean accuracy	Mean distance
Benchmark	-	-	0.807	57.087
Simple	2.000	0.500	0.834	54.057
Simple	4.000	0.500	0.803	46.255
Augmented	2.000	0.100	0.833	13.095
Augmented	4.000	0.100	0.819	3.224
Augmented and restricted	2.000	0.100	0.761	10.842
Augmented and restricted	4.000	0.100	0.749	7.139

Table 5.3: The best performing parameter combinations in terms of either mean accuracy or mean Hausdorff distance. The measurement group has high sampling frequency and low variance.

Method	σ_p	γ
S	4	0.5
A	4	0.1
AR	4	0.1

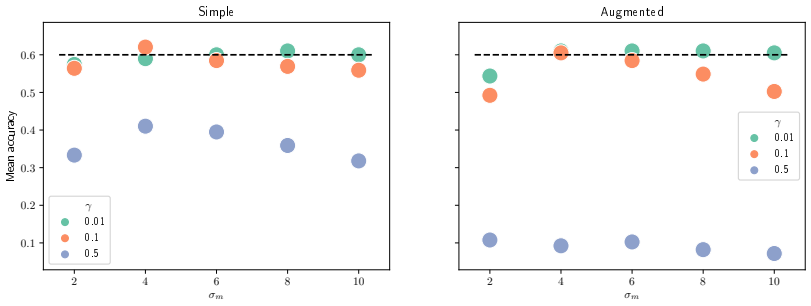
Table 5.4: Optimal parameters for the various methods when applied to measurement sequences obtained using high sampling frequency and low variance.

distance is remarkably low for $\sigma_p = 4$ and $\gamma = \frac{1}{10}$. This parameter choice is clearly the best, seeing as the parameter choices that lead to a slightly higher accuracy gives us a significant increase in mean Hausdorff distance. For the augmented and restricted method we see a slight decrease in overall performance compared to the augmented method, but $\sigma_p = 4$ and $\gamma = \frac{1}{10}$ are still the superior parameter choices. A full overview of the parameters identified as optimal are shown in Table 5.4.

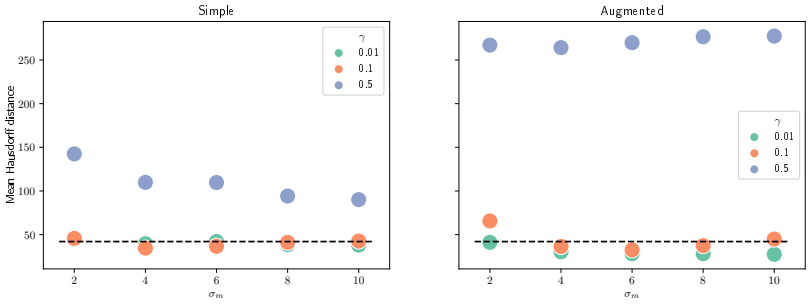
Low sampling frequency and high variance

We now turn our attention to the measurement group where measurements are made with low sampling frequency and high variance. We recall that the augmented and restricted method is not viable for this group, and is therefore not considered. The mean accuracy and Hausdorff distance for the various parameter choices is shown in Figure 5.4a and 5.4b. The summary of the best-performing parameter choices, chosen in the same way as before, is shown in Table 5.5.

We note that, while the accuracy is lower than the case with high sampling frequency and high variance, we have a clear reduction in mean Hausdorff distance for both the benchmark method and the simple method. These methods seem to benefit from the decrease in sampling frequency. The optimal parameter choice for the simple method is $\sigma_p = 4$ and $\gamma = \frac{1}{10}$. The optimal value of γ is reduced when compared to the high sampling frequency case. This makes sense, seeing as the distance between measurements has been increased considerably. For the



(a) Mean accuracy for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.



(b) Mean Hausdorff distance for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.

Figure 5.4: Mean accuracy and mean Hausdorff distance for the various parameter choices in the measurement group with low sampling frequency and high variance.

Method	σ_p	γ	Mean accuracy	Mean distance
Benchmark	-	-	0.600	42.070
Simple	4.000	0.100	0.621	34.563
Simple	4.000	0.100	0.621	34.563
Augmented	8.000	0.010	0.610	28.153
Augmented	10.000	0.010	0.605	27.513

Table 5.5: The best performing parameter combinations in terms of either mean accuracy or mean Hausdorff distance. The measurement group has low sampling frequency and high variance.

Method	σ_p	γ
S	4	0.1
A	8	0.01

Table 5.6: Optimal parameters for the various methods when applied to measurement sequences obtained using low sampling frequency and high variance.

Method	σ_p	γ	Mean accuracy	Mean distance
Benchmark	-	-	0.723	34.009
Simple	6.000	0.010	0.769	34.806
Simple	10.000	0.010	0.728	29.995
Augmented	6.000	0.010	0.779	19.905
Augmented	8.000	0.010	0.759	19.502

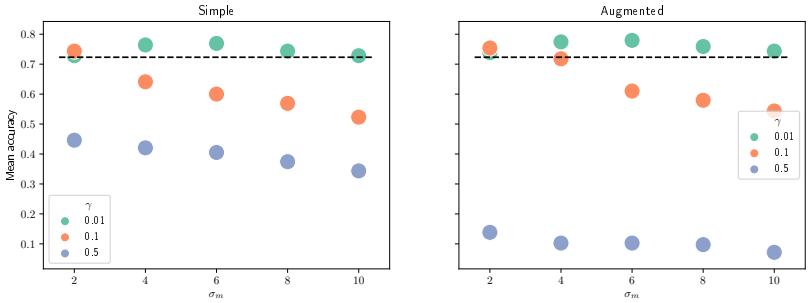
Table 5.7: The best performing parameter combinations in terms of either mean accuracy or mean Hausdorff distance. The measurement group has low sampling frequency and low variance.

augmented method we observe that the decrease in sampling frequency had a negative impact, resulting in both higher mean Hausdorff distance and lower accuracy. The parameter choices $\gamma = \frac{1}{100}$ and $\sigma_p = 8$ yields the lowest distance without sacrificing much in terms of accuracy, and are therefore considered to be the optimal choices. A full overview of the parameters identified as optimal are shown in Table 5.6.

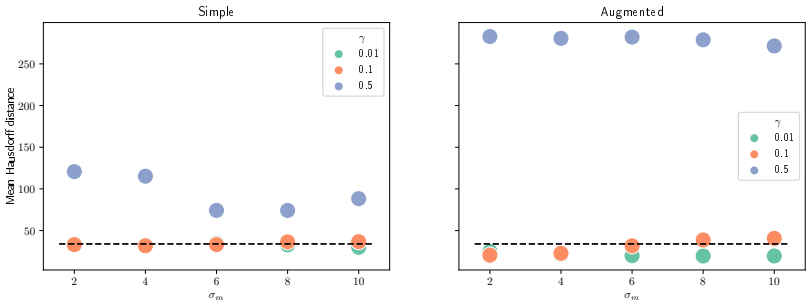
Low sampling frequency and low variance

Finally, we consider the measurement group with low sampling frequency and low variance. The mean accuracy for the different variations are shown in Figure 5.5a, while the distance is shown in Figure 5.4a. The overview is shown in Table 5.7.

In this situation we see that the Hausdorff distance is minimized for the simple method when $\sigma_p = 10$ and $\gamma = \frac{1}{100}$, while $\sigma_p = 6$ and $\gamma = \frac{1}{100}$ maximizes the accuracy. Due to the fact that $\sigma_p = 10$ is a lot higher than what we would expect (based on the other measurement groups and our knowledge of the data-generating process), and that the trade-off in accuracy for this parameter choice is rather high, we claim that $\sigma_p = 6$ and $\gamma = \frac{1}{100}$ is the optimal parameter choice. We choose the same set of parameters for the augmented method since this maximizes the accuracy without changing the Hausdorff distance by any significant amount. A full overview of the parameters identified as optimal are shown in Table 5.8.



(a) Mean accuracy for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.

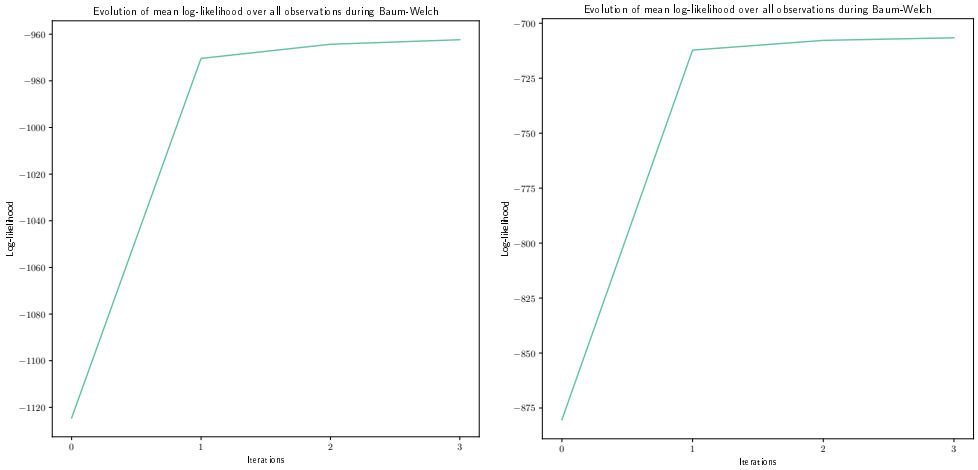


(b) Mean Hausdorff distance for the 15 measurement sequences in the measurement group. Performance of benchmark method indicated by a dashed black line.

Figure 5.5: Mean accuracy and mean Hausdorff distance for the various parameter choices in the measurement group with low sampling frequency and low variance.

Method	σ_p	γ
S	6	0.01
A	6	0.01

Table 5.8: Optimal parameters for the various methods when applied to measurement sequences obtained using low sampling frequency and low variance.



(a) High sampling frequency and high variance.

(b) High sampling frequency and low variance.

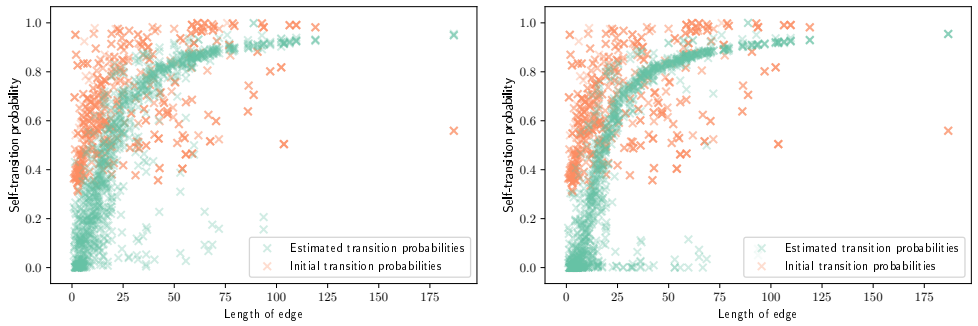
Figure 5.6: Evolution of the mean log-likelihood of the observation sequences over 3 iterations of the *Baum-Welch algorithm*.

5.2 Parameter estimation

In Section 4.3.2 we identified $\sigma_p = 8$ and $\gamma = \frac{1}{10}$ as optimal parameters for the ATR method when both the sampling frequency and variance is high. The parameter choices $\sigma_p = 4$ and $\gamma = \frac{1}{10}$ are considered optimal for the same method when the sampling frequency is high and variance is low. By applying the *Baum-Welch algorithm* to the mentioned measurement groups we observe that the mean log-likelihood of the observation sequences evolves as shown in Figure 5.6. The *Baum-Welch algorithm* (and more generally, the EM algorithm) guarantees an increase in the expected log-likelihood of an observation sequence. In the plots we can see that this results in an increase in the mean log-likelihood. The increase is considerable after the first iteration of the algorithm, but the increase tapers off for the second and third iteration. This likely indicates that the *Baum-Welch algorithm* has found a local maximum for the expected log-likelihood.

In order to get an idea about the effect of applying the *Baum-Welch algorithm* we take a look at the "self-transition" probabilities. The rationale for doing this is explained in Section 4.3.3. The initial and estimated transition probabilities are shown in Figure 5.7. We can see quite clearly that the estimated probabilities tend to 1 as the length of the edge increases. This indicates that the *Baum-Welch algorithm* has been able to learn how long the vehicles stay in the various states.

Another way of getting insights into the effect of applying the *Baum-Welch algorithm* is to compare the performance of the method with estimated transition probabilities to that of the competing methods. This is combined with the overall performance evaluation and presented in section Section 5.3.



(a) High frequency and high variance.

(b) High frequency and low variance.

Figure 5.7: Relation between "self-transition" probability and edge length.

5.3 Performance evaluation and comparison

Overview

Now that we have identified sensible values for the parameters involved in the various methods and estimated the transition probabilities for the ATR method using the *Baum-Welch algorithm*, it is time to compare the various methods in order to gain insights into whether the extra effort and time involved in using the augmented state space, as well as estimating the transition probabilities using the *Baum-Welch algorithm*, yield any performance gains. We will apply each of the methods, with optimal parameter choices, to each of the measurement groups and evaluate the performance by looking at the distribution of the accuracy and Hausdorff distance obtained for the different measurement sequences. We will be presenting histograms and kernel density estimates to better visualize the distribution of the accuracy and Hausdorff distances observed. The kernel density estimates uses a Gaussian kernel, with bandwidth selection done using Scott's rule [22]. The bandwidth of the histograms is determined using the Freedman-Diaconis rule [23].

High sampling frequency and high variance

We start off by considering the group of measurements simulated with high sampling frequency and high variance. In Figure 5.8a we show histograms of the accuracy obtained using the different methods. Here, we see a very clear difference between the methods using the simple state space and the methods using the augmented state space. In fact, it is the only measurement group where we observe significant differences in the accuracy across methods. An even clearer performance advantage for the methods using the augmented state space can be seen in the histograms of the Hausdorff distance in Figure 5.8b. The information displayed in Table 5.9 confirms what we have already seen, namely that using the

augmented state space, along with any changes made to take advantage of this state space formulation, leads to superior performance compared to the use of the simple state space.

It is also worth noting that using the simple method provides no benefits over the benchmark method. In addition, we observe that using the ATR method leads to a slight increase in mean accuracy and a slight decrease in mean Hausdorff distance. The performance gain is, however, not large enough to enable us to make any claims about it being a "better method". It is hard to argue that the ATR method is worse, though, and this is significant in itself, seeing as the ATR method uses transition probabilities that we suspect reflect the traffic flow of the road network better than the alternatives. Overall, the "augmented" methods (A, AR, ATR) all perform very similarly. The most interesting thing with regards to these methods, except their superior performance compared to the simple state, is how the Hausdorff distance of the "restricted" methods (AR and ATR) is more spread out. We expect the "restricted" methods to be more "rigid", and this could be an indication of this. The methods might be less affected by noise, but are also incapable of moving through many states in a short time in order to reach a certain position.

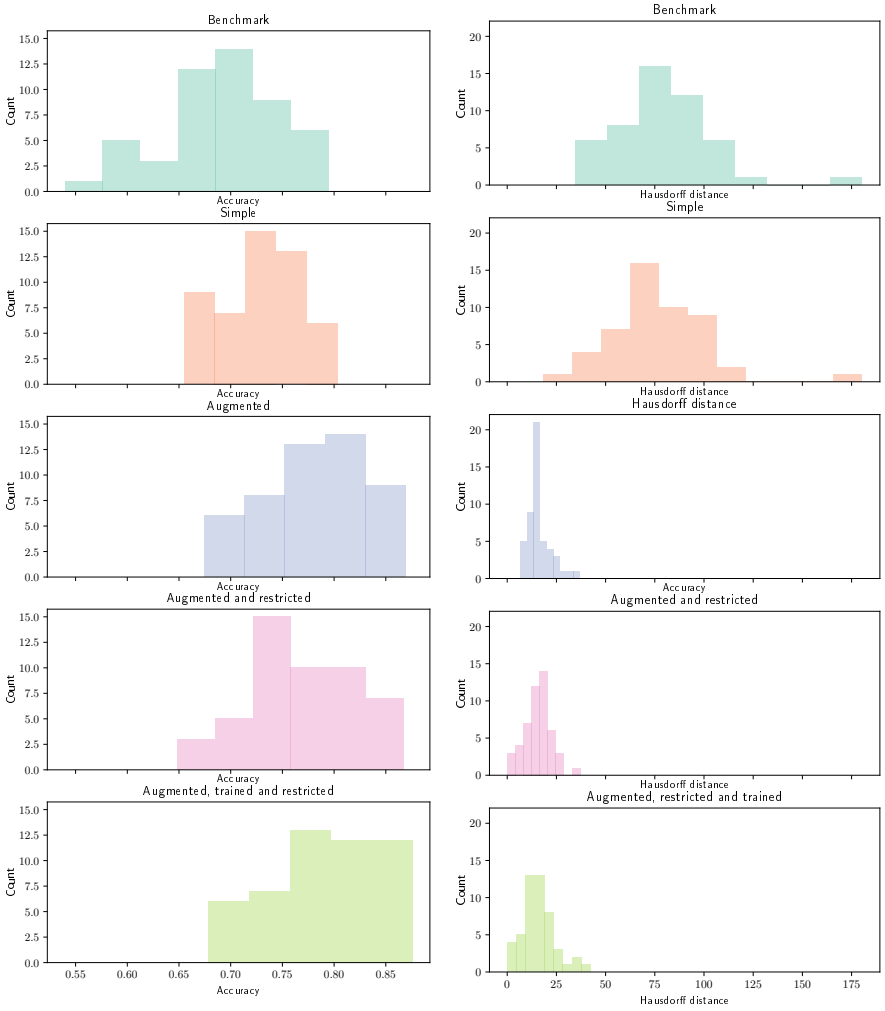
Kernel density estimates for the various methods are shown in Figure 5.9. These serve only to confirm what has been already established — that using the augmented state space provides very clear benefits, but any additional extensions and variations provide only minor benefits, if any, in terms of pure performance.

Method	Mean accuracy	Mean Hausdorff distance
A	0.783	16.393
AR	0.768	15.962
ATR	0.787	15.918
BM	0.693	78.704
S	0.730	76.099

Table 5.9: Mean accuracy and Hausdorff distance obtained for the various methods when applied to measurements made with high sampling frequency and high variance.

High sampling frequency and low variance

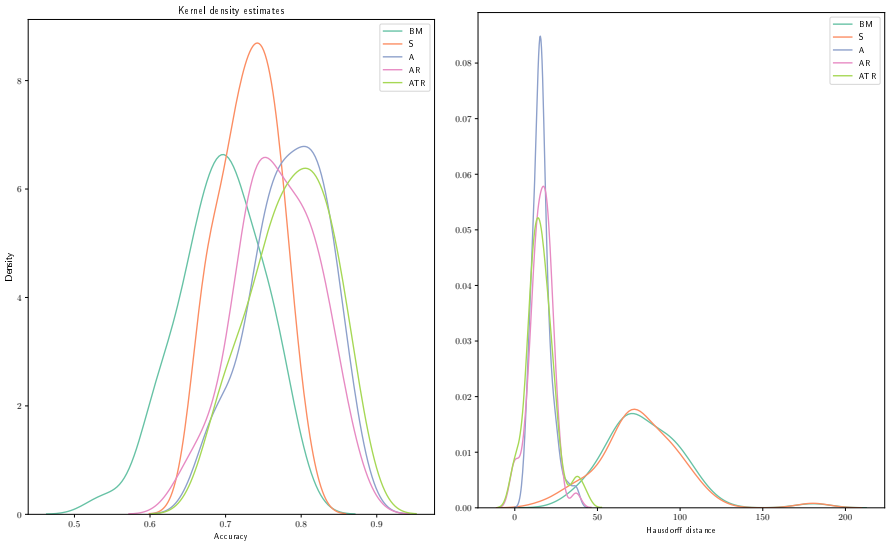
We move on to the measurement group with high sampling frequency and low variance. The associated accuracy histograms are shown in Figure 5.10a. The histograms of the Hausdorff distance are shown in Figure 5.10b. The mean accuracy and mean Hausdorff distance for this measurement group are shown in Table 5.10. Here, as before, one observes a clear difference between the "augmented" methods (A, AR, ATR) and the other methods, while using the S method still provides no benefit over the BM method. We also note that the mean accuracy is virtually



(a) Accuracy.

(b) Hausdorff distance.

Figure 5.8: Histograms of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with high sampling frequency and high variance.



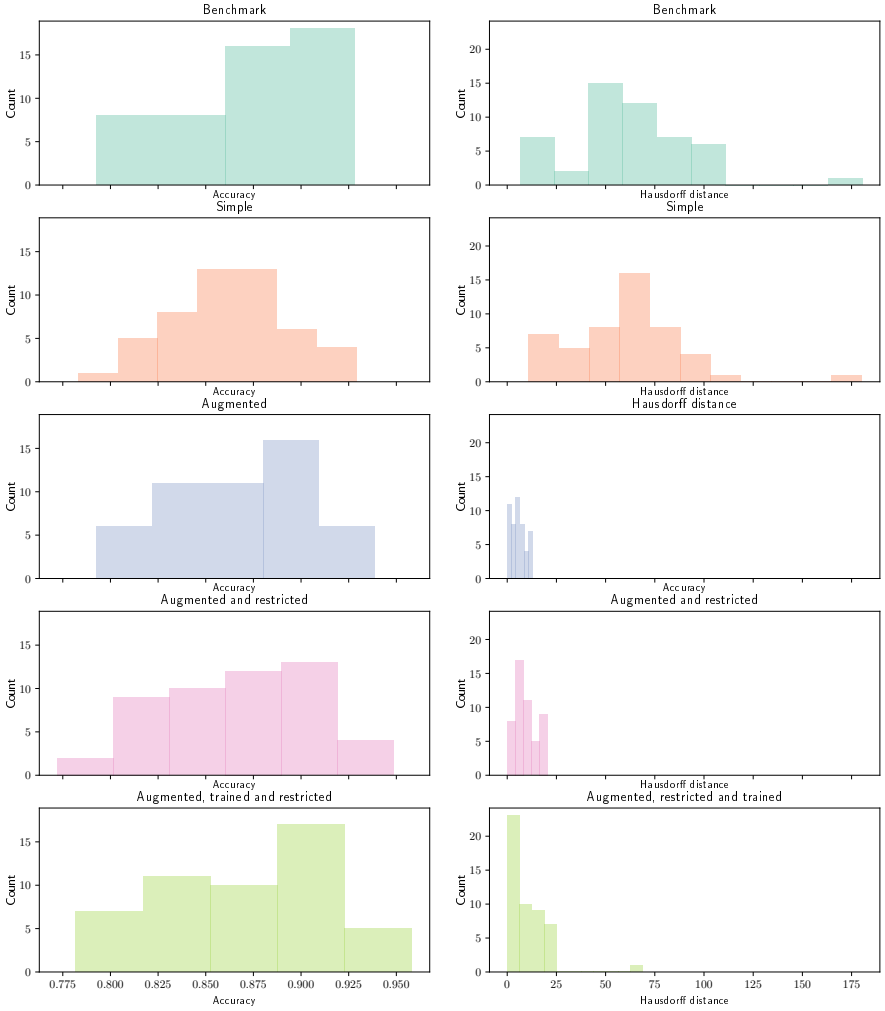
(a) Accuracy. **(b)** Hausdorff distance.

Figure 5.9: Kernel density estimates of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with high sampling frequency and high variance.

identical for all methods in this measurement group, while the difference in mean Hausdorff distance is even more pronounced than before, at least for the A method. This is very interesting, and also displays the advantage of including the Hausdorff distance as a performance metric. It is clear that the A method is superior in this situation, with both lower spread and lower mean for the Hausdorff distance. Moving on, we observe that the reduction in variance increases accuracy and decreases Hausdorff distance across the board. We note that the A method seems to better utilize the decrease in variance compared to the other "augmented" methods (AR and ATR). Disallowing connections between unconnected states does not seem to be beneficial here, and using estimated transition probabilities does not seem to fix the problem. We still observe a large spread in the Hausdorff distance for the "restricted" methods (AR, ATR) when compared to the A method, which strengthens our belief that the method has a more "rigid" behaviour.

Low sampling frequency and high variance

We now move on the measurement group with low sampling frequency and high variance. We will present the same plots as earlier, but we now only consider the "unrestricted" (BM, S, A) methods, since these are the only ones that are applicable in this scenario. The histograms showing the accuracy and Hausdorff distance can be seen in Figure 5.12a and 5.12b respectively. The mean of the performance metrics are shown in Table 5.11. The benefit provided by using the augmented state space has diminished considerably compared to the case with high sampling



(a) Accuracy. **(b)** Hausdorff distance.

Figure 5.10: Histograms of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with high sampling frequency and low variance.

Method	Mean accuracy	Mean Hausdorff distance
A	0.870	5.674
AR	0.870	9.718
ATR	0.873	10.002
BM	0.872	62.874
S	0.861	61.818

Table 5.10: Mean accuracy and Hausdorff distance obtained for the various methods when applied to measurements made with high sampling frequency and low variance.

frequency. The A method is still quite clearly a superior method, but the difference is not as pronounced as it is in the case with high sampling frequency. We also observe that the simple method is finally showing itself as a better option than the benchmark method, and is closer, in terms of Hausdorff distance, to the augmented method than it is to the benchmark method.

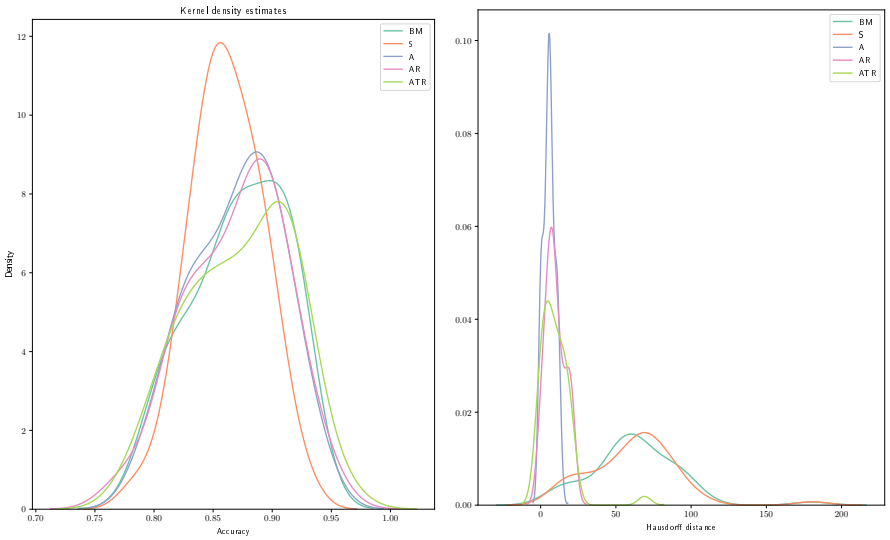
Method	Mean accuracy	Mean Hausdorff distance
A	0.690	36.798
BM	0.683	51.626
S	0.680	41.657

Table 5.11: Mean accuracy and Hausdorff distance obtained for the various methods when applied to measurements made with low sampling frequency and high variance.

Low sampling frequency and low variance

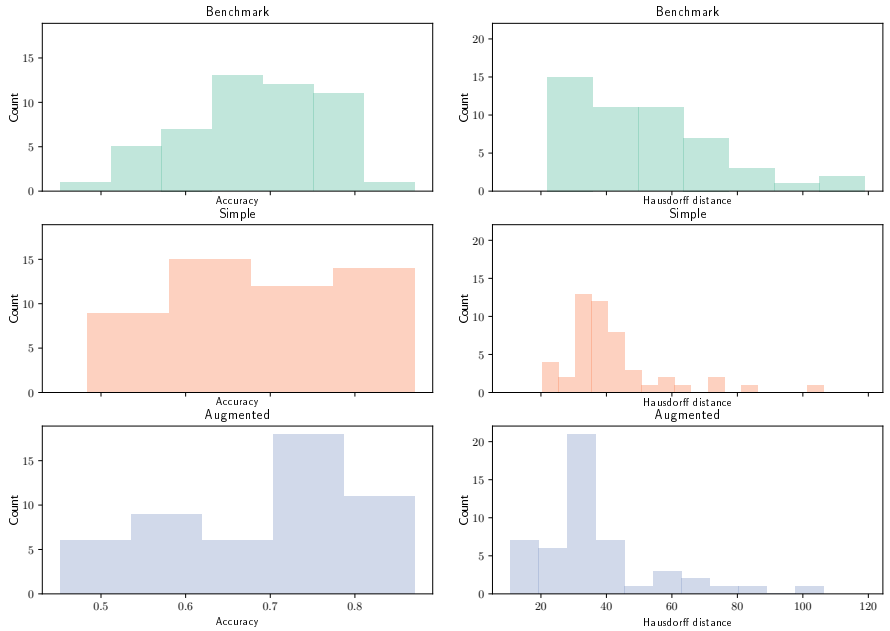
For the final case, we consider measurements made with low sampling frequency and low variance. The histograms are shown in Figure 5.14, the kernel density estimates in Figure 5.15, and the mean of the performance metrics for the various methods in Table 5.12. Here, we observe similar tendencies as seen in the previous case. The benefit provided by the augmented method is clear, but not nearly as large as in the case with high sampling frequency. The performance of the S method is approaching that of the A method, with the accuracy being equal across the board. It is striking that the simple method performs a lot better in this situation, with low sampling frequency, than it does in the situation with high sampling frequency. This was seen in the other measurement group with low sampling frequency as well. One would, at least intuitively, expect more information to translate to better route estimates, but this is clearly not the case for the simple method.

The decrease in measurement variance has increased the accuracy overall, but no method has significantly better accuracy than the others. Again, the Hausdorff distance provides valuable insights that are not visible at all by looking the accuracy.



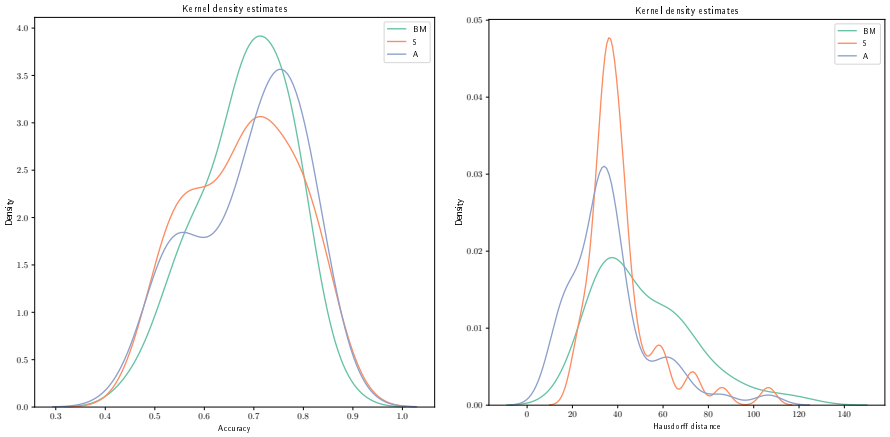
(a) Accuracy. **(b)** Hausdorff distance.

Figure 5.11: Kernel density estimates of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with high sampling frequency and low variance.



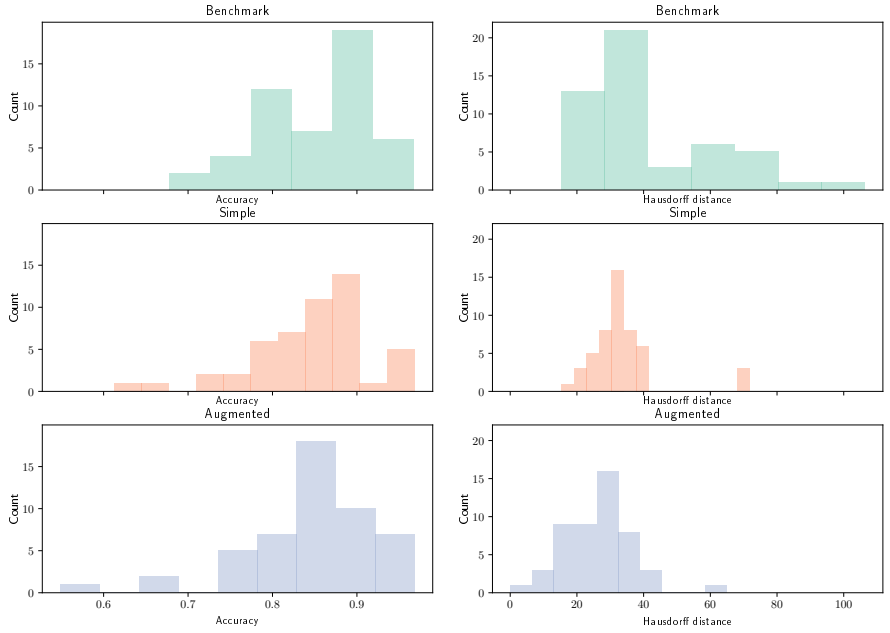
(a) Accuracy. **(b)** Hausdorff distance.

Figure 5.12: Histograms of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with low sampling frequency and high variance.



(a) Accuracy. **(b)** Hausdorff distance.

Figure 5.13: Kernel density estimates of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with low sampling frequency and high variance.

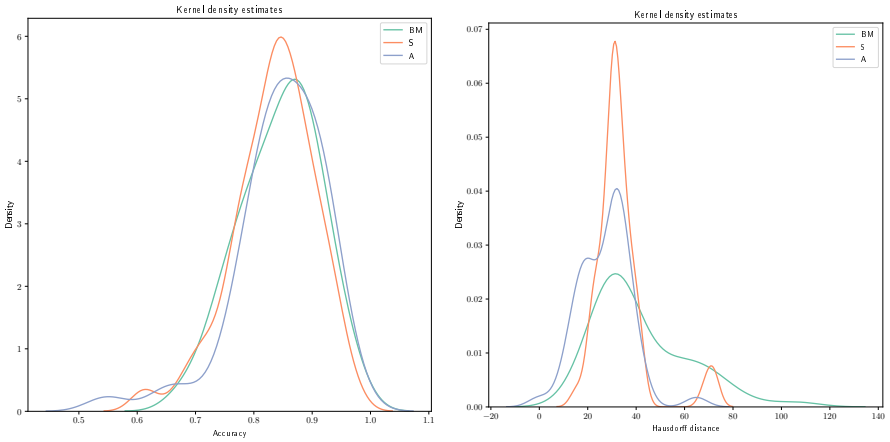


(a) Accuracy. **(b)** Hausdorff distance.

Figure 5.14: Histograms of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with low sampling frequency and low variance.

Method	Mean accuracy	Mean Hausdorff distance
A	0.846	27.286
BM	0.845	41.919
S	0.834	33.357

Table 5.12: Mean accuracy and Hausdorff distance obtained for the various methods when applied to measurements made with low sampling frequency and low variance.



(a) Accuracy.

(b) Hausdorff distance.

Figure 5.15: Kernel density estimates of accuracy and Hausdorff distance for the various methods when applied to measurements from the measurement group with low sampling frequency and low variance.

Chapter 6

Discussion and conclusion

6.1 Discussion

The purpose of this thesis has been to introduce and explore HMMs with the intention of arriving at one or several HMM formulations that can enable high quality map matching. The HMM formulations require that we assume something about the distribution of the distance travelled between each measurement. It also requires assumptions about the distance between the measurement and the road segment one is currently on. We are, however, limited by the requirements of the HMM. One would ideally represent the position of a vehicle on a road network as a point in space that could take on any value in a subset of $\mathbb{R}^2$. This is not possible within an HMM since the state space must be discrete. Furthermore, it is beneficial (for computational reasons) to have a state space that is as small as possible. The fact that "being in a position" is represented as being in some state from a discrete state space has the consequence of making the "distance between measurements" that we must make assumptions about a coarse, and somewhat unpredictable, approximation of the actual distance travelled by the vehicle. The accuracy of this approximation is, in our case, related to the length of the edges in the road network graph. These edges do not have a fixed lengths, and can, at least in theory, vary by several orders of magnitude. It is therefore hard to accurately translate what we know about the actual movement (which is typically given by time between measurements and the speed limit) into an assumption about the "distance between measurements" used in our model. Since the details of the road network representation is expected to have considerable impact on the distribution of the "distances" it seems sensible to experiment with a variety of assumptions and choose the assumption that maximize our performance.

A similar issue is encountered when it comes to the distance between the measurement and the road segment. We do not know exactly what the distribution of this quantity looks like, and the knowledge that is available about measurement error is related to the distance between the true position and the measured position. The most sensible course of action in this scenario was, again, to conduct experiments to test a limited set of assumptions to find the assumption that gives

us the best performance.

During the parameter search we observe that there are significant variations in performance for the various parameter choices. We note that it is challenging to decide which parameter choices are optimal, seeing as we have not found a single metric that is able to quantify the quality of all aspects of the route estimate in a sensible manner. The accuracy does not take into account the whole route, but only evaluates whether the estimated states at the various measurements times are identical to the state we are in when the measurement is made. By using the accuracy we have no way of reliably assessing the "similarity" between the estimated route and the true route. We are restricted to checking whether the endpoints of a path between an estimated state at time $n-1$ and an estimated state at time n match, but can not check whether the path that connect the estimated states matches the true path.

In order to compare the entirety of the estimated route to the true route we employ the Hausdorff distance. This enables us to compare entire routes, but we miss out on information about the ordering of the estimated states. It is also somewhat misleading to judge the quality of a HMM-based method by using a metric that takes into account how the estimated states are "glued" together — especially when the estimated states are glued together using an approach as naive as choosing the shortest path. The metric is very suitable for evaluating a general map matching method, but it is not necessarily fair to use it to evaluate the underlying HMM — signs of bad performance could just as well be because of "bad glueing". This concern is alleviated, at least to a certain degree, when using the "augmented and restricted" method, since this method removes the need for "glueing" unconnected states together.

The choice of parameters is straightforward only in the case when one unique parameter combination both maximizes accuracy and minimizes Hausdorff distance. This only happens in a handful of scenarios. In other cases we must either choose which metric is most important, or decide how much we can accept in a trade-off between the two metrics. This is hard to do well in a consistent manner. Another challenging situation appears when what appears to be the best parameter choice is unexpected, such as when $\sigma_p = 10$ minimizes the Hausdorff distance in the case with low sampling frequency and low variance. This value is not in line with our expectation, seeing as we know the measurement standard deviance to be $\sigma_m = 3$, and also not in line with what was observed in the other low variance group. While we did not expect that σ_p and σ_m would be equal, it was surprising to see a discrepancy as large as the one seen here. One explanation for this could be that there is large discrepancy between our "model world" and the actual scenario. We end up choosing $\sigma_p = 6$ as a better alternative in this specific situation, and do so by primarily appealing to the higher accuracy, but we are also influenced by the fact that this choice is more in line with our expectations. We are not entirely confident about this choice.

The difficulty of determining optimal parameters leads to a scenario where the final performance evaluation is conducted using methods where the parameter

choices might, in some cases, be sub-optimal. It is therefore hard to make definitive claims about the viability of the various methods. Certain observations are, however, so clear that we do not think better parameter choices would have changed the final outcome significantly. Some of these clear observations are:

- The superior performance of the HMM-based methods using the augmented state space,
- The "augmented" method's ability to benefit from low measurement variance, and
- The "simple" method's inability to benefit from high sampling frequency.

It is also worth discussing the performance of the AR method. This method avoids the issue of having to "glue" estimated states together while maintaining performance similar to the other methods using the augmented state space, at least in the high variance case. Another upside of this method is the increased interpretability of the transition probabilities — the transition probabilities represent the probability that a certain directly connected edge is chosen, regardless of whether a measurement is made at this directly connected edge. The downside is that the sampling frequency must be high enough to ensure that there are enough transitions to "keep up with" the true route. This can be especially problematic in scenarios where the route consists of a lot of very short edges. The form of the transition probabilities in the AR method is not only useful because of the increased interpretability — it might also make the transition probabilities easier improve. It is not incomprehensible that the traffic flow of a road network could be estimated using some other method. This could then be translated to transition probabilities for the AR method, and hopefully increase its performance.

The interpretation of the transition probabilities of the ATR method as estimates of the "flow" between connected road segments ensures that there is some, although possibly small, value to using the *Baum-Welch algorithm* in the context explored in this thesis. It can enable us to obtain insight about the traffic flow without having access to ground truth data, given that the data consists of enough observations. The plot shown in Figure 5.7 indicates that the *Baum-Welch algorithm* is able to learn some of the behaviour exhibited by the system. Having said this, it must be admitted that the ATR method did not yield results that were as good as one could have hoped. It did not outperform the other "augmented" methods, which would have been required for us to claim that it had learned something useful about the system. This, combined with the difficulty of evaluating what it had "learned" in a manner we trusted to be correct, left us ensure about how much value it could actually provide.

6.2 Summary

In this thesis we have presented theory and experiments relating to various HMM-based map matching methods, and introduced ideas and concepts that hopefully can be of interest to researchers and practitioners who are engaged in this topic.

The thesis starts off with a thorough review of the theory behind HMMs. We continue by explaining the map matching problem and the context it exists in. By combining the theory of the HMM with our understanding of the problem we arrive at methods that can be used to do map matching. This involves defining the state space by using parts of the road network, creating transition probabilities that lets us utilize knowledge about the restrictions of the road network, and using emission probabilities that enable us to account for the noisy nature of the position measurements. Various methods are presented, with the augmented state space introduced in this thesis being a central component to most of them. Several possible transition probabilities are used, with those estimated using the *Baum-Welch algorithm* perhaps being the most notable, seeing as this approach has not been taken before.

We conduct multiple experiments to evaluate the performance of the methods and assess the viability of using the *Baum-Welch algorithm* in this context. The performance is measured using the Hausdorff distance and the accuracy, which are two metrics that capture different facets of the performance of the methods. The purpose of the first experiment is to determine sensible and well-performing values for the parameters used in the emission and transition probabilities. This is done by assessing how the parameter choices impact the performance of our method when applied to simulated position measurements obtained during simulated road network traversal. The measurement simulation in this experiment is conducted under four different circumstances, with the circumstances dictated by the sampling frequency and variance of the simulated position measurements. The second experiment uses data created in a similar fashion to estimate transition probabilities using the *Baum-Welch algorithm*. An inspection of the resulting estimates show signs that the estimated transition probabilities reflect at least certain aspects of the true situation. The performance of all presented methods, including the one with estimated transition probabilities, is evaluated in the final experiment. This experiment is similar to the parameter search in the first experiment, but with more data, and now only using parameter choices that have been identified as optimal. These experiments enable to us to draw certain conclusions.

- The state space introduced in this thesis contributes to a significant performance increase when compared to benchmark methods and the HMM-based method using a simple state space. This is especially true in situations with high sampling frequency. When comparing the simple method to the augmented method in the case with high sampling frequency and high variance we observe a 5.3% increase in accuracy and a 78.5% decrease in Hausdorff distance. This can be seen in Table 5.9. In the case of high sampling frequency and low variance we observe a 0.9% increase in accuracy and a 90.7% decrease in Hausdorff distance, as seen in Table 5.10.
- The method that uses estimated transition probabilities displays performance that is only slightly worse than the best competing method. Its performance is, under one specific set of circumstances, identical to the best competing method.

- The simplest of the HMM-based methods performs worse with high sampling frequency, with the other HMM-based methods showing an increase in performance.
- Measurement variance has a high impact on both accuracy and Hausdorff distance for all methods.

The second point in the preceding list is significant because of the potentially increased interpretability of this method. There are signs suggesting that the estimated transition probabilities can reflect the traffic flow on the road network, and there might be situations where such an interpretation is desirable, and worth sacrificing low to moderate amounts of performance for.

6.3 Further work

It would be interesting to see a more focused effort at analyzing the viability of using the Baum-Welch algorithm for learning transition probabilities. One issue that could be explored is whether the Baum-Welch algorithm is able to reduce the error observed when missing data is encountered. Data missing at random can easily be dealt with within an HMM framework. In this case the only way to evaluate the probability of various paths is through the transition probabilities. If the estimated transition probabilities are closer to the truth than the initial estimates then one would expect to see that the estimated transition probabilities make the method more robust when faced with missing data.

It could also be interesting to train a HMM with Gaussian emissions. This was briefly explored during initial work with this thesis, but it proved challenging for a variety of reasons. One of the challenges were related to determining sensible initial values for the mean and the covariance for each state. Attempting the learn the mean and covariance also increases the total number of parameters by a lot, and would likely require more data. This would result in a substantial increase in the runtime of the *Baum-Welch algorithm*. This would add to an already significant computational cost, and would likely not be feasible without a more efficient implementation. It is possible, although not necessarily likely, that given solid initial values and real-world data, a trained Gaussian HMM would be able to represent both the traffic flow and the specifics about measurements made in different regions of the road network. The traffic flow would be encoded in the transition matrix, while details about the measurements would be captured by the mean and covariance of the various states. It is possible that edges located in a high-density urban region could, for instance, be found to have higher variance than a region with less obstruction for the GPS signal.

It would, if given large amounts of real-world data, be interesting to incorporate supervised learning algorithms in an effort to create more intelligent map matching methods. The Input-Output HMM (IOHMM) presented in [24] could for instance enable the variance of the measurements to be based on external features such as satellite imagery, WiFi and cellular signals, and the features of the road network.

It is also worth mentioning that using the exact methodology presented in this thesis is probably not advisable for large scale real-world problems. This thesis is focused on presenting the various techniques in a thorough manner, making sure that the methods are mathematically sound and consistent with the assumptions made. A more efficient implementation of the methods used in this thesis would likely require that one constructs a state space consisting only of states that are somewhat likely to be a part of the route estimate. One could also consider using a sequence of edges as a state instead of a single edge. This would make the state space smaller, but would also make the states a worse approximation of a "position". This would, among other things, have an impact on the accuracy of the "distance" $d(x, y)$ used in the transition probabilities, and would likely have an impact on the performance of the method. It would also make it harder to justify that the measurements are conditionally independent given the state.

In previous work, as well as real-world applications, it has been quite common to violate the assumptions of the HMM when formulating the transition probability. One could make such adaptions to the methods presented here as well, and the resulting methods might end up performing well. Such an approach has, however, not been considered to be suitable for this thesis.

Bibliography

[1] 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*, ser. GIS '09, New York, NY, USA: ACM, 2009, pp. 336–343.

[2] R. Raymond, T. Morimura, T. Osogami and N. Hirosue, 'Map matching with Hidden Markov Model on sampled road network', in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov. 2012, pp. 2242–2245.

[3] B. Hummel, 'Map Matching for Vehicle Guidance', in *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, ser. Innovations in GIS, CRC Press, 2006, pp. 175–185.

[4] C. Yang and G. Gidófalvi, 'Fast map matching, an algorithm integrating hidden Markov model with precomputation', *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, Mar. 2018.

[5] S. M. Ross, *Introduction to Probability Models*. Academic Press, Jan. 2014.

[6] S. L. Scott, 'Bayesian Methods for Hidden Markov Models: Recursive Computing in the 21st Century', *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 337–351, 2002.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, Aug. 2016.

[8] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., 1993.

[9] G. Forney, 'The Viterbi algorithm', *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[10] S. Batzoglou, V. Popic and J. Louie, *Lecture notes in Computational Genomics - Lecture 6: Hidden markov models continued*, Jan. 2015. [Online]. Available: `https://web.stanford.edu/class/cs262/archives/notes/lecture6.pdf` (visited on 17/01/2020).

[11] A. P. Dempster, N. M. Laird and D. B. Rubin, 'Maximum Likelihood from Incomplete Data Via the *EM* Algorithm', *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, Sep. 1977.

[12] L. E. Baum and T. Petrie, 'Statistical Inference for Probabilistic Functions of Finite State Markov Chains', *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[13] W. Zucchini, I. L. MacDonald, R. Langrock, I. L. MacDonald and R. Langrock, *Hidden Markov Models for Time Series : An Introduction Using R, Second Edition*. Chapman and Hall/CRC, Dec. 2017.

[14] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[15] J. Bilmes, 'A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models', *Technical Report ICSI-TR-97-021, University of Berkeley*, vol. 4, Jun. 2000.

[16] S. Tu, *Derivation of Baum-Welch Algorithm for Hidden Markov Models*. [Online]. Available: `https://stephentu.github.io/writeups/hmm-baum-welch-derivation.pdf` (visited on 17/01/2020).

[17] Xiaolin Li, M. Parizeau and R. Plamondon, 'Training hidden Markov models with multiple observations-a combinatorial method', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 371–377, Apr. 2000.

[18] J. P. Snyder, *Map projections - a working manual*, ser. U.S. Geological Survey professional paper; 1396. U.S. Government Printing Office, 1987.

[19] E. W. Dijkstra, 'A note on two problems in connexion with graphs', *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

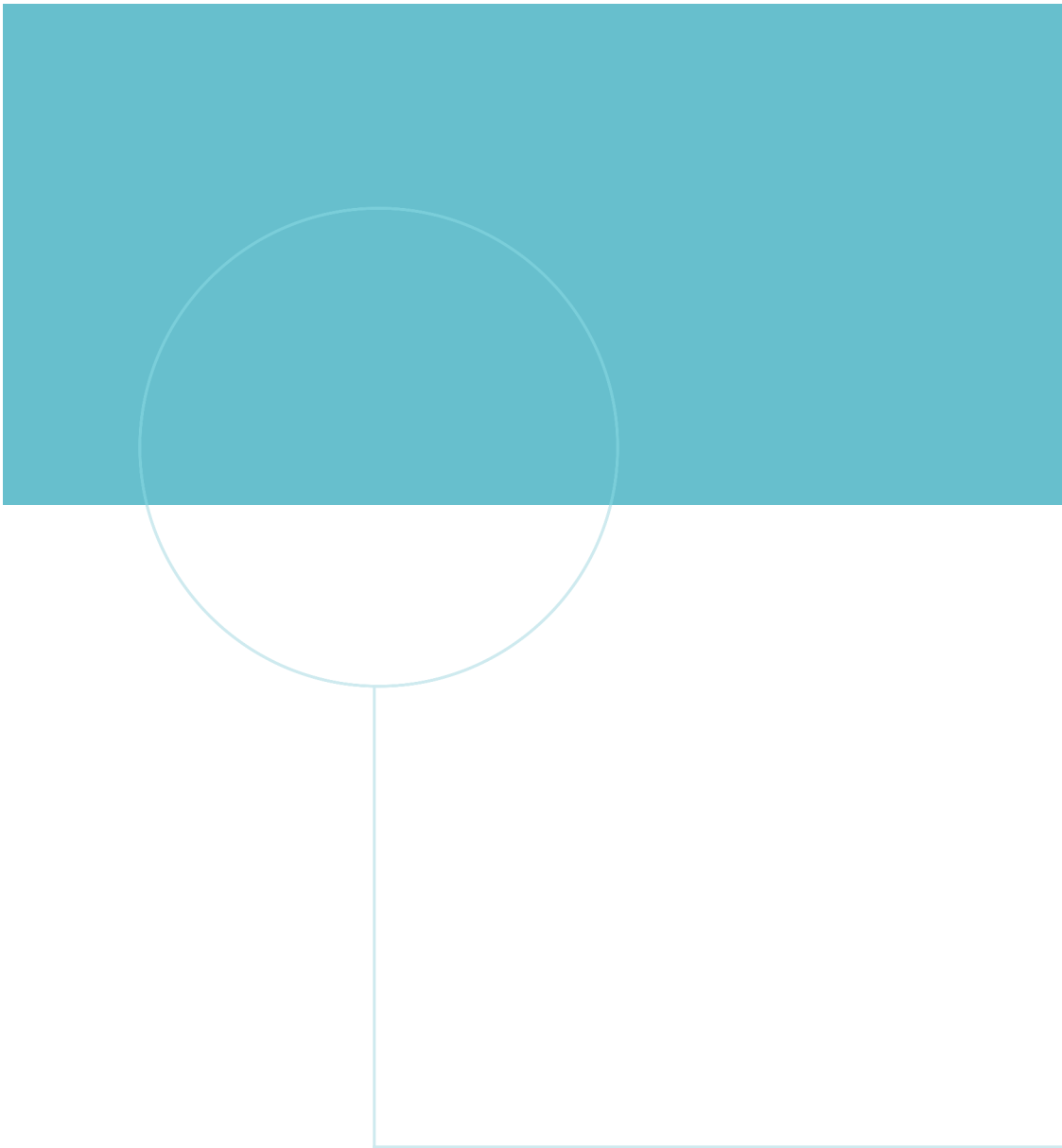
[20] A. A. Taha and A. Hanbury, 'An efficient algorithm for calculating the exact Hausdorff distance', *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 11, pp. 2153–2163, Nov. 2015.

[21] *Global Positioning System Standard Positioning Service Performance Standard*, Sep. 2008. [Online]. Available: `https://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf` (visited on 14/12/2019).

[22] D. W. Scott, 'Sturges' and Scott's Rules', in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer, 2011, pp. 1563–1566.

[23] D. Freedman and P. Diaconis, 'On the histogram as a density estimator:L2 theory', *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, Dec. 1981.

[24] Y. Bengio and P. Frasconi, 'An Input Output HMM Architecture', in *Proceedings of the 7th International Conference on Neural Information Processing Systems*, ser. NIPS'94, Cambridge, MA, USA: MIT Press, 1994, pp. 427–434.



NTNU

Norwegian University of
Science and Technology