Linköping University | Department of Computer and Information Science
Master thesis, 30 ECTS | Computer Science
202018 | LIU-IDA/LITH-EX-A--2018/XXX--SE

Gaussian Process Regression-based GPS Variance Estimation and Trajectory Forecasting

Regression med Gaussiska Processer för Estimering av GPS Varians och Trajektoriebaserade Tidtabellsprognoser

Linus Kortesalmi

Supervisor : Mattias Tiger
Examiner : Fredrik Heintz

External supervisor : Simon Johansson

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

Spatio-temporal data is a commonly used source of information. Using machine learning to analyse this kind of data can lead to many interesting and useful insights. In this thesis project, a novel public transportation spatio-temporal dataset is explored and analysed. The dataset contains 300 GB of positional events, spanning two weeks of time, from all public transportation vehicles in Östergötland county, Sweden. From the data exploration, three high-level problems are formulated: bus stop detection, GPS variance estimation, and arrival time prediction, also called trajectory forecasting. The bus stop detection problem is briefly discussed and solutions are proposed. Gaussian process regression is an effective method for solving regression problems, which is a typical class of problems. The GPS variance estimation problem is solved via the use of a mixture of Gaussian processes. A mixture of Gaussian processes is also used to predict the arrival time for public transportation buses. The arrival time prediction is from one bus stop to the next, not for the whole trajectory. The result from the arrival time prediction is a distribution of arrival times, which can easily be applied to determine the earliest and latest expected arrival to the next bus stop, alongside the most probable arrival time. The naïve arrival time prediction model implemented has a root mean square error of 5 to 19 seconds. In general, the absolute error of the prediction model decreases over time in each respective segment. The results from the GPS variance estimation problem is a model which can compare the variance for different environments along the route on a given trajectory.

Acknowledgments

I would like to thank my examiner Docent Fredrik Heintz at the Department of Computer and Information Science (IDA) at Linköping University for making this thesis project possible. A huge thanks to my supervisor Mattias Tiger at IDA at Linköping University for his endless help with Gaussian Processes and keeping my spirit high throughout the project.

I would also like to thank Attentec AB for the opportunity to do this thesis work and for supplying me with office space and an external supervisor. A big thank you to Simon Johansson at Attentec AB, Josef Fagerström, and Simon Lindblad for providing support and proof reading of the report.

I would finally like to thank Östgötatrafiken AB for providing the dataset, which this whole thesis project is based upon.

Contents

List of Figures

List of Tables

Todo list

1 Introduction

Today, there is an increasing number of services built upon the use of spatio-temporal datasets. For example, public transportation companies use such data for route planning and arrival time prediction. Taxi companies use spatio-temporal data to track vehicles and, for example, assign the closest available vehicle to a new customer. Spatio-temporal datasets can also be used for city planning and analysing effects of rapid urbanisation [1, 2]. Service vehicles are today equipped with GPS sensors which collect behavioural data. The availability and usability of spatio-temporal datasets will only increase with the development of new technologies. For example, the field of autonomous vehicles utilises spatio-temporal dataset for various applications [3, 4].

The purpose of this thesis project is to perform analysis on a novel spatio-temporal dataset collected by Östgötatrafiken AB. The goal is to analyse the dataset and define problems inherent in it. By solving the inherent problems of the dataset, the dataset can be used to provide new or improved services. For example, further knowledge can be gained about the flow of traffic. The arrival time prediction or vehicle position estimation for service vehicles can also be improved. The provided documentation for the dataset is minimal, which makes the insights from the analysis even more valuable.

1.1 Aim

The aim of this thesis project is to explore a novel dataset and formulate problems and solutions which can be interesting not only to the dataset provider but also in a broader context, e.g., for services relying on GPS positioning. The dataset is currently used to provide time forecasts on the public transportation available in Östergötland county. The aim is not to fully solve all possible problems with the dataset, but rather formulate outlines for a few interesting problems. Some of the outlines are implemented in this thesis project and their results presented.

1.2 Research Questions

The specific research questions treated in this thesis project are presented in this section. They are divided into three groups: *Metadata Questions*, *High-Level Problem Questions*, and *Problem*

Specific Questions. The *Metadata Questions* investigate and explore the provided dataset in depth, using Exploratory Data Analysis (EDA). They cover issues such as pre-processing, data noise and outliers. These questions can be seen as a pre-study before data analysis and machine learning techniques are applied. The insights from these questions are the basis for the *High-Level Problem Questions*. The *High-Level Problem Questions* formulate general problems which span individual applications by using the information available both internally in the dataset and externally from sources outside the provided dataset. High-level problems are problems which are not inherent in the dataset but rather problems which can be solved by using the information available in the dataset. They also aim to suggest solutions for these formulated problems. The *Problem Specific Questions* are questions related to a specific problem and solution described by the *High-Level Problem Questions*.

Metadata Questions:

1. *What kind of information is available in the dataset provided by Östgötatrafiken AB?*
   Which features exist in the dataset, such as different event types, event parameters, and event structures? What do these features mean and how do they relate to each other? Are some features more important than others? Are there any features missing from the given dataset?
2. *What pre-processing needs to be done in order to solve high-level problems?*
   The pre-processing focuses on solving problems inherent in the dataset, such as processing events and extracting relevant information from them. What are some pre-processing methods that can be applied to this dataset? How can the raw data in the dataset be transformed into useful features? How does the ordering of the raw data affect the solutions for high-level problems? How can the raw data be transformed into trajectories? How can information from different event types be added to the trajectories?
3. *How can noisy measurements be detected?*
   Noisy measurements can affect the solutions for high-level problems negatively. Various anomaly-detection algorithms can be applied to find such cases, but they can also be solved using dataset-tailored algorithms. Which manual inspection methods can be used to detect noisy measurements? Which general anomaly detection algorithms can be applied? How can a dataset-tailored algorithm be implemented? How does the general anomaly detection algorithm differ from the dataset-tailored algorithm?
4. *How is the provided dataset related to external data sources and how can they be combined?*
   There are external data sources available which can complement the data in the provided dataset, such as Google Roads API[1], the website of Östgötatrafiken[2] and the Traffic Lab API[3]. How compatible is the data from the external sources with the data in the given dataset? Does the external sources contain any of the features that are deemed missing from the given dataset (in *Metadata Question* 1)? How can the data from the external sources be used?

High-Level Problem Questions:

1. *What interesting high-level problems can be investigated and solved based on the available data?*
   The high-level problems can utilise both the data available in the provided dataset and any complementary external data. What problems span over the areas of GPS Positioning and Trajectory Forecasting? What are the core problems for each area? How can the existing system that generated the dataset be improved?

[1] https://developers.google.com/maps/documentation/roads/intro
[2] https://www.ostgotatrafiken.se/
[3] https://www.trafiklab.se/

2. *How can these problems be solved?*
   The solutions tackle the core of each problem. Each solution explicitly states if there is a baseline available for comparison or if one can be easily created. How can problems with GPS Positioning and Trajectory Forecasting be solved?
3. *How do solutions to these problems benefit society and the industry?*
   The solutions are analysed in a broader context, e.g., from a societal or ethical point of view. What value do the solutions offer to the industry? How is consumer privacy affected by the solutions?

Problem Specific Questions

1. *How can the spatio-temporal varying GPS variance be estimated from sets of observed trajectories over extended periods of time?* Can recent Gaussian Processes Regression (GPR) based trajectory modelling approaches be used to solve this problem? How can the GPR-based approach scale with multiple trajectory models? How can potential periodicity of the data be handled in the approach? How can the model be trained on extended periods of time?
2. *How can the approach to estimate the spatio-temporal varying GPS variance be evaluated?* What assumptions are made regarding the inherent noise of the data? How can kernels be evaluated if a GPR-based approach is applied? How can trajectories be evaluated? Which evaluation criteria can be applied to evaluate the approach?
3. *How can Trajectory Forecasting be implemented from sets of observed trajectories?*
   Can GPR-based trajectory modelling approaches be used to solve this problem? Which modifications need to be done to the approach in order to produce trajectory forecasts? How can features from observed trajectories over extended periods of time be used to improve the forecasts?
4. *How can the Trajectory Forecasting model be evaluated?*
   Can the existing forecasts from Östgötatrafiken AB be used to create an evaluation baseline? How can new forecasts be compared with the baseline? Which insights can be made from the information available in the output of the Trajectory Forecasting model? How can the model respond to immediate real-time changes? How can the performance of a single forecast be evaluated? How can the overall performance of the model be evaluated? Which evaluation criteria can be applied to evaluate the forecasts? Can the new model be combined with the existing model in order to produce more precise forecasts? Are there any benefits to returning a probability distribution over arrival times compared to returning the expected arrival time?

1.3 Delimitations and Limitations

The dataset is provided by Östgötatrafiken AB and is not available for public use. This thesis project only focuses on the data from buses; data from public transportation trains are ignored. In order to support manual inspection of the data in the dataset, the data is filtered to only contain data from a certain geographical area. The data gathering process cannot be changed or affected during the span of this thesis project; the dataset is thus as-is, which means that information missing in the dataset cannot be retrieved from other sources. Bugs in the data gathering process cannot be solved either.

The models used in the thesis project are trained and evaluated on a MacBook Pro (early 2015), with a 2.7 GHz Intel Core i5 processor and 8GB 1867 MHz DDR3 memory. The setup is by no means optimal, as the MacBook Pro used does not have an NVIDIA GPU. The TensorFlow library is thus not fully utilised, as there is no GPU acceleration.

1.4 Structure of the Report

Chapter 2 covers the background concepts used in the report. Relevant work is also described, alongside the methodology employed in the thesis project. Chapter 3 analyses the data in depth. The data gathering process is explained together with the structure and characteristics of the dataset. In the dataset pre-study, problems inherent in the dataset are explained and solutions to the problems are either outlined or implemented. Interesting problems from real-world scenarios are visualised and used to explain various pre-processing steps. The data chapter acts as a stepping stone to the high-level problems described in Chapters 4-6. Chapter 4 covers the problem of poorly detected bus stops, which causes a large proportion of the problems in the dataset. The chapter explains the problem and proposes a solution to it. Chapter 5 focuses on the positional data in the dataset. It attempts to estimate the GPS variation of journeys for a bus line. If the proposed methodology yields good results, the overall precision of positional data can be improved for the dataset. Chapter 6 proposes a method to predict the arrival time of buses, using processed data from the spatio-temporal dataset. The chapter covers the description of the problem, the methodology, and the results of the implemented solution.

2 Background

This chapter briefly covers the binary search algorithm, the background of Gaussian Processes, and relevant studies using Gaussian Processes. Various evaluation metrics are explained, alongside the GPflow framework used for the implementation. Finite-state machines are defined and various techniques in the area of Exploratory Data Analysis are explained.

2.1 Binary Search

Binary search is a common algorithm for searching through a sequence for a target value. The algorithm utilises a binary tree for the search. The algorithm can be described by the following steps, from Knuth [5]:

1. If the tree has no root node, the search is unsuccessful. The target value is not in the tree, as the tree is empty.
2. If the root node has the target value, the search is successful. The position of the target value is found.
3. If the target value is smaller than the root value, the left side of the binary tree is explored. The algorithm returns to the first step of the procedure. This time the tree is the left half of the binary tree, with the new root node being the left child of the previous root node.
4. If the target value is greater than the root value, the right side of the binary tree is explored. The algorithm returns to the first step, but this time the tree is the right subtree of the binary tree. The new root node is the right child of the previous root node.

The procedure is illustrated in Figure 2.1. The algorithm has a complexity of $O(log_2 n)$, as the search space is in the binary case divided in two after each comparison. However, the algorithm requires a sorted sequence of values.

2.2 Gaussian Processes

The Gaussian process [6] (GP) is a Bayesian non-parametric approach to modelling distributions over functions. GPs have throughout the years proved themselves useful for solving a

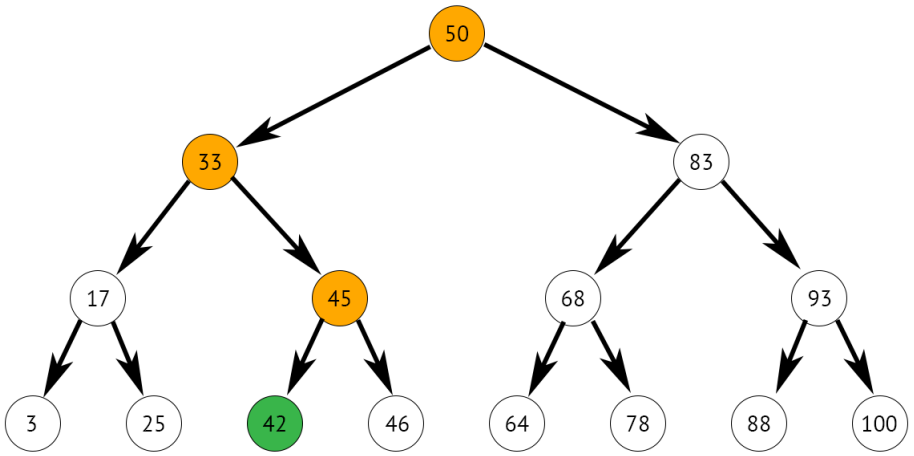


Figure 2.1: Example of a binary search algorithm run. The target value is 42. First the root node 50 is compared, but as $42 < 50$ the left subtree is explored. The new root node is 33 and $42 > 30$, so the right subtree is explored. The new root node is 45 and $42 < 45$. The algorithm should search the left subtree, which in this case only consists of one node, with the target value. The search is successful.

wide variety of real-world problems in various fields. Recent examples of GPs being used are in the field of astronomy to infer intrinsic spectra and stellar radial velocities [7] or to infer stellar rotation periods [8]. GPs have also been successfully used in the field of Biomedical Engineering to develop a real-time algorithm which scores the risk of intensive care unit admission for ward patients [9]. Finley et al. use GPs to predict forest canopy height [10]. In [11], large-scale GPs are used to improve remote sensing image classification and in [12] GPs are used to infer linear equation parameters. In [13], van der Wilk et al. improve the practical aspects for a convolutional structure in GP in order to improve image detection. In [14], Stein conducts a large-scale spatio-temporal data analysis with models based on GPs and in [15], Dong et al. perform continuous-time trajectory estimation using GPs.

Definition

Definition 1 is the formal definition of a GP given by Rasmussen et al. in [6].

Definition 1 *A Gaussian process is a collection of random variables, any infinite number of which have a joint Gaussian distribution.*

A GP is denoted by the equation

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')), \tag{2.1}$$

where the stochastic function $f(\cdot)$ is distributed according to a GP (prior). The GP is fully specified by

- $m(x) = \mathbb{E}[f(x)] = 0$ is the mean function, where the mean function is taken to be zero for notational simplicity [6];
- $k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$ is the covariance function. x and x' are two inputs.

It can be seen as a probability distribution over functions. The GP approach is Bayesian as $f(x) \sim \mathcal{GP}$ encodes a prior belief over the functions. The covariance function can be

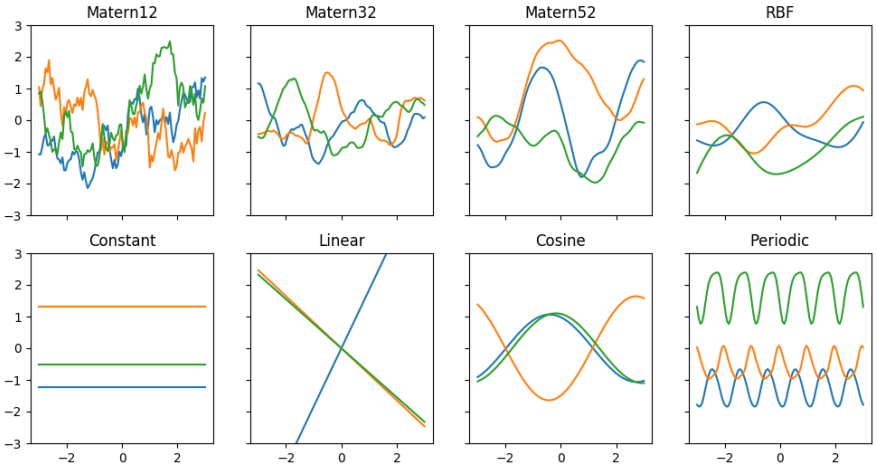


Figure 2.2: Samples from GP priors defined by common covariance functions. The functions exhibit different characteristics depending on the choice of kernel and its kernel parameters.

expressed in the form of a *kernel*, where different kernels exhibit different characteristics. A comparison between common kernels is illustrated in Figure 2.2. The radial basis function (RBF or SE) kernel and Matérn kernel are given by equations 2.2 and 2.3, respectively.

$$k_{\text{SE}}(r) = \sigma^2 \exp\Big(-\frac{r^2}{2\ell^2}\Big) \tag{2.2}$$

$$k_{\text{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big)^{\nu} K_\nu\Big(\frac{\sqrt{2\nu}r}{\ell}\Big), \tag{2.3}$$

where ν and ℓ are positive parameters, K_ν is a modified Bessel function and $r = |x - x'|$ [6].

If $\nu = 3/2$ in the Matérn kernel, the equation can be simplified to eq. 2.4. When $\nu \to \infty$, the Matérn kernel becomes the RBF [6].

$$k_{\nu=3/2}(r) = \Big(1 + \frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{2.4}$$

Gaussian process regression (GPR) aims at reconstruct the underlying, unknown function without the observation noise ϵ. The GPR model is given by

$$y = f(x) + \epsilon, \tag{2.5}$$

where

$$\epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_{noise})$$

Here ϵ represents the noise in each observation, where ϵ is assumed to be additive independent identically distributed (iid) Gaussian noise.

The posterior of a GP is also a GP [6], with a Gaussian predictive distribution given by the posterior GP evaluated at the new point $x_\star$. The GP prediction (posterior) in a point x^* is defined by the equation

$$p(y^*|x, y, x^*) \sim \mathcal{N}(\mu_f(x^*), \sigma^2_f(x^*)), \tag{2.6}$$

where

$$\mu_f(x^*) = K(x^*, x)[K(x, x) + \sigma^2_{noise} I]^{-1} y \tag{2.7}$$

$$\sigma^2_f(x^*) = K(x^*, x^*) - K(x^*, x)[K(x, x) + \sigma^2_{noise} I]^{-1} K(x, x^*), \tag{2.8}$$

where $K(x^*, x), K(x, x)$, and $K(x, x^*)$ each are a covariance (Gram) matrix [6].

Related Work

GPs have been used with spatio-temporal datasets before in order to create local trajectory models [16, 17, 18]. In [16], GPs are used to recognise common activities from spatio-temporal datasets. In [17], sparse GPR is used in order to perform trajectory modelling. The methodology conducted can be adapted and implemented in this thesis project. For example, the equations to aggregate trajectory models can be directly applied in this thesis work. Tiger et al. specifies two ways to aggregate trajectory models by either Fusion or Combining [17]. The aggregation of trajectory models is useful in order to provide coherent predictions from multiple GPs. The Combining formula on the predictive mean and variance is given by

$$\mu(x^*) = \frac{\sum_{j=1}^{J} N_j \mu_j(x^*)}{\sum_{j=1}^{J} N_j} \tag{2.9}$$

$$\sigma^2(x^*) = \frac{\sum_{j=1}^{J} N_j \left(\sigma_j^2(x^*) + \mu_j^2(x^*)\right)}{\sum_{j=1}^{J} N_j} - \mu(x^*)^2 \tag{2.10}$$

where N_j is the number of trajectories used to train the j trajectory model. The predictive mean function and predictive variance function are given by $\mu(x^*)$ and $\sigma^2(x^*)$, respectively.

The two approaches are visualised in Figure 2.3. The red Gaussian has mean 0 and variance 0.2, while the blue Gaussian has mean 2 and variance 0.5. As shown in the figure, the Combining approach creates a Gaussian which captures the variance of all local Gaussians.

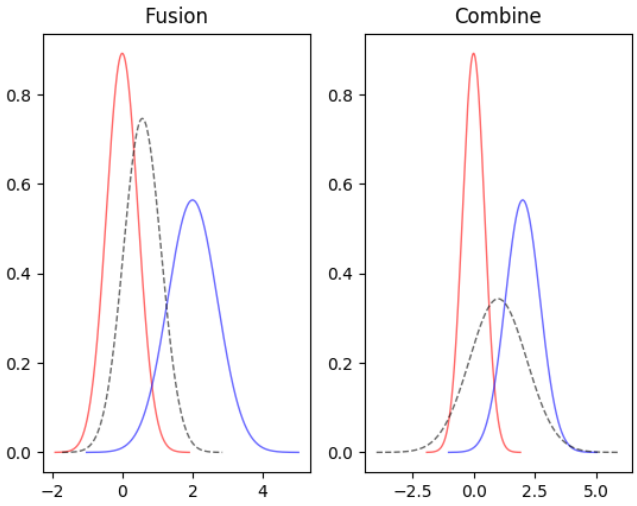


Figure 2.3: Fusion and Combining two Gaussian distributions, with $N_1 = N_2 = 1$. The red Gaussian distribution has mean 0 and variance 0.2. The blue Gaussian has mean 2 and variance 0.5.

2.3 Evaluation Metrics

Common evaluation metrics when comparing regression models are mean square error (MSE), mean squared logarithmic error (MSLE), and mean absolute error (MAE).

$$MSE(y, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.11}$$

$$MSLE(y, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}\big(\ln(1+y_i) - \ln(1+\hat{y}_i)\big)^2 \tag{2.12}$$

$$MAE(y, \hat{y}_i) = \text{median}(|y_1 - \hat{y_1}|, \ldots, |y_n - \hat{y_n}|) \tag{2.13}$$

Typically, the root of the MSE (RMSE) is used as a metric

$$RMSE(y, \hat{y}_i) = \sqrt{MSE(y, \hat{y}_i)} \tag{2.14}$$

RMSE is susceptible to outliers affecting the performance of the metric, as it causes a large variability of the errors [19]. MSLE partly solves the variability of large errors by applying the natural logarithm to the targets y_i and $\hat{y}_i$, respectively. MAE is resistant to outliers, as it only looks at the median of all errors. MAE is praised in [19] by Willmott et al., while Chai et al. defend the use of RMSE in [20]. For example, RMSE is shown in [20] to be a useful metric if the errors follow a Gaussian distribution.

2.4 GPflow

GPflow[1] [21] is a GP framework for the Python[2] programming language, using TensorFlow[3]. It is an open-source software which has its origins from the contributors of GPy[4], another famous framework for implementing GPs in Python. As GPflow is built upon TensorFlow, it achieves better performance when using GPUs for processing [21]. The GPy background and TensorFlow foundation contribute with quality of life features which facilitates easier development, such as automatic differentiation. Building GPflow on top of TensorFlow also allows the benefits of using TensorFlow methods for calculations, e.g., most gradient computations are handled by TensorFlow. All models in this thesis project are implemented using the GPflow framework.

2.5 Finite-State Machines

Finite-state machines (FSM) are well-defined mathematical models which can have numerous representations. The formal notation used here are from the book "Automata and Computability" by Kozen [22]. An FSM is defined as a set of states with potentially multiple transitions between each state. The FSM can only be in one given state at any time.

Formally an FSM is given by the structure

$$M = (Q, \Sigma, \delta, s, F),$$

where

- Q is a finite set of states.
- Σ is the input alphabet of the FSM.
- $\delta : Q \times \Sigma \to Q$ is the transition function for the FSM. It can intuitively be seen as the function which tells the FSM which state to transition to in response to an input. For example, if M is in state q and receives input x it should move to state $\delta(q, x)$.

[1] https://gpflow.readthedocs.io/en/latest/index.html
[2] https://www.python.org/
[3] https://www.tensorflow.org/
[4] https://sheffieldml.github.io/GPy/

- $s \in Q$ is the start state.
- $F \subset Q$; where the elements of F are the final states of the FSM.

An FSM can be modelled with various representations [22], e.g, in tabular form, as a transition diagram or as a regular expression.

2.6 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a broad concept comprising various techniques for exploring and analysing data [23, 24, 25, 26, 27]. Early popular techniques include *box plots* and *stem-and-leaf* displays. A stem-and-leaf plot takes numbers and splits them into two groups. The first group contains the leading digit(s) and the second group contains the trailing digit(s). Figure 2.4 is an example of a stem-and-leaf plot with one leading and one trailing digit. The grouping helps when sorting batches of data and visualising important features, without losing the information of any single data point used [27].

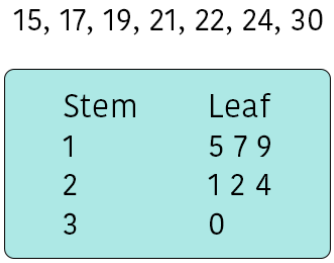


Figure 2.4: Example of a stem-and-leaf plot. The numbers above the plot is the input. The first digit of the number is the *stem*, the following digits are the *leafs*.

EDA can be seen as the application of tools and statistical methods to analyse data in a meaningful way, e.g., it can be applied to the detection of outliers, smoothing the data, or performing a variance analysis [23, 25, 26, 27]. EDA can also reveal subtle practical problems with the chosen model that can easily be missed when performing statistical analysis of the model [24].

In [26], Tukey describes how EDA can be used to answer research questions such as "What is the age distribution for the Vietnamese population?" and "Are there differences in the annual household per capita expenditures between the rural and urban populations in Vietnam?". Tukey uses plots to compare different groups and estimators, such as the sample mean estimator, or the *winsorised* mean, to quantify the difference. The winsorised mean handles the case where tails of a distribution dominates the value space. This causes the sample mean estimator to poorly reflect on the "typical" data point, as it is skewed by the small tail population [26].

In [27], Velleman et al. present different EDA techniques and highlights four key areas of EDA: displays (plots), residuals, re-expressions and resistance. Residuals are what remain after data analysis is performed. Residuals can, for example, be what remains after fitting data to a model (the error of the fit) [27]. Re-expression is the notion of applying mathematical functions to the data. Re-expressing the data can help with the data analysis [25, 27]. Examples of mathematical functions that can be applied are: logarithm, square root, reciprocal square function or generally raising the data to some power p. Resistance is the concept that outliers should not disproportionately affect the data analysis [25, 27]. For example, the winsorised mean estimator is less sensitive to localised misbehaviour than the sample mean estimator [26].

Smoothing data is important for many different applications [28, 29, 30, 27]. This can, for example, be done by applying *running median smoothers*. The running median smoothers

go through all the data points in sequence and calculate only the median for the n closest values near each point [27]. Another approach is the *running weighted average* [27]. Instead of taking the median of the n values, the average is calculated. The average can also be weighted with different functions, like hanning smoothing [27]. The hanning smoothing for three data points is shown in Eq. 2.15. It is worth noting that a single outlier heavily affects the hanning smoothing and that in practice it is common to first apply a running median smoothing to remove outliers [27].

$$\hat{y}_t = \frac{1}{4}y_{t-1} + \frac{1}{2}y_t + \frac{1}{4}y_{t+1} \tag{2.15}$$

3 Data

This chapter describes the spatio-temporal dataset used in the thesis project. The dataset provider is briefly mentioned alongside the data gathering process, followed by the structure of the data. The different event types in the dataset are described and how they are used in the thesis project. A pre-study of the dataset is detailed after the basic characteristics of the dataset have been described. The dataset pre-study analysed the data in-depth and identified problems which occur because of the basic characteristics and their limitations. The problems are illustrated by real-world examples. The pre-processing section covers methods employed to transform the data into usable features. The pre-processing section also mentions solutions to some of the problems identified in the dataset pre-study.

3.1 Background

The dataset is provided by Östgötatrafiken AB and contains 282 GB of data. Östgötatrafiken AB is owned by Östergötland County and is responsible for the public transportation in the county. During this thesis project, only the bus data available in the given data set is analysed. The dataset is a collection of documents, where one document represents a full day of data. The dataset contains 62 days. A typical day has a document size of around 5 GB, see Figure 3.2.

Traffic Lab

Traffic Lab API[1] contains complementary information to the data in the provided dataset by Östgötatrafiken AB. For example, one can retrieve coordinates for all bus stops and the arrival times for buses at each bus stop. However, the arrival times are not updated in real-time for the buses of Östgötatrafiken AB.

Data Gathering

The process of gathering the data used in this thesis project can be generally described by the following simplified procedure:

[1] https://www.trafiklab.se/

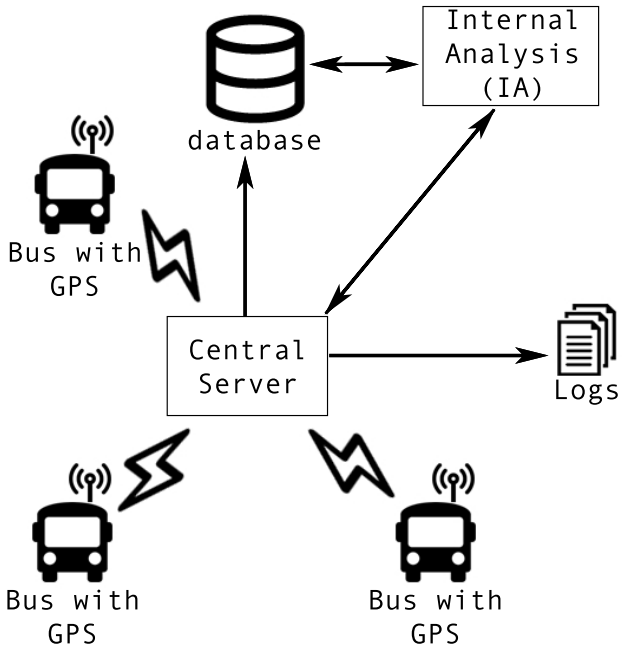


Figure 3.1: Simplified graph illustrating the data gathering process. Each bus is equipped with a GPS sensor and transmits its position to a central server, which is propagated to a database. The dataset used in this thesis project is a collection of logs, where each log is a collection of events. The logs serve as a historical timeline of all events occurring at the central server. Examples of events occurring are the central server receiving the GPS position of a bus or events produced by the internal analysis component.

1. Each Östgötatrafiken AB bus is running a system collecting data from sensors installed inside the bus.
2. The system collects the sensor data and transmits it to a central server or database.
3. A log containing all events for a full day is created and stored as a document in a collection.
4. The central server processes and analyses the data. The results from the data analysis is stored in the log.

Figure 3.1 illustrates the procedure. The collection of logs is the dataset used in this thesis project. The logs contain the GPS data from the buses and also events created by the internal analysis (IA) component in the system. The precise implementation of the IA component is unknown.

3.2 Structure

A document in the dataset is made up of a large number of events representing a single day. A single day typically contains about 23.5 million events, as shown in Figure 3.2. The orange bar is the median value, while the green triangle is the mean value. This shows that there are a few cases where the number of events in a log are much fewer compared to the other log files. Each event is represented by a single line of text. An event can be split into two parts: a header and a body. There are different types of events reported during the span of a single day. Each type has its own header and body structure.

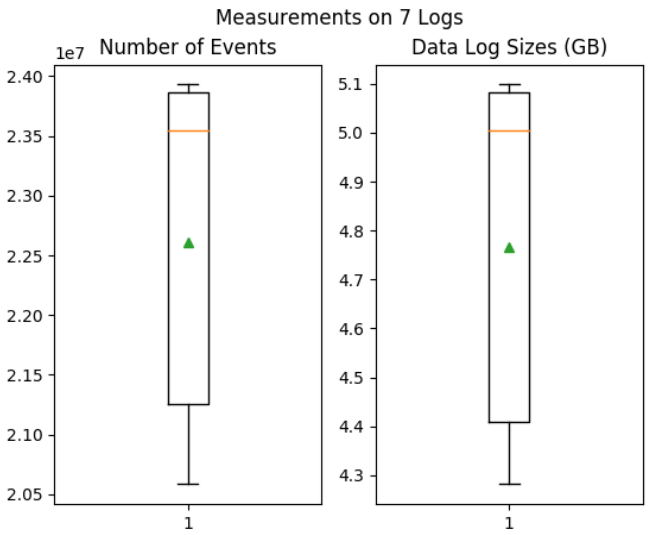


Figure 3.2: Average size of logs. Measurements are made on 7 logs (days) in the dataset. Naturally, there is a relationship between the number of events in a log and the size of the log.
The orange bar is the median, while the green triangle is the mean value.

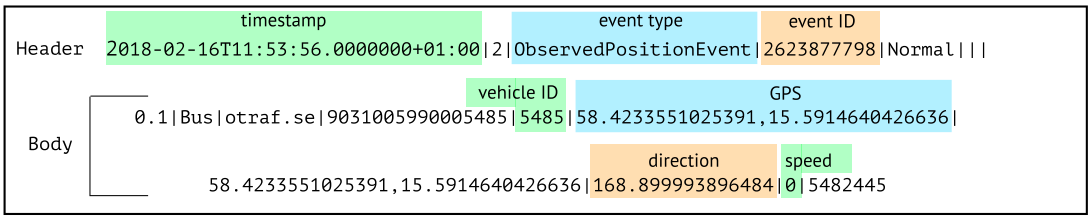


Figure 3.3: Example of a raw `ObservedPositionEvent` entry. The header and the body are separated by `|||`. Each parameter in the header and body is separated by a single `|`. Key parameters for the `ObservedPositionEvent` event type is highlighted.

Event Example

Figure 3.3 illustrates an event of the type `ObservedPositionEvent`. The header is defined as all the parameters before the `|||` separator. All the parameters after the separator are defined as the body of the event. In this example the header and body contain seven key parameters:

- *Timestamp:* A timestamp (`2018-02-16T11:53:56.0000000+01:00`), which is the timestamp from the system running on the bus.
- *Event Type:* The event type (`ObservedPositionEvent`).
- *Event ID:* The event id (`2623877798`). This is a number set by the system responsible for collecting the data from all buses. It is incremented for every event added to the log by either the database system or the IA component in Figure 3.1.
- *Vehicle ID:* Unique ID for the bus transmitting its position.
- *GPS:* The GPS position of the bus in latitude and longitude.
- *Direction:* The angle of rotation of the bus, in degrees.
- *Speed:* The current speed of the bus, in metres per second.

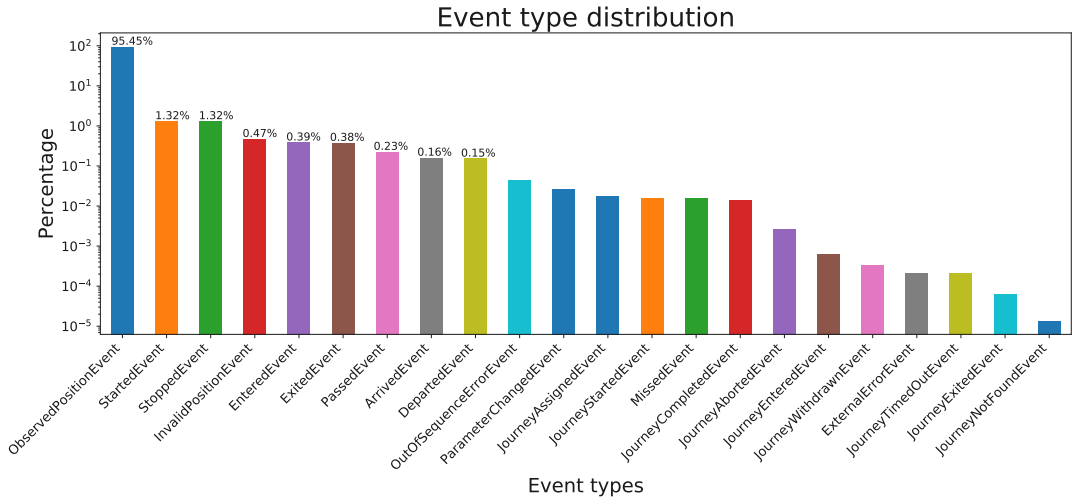


Figure 3.4: The distribution of event types for an arbitrary day in the dataset.

Event Types

The dataset contains 22 unique event types. Figure 3.4 visualises the distribution of event types for an arbitrary day in the dataset. Only event types from buses are visualised, events from trains are discarded. The figure only gives an indication of what the true distribution can be, as it is computationally expensive to calculate the true distribution for the given dataset, due to its size. Knowledge about the true distribution is not required in order to reason about the event types. As the figure shows, the majority of events that occur are of the type `ObservedPositionEvent`, which is the event containing an updated GPS position for a vehicle.

Of the 20 event types available in the dataset, this thesis project only uses 12. The events used are chosen by analysing the log for a single day in great detail. Event types which occur rarely and seemingly at random are discarded, as no use can be determined for them within the scope of this thesis. For example, the `InvalidPositionEvent` type only contains the GPS position of the vehicle. The `ObservedPositionEvent` type also contains the *Speed* and *Direction* parameters. All of the `InvalidPositionEvents` events have the same coordinates. In this thesis project the `InvalidPositionEvent` type is discarded, due to the missing parameters and static GPS position.

The 12 event types used in this thesis project are:

- `ObservedPositionEvent`: This event type contains the information highlighted in Figure 3.3. It is the most prevalent event type in the provided dataset. This event type is contextless, as it contains no information about which public transportation line the vehicle is currently serving, if any.

- `StartedEvent` and `StoppedEvent`: These two event types provide context to a sequence of observed position events. They denote when the vehicle has started or stopped moving, respectively. For example, they can be used to identify road intersections, bus stops, traffic or driver breaks.

- `EnteredEvent` and `ExitedEvent`: These two event types are used by the IA component to identify bus stops. The `EnteredEvent` is produced by the system when the vehicle is within a certain predefined distance of a bus stop. The `ExitedEvent` is similarly produced when the vehicle leaves the predefined distance of the bus stop. These event types can, for example, be used in an algorithm which improves the bus stop detection.

- `PassedEvent`, `ArrivedEvent`, and `DepartedEvent`: These three event types are used to provide information regarding which bus stop a particular bus is at. The `PassedEvent` type denotes when a particular bus, serving a specific public transportation line, passed a bus line stop. It contains information about the predefined position of the stop, the public transportation line the particular bus is currently serving and the time of the passing. Similarly, the `ArrivedEvent` and `DepartedEvent` types denote when a particular bus arrives at or departs from a particular bus stop. These event types provide context to the observed positions. In this thesis project they are used to group a sequence of observed positions into a segment between two stops for a particular bus line.

- `ParameterChangedEvent`: The `ParameterChangedEvent` type is the most dynamic event type in the data set. For example, it can be used to inform the system when the doors on a particular bus open or close or when a bus changes journeys. In this thesis project it is only used to identify bus lines and give context to observed positions.

- `JourneyStartedEvent` and `JourneyCompletedEvent`:
  The `JourneyStartedEvent` type is produced by the IA component when a bus has reached the starting bus stop of a bus line. The `JourneyCompletedEvent` event type is produced when the bus has reached the final bus stop of a bus line.

- `JourneyAssignedEvent`: This event type is in most cases accompanied by a `ParameterChangedEvent` to denote when a bus changes its journey. It contains the time of the assignment, the position of the bus and the bus line the bus is assigned. The information is thus similar to the `ParameterChangedEvent`.

3.3 Dataset Pre-Study

The dataset pre-study is conducted after the basic characteristics are identified. It analyses the dataset in greater depth, identifying interesting scenarios that occurs in the data. The analysis is conducted by manually parsing the data. This manual task uses EDA methods to simplify the process; the focus is put on visualising the data in order to detect patterns and outliers. All events during an arbitrary day are initially used for this task. Anomalies that occurred are isolated and categorised by applying the rationale behind the binary search algorithm. The log is divided in two equal halves in order to see which half inhibits the anomalies. The procedure is repeated as long as anomalies are present. This halves the search space at each iteration, as long as anomalies are not present in both halves. If an anomaly is missing from both halves it is an indication that the anomaly occurs by combining the events around the split.

The search is more complex than one simple binary search algorithm run, as anomalies can occur in both splits. If anomalies are present in both splits, the split intervals are saved and then the search proceeds in one of the splits. When the search exhausts the arbitrarily chosen split, it starts searching the other split using the saved interval. The method can thus be seen as a binary search algorithm which branches and starts another binary search processes if anomalies are detected in both splits.

Three interesting scenarios are identified in the dataset pre-study: human error induced early stopping, imprecise algorithms causing final bus stops to be missed, and self-overlaps.

Human Error: Early Stopping

Figures 3.5 and 3.6 illustrate two scenarios when the `JourneyCompletedEvent` type for a started journey is missing. In Figure 3.5, the bus driver is supposed to follow the red line and visit the three markers in order to complete the journey for a particular line. Instead, the bus follows the green line and the three markers are skipped. In Figure 3.6, the bus drives along

the planned path (the green line). When the bus reaches the crossing marked by the arrow, the bus turns and drives along the blue line instead of following the planned route (the red dashed line). In both these real-world examples, the bus drivers ignore the last few bus stops of the bus lines.

The IA component never deems the journey as completed in these scenarios, which results in the `JourneyCompletedEvent` type never being produced. This scenario is easy to detect in historical data, but in real-time certain assumptions need to be made. For example, if the journey is compared with an average journey, the anomaly can be detected early. However, it is uncertain if the anomaly is due to a journey being stopped early or if the bus driver took a wrong turn on the highway. A time constraint threshold can be introduced to the system in order to separate these two cases. In the case of a wrong turn, the bus eventually converges to the average journey, while in the early-stopping scenario the journey most likely never converges.

Figure 3.5: Example of early stopping in a journey. The three markers are the final three stops of a particular bus line. The green line is the route the bus drives. Instead of following the planned, red route of the bus line, the final three stops are skipped. This results in the journey never being deemed completed.

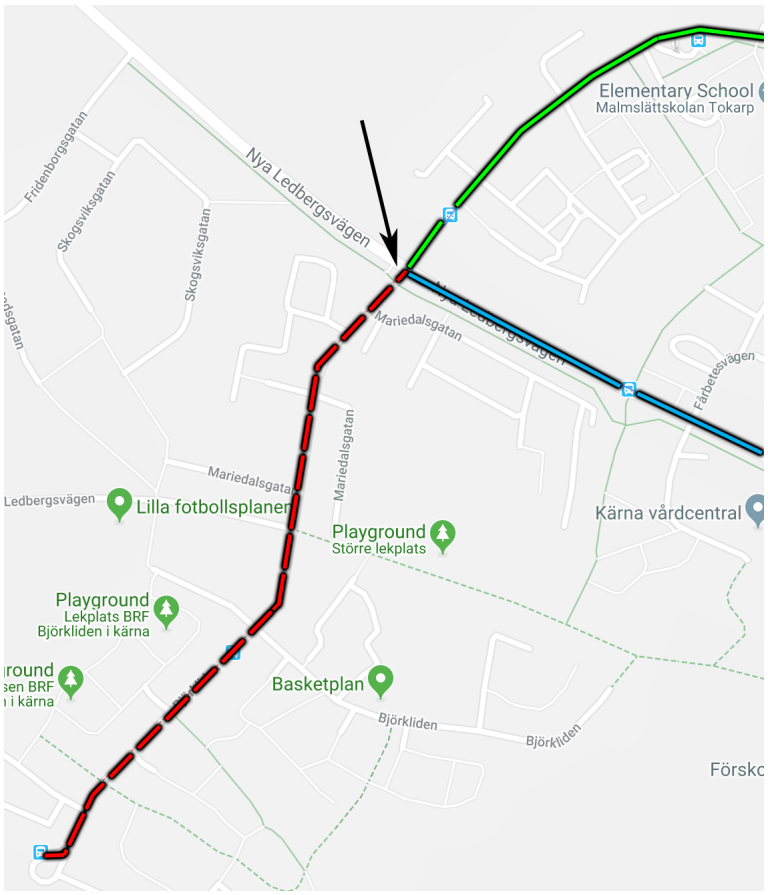


Figure 3.6: Another example of early stopping in a journey. The bus drives on the green line and reaches the crossing marked by the arrow. The planned route for the bus line is to follow the red dashed line. Instead of following the planned route, the bus drives along the blue line. This results in the journey never being deemed completed, creating an erroneous ordering of contextual events (`JourneyCompletedEvent` never received).

Imprecise Algorithms: Final Bus Stop Missed

This scenario occurs due to a mix of imprecision in the bus stop detection algorithm and human error. The scenario is illustrated in Figure 3.7. The bus driver completes the journey of a particular bus line and reaches the final bus stop. However, the IA component does not detect that the bus stop has been reached. This is due to the bus driver stopping the bus roughly 45 meters before the final bus stop. The bus stops at the red marker for a few minutes, before it starts driving towards the first bus stop of the next planned journey. The route after the red marker is erroneously included in the journey for the bus line, as the journey is never completed.

This systematically occurs at certain bus stops, due to there being a "waiting" space commonly used by bus drivers while waiting for a new journey to be assigned. The scenario highlights a problem with the implemented bus stop detection algorithm. The bus stop detection algorithm can be improved to both handle these scenarios and yield more precise bus stop detection. An improved bus stop detection method is proposed in Chapter 4.

Missed final bus stops cause problems for both the data analysis and for specific applications. For example, journeys contain more `ObservedPositionEvents` than they should, as the journeys are never deemed completed. This results not only in journeys being discarded as erroneous, but also affects other events and the ordering of events. The pre-processing step has to adapt to these problems, which increases the complexity of the data pre-processing.

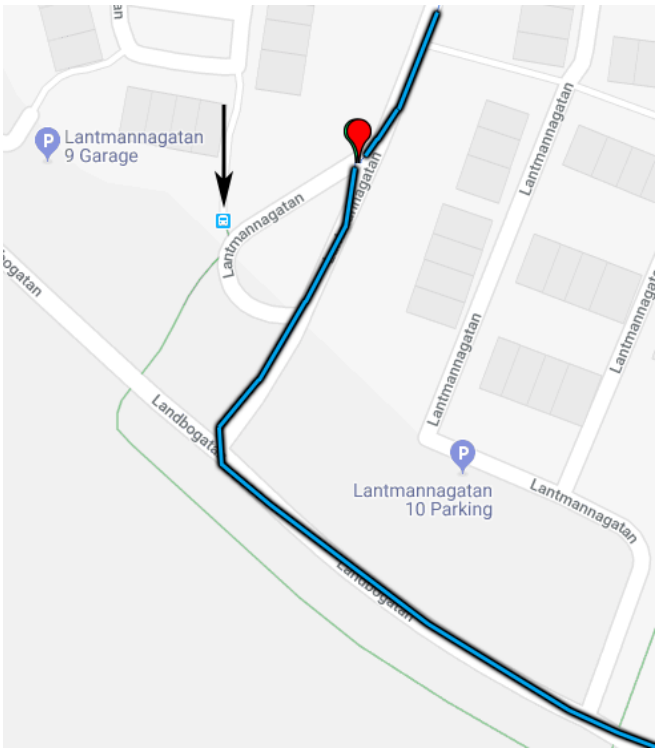


Figure 3.7: Real-world example of a bus driver stopping roughly 45 meters before reaching the final bus stop. The final bus stop of the bus line is marked on the map by the blue bus icon (indicated by the black arrow). The red marker is the GPS position where the bus stopped. The blue line is the route the bus drives. The bus stops at the marker for a few minutes before it continues to the first bus stop for the next journey. The system does not register the bus as having passed the bus stop indicated by the black arrow. Thus, the system does not determine the journey as completed. This causes the route after the red marker to be included in the trajectory. The bus stop detection algorithm systematically does not identify the cases where a bus stops at a "waiting" space.

Self-Overlap

Self-overlap causes issues when used as training data for GP models [18]. Self-overlapping causes instabilities. In the specific applications of this thesis project, self-overlapping causes problems with the trajectory forecasting. For example, overlapped GPS positions results in ambiguous arrival time predictions.

An example of self-overlap is shown in Figure 3.8. Self-overlap can be detected manually by visually looking at the trajectories, which is the method used in this thesis project. In [18], an approach to handling and detecting self-overlaps is explained. The distances between subsequent points on a latent function f of the trajectory is calculated with a given resolution θ. It is possible, using this approach, to find self-overlaps down to the given resolution [18]. Generally, f is unknown and needs to be estimated using a GP model trained on the full trajectory [18]. If the pair-wise distance between non-subsequent interpolated points is under a threshold, then the self-overlap is detected.

3.4 Pre-Processing Events

The first step deemed necessary in order to use the data in the provided dataset is to transform it from strings to objects with attributes. Figure 3.3 visualises which attributes an `ObservedPositionEvent` object contains. Similar structures are created for each of the mentioned event types.

Figure 3.8: Example of self-overlap in a bus journey. The red lines show the trajectories of the buses driving for the same bus line.

During this pre-process step all the events from a vehicle type other than "Bus" are ignored. A *geo-fence* is applied in order to facilitate and support manual inspection and visualisation of the data in the dataset. A geo-fence is a virtual polygon which establishes a virtual perimeter for a real-world geographical area. The geo-fence applied in this thesis project is shown in Figure 3.9.

After the parsing and filtering step the idea is to provide context to `ObservedPositionEvent`s. Only using this one contextless event type greatly reduces the span of potential problems one can solve with the provided dataset. Context is provided by constructing a FSM.

Context-Providing Finite-State Machine

The context-providing FSM constructed in this thesis project is shown in Figure 3.10. The shown state machine is illustrating the best-case scenario, when the actual order of events is equal to the logical ordering of events, see Figure 3.11 for a real-world example. However, this is not always the case when working with real-world data, as shown in Figures 3.12 and 3.13. Occasionally the timing of events gets mixed up, e.g. a bus in the "Started" state receives a `JourneyAssignedEvent` before it receives a `JourneyCompletedEvent`. This ordering breaks the logical ordering of events: a journey needs to be completed before a new one can assigned.

The problem is partly solved by changing the ordering of events from *Timestamp* to *Event ID*. This is a feasible approach when the data is batched into separate files, where one file is a full day of events. Unfortunately, this approach does not work when dealing with streamed real-time data, as the ordering cannot be changed without introducing a buffer. The context-providing FSM is slightly altered in order to support the processing of streamed real-time data. Another motivation behind the change is to remove the assumption that the sequence of *Event ID*s is always correct. The `ObservedPositionEvent`s are instead placed in a temporary buffer once an anomaly is detected in the event ordering. For example, if a new `JourneyAssignedEvent` is received before the supposed `JourneyCompletedEvent`

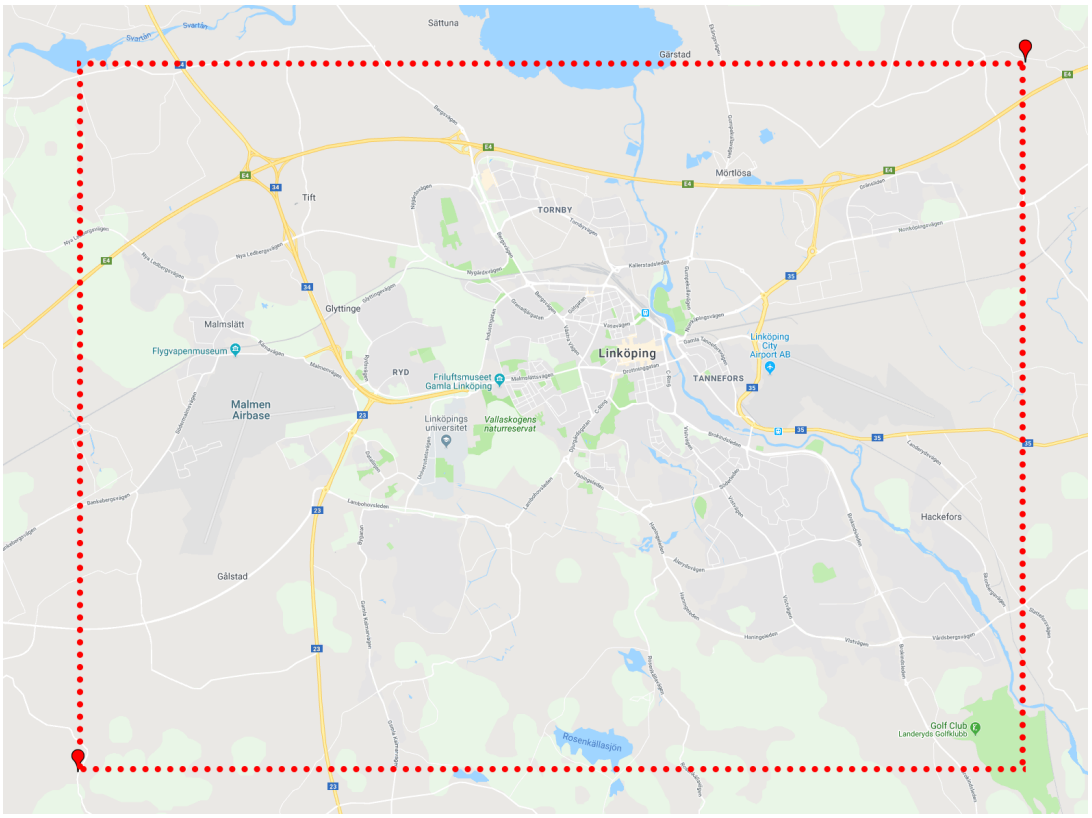


Figure 3.9: A geo-fence is constructed to filter out events occurring outside the virtual perimeter. The two red markers create a rectangular boundary, which is illustrated with the red-dotted line. The geographical area is the city of Linköping.

and the system is in the "Started" state. Once the `JourneyCompletedEvent` type is received, after some delay, the data in the buffer is retroactively added to the "Started" state. The "Started" state then correctly contains all `ObservedPositionEvent`s received from the starting bus stop to the final bus stop. However, it is not always the case that the `JourneyCompletedEvent` type is contained in the journey. Occasionally, the type is missing from the sequence of events due to either human errors or imprecise algorithms in the IA component. These scenarios can easily be detected by utilising the context-providing FSM with the added buffer extension.

Bus Stops

Using the FSM provides context to the `ObservedPositionEvent` types. The states introduced yield a simple way to visualise contextual paths, e.g., actual journeys for a particular bus line or the path a bus drives to start a journey under a new bus line number. However, the context-providing FSM solution described does not handle events about a bus arriving, departing or passing a bus stop on the journey. Handling this type of data is a critical step in detecting imprecisions in the bus stop detection algorithm or early stopping due to human error. The "Started" state in the FSM can be extended to not only include `ObservedPositionEvent` types, but also `ArrivedEvent`, `DepartedEvent`, and `PassedEvent` types. A missed final bus stop can then, for example, be identified by looking at all the bus stops added to the "Started" state for that journey and compare them to the bus stops in other journeys for that particular bus line. Extending the FSM to handle the bus stop data causes the complexity of the FSM to increase.

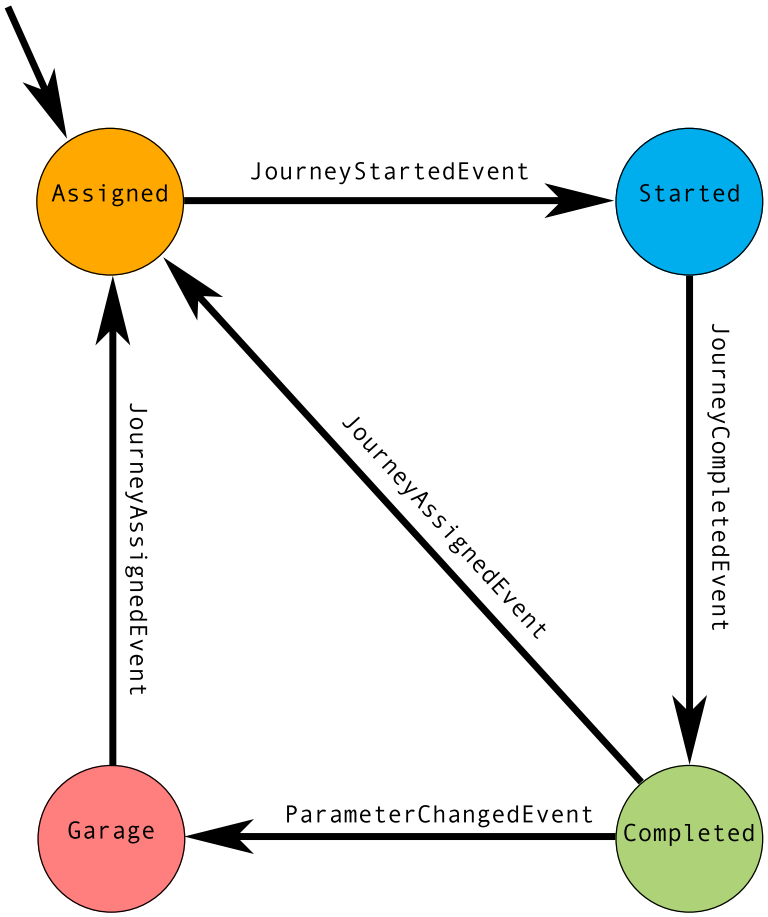


Figure 3.10: Finite-state machine providing context to `ObservedPositionEvent`s. The constructed FSM is simplified to illustrate the best-case scenario. The "Assigned" state is the starting state. `ObservedPositionEvent`s are assigned to the current FSM state.

An alternative approach is implemented instead, as the existing context-providing FSM can easily be verified to produce correct output. The approach extracts all events of the types `StartedEvent`, `StoppedEvent`, `PassedEvent`, `ArrivedEvent`, and `DepartedEvent` for each specific *Vehicle ID*. These events are then paired with the journeys for each respective *Vehicle ID*. Any events of the type `StartedEvent` or `StoppedEvent` occurring before the first `ObservedPositionEvent` of the journey are discarded. Events of the type `PassedEvent`, `ArrivedEvent` or `DepartedEvent` are included in the journey if they occur within a specific period of time to the first `ObservedPositionEvent`. When all `ObservedPositionEvent`s of a journey are processed, the procedure continues with the next journey. The set of extracted events thus naturally reduces in size for each journey processed. Events occurring in-between journeys are discarded. When adding a bus stop to the journey, the name of the bus stop is also added to a separate array, in order to simplify the process of detecting erroneous journeys.

Results

The result of the pre-processing step is a collection of journeys for each bus line. A journey consists of all the `ObservedPositionEvent`s sent by the bus while the FSM is in the "Started" state and all the bus stops the bus arrived at, departed from or passed by. It also contains all the stops and starts made by the bus during the journey. Erroneous event type ordering is solved by sorting the events based on *Event ID* and using a buffer to retroactively

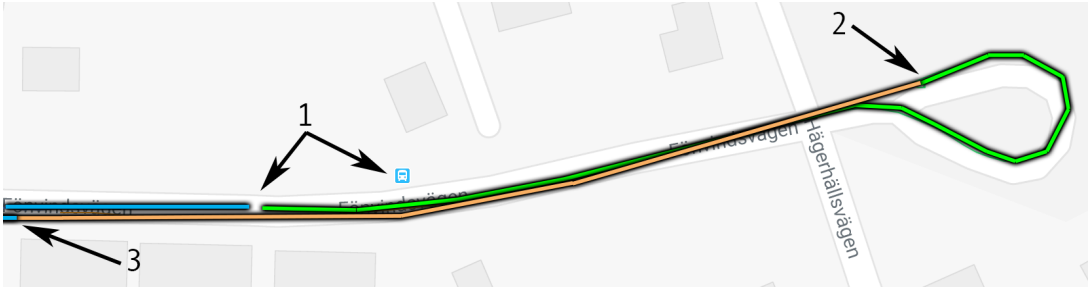


Figure 3.11: Real-world scenario illustrating when events occur in a logical ordering. The blue line is `ObservedPositionEvents` in the "Started" state. Upon reaching the final bus stop for the line the state changes to "Completed" (1). The `ObservedPositionEvents` for this state is drawn with a green line to denote the "Completed" state. The bus turns around and stops for a period of time until a new bus line is assigned to it (2). In this particular scenario, the bus is assigned the same bus line number, but in the opposite direction. The orange line denotes the `ObservedPositionEvents` in the "Assigned" state. Shortly after passing the first bus stop the orange line changes to blue (3), which denotes the "Started" state.

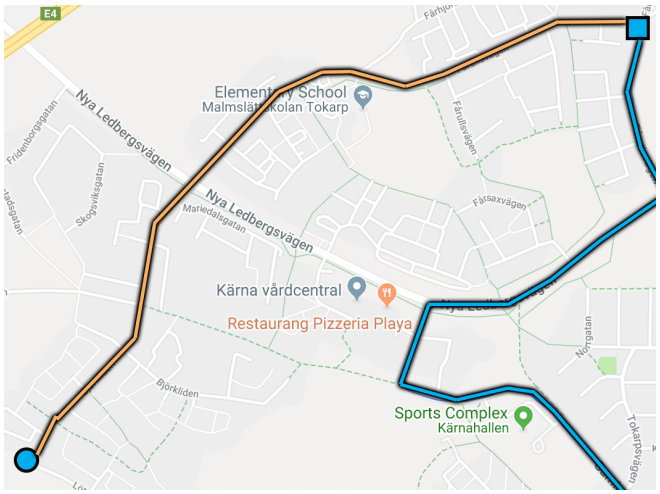


Figure 3.12: Example of early stopping in a journey. The bus is assigned a new bus line long before reaching the final bus stop. The final bus stop is marked with a circle. The rectangle marks the position of the bus when it is assigned a new bus line. The last part of the journey (the path between the rectangle and the circle) is thus assigned to a new state "Assigned", instead of the actual, logical state "Started".

add `ObservedPositionEvent` in the "Assigned" state to the "Started" state in the case of an early journey assignment. Journeys with early stopping or missed final bus stops are still prevalent in the collection of journeys. These can be detected by analysing the bus stops registered during a journey. In this thesis, the erroneous journeys are only marked as anomalies and discarded. An index-tree is constructed to quickly access all journeys for a bus with a particular *Vehicle ID*. A number of high-level problems can now be formulated by using the result from the pre-processing steps.

Discussion

The erroneous journeys are only marked as anomalies and discarded in this thesis. The fraction of erroneous journeys can be calculated and used as a baseline when comparing an improved bus stop detection method with the existing one. Erroneous journeys can be categorised based on the error in the journey, e.g., a missed starting bus stop is a different error compared to a missed final bus stop. The categorisation of faults can provide deeper insights

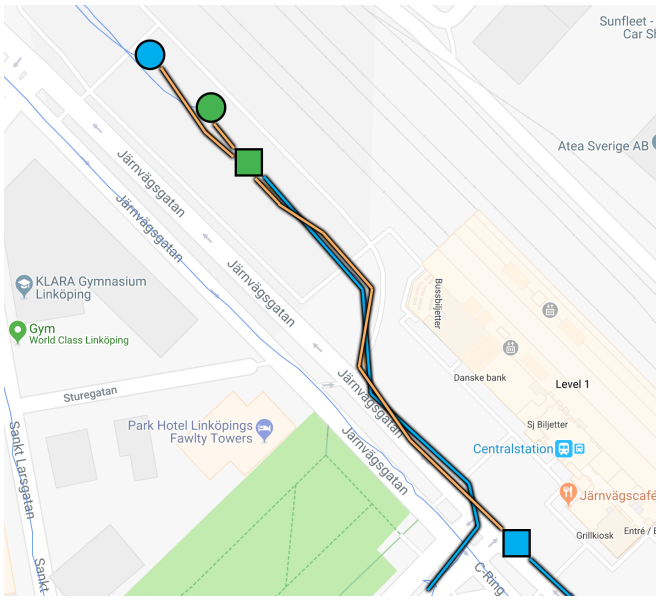


Figure 3.13: Another example of early stopping in a journey. The two circles (green and blue) denote the final bus stops for the respective bus lines. One of the buses is assigned a new bus line at the blue rectangle in the bottom-right corner, long before reaching the final bus stop. The other bus is assigned a new bus line at the green rectangle, which is closer to the final bus stops. This example demonstrates that the assignment to a new bus line is independent of the distance to the final bus stop.

into the dataset. For example, the insights can be used to improve the bus stop detection algorithm or identify journeys where a particular error occurs regularly.

Discarded Event Types

The discarded event types can be investigated in-depth with regard to the whole dataset and not only for an arbitrary day. Some of the events can perhaps be used to easier detect or handle the erroneous journeys. During the initial manual processing of the data, no use is discovered for any of the excluded event types. This may be an indication that the rarer event types are wrongly produced by the IA component. A more probable explanation is that the method of manually investigating the context different event types occur in is subpar and prone to human errors. An automated approach tailored to focus on rare event types probably yields better results and a use for the discarded event types.

Train Events

It would be interesting to perform the same analysis done in this chapter on the data from public transportation trains. The trains exhibit different behaviour than busses, as traffic is different and perhaps more static, with dedicated waiting tracks.

3.5 Future Work

The discarded event types should probably be explored further. In a setup with more processing power, the requirement of a geo-fence can be removed. Different environments can easily be analysed by allowing events from a larger geographical area. The context-providing FSM probably has to be changed after analysing discarded event types, if any of them are deemed useful. Perhaps a combination of discarded events can result in a simplification to be possible in the FSM. However, the risk is that the complexity of the FSM increases with more events types processed.

4 Bus Stop Detection

The first high-level problem is the issue of inaccurate bus stop detection. This causes the system to occasionally miss bus stops, which results in erroneous or incomplete journeys. Incomplete journeys require more pre-processing to be done in order for the data to be usable. An example of an erroneous journey is shown in Figure 4.1, where a bus passes a bus stop before the system registers that the bus departed from the previous bus stop.

```
([491, 491],
 'Carl Cederströms gata',
 [(datetime.datetime(2018, 2, 18, 2, 25, 50, tzinfo=tzoffset(None, 3600)),
  'ArrivedEvent'),
  (datetime.datetime(2018, 2, 18, 2, 26, 37, tzinfo=tzoffset(None, 3600)),
  'DepartedEvent')]),
([528, 528],
 'Flygvapenmuseum',
 [(datetime.datetime(2018, 2, 18, 2, 26, 27, tzinfo=tzoffset(None, 3600)),
  'PassedEvent')])
```

Figure 4.1: Example of poor bus stop detection. Real-world example of a bus being registered by the system to pass the "Flygvapenmusem" bus stop at 26:27, while it later registers the bus as having departed from "Carl Cederströms Gata" at 26:37, 10 seconds later.

Another problem with the existing bus stop detection algorithm is that bus stops provided by Östgötatrafiken AB and the Traffic Lab API do not always correspond to the actual bus stops. This means that erroneous bus stop positions are used in the algorithms of the IA component.

Figure 4.2 shows the results from the existing bus stop detection algorithm of the IA component. Each bus produces seven GPS coordinates for each bus stop: one blue, two red, two yellow, and one green.

- **Green:** The green GPS coordinate is always the same between all buses for each bus stop; it is the pre-determined coordinate for the bus stop.
- **Blue:** The blue GPS coordinate is the position of the bus when it departs from the bus stop (the last positional update within a pre-determined radius of the bus stop).

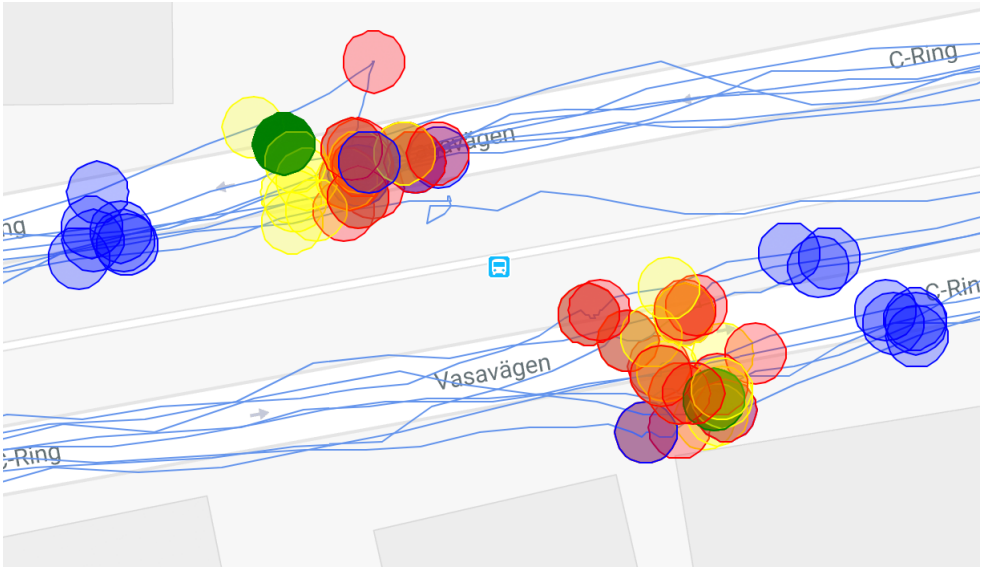


Figure 4.2: Output from the bus stop detection algorithm from the IA component. The green circles are the pre-determined coordinates for the bus stop. The bus stop shown has two platforms, depending on if the bus travels west (top road) or east (bottom road). The red and yellow clusters are the GPS positions of the bus when it arrives to the bus stop. The blue clusters are the GPS positions of the buses departing from the bus stop.

- **Red and Yellow:** The red and yellow coordinates are the positions of the buses when the system determines that they have arrived and stopped at the bus stop. Due to lacking documentation, the difference between the red and yellow coordinates are unclear as they are occasionally identical, but not always (as shown in the figure).

The existing bus stop detection algorithm thus checks within a pre-determined radius around the green GPS coordinate for updates from buses. When the first positional update from the bus is received, the system tracks the bus and waits for it to stop. Upon stopping, the bus is determined to have arrived at the bus stop. When the bus starts again, the system continues to track it and reports when the bus has left the radius of the bus stop. The blue GPS coordinate shows where this happens.

4.1 Methodology

This section covers the outline on how to improve the bus stop detection algorithm. However, due to time limitations for this thesis project, the steps in the outline merely act as a hypothesis on how the bus stop detection can be improved. The steps are not implemented and the method is thus not evaluated.

Data Pre-Processing

Data pre-processing is the first step required. The procedure described in Section 3.4 is executed in order to pre-process the data. The result of the pre-processing step is a collection of journeys, extended with information regarding bus stops. In addition to these results, the journeys are also extended with `EnteredEvent` and `ExitedEvent` types.

Estimate Bus Stop Position and Detection Radius

One of possible improvements to make is to introduce a dynamic radius to bus stops, i.e., from data estimate the position and detection radius for each bus stop individually. The

radius can be based on analysis of the positional data from buses passing by the bus stop. Instead of using a pre-determined radius, the radius can be decided my a model trained on observed positional data. This only works for bus stops that detect buses approaching.

Certain bus stops are prone to be exploited by bus drivers, e.g., the example in Figure 3.7. The errors can be detected by analysing bus stops visited during a journey and compare the list against the expected list. Manual inspection is then required, in a pre-study step, to categorise the errors. Some errors, like the one in Figure 3.7, can be added as exceptions to the bus stop detection algorithm. In this scenario, the bus stops and waits for a new journey assignment close to the final bus stop. This particular scenario happens more than once throughout a single day, so it is a common bus driver behaviour. The bus stop detection algorithm can adapt to this and increase the detection radius for this particular bus stop.

Multimodal Bus Stops

Occasionally, bus stops have space for more than one bus dropping off and picking up commuters. This is not reflected in the dataset; the bus stops in the dataset are unimodal. There is only one green circle for each platform of the bus stop. An improvement to this is to look at the stopped positions of buses at every bus stop. The positional data creates a distribution with (longitude, latitude)-pairs denoting the different modes for each platform. For bus stops with space for multiple buses on a single platform, the expected result is a multimodal distribution over the actual coordinates a bus stops at. Figure 4.3 shows a simulated example of a bus stop having two modes.

One of the benefits when adding information about multimodality to the system is for the possibility of analysing the design of bus stops. The system produces a distribution over positions of stopped bus at each platform for each bus stop. A multimodal distribution for a bus stop with space for only one bus implies that the design of the bus stop can benefit by being changed. A wide multimodal distribution can, respectively, imply the need for certain changes to the bus schedules. For example, at specific times it can be the case that multiple buses simultaneously arrive at the same bus stop due to the nature of the time schedule. This either causes a queue at the bus stop, or potentially the unsafe loading and unloading of commuters. A slight alteration in the schedule for a few of these buses can prevent this from occurring as often. Arrival time prediction can thus be used as an assisting tool to predict bus stop queues.

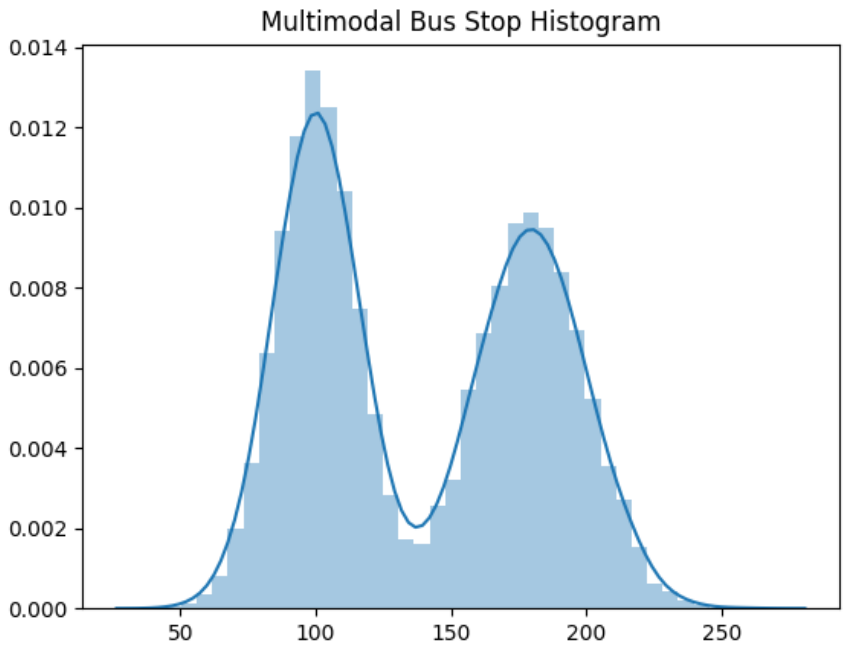


Figure 4.3: Simulated example of a multimodal bus stop. The buses mainly stop at two different positions of the platform (the two modes).

5 GPS Variation Estimation

The main component of the dataset is the GPS positional update events from the vehicles. They contribute to 95% of the events in the data, as shown by Figure 3.4 in Section 3.2. An interesting high-level problem is thus the estimation of the variance in the GPS positions. Manual inspection shows that the quality varies between trajectories and for different parts of the road network. The quality also seems to change over time. If there is a measurement for the variation in variance, one could make more informed decisions regarding the detection of bus stops and the determination of bus positions.

The hypotheses are that the GPS positions vary depending on:

1. *Environment (Spatial Locality):* The surrounding environment impacts the visibility of GPS satellites. For example, in an environment with a lot of reflections and obstacles, the precision of the GPS position will decrease. On the other hand, in open fields the precision will increase, as there are fewer obstacles blocking the GPS satellite signals. The environment can vary between different road segments, but also inside one road segment, e.g., a road can be partly covered by houses or trees. The variance can either be seen as continuous, where each point on the journey has its own variance, or as road segments, where each road segment has its own variance. In the continuous case, neighbouring pairs of points are connected and thus exhibit more similar variance, depending on the distance between the points (granularity) in a journey. In the case of road segments, different segments exhibit a potentially larger difference in variance.

   Road segments can either be created geographically, e.g., a segment is the road between two bus stops or between crossings, or contextually from the output of the continuous case. A geographical road segmentation most likely results in a higher average variance, as the environment potentially changes within the segment. The contextual segmentation can look at sequences of points in the journey, and group the sequential points which exhibit similar variance. The result is potentially segments where the inherent variance of each segment is lower than between different segments.

   Figure 5.1 illustrates how the GPS variance can differ for different road segments. In the shown scenario, the difference between geographical segmentation between crossings and contextual segmentation would be minor, as the variance seems to be similar for a given road segment. However, the difference is large if the geographical segmentation is done between bus stops. The number of road segments is higher in the

crossing-geographical approach than in the contextual approach, as, for example, "Järnvägsgatan" has many crossings but exhibit similar variance.

2. *GPS Sensor Calibration:* Each bus has its own GPS sensor, where different buses can have different sensor models with their own calibrations. In the general case the different GPS sensors show small variance due to different calibrations, environments, and satellite positions. However, occasionally they can vary, e.g., by some constant factor, as shown in Figure 5.2. The single orange-coloured bus journey seems to have a constant offset compared to the rest of the journeys for that particular bus line. The shown scenario is quite extreme, as the offset is large enough to cause problems with the Bus Stop Detection Algorithm in the IA component. No bus stops are detected in this particular scenario. The extreme situations can thus be detected by combining the positional event data with the bus stop event data. However, differences in calibrations which yield smaller offsets cannot be detected with this combining approach and are thus latent in the GPS variation estimation model.

3. *Time:* The timestamps of the position events affect the precision of the GPS sensor. At a single given point of the journey the precision could vary depending on the positions of satellites and the amount of satellites visible. The hypothesis is that there is a constant bias to the GPS positions depending on the time. The positions of the satellites thus cause a small offset to occur in the GPS position. This parameter probably has some periodicity, which could be fitted using a Gaussian Process with a periodical kernel.

Figure 5.1: Visualisation of multiple journeys by different buses in the same area. Each blue line is a journey in the "Started" state. The variance differs between different road segments, as shown around "World Class Linköping" compared to "Järnvägsgatan".

In this thesis project, 37 trajectories from a single day is used. Sections 5.1-5.4 cover the methodology used for the GPS variation estimation. The results are presented in Section 5.5 and discussed in Section 5.6.

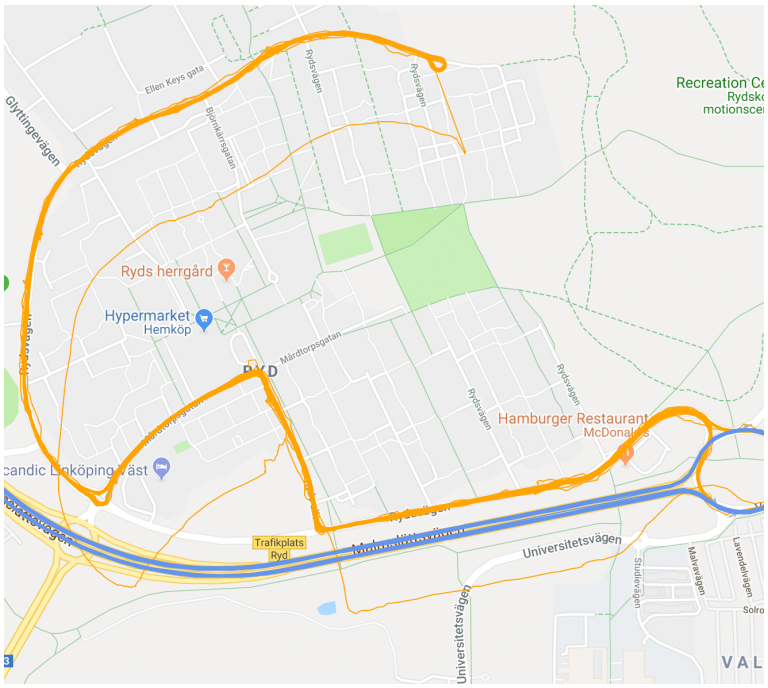


Figure 5.2: Illustrates the case of a poorly calibrated GPS sensor on an individual bus. The orange lines creating the thicker line are all the other journeys for the same bus line, but done by other buses. The thin orange line is one bus with an extremely poorly calibrated GPS sensor. It seems to have a constant offset compared to the rest. This real-world scenario shows that the calibration of GPS sensors is important in order to produce useful data.

5.1 Stop Compression

Each stop present in the journey causes an imbalance of points being clustered around that stop. The longer the stop is, the more points are present. This results in problems when using a GP model, as certain coordinates would thus be denser than others. The GP would model the noise of the GPS in these denser areas, instead of the variance due to the environment.

Stop compression can be achieved in multiple ways, e.g., stopped positions could be either averaged or ignore completely. The time of the stop is also adjusted. In this thesis project, all `ObservedPositionEvents` with a *Speed* value lower than 0.1 m/s are discarded and the time adjusted. The result is a journey (trajectory) consisting of a sequence of coordinates where stops are invisible time-wise.

Examples of the procedure are shown in Figures 5.3 and 5.4. Both figures show the same trajectory. Figure 5.3 shows the trajectory without stop compression, where the stops are highlighted by drawn lines. Figure 5.4 shows the stop compressed speeds and coordinates of the trajectory. The stop compression is implemented using a *Speed* threshold value of 0.1 m/s.

5.2 Synchronising Input Space

Before GPs can be trained on the journey data, the journeys need to be synchronised in their input space. A journey can be seen as a sequence of (longitude, latitude, time)-tuples, where *time* is the timestamp of the bus when it is at coordinate (longitude, latitude). However, at identical values for the *time* parameter, different buses driving for the same bus line have progressed differently on the journey. For example, buses drive at different speeds and they may stay stopped for different periods of time at each bus stop. The *time* parameter needs thus to

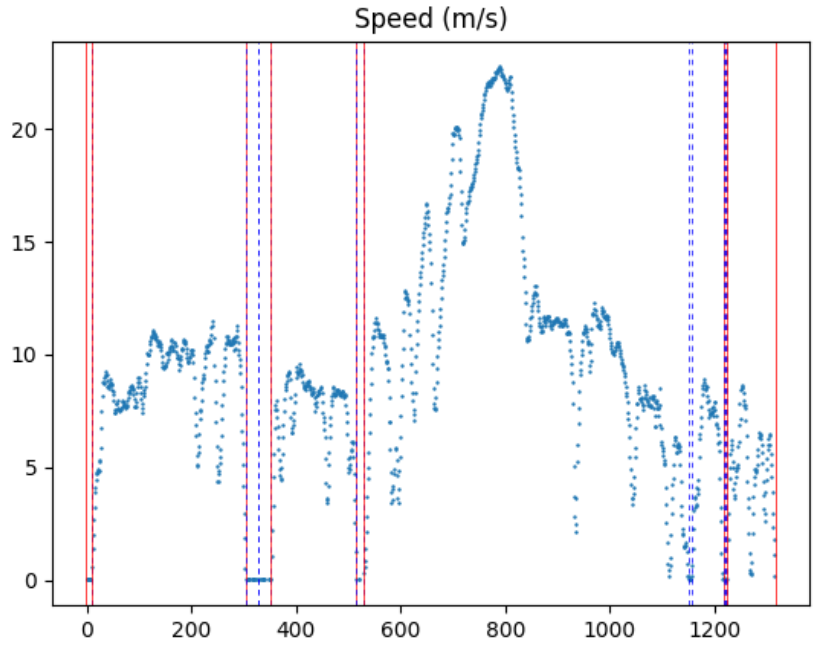


Figure 5.3: Example of a trajectory with uncompressed speeds. The red lines are `ArrivedEvent`s and `DepartedEvent`s. The dashed blue lines are `StartedEvent`s and `StoppedEvent`s. The red and blue lines usually coincide.

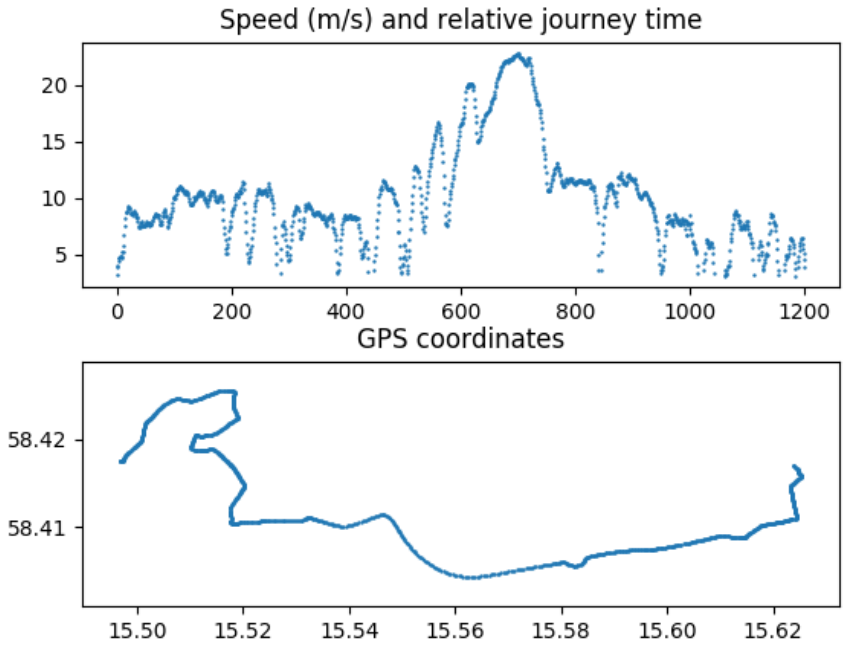


Figure 5.4: Example of a stop compressed trajectory. The *Speed* threshold is set to 0.1 m/s. The various speeds are shown in the top graph, while the trajectory is drawn in the bottom graph.

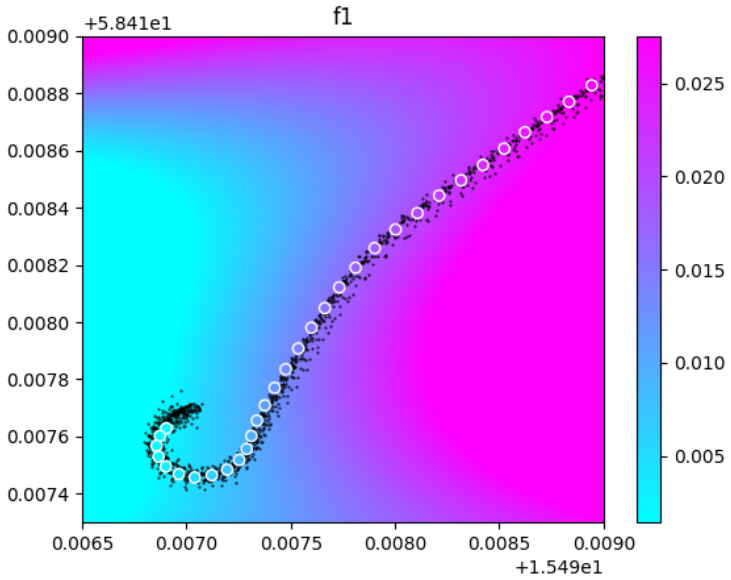


Figure 5.5: f_1 GP from the first approach. The coloured, outlined circles are training data points. The black data points are observed `ObservedPositionEvent`s from all trajectories. The regression field is visualised by the gradient background.

be synchronised, so that (longitude, latitude)-pairs can be compared. The synchronisation is achieved by introducing a global bus line GP f_1, given by

$$f_1 : (x, y) \longmapsto \tau, \tag{5.1}$$

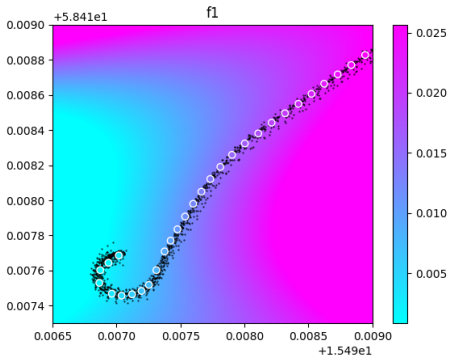
where (x, y) is the (longitude, latitude)-pair and τ the synchronised *time* parameter, i.e, the percentage of journey completed. Each bus line has its own f_1 GP, which is trained on a tailored trajectory for that bus line. The trajectory is tailored by pre-processing it with a stricter stop compression algorithm. Two approaches are tested and implemented:

1. ***Speed* value of 3**: Instead of discarding `ObservedPositionEvent`s with a *Speed* value lower than 0.1 m/s, the threshold is increased to 3 m/s. The reason behind this is to reduce the impact from stops and traffic on the trained model. The trajectory used for the training is arbitrarily chosen from a pool of trajectories occurring in the middle of the night, as during night-time the buses stop at fewer bus stops and the traffic is generally lighter.

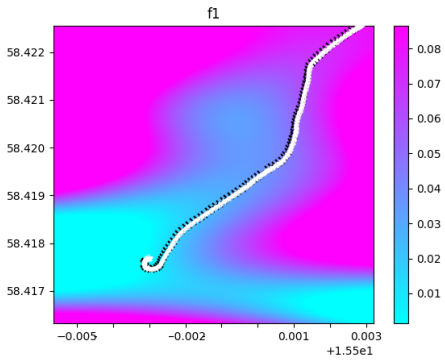
2. **Distance Filtering**: With the distance filtering approach, all `ObservedPositionEvent` types are included, regardless of their respective *Speed* values. Instead, the filtering is done by looking at the distance between each `ObservedPositionEvent`. The fifth decimal roughly corresponds to changes in metres [1]. The distance filter is set to remove a point with a distance smaller than 6×10^{-5} (around 4 metres) to the previous, accepted point.

The first approach achieved poor results, see Figure 5.5, as points in the beginning of the trajectory are not properly modelled by the f_1 GP, since most points had a *Speed* value less than 0.1 m/s. The coloured, outlined circles are data points used for training the GP. The black points are `ObservedPositionEvent`s from trajectories. The regression field is shown in the gradient background; the colour maps to a τ value. The second approach achieves better

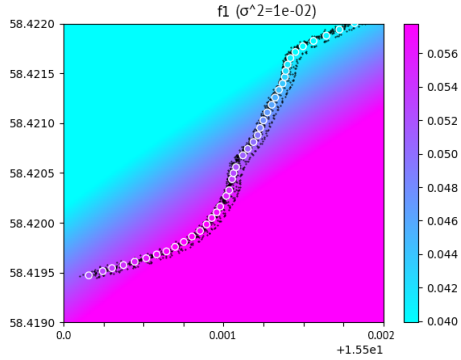
[1] World Geodetic System: http://epsg.io/7030-ellipsoid



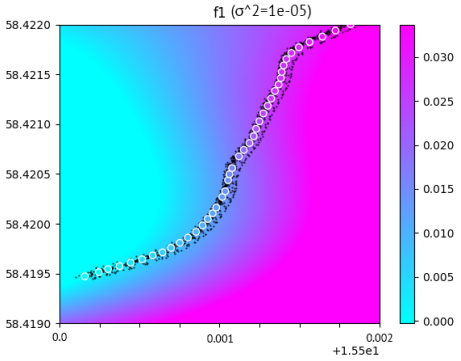
(a) f_1 GP trained with a distance filter of 6×10^{-5}, which corresponds to around 4 metres.



(b) Same f_1 GP as in 5.6a, but visualised on a bigger grid.



(c) The first segment has been removed from all trajectories. Observation noise σ_n^2 fixed to 10^{-2}.



(d) The first segment has been removed from all trajectories. Observation noise σ_n^2 fixed to 10^{-5}.

Figure 5.6: f_1 GPs from the second approach. The coloured circles are the training data points. The small black points are observed data points from other trajectories. The background shows how points are assigned to different τ values.

results. Figures 5.6a and 5.6b show the same model visualised on different grids. The model does not generalise well, as seen in the larger grid by the blue coloured area in the bottom right corner of 5.6b.

Figure 5.6c shows an attempt to improve the general performance of the f_1 GP. The distance filter is the same as in the model shown in 5.6a and 5.6b. However, the first segment has been removed from all trajectories, as the curvature created instabilities in the model. The effect of the curvature are similar to those which stem from trajectories with self-overlap. The variance is fixed to 10^{-2}, which corresponds to a standard deviation of around 6 kilometres in an attempt to prevent overfitting, as many points in the trajectory are close to each other. This causes the model to overestimate the variance of observations. Similar results are achieved if the trajectory is downsampled to only use every n data point. The fixed variance is too large for the model, which causes extremely poor regression.

Figure 5.6d locks the variance to 10^{-5}, which corresponds to a few metres in longitude and latitude, similar to the distance filtering. The specific GPS sensors used in the buses are unknown, but they are in this thesis assumed to have similar precision as GPS-enabled smartphones. The precision is typically around the range of ca. 5 metres [31].

GP models also typically assume zero mean for the input data, which in this thesis project is achieved by fitting a global re-scaling model to the input data. The same trajectory used for f_1 is used to fit this global re-scaling model. All trajectory coordinates are centralised and

normalised by the global re-scaling model. The f_1 GPs are trained with a RBF kernel and the hyperparameters are automatically optimised using the GPflow framework. Automatic Relevance Determination [6] is used with the RBF kernel to allow for different length-scales in different dimensions.

5.3 Segment Self-Overlapping Journeys

After the input space synchronisation, each trajectory if fitted with two GPs. Journeys are analysed in order to detect self-overlapping, which would cause problems for the GP models, as described in Section 3.3. Journeys with self-overlap are split at the overlapping points into two segments. The process recursively analyses each resulting segment and proceeds with splitting the segments as long as self-overlapping is present. If a journey has multiple segments, each segment is fitted with its own two GPs

$$f_2 : \tau \longmapsto x \tag{5.2}$$

$$f_3 : \tau \longmapsto y \tag{5.3}$$

τ is the relative time of the journey, denoting the progress the bus has made on its journey, x the longitude, and y the latitude. The τ values are from the f_1 GP, evaluated on the trajectory data.

Once f_{2_k} and f_{3_k} have been trained for each trajectory k, the GPs are evaluated on a grid of longitude and latitude points. The grid is fed to the f_1 GP in order to produce a grid of τ values. The τ values are given to the f_2 and f_3 models and the result is a grid where each point has a longitude and latitude mean and variance, respectively. The grid is then visualised alongside the observed `ObservedPositionEvent`s, in order to compare the variance variation. The f_2 and f_3 models have a locked data noise σ_n^2 of 10^{-3}, which corresponds to a standard deviation of around 1500 metres, separate from the data noise used in GP f_1.

5.4 Probability Models

The GPs f_2 and f_3 each describe the mean (μ_x and μ_y, respectively) and variance (σ_x^2 and σ_y^2, respectively) at each point in the fitted segment or journey. The probability for various coordinate pairs $p((x_\star, y_\star))$ is of interest in this thesis project, which can be calculated with two different approaches.

Mixture of Gaussian Process Model with Uniform Prior

The first approach is to see all of the trajectory models for the same bus line as a Mixture of Gaussian Process (MoGP) model. An observed $(x_\star, y_\star)$ coordinate is thus assumed to be generated from one of the GP models in the MoGP. The prior on the MoGP model is assumed to be uniform. The probability of $(x_\star, y_\star)$ for each trajectory model k is $p_k((x_\star, y_\star)) = \mathcal{N}((x_\star, y_\star); \mu_k, \Sigma_k)$, where

$$\mu_k = \begin{bmatrix} \mu_{k_x} \\ \mu_{k_y} \end{bmatrix} \tag{5.4}$$

$$\Sigma_k = \begin{bmatrix} \sigma_{k_x}^2 & 0 \\ 0 & \sigma_{k_y}^2 \end{bmatrix} \tag{5.5}$$

is the mean and covariance of the point evaluated at $(x_\star, y_\star)$. In the covariance matrix Σ_k, longitude and latitude is assumed to be independent. As the MoGP prior is uniform, the probability of $(x_\star, y_\star)$ is given by

$$p((x_\star, y_\star)) = \frac{1}{K} \sum_{k=1}^{K} p_k((x_\star, y_\star)), \tag{5.6}$$

where K is the number of trajectory models.

Unimodal Mixture of Gaussian Process Model by Combining

In order to make a unimodal approximation of the MoGP model, f_{2_k} and f_{3_k} are point-wise combined for each trajectory k. However, it is not an approximation with respect to assumptions made in this thesis project. The assumption is that the observed trajectories are drawn from a single unimodal distribution over functions with Gaussian noise with varying variance. The aggregation is done by using the Combining equations 2.9 and 2.10 from Section 2.2. The Combining approach is preferred over the Fusion approach in this particular problem, as the model is trying to capture the variation of GPS positions. It is thus beneficial to use an aggregation approach which tries to capture and aggregate the variance of each local trajectory model. They are applied as follows (with the notation changed to represent the notation used in this chapter):

$$\mu_x(x^*) = \frac{\sum_{k=1}^{K} M_k \mu_{k_x}(x^*)}{\sum_{k=1}^{K} M_k}, \tag{5.7}$$

$$\mu_y(y^*) = \frac{\sum_{k=1}^{K} M_k \mu_{k_y}(y^*)}{\sum_{k=1}^{K} M_k}, \tag{5.8}$$

$$\sigma_x^2(x^*) = \frac{\sum_{k=1}^{K} M_k \left(\sigma_{k_x}^2(x^*) + \mu_{k_x}^2(x^*)\right)}{\sum_{k=1}^{K} M_k} - \mu_x(x^*)^2, \tag{5.9}$$

$$\sigma_y^2(y^*) = \frac{\sum_{k=1}^{K} M_k \left(\sigma_{k_y}^2(y^*) + \mu_{k_y}^2(y^*)\right)}{\sum_{k=1}^{K} M_k} - \mu_y(y^*)^2, \tag{5.10}$$

where

- M_k is the number of trajectories used to train f_{2_k} and f_{3_k}. In this thesis project, each trajectory k is trained on its own f_{2_k} and $f_{3_k} \Rightarrow M_k = 1$ for all $k = 1, \ldots, K$ trajectories.
- x^* and y^* are index slices (points) to evaluate the MoGP model at.
- μ_{k_x} and μ_{k_y} is the predicted mean of the evaluated point at the index slice of x (longitude) and y (latitude), respectively.
- $\sigma_{k_x}^2$ and $\sigma_{k_y}^2$ is the predicted variance of the evaluated point at the index slice.

Equations 5.7-5.10 can thus be simplified to

$$\mu_x(x^*) = \frac{\sum_{k=1}^{K} \mu_{k_x}(x^*)}{K} \tag{5.11}$$

$$\mu_y(y^*) = \frac{\sum_{k=1}^{K} \mu_{k_y}(y^*)}{K} \tag{5.12}$$

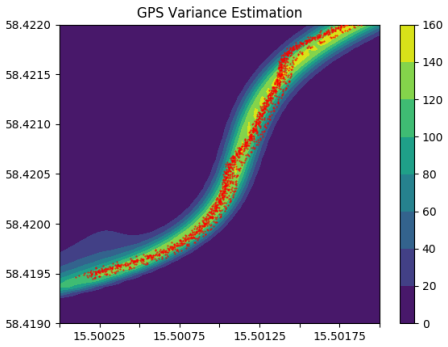
$$\sigma_x^2(x^*) = \frac{\sum_{k=1}^{K} \left(\sigma_{k_x}^2(x^*) + \mu_{k_x}^2(x^*)\right)}{K} - \mu_x(x^*)^2 \tag{5.13}$$

$$\sigma_y^2(y^*) = \frac{\sum_{k=1}^{K} \left(\sigma_{k_y}^2(y^*) + \mu_{k_y}^2(y^*)\right)}{K} - \mu_y(y^*)^2 \tag{5.14}$$

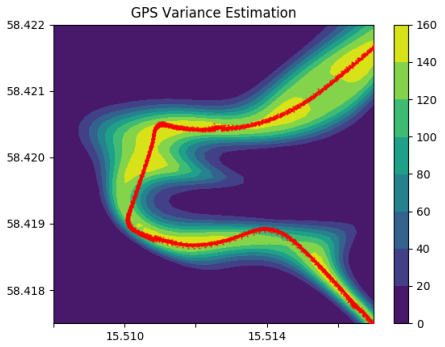
The equations 5.11-5.14 give the combined unimodal MoGP model at index slice (x^*, y^*)

$$\mu((x^*, y^*)) = \begin{bmatrix} \mu_x(x^*) \\ \mu_y(y^*) \end{bmatrix} \tag{5.15}$$

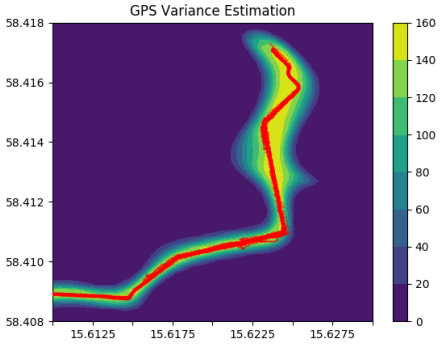
$$\Sigma((x^*, y^*)) = \begin{bmatrix} \sigma_x^2(x^*) & 0 \\ 0 & \sigma_y^2(y^*) \end{bmatrix} \tag{5.16}$$



(a) Segment 1 - Beginning of bus line.



(b) Segment 2 - Middle of the bus line.



(c) Segment 3 - End of the bus line.

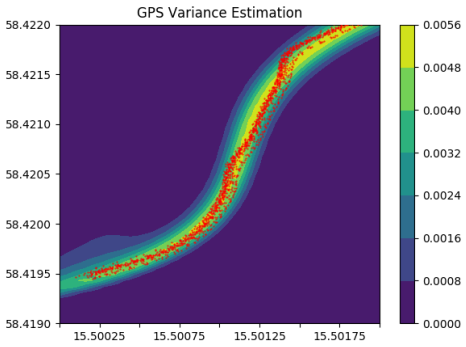
Figure 5.7: GPS variance variation estimation for the MoGP model. Shows the PDF of GPS positions for a given bus line. The model uses a fixed data noise of 10^{-3}. The red dots are `ObservedPositionEvents` from the data set. The Y-axis is the latitude coordinates and the X-axis is the longitude coordinates.

which gives the probability of an arbitrary point $(x_\star, y_\star)$: $p(x_\star, y_\star) = \mathcal{N}((x_\star, y_\star); \mu, \Sigma)$

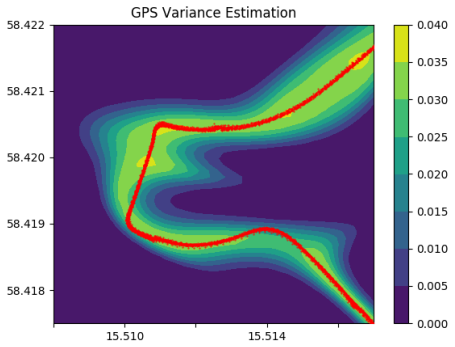
5.5 GPS Variance Estimation Results

The MoGP model is visualised by evaluating the model on a grid of longitude and latitude values. The results from the MoGP model with uniform prior is visualised in Figure 5.7 for three segments of a bus line. The grid is fed to the f_1 GP and thus converted into a grid of τ values. The τ values are given to the f_2 and f_3 GPs, which returns the predicted longitude and latitude mean and variance, respectively. The predicted means and variances are used to create Gaussian distributions for each longitude and latitude pair in the grid. The probability density function (PDF) is calculated for each Gaussian distribution and divided by the number of GPs in the MoGP model.

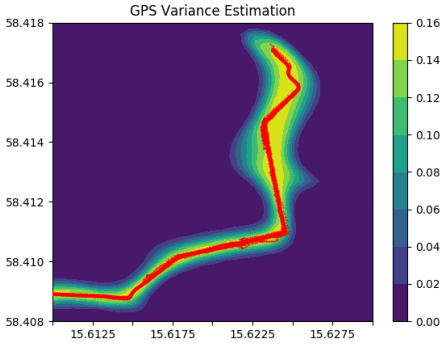
The unimodal MoGP model is visualised by evaluating the MoGP model on a grid of longitude and latitude values. The same approach as in the MoGP model with uniform prior is used. The f_2 and f_3 GPs are combined using the Combining formula. The PDF is calculated for the combined model and is integrated with the grid resolution in order to create a probability distribution. Figure 5.8 shows the GPS variation for three segments of a bus line. The results are similar to the MoGP model with uniform prior.



(a) Segment 1 - Beginning of bus line.



(b) Segment 2 - Middle of the bus line.



(c) Segment 3 - End of the bus line.

Figure 5.8: GPS variance variation estimation for the unimodal model. Shows the probability that a GPS position is observed for a given bus line. The model uses a fixed data noise of 10^{-3}. The red dots are `ObservedPositionEvent`s from the data set. The Y-axis is the latitude coordinates and the X-axis is the longitude coordinates.

5.6 Discussion

The ridges of the combined probability distributions presented in Figure 5.8 all follow the observed data points well. However, the probability of observing certain GPS positions are low, even at the ridge of the probability distribution, which is surrounded by the observed data points. This is probably due to a few things:

1. Though the f_1 GP is improved compared to the first approach, it is still not perfect and generalises poorly. It creates large areas of uncertainty, where similar values are mapped to the same τ value. This affects the training of the f_2 and f_3 GPs, as the data is dependent on the transformation (synchronisation) from f_1.

2. Individual longitude or latitude coordinates close to a cluster of observed positions achieve a probability of ca. 0.8-0.9. However, it is generally the case that if the probability of a longitude coordinate is high, then the probability of the latitude coordinate for that grid point is low. As the longitude and latitude coordinates are assumed independent, the probability for a grid point is $p(lat_{point}) * p(lng_{point})$, where one is close to 1 and the other one small (< 0.1). This is partially solved if the PDF is integrated on a larger interval, as more of the probability density is captured in one point.

The variance variation is larger at certain locations on the bus line. In the end segment shown in Figure 5.8c, this occurs in an environment surrounded by tall buildings. This might be a reason for the increased variation, though it is not adequately reflected in the observed training data. A more likely reason for the increased variance in the figure is due to a poor f_1, f_2, and f_3 models unable to handle the sharp trajectory turns. The f_1 GP might perform better if it is trained on segments of the trajectory, instead of trying to learn the full trajectory. The removal of the first segment from all trajectories is a necessary step in order to improve the generalisation of the f_1 GP. However, as discussed above, it might not have been enough in order to create a generally robust model.

Evaluation and Assumptions

In this thesis project, the variance variation is evaluated visually. Ideally, other metrics would be applied, such as determining locations where the variation is the smallest and the biggest. These metrics could be useful when looking at the impact of the surrounding environment on the GPS variance. The kernels used are evaluated both visually and by looking at the resulting hyperparameters after the optimisation step. If the likelihood of the data is optimised to be small (10^{-6}), it could be a result of overfitting to the linear trend of the data.

The evaluated data is assumed to have an inherent noise of 10^{-5}, which is the measured accuracy of GPS-enabled smartphones. The noise specified for the f_2 and f_3 GPs are fixed to 10^{-3}. They are tested with a data noise of 10^{-5}, as in the case of f_1, but this resulted in barely any visible variation. The variation shown in Figure 5.8 is thus dependent on the fixed noise of the observed data. This parameter needs further investigating, as the current choice results in perhaps larger variation than expected.

Model Optimisation

The models in this thesis project are optimised using the L-BFGS-B algorithm implemented in open-source software Scipy[2]. It would perhaps have been better to estimate the hyperparameters through Markov chain Monte Carlo (MCMC) using Hamiltonian Monte Carlo (HMC) sampling, implemented in the GPflow framework. The use of sampling to estimate hyperparameters, and further optimise them with the maximum a posteriori (MAP) of the sampled posterior as initial value would definitely be useful to investigate in future work.

Trajectories

Due to limited time and limited processing power, the number of trajectories used for the models is restricted to 37. The trajectories are all collected from the same day. It took, on average, eight hours to retrain all the models, which caused changes to be expensive time-wise. It is not always immediately clear if a change generalises well or not. For example, some configurations show extremely poor results on only a small number of trajectories, while the results are average on the others. The training time proved to be a huge drain on the effectivity of the feature extraction and model optimisation.

The aim of the thesis project is to use trajectories from multiple days in order to determine an eventual bias of the positions depending on day and time. This thesis project creates a foundation where this analysis is possible in a future work.

Only trajectories from buses are analysed in this thesis project. It would have been interesting to see how the data from trains differed and how their variance variation compared to the variation from the buses.

[2]https://docs.scipy.org/doc/scipy/reference/index.html

5.7 Future Work

There are a few aspects which could be improved in future work. The improvement of the f_1 GP would be a main priority, given more time to the thesis project. The f_1 GP is the backbone of the f_2 and f_3 models trained for the GPS variance variation estimation. The idea of splitting a trajectory into smaller segments and train one f_1 GP for each segment is sound.

Variation Over Multiple Days

Future improvements of the GPS variation estimation would be to train the models on data from different days, using the day and time of the day as a feature. This would allow for the analysis of how the GPS variation changes throughout a day and over a span of days. The use of a periodic kernel would perhaps also be useful, in order to capture the periodicity of satellites. Improvements made to the dataset could further benefit this analysis. For example, information about the number of satellites used, which satellites are available, and their respective positions, would give information about the potential periodicity of the variation.

Figure 5.9: Real-world example of multiple buses driving on a road not correctly modelled by Google Maps. For example, at "Sjukhusvägen" the buses are driving through a building according to Google Maps. In this particular scenario the area has received major road re-structuring and renovation. The Google Maps representation is thus likely outdated.

Road Segments

Another area not investigated in this thesis project is the use of the Google Maps Roads API[3]. The API maps GPS coordinates to the closest geographical road segment. The road segments could, for example, in future work be used as a "truth" when determining the variation of GPS signals. However, manual inspection of journeys done in the pre-processing step highlights

[3]https://developers.google.com/maps/documentation/roads/intro

scenarios with discrepancies between roads in Google Maps and actual roads the buses travelled. The Google Maps roads are in these scenarios usually incorrectly drawn or has missing roads. Figure 5.9 shows such a real-world example. The discrepancies would cause problems if used naïvely together with the Google Maps Roads API. These journeys would have to be detected and either handled or ignored. The detection could be done by looking at the distances to the closest Google Maps Roads segments. The distances would be roughly similar for all the journeys, as in Figure 5.9. Since the distances would be roughly similar they could be averaged and interpreted as a bias to the roads. Applying the bias-interpretation methodology would result in a more generalised model using the Google Maps Roads API. Ignoring the journeys with discrepancies could also be seen as feasible, given that the occurrence is rare. Before using the simpler approach of ignoring journeys with discrepancies, a deeper analysis needs to be conducted.

6 Arrival Time Prediction

Trajectory forecasting is the process of predetermining when a bus arrives at a particular bus stop. The existing arrival time prediction of Östgötatrafiken AB is based purely on long-time historical data. It provides a single timestamp when it predicts the bus to arrive at a certain bus stop. The prediction is the most likely arrival time for the bus.

This chapter uses a methodology similar to the previous Chapter 5. The aim is to provide a probability distribution of arrival times for the next bus stops. This thesis project proposes a Trajectory Forecasting model based on multiple GPs. It provides time-to-arrival probability distributions for all the remaining bus stops in the journey. The probability distribution allows for queries such as:

- *What is the most likely time of arrival at bus stop p from the previous bus stop?*
- *What is the probability of the bus arriving at the earliest/latest possible time?*

6.1 Initial Methodology

The first steps of the proposed approach is similar to the steps covered in the Sections 5.1-5.3. Below is a brief description of the steps already covered, with alterations highlighted:

1. *Stop Model and Compression:* Stops during the trajectory are modelled as bus stops and red lights, i.e., common stops during a bus journey. GPS positions of stopped buses need to be compressed in order to support more robust GPs [18]. The result of this step is a collection of trajectories for each bus line. The trajectories include common stops and compressed GPS positions during stops. As in Section 5.1, a *Speed* threshold value of 0.1 m/s is used for the trajectory forecasting. However, while stops due to traffic and red lights are compressed by removing positions occurring during a stop, they are not compressed time-wise. This means that the position before the stop and after the stop have potentially wildly different timestamps. The reason behind this is to indirectly model traffic and red lights as latent variables. The effects of traffic and red lights are neither explicitly modelled nor discarded.

2. *Synchronising Input Space:* The input space is synchronised using by training a global GP model f_1, given by eq. 6.1 in Section 5.2. The model is thus

$$f_1 : (x, y) \longmapsto \tau, \tag{6.1}$$

and is implemented using the GPflow [21] framework. The training trajectory is distance filtered with a threshold of 6×10^{-5}, which roughly corresponds to a few metres. The GP is implemented using a RBF kernel with Automatic Relevance Determination (ARD) for the length-scale parameter. The data noise variance parameter σ_n^2 is fixed at 10^{-5}. The f_1 GP is thus the same as the one used in Chapter 5.

3. *Segment Self-Overlapping Journeys:* Self-overlapping trajectories are segmented until no self-overlapping is occurring for the trajectory. The segmentation is required in order to avoid problems with instability, as mention in Section 3.3, and the problem of overlapped positions in a model being trained to predict completely different arrival times.

   Each segment can potentially have a small model overlap, which means that neighbouring segments share a few data points. This improves the robustness of the model, as it retains the desired shape close to the borders [18]. The overlap used in this thesis project is set to five `ObservedPositionEvent`s, which is roughly 6.5% of an average-length segment (around 77 `ObservedPositionEvent`s).

4. *Remove First Segment:* As briefly discussed in Chapter 5, the performance of the f_1 GP is poor when training on the start of the trajectory. Therefore, the first segment is removed from all trajectories.

Once these three steps are executed, the approach diverges compared to the approach described in Chapter 5.

6.2 Segment Bus Stops

The trajectory is segmented at bus stops, which means that the `ObservedPositionEvent`s occurring between two bus stops are segmented into one *trajectory segment*. For example, a bus line $A \rightarrow B \rightarrow C$ is segmented into two segments

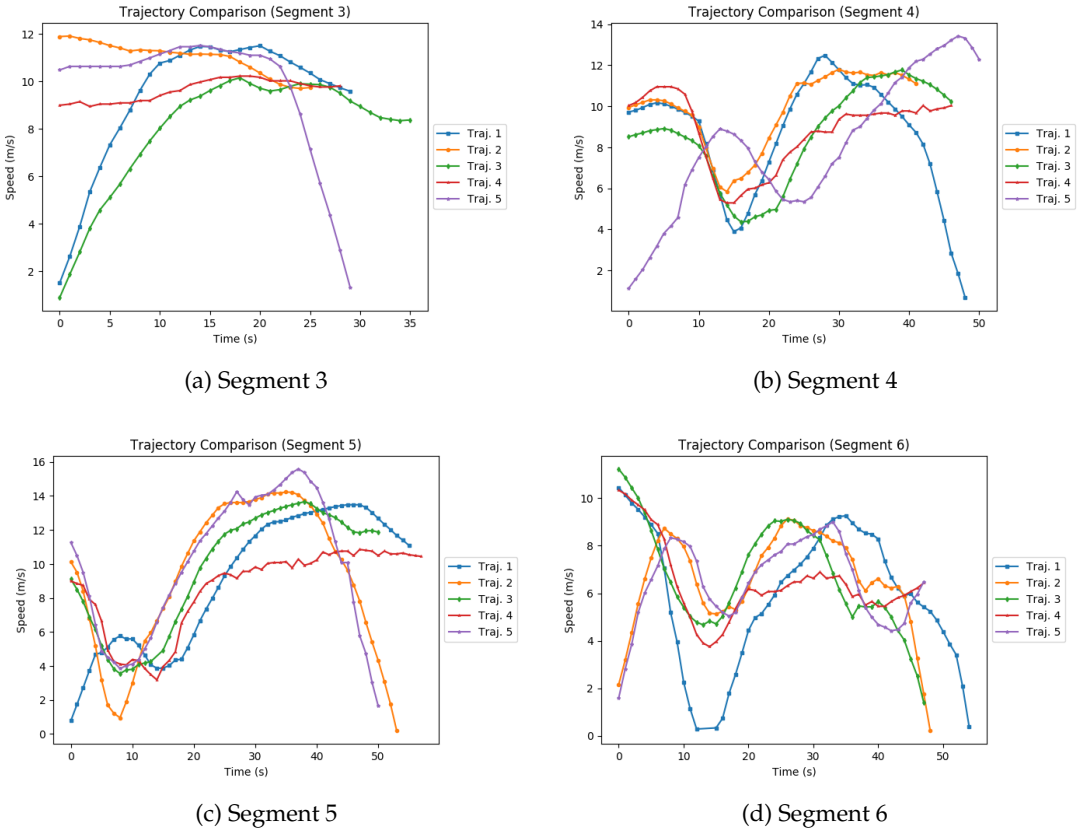
1. $A \rightarrow B$
2. $B \rightarrow C$

The positions occurring during a bus stops are already compressed by Step 1 in Section 6.1. While creating the segments, the arrival time for each segment is extracted from the trajectory. The arrival time is the timestamp of the event where the system determines that the bus has reached the bus stop, e.g., the date when the bus reaches

1. B, after departing from bus stop A
2. C, after departing from bus stop B

Figure 6.1 shows an example of the bus stop segmentation for an arbitrary trajectory. Four sequential segments are visualised, with the relative time (in seconds) for each segment on the X-axis and the speed (m/s) of `ObservedPositionEvent`s on the Y-axis.

The result of the bus stop segmentation step is a collection of segments and segment arrival times for each trajectory.



(a) Segment 3

(b) Segment 4

(c) Segment 5

(d) Segment 6

Figure 6.1: Example of four trajectory segments with zero segment overlap. The X-axis is the relative time (in seconds) of the trajectory segment, the Y-axis is the speed of each `ObservedPositionEvent` point. Five trajectories are visualised for each segment.

6.3 Trajectory Segment Forecasting Models

The collection of segments and arrival times are used to create features for the arrival time prediction model. Two different GP models are created and are given by

$$f_A : \tau \mapsto t_s \tag{6.2}$$
$$f_B : (\tau, v) \mapsto t_s \tag{6.3}$$

- τ is the trajectory progress for the bus. τ is calculated by using GP f_1 from eq. 6.1.
- t_s is the predicted arrival time to the bus stop at the end of segment s.
- v is the speed of the bus at τ.

Model Comparison f_A and f_B

A comparison between GP models f_A and f_B is executed in order to find the best model for arrival time prediction. In the comparison, both GP models use a Matérn kernel with $\nu = 3/2$.

Table 6.1 shows the evaluation results of GP model f_B. GP model f_B is used to predict the arrival time for a particular bus driving in segment S_0. GP model $f_{B_0}^{(1)}$ is trained with the same trajectory as both GP models are evaluated with. GP model $f_{B_0}^{(2)}$ is trained on a different trajectory. A comparison between the two trajectories used for the training of $f_{B_0}^{(1)}$ and $f_{B_0}^{(2)}$, respectively, is shown in Figure 6.2. Trajectory 1 is used for the training of GP model $f_{B_0}^{(1)}$,

Table 6.1: Arrival time prediction with the f_B GP model. The GP model predicts the arrival time for a bus driving in segment S_0. This table contains only a part of segment S_0 . Two GP models trained on segment S_0 are used for the prediction: $f_{B_0}^{(1)}$ and $f_{B_0}^{(2)}$. The right column contains the true arrival time (in seconds remaining) for the bus.

Prediction $f_{B_0}^{(1)}(S_0)$	Prediction $f_{B_0}^{(2)}(S_0)$	Truth S_0
48.00032032	-176.47767445	48.0
46.99899913	-207.23221327	47.0
46.00140604	-233.29939699	46.0
44.99743229	-263.30660917	45.0
44.00544805	-291.29058078	44.0
42.97906806	-326.62038041	43.0
42.0257244	-339.44856005	42.0
40.97131472	-358.31354569	41.0
40.15149734	-374.08468248	40.0
39.18597533	-377.21521534	39.0
38.1260134	-385.63015598	38.0
36.86945791	-383.52959391	37.0
35.6697429	-378.47844965	36.0
35.02909338	-382.12072838	35.0
33.72776512	-408.53070845	34.0
33.26752095	-410.33279089	33.0
31.99396506	-441.67817682	32.0

Table 6.2: Arrival time prediction with the f_A GP model. The GP model predicts the arrival time for a bus driving in segment S_0. This table contains only a part of segment S_0. Two GP models trained on segment S_0 are used for the prediction: $f_{A_0}^{(1)}$ and $f_{A_0}^{(2)}$. The right column contains the true arrival time (in seconds remaining) for the bus.

Prediction $f_{A_0}^{(1)}(S_0)$	Prediction $f_{A_0}^{(2)}(S_0)$	Truth S_0
27.95503975	65.03012452	48.0
27.37264309	65.60988244	47.0
26.79024643	66.27802379	46.0
26.20784977	66.57825337	45.0
25.6254531	66.56967009	44.0
25.04305644	66.45871024	43.0
24.46065978	66.43164454	42.0
23.87826312	66.55277654	41.0
23.29586646	66.51186703	40.0
22.7134698	64.80122062	39.0
22.13107314	60.54143875	38.0
21.54867647	56.10481638	37.0
20.96627981	52.76073634	36.0
20.38388315	50.551963	35.0
19.80148649	49.37658565	34.0
19.21908983	48.68000806	33.0
18.63669317	47.65352869	32.0

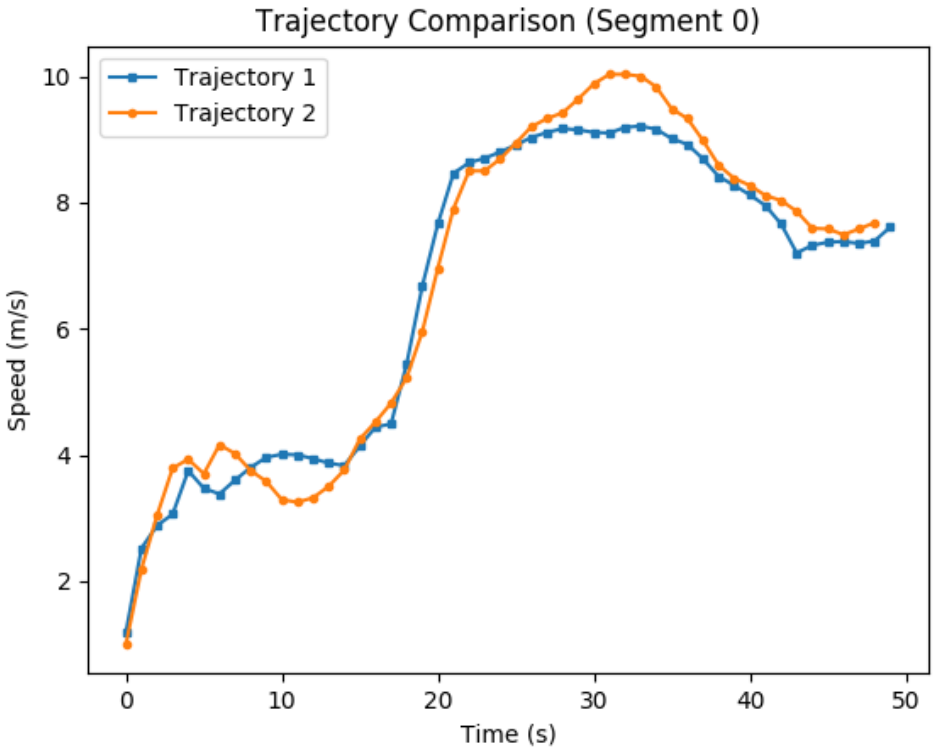


Figure 6.2: Trajectory comparison for segment S_0. Trajectory 1 is used to train $f_{B_0}^{(1)}$. Trajectory 2 is used to train $f_{B_0}^{(2)}$. Trajectory 1 is also used to evaluate both $f_{B_0}^{(1)}$ and $f_{B_0}^{(2)}$.

while trajectory 2 is used for the training of $f_{B_0}^{(2)}$. Trajectory 1 is used for the evaluation of both models. The results from the GP model $f_{B_0}^{(0)}$ does not come with any surprises, as the model is trained on the same data used to evaluate it. However, GP model $f_{B_0}^{(2)}$ performs extremely poorly on unseen data; which is an indication that the model does not generalise well. On the other hand, the two trajectories do not differ by much, as shown in Figure 6.2. This suggests that the model, kernel and/or kernel parameters are chosen poorly for the features used in f_B.

Table 6.2 show the evaluation results of GP model f_A. The same approach is used in the evaluation:

- Both models are evaluated on the same data
- GP model $f_{A_0}^{(1)}$ is trained on the same data used for the evaluation.
- GP model $f_{A_0}^{(2)}$ is trained on different data, but for the same segment S_0.
- The right column contains the true arrival times for the evaluation data.

It is surprising that the $f_{A_0}^{(1)}$ performs poorly when evaluated on training data. However, as both models perform moderately well on both testing and training data, the f_A model is the focus for the remainder of this thesis project. Different kernels are tested and evaluated for the f_A, together with varying the hyperparameters. The follow kernels are tested:

Table 6.3: Metric scores for f_A GP models. The GP models predict the arrival time for a bus driving in segment S_0. The metrics are calculated by comparing the predicted arrival time to the true arrival time. Two GP models are trained for each choice of kernel. The model $f_{A_0}^{(1)}$ is trained on the same data as it is evaluated by. Model $f_{A_0}^{(2)}$ is evaluated on the same data as $f_{A_0}^{(1)}$, but is trained on different data.

Kernel	RMSE $f_{A_0}^{(1)}$	RMSE $f_{A_0}^{(2)}$	MAE $f_{A_0}^{(1)}$	MAE $f_{A_0}^{(2)}$
Linear	24.85	25.26	20.56	21.49
White	26.80	26.80	22.38	22.38
RBF	**1.56**	**13.53**	**0.84**	**11.69**
RBF+Linear	**1.56**	13.60	**0.84**	11.84
Matern32	11.19	13.56	9.34	11.75

1. Linear
2. White
3. RBF
4. RBF+Linear
5. Matern32

The different f_A models are compared by calculating the following two metrics for each model:

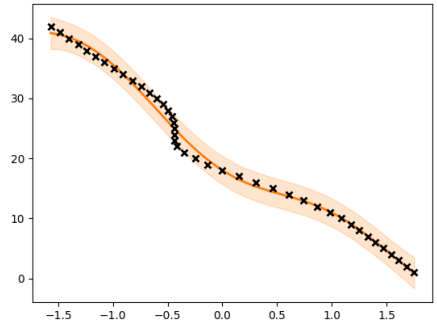
1. Root Mean Square Error (RMSE)
2. Mean Absolute Error (MAE)

The evaluation is similar to the evaluation between GP models f_B and f_A.

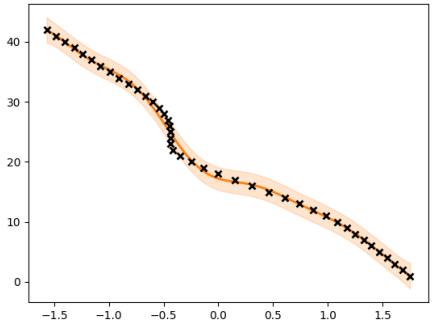
The metric scores and the visual representation of the different GPs are taken into consider when choosing the best, appropriate model. The best setting is used for the remainder of the thesis project. GPflow also supports a wide range of optimisation algorithms. This thesis project uses the standard L-BFGS-B algorithm.

Model Evaluation f_A

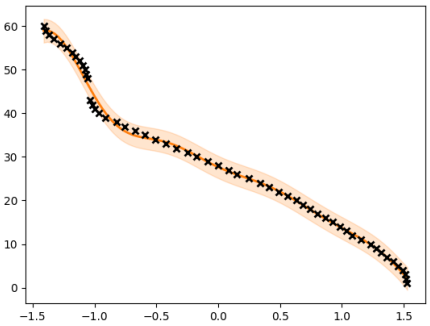
The results of the f_A GP model evaluation are presented in Table 6.3. The RBF kernel and the combined RBF+Linear kernel achieved the best scores. While GPflow optimises the hyperparameters of the model, the optimisation occasionally gets stuck in a local minima. The choice of initial kernel parameters is important, as a poor local minima in the optimisation function can result in poor robustness of a GP model. While the kernels work equally well in the general case, there are segments where neither work, depending on the trajectory and initialisation, which is expected. The problem with poor performance can be solved by, for example, changing the hyperparameter initialisation or varying it by using random restarts. Figure 6.3 shows a comparison between the two kernels for two different segments, both with optimised initial parameters, which results in a better model comparison. Both kernels produce similar models in the optimal case, with few differences. Figure 6.3b seems slightly better than the RBF kernel counterpart in Figure 6.3a, as the RBF+Linear kernel captures the linearity better in the start of the segment. Manual analysis of the different segments show that most segments follow a linear trend with occasional curvature. This suggests that a RBF+Linear kernel can be more suitable in the general case.



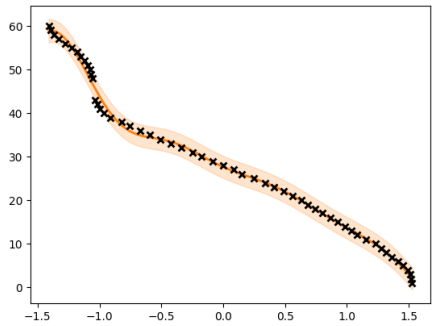
(a) Model of trajectory 10, segment 7, using the RBF kernel.



(b) Model of trajectory 10, segment 7, using the RBF+Linear kernel.



(c) Model of trajectory 22, segment 12, using the RBF kernel.



(d) Model of trajectory 22, segment 12, using the RBF+Linear kernel.

Figure 6.3: Visualisation of f_A models with different kernels. Each figure contains the predictive mean and variance of the model, alongside the training data points. The X-axis is the τ values of the trajectory segment and the Y-axis is the arrival time to the end of the segment, in seconds.

Final Model

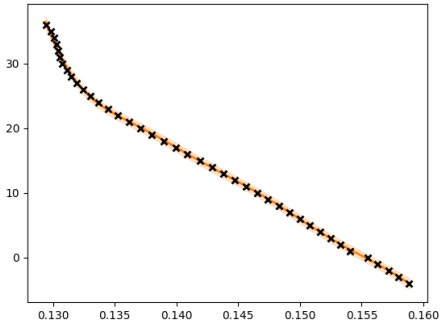
Figure 6.4 shows the final RBF+Linear model with the following initial hyperparameters:

1. $l_{RBF} = 0.1$, where l_{RBF} is the length-scale of the RBF kernel in the combined RBF+Linear model.
2. $\sigma^2_{Linear} = 500$, where σ^2_{Linear} is a prior on the variance of the Linear kernel in the combined RBF+Linear model.

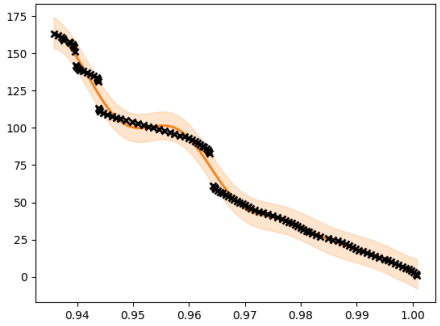
The arrival time prediction model is evaluated with the Leave-One-Out (LOO) approach:

1. One trajectory from the pool of 38 trajectories used in the thesis is randomly chosen as a test trajectory.
2. The test trajectory is used for evaluation on all $K - 1$ models trained on different data. In this case, the test trajectory is used to evaluate 37 models.
3. The first step is repeated, but with another trajectory.

The evaluation is done by predicting the arrival time at each `ObservedPositionEvent` p_i in the test trajectory. Each model gives a predictive mean and variance for the arrival time.



(a) Model of trajectory 16, segment 4, using the RBF+Linear kernel.



(b) Model of trajectory 26, segment 18, using the RBF+Linear kernel.

Figure 6.4: Visualisation of segments with the RBF+Linear kernel. Each figure contains the predictive mean and variance of the model, alongside the training data points. The X-axis is the τ values of the trajectory segment and the Y-axis is the arrival time to the end of the segment, in seconds.

The prediction is thus a mixture of Gaussian distributions. The mixture is in this thesis project assumed to have a uniform prior. The predicted arrival time is chosen to be the mean of the mixture means.

The evaluation metrics for the model are presented in Table 6.4. The RMSE and MAE is calculated for each bus stop segment and the table presents the mean and standard error of the metrics for all segments in a trajectory. The score is calculated by taking the mean and standard error RMSE and MAE from all segments. The standard error is given by

$$SE = \frac{\sigma}{\sqrt{n}}, \tag{6.4}$$

where σ is the standard deviation calculated using Bessel's correction

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{6.5}$$

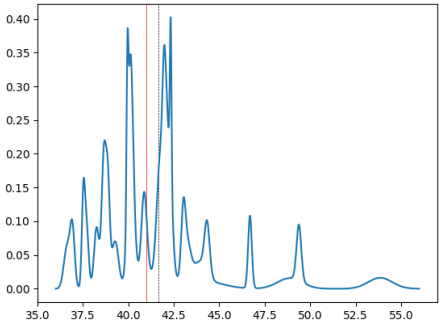
6.4 Results

The arrival times for an arbitrary `ObservedPositionEvent` are shown in Figure 6.5. The dashed line is the predicted arrival time, the red line is the true arrival time. The X-axis is the arrival time to the next bus stop, in seconds. The Y-axis is the PDF of the Gaussian mixture.

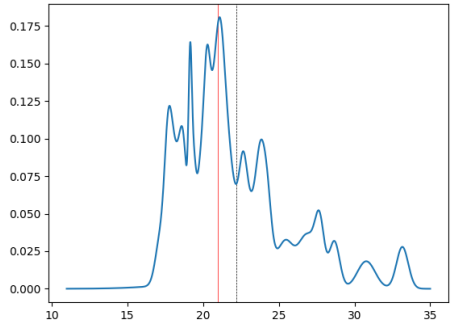
Table 6.4 shows the evaluation done at a segment level. A segment is the route between two bus stops. Each segment only predicts the arrival time to the end of that segment. The RMSE and MAE is calculated for each segment, for each respective trajectory. In these results the bus line has 18 segments. The RMSE and MAE scores presented in Table 6.4 is thus the mean RMSE and mean MAE of all 18 segments for each respective trajectory. The standard error is calculated using Bessel's correction.

Table 6.5 presents the RMSE and MAE scores on a trajectory level. The prediction is still done at a bus stop segment level. However, instead of calculating the mean RMSE and MAE for all segments, the RMSE and MAE is calculated for the whole trajectory.

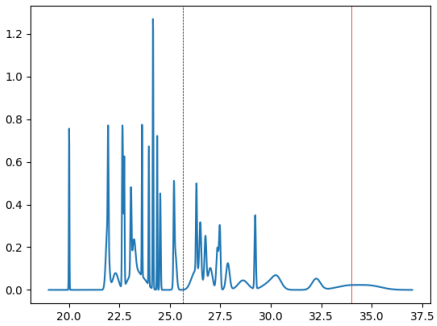
Figure 6.6 shows how the absolute errors of the predicted arrival times decrease over time for arbitrary segments.



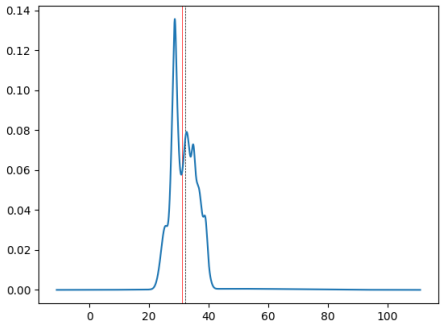
(a) Predicted arrival time for trajectory 32 at segment 4. The black dotted line is the mean predicted arrival time of the mixture. The red line is the true arrival time at that point in the segment.



(b) Predicted arrival time for trajectory 13 at segment 10. The black dotted line is the mean predicted arrival time of the mixture. The red line is the true arrival time at that point in the segment.



(c) Predicted arrival time for trajectory 17 at segment 3. The black dotted line is the mean predicted arrival time of the mixture. The red line is the true arrival time at that point in the segment.



(d) Predicted arrival time for trajectory 13 at segment 7. The black dotted line is the mean predicted arrival time of the mixture. The red line is the true arrival time at that point in the segment.

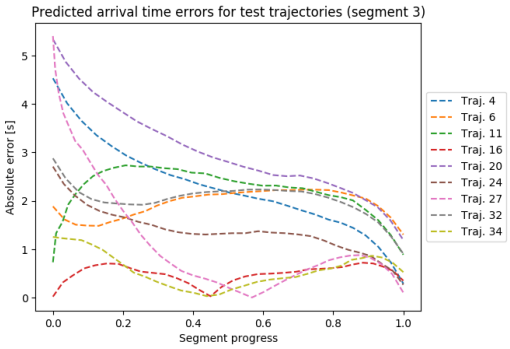
Figure 6.5: Predicted arrival times. Each figure contains the predictive and true arrival time for the test trajectory, evaluating on 37 models trained on different trajectories. The dashed line is the predicted arrival time from the mixture of Gaussian distributions. The red line is the true arrival time. The X-axis is the arrival time to the end of the segment, in seconds. The Y-axis is the PDF of the Gaussian mixture.

Table 6.4: Evaluation of arrival time prediction model. The RMSE and MAE is calculated for each segment in each trajectory. A segment is the route between two bus stops. The average RMSE and MAE score is shown, together with the standard error using Bessel's correction. This is the average for all segments in a trajectory.

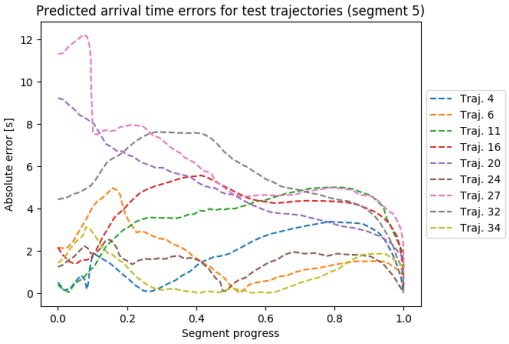
Test Trajectory #	RMSE [s]	MAE [s]
13	6.14 ± 1.9	5.18 ± 1.56
17	4.71 ± 0.64	4.18 ± 0.57
18	7.22 ± 2.03	6.55 ± 2.0
22	3.86 ± 0.88	3.35 ± 0.78
32	7.47 ± 2.09	6.54 ± 1.81
35	5.68 ± 1.17	4.83 ± 1.06

Table 6.5: Evaluation of arrival time prediction model. The RMSE and MAE is calculated for each trajectory. The arrival time predictions are still for segments in the trajectory. However, instead of calculating the scores for each segment, the scores are predicted for all arrival time predictions done during the trajectory.

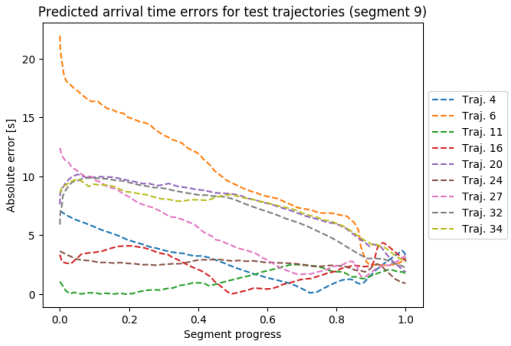
Test Trajectory #	RMSE [s]	MAE [s]
13	16.94	10.8
17	5.93	4.84
18	19.52	12.79
22	5.92	4.0
32	19.04	12.44
35	12.4	8.5



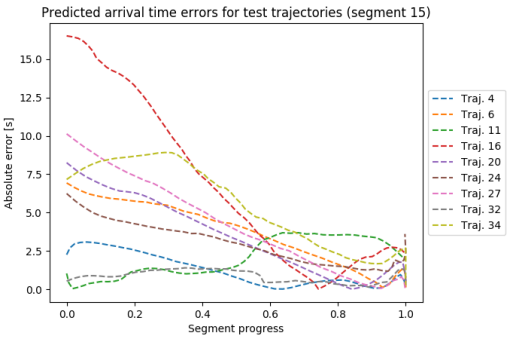
(a) Absolute error for segment 3.



(b) Absolute error for segment 5.



(c) Absolute error for segment 9.



(d) Absolute error for segment 15.

Figure 6.6: Absolute error of predicted arrival times for test trajectories. Each figure contains the absolute error of the predicted arrival time for each test trajectory in arbitrary segments. The X-axis is the segment progress from start of the segment to the end of the segment. The start and end of the segment are bus stops, as the segmentation is done on a bus stop level.

6.5 Discussion

The different evaluation methods highlight different aspects of the implemented model. Table 6.4 presents the RMSE and MAE scores at a segment level. The segments are the route between two bus stops. A trajectory is thus a collection of segments. The scores presented in the table give an average measurement of the scores for all segments in a trajectory. For certain test trajectories (17 and 22), all segments have a similar score, which causes the standard error to be lower. Other test trajectories have segments with varying scores, resulting in higher standard error. This is also reflected in Table 6.5, which presents the RMSE and MAE scores at a trajectory level. Individual segments are not used to calculate the scores. The

evaluation measures are thus an indication on how well, in general, the model predicts the arrival times for the tested trajectories. The tested trajectories with higher RMSE and MAE scores in Table 6.5 also have higher standard errors in Table 6.4.

The use of RMSE and MAE to evaluate the performance of the arrival time prediction model does not cover all aspects of the model. When calculating the overall RMSE and MAE scores for the whole trajectory, or for each segment respectively, they average errors occurring over time. Figure 6.6 shows how the absolute error decreases, in general, over time for each segment. The model achieves better arrival time prediction as the buses travel the segment. The mean scores reflect poorly on the improvement over time, as the mean scores are heavily affected by the first few arrival time predictions done for each segment. The poor scores in the beginning of each segment can be indicative of poor structure for the GP models.

The arrival time distribution created by the system has numerous benefits compared to a single, most probable, arrival time. For example, the most likely arrival time can be calculated from the distribution either as a mixture mean, as in the naïve approach, or by using various methods [32, 33, 34] Furthermore, the probabilities of early arrival times and late arrival times can be calculated by integrating over the combined PDF from the model.

Arrival Time Prediction Aggregation

The aggregation of predicted arrival times used in this thesis project is too simple. It is heavily affected by the number of models used and captures only the average of the predicted arrival times. The assumption that the mixture of Gaussian distributions have a uniform prior is too naïve to give one correct prediction. This is obvious in the results, as the quality of the prediction wildly vary depending on test trajectory. The reason behind this is that while models do capture different trajectory behaviours for a single segment, e.g., heavy traffic compared with no traffic, the final prediction is only based upon the average of all models. If the typical model is traffic-free, then the predicted time is skewed towards a traffic-free arrival time. However, if a few trained models capture extreme traffic, then those predictions also affect the mean of all predicted arrival times. The use of a mixture with an assumed uniform prior is thus too simple to result in good predictions.

A better approach is to calculate the *data log likelihood* of a set of observed data points, given a GP model. The data log likelihood can then be used as a weight for each model. The larger the weight, the bigger the likelihood that the observed data is from a particular GP model. In the example above, where the majority of models are traffic-free and a few model heavy traffic, a test trajectory with heavy traffic increases the weights of the predictions from the few models and reduces the weight of the predictions from the majority of the models. The result is a better prediction of the arrival time. This method can be used either on each observed data point, in order to see which model best describes that point, or on a trace of observer data points. If a trace of observed data points are used instead, the weights would, over time, gravitate towards the models which best explain the observed data. For example, in the predicted arrival time distribution shown in 6.5c, the naïve approach performs poorly compared to the other visualised results. If the weighted approach is used instead, the idea would be that the modes closer to the red line would receive a larger weight than the majority of modes around the 20-27.5 second interval.

Kernels

Due to time limitations in the thesis project it is necessary to narrow down the model search space as development continued. The choice of kernel is an important one. The performance of the RBF kernel is similar to the RBF+Linear kernel. The hyperparameter initialisation might have caused the RBF+Linear kernel to perform slightly better. The linear trend in the data can be removed by applying a de-trending method, e.g., differencing or model fitting.

This pre-processing step would have been beneficial for the RBF kernel, which might have caused better performance overall.

The hyperparameters of the kernels can also be optimised using HMC sampling, instead of the L-BFGS-B algorithm. This could have avoided the local minima problems with the hyperparameters. It is a non-trivial problem to find a good initialisation for the hyperparameters, as they are used for over 700 GP models. Any one of those GPs might get stuck in local minima, which would require changes to the hyperparameter initialisation. The arrival time models took, on average, seven hours to train.

In hindsight, the use of ARD should have been investigated for the f_B model, where the *Speed* is a part of the feature. Using different kernels for different dimensions is also interesting to experiment with, as the model might capture new patterns.

Common Stops

Stop compression is the process of compressing common stops. This can be done at different levels, depending on the definition of a common stop. In this thesis project only bus stops are defined as common stops during the segmentation process, due to time limitations. Only bus stops are considered as common stops since this is the simplest approach possible (beyond no segmentation at all). The approach results in a probability distribution of arrival time predictions for the next bus stop. With the improved arrival time aggregation, this approach also models red lights and traffic as latent variables. The simplicity of the model is maintained, which makes the model easy to understand. Unfortunately the complexity is increased in the arrival time aggregation.

Other possible definitions of common stops are:

1. *Bus Stops and Red Lights:* Modelling common stops using both bus stops and red lights would probably be the optimal approach. Red lights are static phenomenons which either causes a change to the expected arrival time (if red light) or not (green light). However, deciding whether or not a stop occurred due to red lights or traffic is not trivial with the data available in the dataset. Stops in all trajectories for the same bus line could be compared and a common ground could be established, but what would the comparison be? For example, if the same stop occurs in 90% of all trajectories and it is not a bus stop, then it is most likely a red light, but how often does this occur? Maybe a more reasonable approach would be to combine the data with information from Google Maps Roads API or OpenStreetMap[1], as they contain information about crossings with red lights. A crossing would then be more accurately labelled as having a red light or not, which could be used as input by the system. Nevertheless, traffic would still be a latent variable not modelled by the extended approach.

2. *All Stops:* This extreme approach would model all stops in each trajectory as a common stop. The approach would create a larger number of segments compared to the previous, more restrictive, approaches. A trajectory suffering from heavy traffic would create multiple segments over a potentially short distance. The complexity of the model would thus explode as the number of segments grow. However, with a large dataset this approach could be effective to model the effects of traffic, but experiments are needed in order to determine the requirement on the dataset size.

Red Lights

If the second approach is applied to the arrival time prediction model, then valuable information could be gained by analysing the crossing with and without red lights. Dedicated red lights exist for public transportation vehicles, which instantly turn green once a bus is arriving at the crossing. By modelling crossings, the usefulness of such dedicated crossings could

[1] https://www.openstreetmap.org/

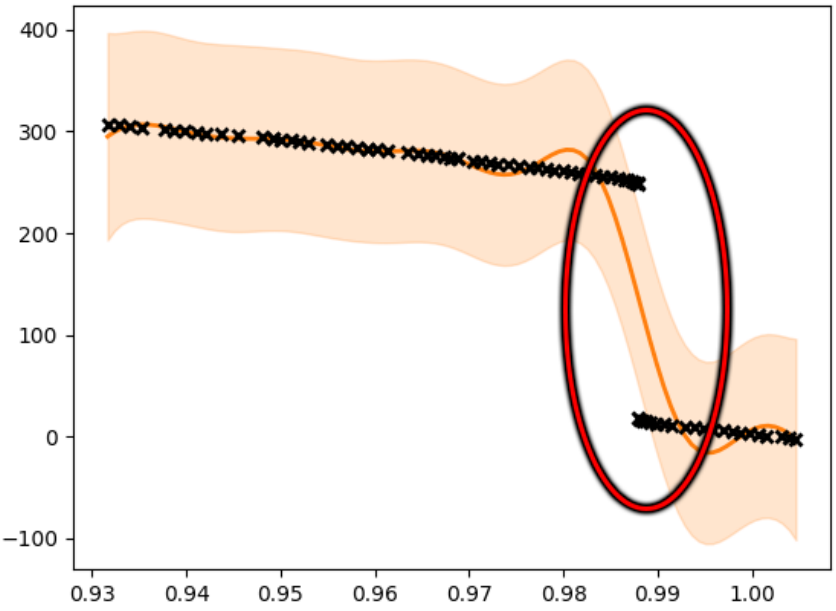


Figure 6.7: Example of poor bus stop detection for a bus stop with multiple modes. The Y-axis is the predicted arrival time. The X-axis is the segment progress. The black points are the true arrival times at each observed data point. The orange region is the predictive mean and variance of the trajectory forecasting model. The bus arrives to a platform dedicated to dropping off passengers. The bus stops at this platform for around 250 seconds. This platform is not registered as a part of the bus stop by the system, which causes a large offset in the predicted arrival times, as indicated by the red circle. The system registers the bus at the bus stop once the bus drives past the main mode (the main platform) of the bus stop.

be investigated. For example, crossings with abnormally long waiting times could be detected and dedicated traffic lights could be either tested or simulated for these crossings. This information could prove to be a major benefit to the planning of roads and public transportation routes.

Poor Bus Stop Detection

The poor bus stop detection algorithm resulted in bad arrival time prediction models when using the naïve arrival time aggregation method. Figure 6.7 illustrates a scenario where the position of the final bus has multiple modes, but the system does not register one of the modes as being part of the bus stop. As a result, the bus stops at the bus stop for slightly over 3 minutes before driving past a mode which the system recognises. As shown, this causes a large offset to occur in the arrival time prediction for the trained model. The platform (mode) where the bus stops at is a dedicated waiting space and drop-off space for passengers. As such, this platform should be included in the bus stop detection. With an improved bus stop detection algorithm, such problems could have been avoided.

Time and Day Model

An interesting concept that should be explored is the addition of time and day to the model as a feature. Different times of the day and different days are expected to exhibit different arrival time patterns. For example, traffic is expected to be heavier in certain areas during

rush hours and before holidays, while lighter on evenings and nights. Modelling this could improve the accuracy of the arrival time prediction models.

6.6 Future Work

Future work on arrival time prediction (trajectory forecasting) should be focused on improving the arrival time prediction aggregation. The weighted mixture method should be implemented and tested as a next step. One could also experiment with the different approaches to model common stops, by adding information from external sources, as this could be beneficial information to have when planning routes and road improvements. Naturally, more days from the dataset should be processed and added to the model, as this thesis project only utilised the data from one day. The model should also be tested in comparison with the baseline available by Östgötatrafiken AB, by predicting the arrival time of buses in real-time, and compare with the prediction made by the existing system.

7 Conclusion

The effort involved in analysing a larger dataset with poor documentation is tremendous, but worthwhile. Interesting characteristics are mentioned, such as imprecise algorithms and problems due to human errors. Real-world data is typically never perfect; pre-processing is a major part of transforming raw data into useful features. One of the first problems that needs to be solved is to connect the different types of data available in the dataset. In this thesis project, this is done via a context-providing finite-state machine. The result of the pre-processing step is a collection of trajectories for each bus line, which proves to be a useful resource when solving higher-level problems.

The problem of poor bus stop detection is also discussed and explained. An approach on how to solve this problem is proposed and outlined, with the steps briefly covered. The problem of poor bus stops also influenced the arrival time prediction solution. The arrival time prediction models are based on Gaussian process regression (GPR), creating a mixture of Gaussian process model (MoGP). One Gaussian process (GP) model is trained for each trajectory in a bus line. The models are naïvely aggregated using the means and a uniform prior. The results show that the quality of the arrival time prediction greatly varies depending on the tested trajectory and segment. The mean absolute error (MAE) scores range from around 4 to 13 seconds when looking at full trajectories. The MAE scores are better if evaluated on segments for a trajectory. The scores then range from around 3 to 7 seconds. The model achieves better arrival time predictions as the buses travel the segments, i.e, in general, the absolute error decreases over time for each segment. The GPS variance estimation problem is also solved using a MoGP with GPR. The model is able to produce an estimation of the variance, which varies based the environment and the shape of the trajectory. More models need to be trained in order to improve both the GPS variance estimation model and the arrival time prediction model. Improvements can also be made to both models, which should be explored in future work.

Bibliography

[1] Jin S Deng, Ke Wang, Yang Hong and Jia G Qi. 'Spatio-temporal dynamics and evolution of land use change and landscape pattern in response to rapid urbanization'. In: *Landscape and urban planning* 92.3-4 (2009), pp. 187–198.

[2] Jiyuan Liu, Jinyan Zhan and Xiangzheng Deng. 'Spatio-temporal patterns and driving forces of urban land expansion in China during the economic reform era'. In: *AMBIO: a journal of the human environment* 34.6 (2005), pp. 450–455.

[3] Ernst D Dickmanns, Birger Mysliwetz and Thomas Christians. 'An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles'. In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.6 (1990), pp. 1273–1284.

[4] Zhe Peng, Shang Gao, Zecheng Li, Bin Xiao and Yi Qian. 'Vehicle Safety Improvement through Deep Learning and Mobile Sensing'. In: *IEEE Network* 32.4 (2018), pp. 28–33.

[5] D. E. Knuth. 'Optimum binary search trees'. In: *Acta Informatica* 1.1 (Mar. 1971), pp. 14–25. ISSN: 1432-0525. DOI: 10.1007/BF00264289. URL: https://doi.org/10.1007/BF00264289.

[6] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN: 026218253X.

[7] Ian Czekala, Kaisey S. Mandel, Sean M. Andrews, Jason A. Dittmann, Sujit K. Ghosh, Benjamin T. Montet and Elisabeth R. Newton. 'Disentangling Time-series Spectra with Gaussian Processes: Applications to Radial Velocity Analysis'. In: *The Astrophysical Journal* 840.1 (2017), p. 49. URL: http://stacks.iop.org/0004-637X/840/i=1/a=49.

[8] Ruth Angus, Timothy Morton, Suzanne Aigrain, Daniel Foreman-Mackey and Vinesh Rajpaul. 'Inferring probabilistic stellar rotation periods using Gaussian processes'. In: *Monthly Notices of the Royal Astronomical Society* 474.2 (2017), pp. 2094–2108.

[9] A. M. Alaa, J. Yoon, S. Hu and M. van der Schaar. 'Personalized Risk Scoring for Critical Care Prognosis Using Mixtures of Gaussian Processes'. In: *IEEE Transactions on Biomedical Engineering* 65.1 (Jan. 2018), pp. 207–218. ISSN: 0018-9294. DOI: 10.1109/TBME.2017.2698602.

[10] Andrew O. Finley, Abhirup Datta, Bruce C. Cook, Douglas Morton, Hans E. Andersen and Sudipto Banerjee. 'Applying Nearest Neighbor Gaussian Processes to Massive Spatial Data Sets: Forest Canopy Height Prediction Across Tanana Valley Alaska'. In: (Feb. 2017). URL: http://arxiv.org/abs/1702.00434.

[11] P. Morales-Álvarez, A. Pérez-Suay, R. Molina and G. Camps-Valls. 'Remote Sensing Image Classification With Large-Scale Gaussian Processes'. In: *IEEE Transactions on Geoscience and Remote Sensing* 56.2 (Feb. 2018), pp. 1103–1114. ISSN: 0196-2892. DOI: 10.1109/TGRS.2017.2758922.

[12] Maziar Raissi, Paris Perdikaris and George Em Karniadakis. 'Machine learning of linear differential equations using Gaussian processes'. In: *Journal of Computational Physics* 348 (2017), pp. 683–693. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2017.07.050. URL: http://www.sciencedirect.com/science/article/pii/S0021999117305582.

[13] Mark van der Wilk, Carl Edward Rasmussen and James Hensman. 'Convolutional Gaussian Processes'. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett. Curran Associates, Inc., 2017, pp. 2849–2858. URL: http://papers.nips.cc/paper/6877-convolutional-gaussian-processes.pdf.

[14] Michael Stein. 'Spatio-Temporal Data Analysis at Scale Using Models Based on Gaussian Processes'. In: (Mar. 2017). DOI: 10.2172/1346562.

[15] Jing Dong, Byron Boots and Frank Dellaert. 'Sparse Gaussian Processes for Continuous-Time Trajectory Estimation on Matrix Lie Groups'. In: *CoRR* abs/1705.06020 (2017). URL: http://arxiv.org/abs/1705.06020.

[16] Mattias Tiger and Fredrik Heintz. 'Towards Unsupervised Learning, Classification and Prediction of Activities in a Stream-Based Framework'. In: *Proceedings of the Thirteenth Scandinavian Conference on Artificial Intelligence (SCAI) :* 2015. ISBN: 978-1-61499-588-3. URL: https://www.ida.liu.se/divisions/aiics/publications/SCAI-2015-Towards-Unsupervised-Learning.pdf.

[17] M. Tiger and F. Heintz. 'Online sparse Gaussian process regression for trajectory modeling'. In: *2015 18th International Conference on Information Fusion (Fusion)*. July 2015, pp. 782–791.

[18] Mattias Tiger and Fredrik Heintz. 'Gaussian Process Based Motion Pattern Recognition with Sequential Local Models'. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-148724.

[19] Cort J Willmott and Kenji Matsuura. 'Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance'. In: *Climate research* 30.1 (2005), pp. 79–82.

[20] T. Chai and R. R. Draxler. 'Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature'. In: *Geoscientific Model Development* 7.3 (2014), pp. 1247–1250. DOI: 10.5194/gmd-7-1247-2014. URL: https://www.geosci-model-dev.net/7/1247/2014/.

[21] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo Le'on-Villagr'a, Zoubin Ghahramani and James Hensman. 'GPflow: A Gaussian process library using TensorFlow'. In: *Journal of Machine Learning Research* 18.40 (Apr. 2017), pp. 1–6. URL: http://jmlr.org/papers/v18/16-537.html.

[22] Dexter C. Kozen. *Automata and Computability*. 1st. 233 Spring Street, New York, NY 10013, USA: Springer Science+Business Media, Inc., 1997. ISBN: 0-387-94907-0.

[23] Luc Anselin. 'Interactive techniques and exploratory spatial data analysis'. In: *Geographical Information Systems: principles, techniques, management and applications* 1 (1999), pp. 251–264.

[24] Andrew Gelman. 'A Bayesian Formulation of Exploratory Data Analysis and Goodness-of-fit Testing*'. In: *International Statistical Review* 71.2 (2003), pp. 369–382. ISSN: 1751-5823. DOI: 10.1111/j.1751-5823.2003.tb00203.x. URL: http://dx.doi.org/10.1111/j.1751-5823.2003.tb00203.x.

[25] David C. Hoaglin. 'John W. Tukey and Data Analysis'. In: *Statistical Science* 18.3 (2003), pp. 311–318. ISSN: 08834237. URL: http://www.jstor.org/stable/3182748.

[26] John W Tukey. *Exploratory data analysis*. Reading, Mass., 1977.

[27] Paul F Velleman and David C Hoaglin. *Applications, basics, and computing of exploratory data analysis*. Duxbury Press, 1981. ISBN: 0-87150-409-X. URL: http://hdl.handle.net/1813/78.

[28] Andrew P. Bradley. 'The use of the area under the ROC curve in the evaluation of machine learning algorithms'. In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(96)00142-2. URL: http://www.sciencedirect.com/science/article/pii/S0031320396001422.

[29] Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. 'Thumbs Up?: Sentiment Classification Using Machine Learning Techniques'. In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86. DOI: 10.3115/1118693.1118704. URL: https://doi.org/10.3115/1118693.1118704.

[30] John R Quinlan et al. 'Learning with continuous classes'. In: *5th Australian joint conference on artificial intelligence*. Vol. 92. Singapore. 1992, pp. 343–348.

[31] Frank van Diggelen and Per Enge. 'The worlds first gps mooc and worldwide laboratory using smartphones'. In: *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*. 2015, pp. 361–369.

[32] M. A. Carreira-Perpinan. 'Mode-finding for mixtures of Gaussian distributions'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (Nov. 2000), pp. 1318–1323. ISSN: 0162-8828. DOI: 10.1109/34.888716.

[33] Dorin Comaniciu and Peter Meer. 'Mean shift: A robust approach toward feature space analysis'. In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002), pp. 603–619.

[34] Seppo Pulkkinen, Marko Mikael Mäkelä and Napsu Karmitsa. 'A continuation approach to mode-finding of multivariate Gaussian mixtures and kernel density estimates'. In: *Journal of Global Optimization* 56.2 (2013), pp. 459–487.