

Gaussian Process Regression based GPS Variance Estima- tion and Trajectory Forecast- ing

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

Linus Kortesalmi

Supervisor : Mattias Tiger
Examiner : Fredrik Heintz

External supervisor : Simon Johansson

Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innehåller rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>.

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.

Abstract

Many services today rely on information gained from analysing spatio-temporal datasets. New insights can also be made and different problems can be tackled from the analysis. Gaussian process regression is an effective method for solving regression problems, which is a typical class of problems. 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. 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 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, alongside the most probable arrival time. The naïve arrival time prediction model implemented has a root mean square error of 5 to 19 seconds. The results from the GPS variance estimation problem is a model which can compare the variance for different environments on a given trajectory.

Acknowledgments

I would like to thank my examiner 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 Linköping University for his endless help with the Gaussian Processes and keeping my spirit high throughout the project.

Osäker hur
Heintz bör
titularas

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 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

Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 Aim	1
1.2 Research Questions	1
1.3 Delimitations and Limitations	3
1.4 Structure of the Report	4
2 Background	5
2.1 Binary Search	5
2.2 Gaussian Processes	5
2.3 Evaluation Metrics	9
2.4 GPflow	9
2.5 Finite-State Machines	9
2.6 Exploratory Data Analysis	10
3 Data	12
3.1 Background	12
3.2 Structure	13
3.3 Dataset Pre-Study	16
3.4 Pre-Processing Events	18
3.5 Future Work	24
4 Bus Stop Detection	25
4.1 Methodology	26
5 GPS Variation Estimation	29
5.1 Stop Compression	31
5.2 Synchronising Input Space	31
5.3 Segment Self-Overlapping Journeys	35
5.4 Probability Models	35
5.5 GPS Variance Estimation Results	38
5.6 Discussion	38
5.7 Future Work	40
6 Arrival Time Prediction	41

6.1	Initial Methodology	41
6.2	Segment Bus Stops	42
6.3	Trajectory Segment Forecasting Models	43
6.4	Results	49
6.5	Discussion	49
6.6	Future Work	52
7	Conclusion	53
	Bibliography	54

List of Figures

2.1	Example of a binary search algorithm run	6
2.2	Samples from GP priors defined by common covariance functions	7
2.3	Fusion and Combining two Gaussian distributions	8
2.4	Example of a stem-and-leaf plot	10
3.1	Simplified graph illustrating the data gathering process	13
3.2	Average size of logs	14
3.3	Example of a raw ObservedPositionEvent entry	14
3.4	The distribution of event types for an arbitrary day in the dataset	15
3.5	Example of early stopping in a journey	17
3.6	Another example of early stopping in a journey	18
3.7	Real-world example of a bus driver stopping roughly 45 meters before reaching the final bus stop	19
3.8	Example of self-overlap in a bus journey	19
3.9	A geo-fence is constructed to filter out events occurring outside the virtual perimeter	20
3.10	Finite-state machine providing context to ObservedPositionEvents	21
3.11	Real-world scenario illustrating when events occur in a logical ordering	22
3.12	Example of early stopping in a journey	22
3.13	Another example of early stopping in a journey	23
4.1	Example of poor bus stop detection	25
4.2	Output from the bus stop detection algorithm from the "Internal Analysis" component	26
4.3	Simulated example of a multimodal bus stop	28
5.1	Visualisation of multiple journeys by different buses in the same area	30
5.2	Illustrates the case of a poorly calibrated GPS sensor on an individual bus	31
5.3	Real-world example of multiple buses driving on a road not correctly modelled by Google Maps	32
5.4	Example of a trajectory with uncompressed speeds	33
5.5	Example of a stop compressed trajectory	33
5.6	Another example of a stop compressed trajectory segment	34
5.7	f_1 GP from the first approach	35
5.8	f_1 GPs from the second approach	36
5.9	GPS variance variation estimation	38
6.1	Example of trajectory segments	43
6.2	Visualisation of f_A models with different kernels	46
6.3	Visualisation of arbitrary segments with the RBF+Linear kernel	47
6.4	Visualisation of arbitrary segments with the RBF+Linear kernel	47
6.5	Predicted arrival times	48
6.6	Example of poor bus stop detection for a bus stop with multiple modes	52

List of Tables

6.1	Arrival time prediction with the f_B GP model	44
6.2	Arrival time prediction with the f_A GP model	44
6.3	Metric scores for f_A GP models	46
6.4	Evaluation of arrival time prediction model	49
6.5	Evaluation of arrival time prediction model	49

Todo list

Osäker hur Heintz bör tituleras	iv
Jag är inte helt säker på vilken score som är bäst att använda. En kollar på medelscores från alla segment, medan den andre kolla på en trajektorie nivå.	49



1 Introduction

Today, there are an increasing number of services built upon the use of spatio-temporal datasets. For example, public transportation companies use the 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, e.g., the field of autonomous vehicles utilise spatio-temporal dataset for various applications [3, 4].

The purpose of this thesis project is to perform analysis on a novel spatio-temporal dataset received from 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 could be gained about the flow of traffic, the arrival time prediction for service vehicles could also be improved or better position detection of vehicles. 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 could be interesting not only to the dataset provider but also in a broader context, such as 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 will be implemented with their outcomes presented in the thesis.

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*, *Higher-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 machine learning techniques are applied. The insights from these questions are the basis for the *Higher-Level Problem Questions*. The *Higher-Level Problem Questions* formulate various high-level problems 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 *Higher-Level Problem Questions*.

Metadata Questions:

1. *What kind of information is available in the dataset provided by Östgötatrafiken AB?*
What kind of data does the dataset contain? Which features exist in the dataset, such as different event types, event parameters, and event structures? What does 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 higher-level problems?*
The pre-processing focuses of solving problems inherent in the dataset, such as processing events and extracting relevant information from them or building a knowledge base by looking at the ordering of events. What are some pre-processing methods that could be applied to this dataset? How can the raw data in the dataset be transformed to useful features? How does the ordering of the raw data affect the solutions for higher-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 higher-level problems negatively. Various anomaly detection algorithms can be applied to detect such cases, but they could also be solved using dataset-tailored algorithms. Which manual inspection methods could be used to detect noisy measurements? Which general anomaly detection algorithms could be applied? How could 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 could complement the data in the provided dataset. This question explores if these sources could be combined in order to solve problems on a higher level. Which external sources exist and what further information could be gathered from the external data sources? 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 were deemed missing from the given dataset (in *Metadata Question 1*)?

Higher-Level Problem Questions:

1. *What interesting higher-level problems can be investigated and solved based on the available data?*
The higher-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 could the existing system that generated the dataset be improved?

2. *How can these problems be solved?*

The solutions will tackle the core of each problem. Each solution will explicitly state if there is a baseline available for comparison or if one could be easily created. How can problems with GPS Positioning and Trajectory Forecasting be solved?

3. *How does solutions to these problems benefit society and the industry?*

This question analyses the solutions 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 could 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 return 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 will only focus on the data from buses; data from public transportation trains will be 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 can not 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 can not be retrieved from other sources. Bugs in the data gathering process can not be solved either.

The models used in the thesis project will be 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 cannot fully utilise the TensorFlow library, as it benefits from 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 higher-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 could 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

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

Gaussian processes [6] (GPs) are a Bayesian non-parametric approach to modelling. GPs have throughout the years been proved useful when solving a 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 risk scores ward patients [9]. Finley et al. showed in [10] how GPs could be used to predict the forest canopy height. In [11], large-scale GPs are used to improve remote

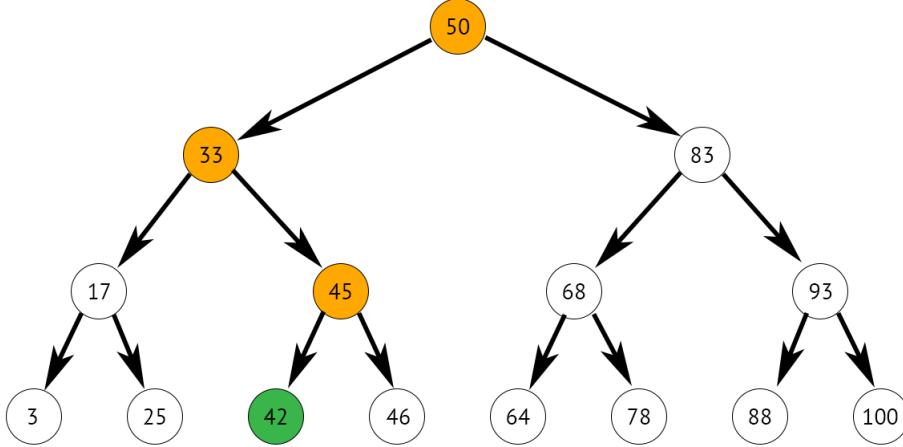


Figure 2.1: Example of a binary search algorithm run. The target value is 42. First the root node 50 is compared, but as $42 < 50$ the left subtree is explored. The new root node is 33 and $42 > 30$, so the right subtree is explored. The new root node is 45 and $42 < 45$. The algorithm should search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

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

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

where

- $m(x) = \mathbb{E}[f(x)]$ is the mean function;
- $k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$ is the covariance function. x and x' are two inputs.

A GP is thus completely described by its mean and covariance function, where the mean function is taken to be zero for notational simplicity [6]. It can be seen as a probability distribution over functions. The covariance function can be expressed in the form of a *kernel*, where 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

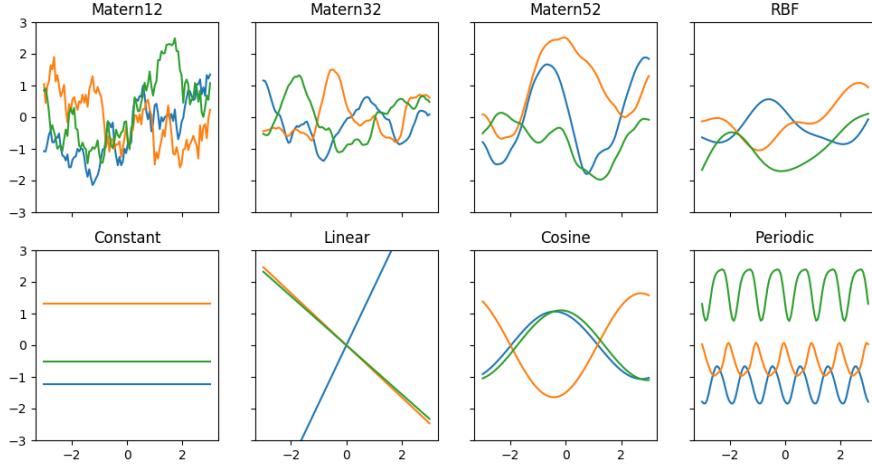


Figure 2.2: Samples from GP priors defined by common covariance functions. The functions exhibit different characteristics depending on the choice of kernel.

equations 2.2 and 2.3, respectively

$$k_{SE}(r) = \exp\left(-\frac{r^2}{2\ell^2}\right) \quad (2.2)$$

$$k_{\text{Matérn}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}r}{\ell}\right), \quad (2.3)$$

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 $\nu \rightarrow \infty$, the Matérn kernel becomes the RBF [6].

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

The GP approach is Bayesian as $f(x) \sim \mathcal{GP}$ encodes a prior belief about the unknown $f(\cdot)$.

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

$$y = f(x) + \epsilon, \quad (2.5)$$

where

$$\epsilon \stackrel{iid}{\sim} \mathcal{N}(0, \sigma_{noise}^2)$$

and $f(\cdot)$ is a prior. Here ϵ represents noisy observations, 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_* . The GP prediction (posterior) is defined by the equation

$$p(f_*|x, y, x_*) \sim \mathcal{GP}(\bar{f}_*, cov(f_*)), \quad (2.6)$$

where \bar{f}_* and $cov(f_*)$ are given by the equations

$$\bar{f}_* = K(x_*, x)[K(x, x) + \sigma_{noise}^2 I]^{-1}y \quad (2.7)$$

$$cov(f_*) = K(x_*, x_*) - K(x_*, x)[K(x, x) + \sigma_{noise}^2 I]^{-1}K(x, x_*), \quad (2.8)$$

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 Fusion formula on the predictive mean and variance is given by [17]

$$\sigma^2(x^*) = \frac{\sum_{j=1}^J N_j}{\sum_{j=1}^J N_j (\sigma_j^2(x^*))^{-1}}, \quad (2.9)$$

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

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 Combining formula on the predictive mean and variance is given by

$$\mu(x^*) = \frac{\sum_{j=1}^J N_j \mu_j(x^*)}{\sum_{j=1}^J N_j} \quad (2.11)$$

$$\sigma^2(x^*) = \frac{\sum_{j=1}^J N_j (\sigma_j^2(x^*) + \mu_j^2(x^*))}{\sum_{j=1}^J N_j} - \mu(x^*)^2 \quad (2.12)$$

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.

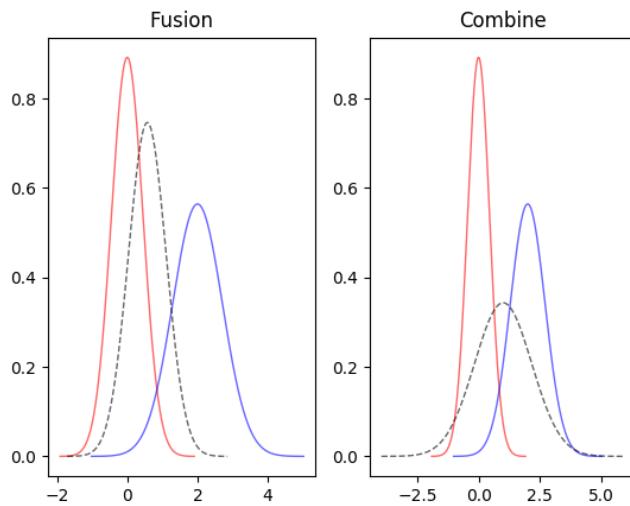


Figure 2.3: Fusion and Combining two Gaussian distributions, with $N_1 = N_2 = 1$. The red Gaussian distribution has mean 0 and variance 0.2. The blue Gaussian has mean 2 and variance 0.5.

[18].

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).

$$MSE(y, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.13)$$

$$MSLE(y, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^n (\ln(1 + y_i) - \ln(1 + \hat{y}_i))^2 \quad (2.14)$$

$$MAE(y, \hat{y}_i) = \text{median}(|y_1 - \hat{y}_1|, \dots, |y_n - \hat{y}_n|) \quad (2.15)$$

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

$$RMSE(y, \hat{y}_i) = \sqrt{MSE(y, \hat{y}_i)} \quad (2.16)$$

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 useful metric if the errors follow a Gaussian distribution.

2.4 GPflow

GPflow¹ [21] is a GP framework for the Python programming language² using TensorFlow³. It is an open-source software which has its origins from the contributors of GPy⁴, another famous framework for implementing GPs in Python. As GPflow is built upon TensorFlow, it achieves better performance when using GPUs for processing [21]. 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.

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 excellent book "Automata and Computability" by Kozen [22]. A FSM is defined as states with potentially multiple transitions between each state. The FSM can only be in one given state at any time.

Formally a 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.
- $\delta : Q \times \Sigma \rightarrow Q$ 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.

A 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 containing various techniques [23, 24, 25, 26, 27] for exploring and analysing data. 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 every single data point used [27].

15, 17, 19, 21, 22, 24, 30

Stem	Leaf
1	5 7 9
2	1 2 4
3	0

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 applying tools and statistics to analyse data in a meaningful way, e.g., it could be applied to the detection of outliers, smoothing the data, and 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 theory 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 to quantify the difference. For example, the sample mean estimator, or the *winsorised* mean can be used [26]. The winsorised mean handles the case where tails of a distribution dominates the value space. This would cause 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 is what remains after data analysis is performed. Residuals could, 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 would be 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. 2.17. It is worth noting that a single outlier will heavily affect the hanning smoothing and that in practice it is common to first apply a running median smoothing to remove outliers [27].

$$\hat{y}_t = \frac{1}{4}y_{t-1} + \frac{1}{2}y_t + \frac{1}{4}y_{t+1} \quad (2.17)$$



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 structure of the data describes the different event types in the dataset and how they were used in the thesis project. A pre-study of the dataset is conducted after the basic characteristics of the dataset have been described. The dataset pre-study analyses the data in-depth and identifies 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 was provided by Östgötatrafiken AB and contains ca 300 GB of data. Östgötatrafiken AB is owned by Östergötland County and is responsible for the public transportation in the county. This thesis project only analysed the bus data available in the given data set. The dataset is a collection of documents, where one document represents a full day of data. A typical day has a document size of around 5 GB, see Figure 3.2.

Data Gathering

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

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.

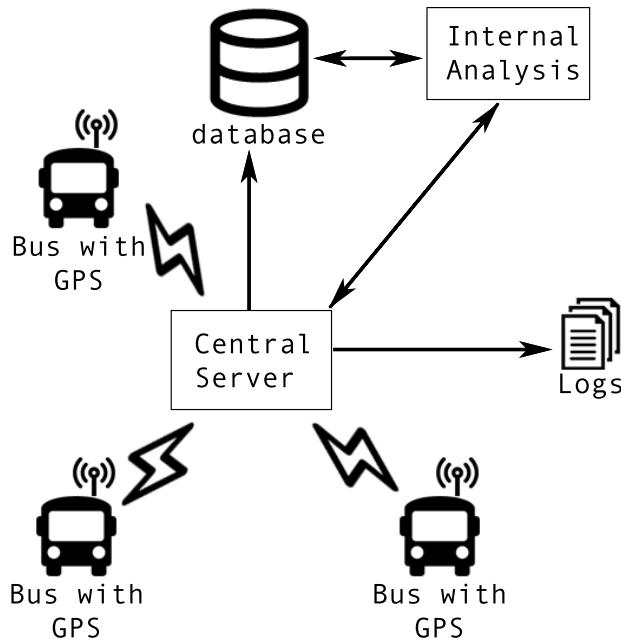


Figure 3.1: Simplified graph illustrating the data gathering process. Each bus is equipped with a GPS sensor and transmits its position to a central server, which is propagated to a database. The dataset used in this thesis project is a collection of logs, where each log is a collection of events. The 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.

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" component in the system. The precise implementation of the "Internal Analysis" 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 ca. 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 rest. 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.

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 (2018-02-16T11:53:56.000000+01:00), which is the timestamp from the system running on the bus.

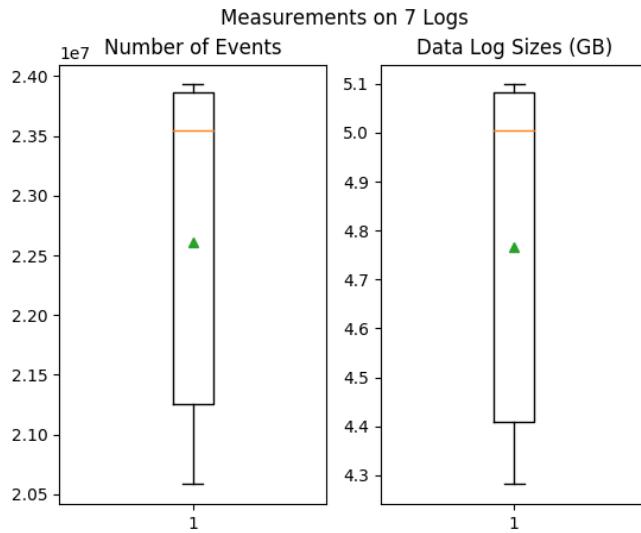


Figure 3.2: Average size of logs. Measurements are made on 7 logs (days) in the dataset. Naturally, there is a relationship between the number of events in a log and the size of the log.

The orange bar is the median, while the green triangle is the mean value.

Header	timestamp	event type	event ID
	2018-02-16T11:53:56.000000+01:00 2 ObservedPositionEvent 2623877798 Normal		
Body	vehicle ID	GPS	
	0.1 Bus otrafa.se 9031005990005485 5485 58.4233551025391,15.5914640426636	direction	speed
	58.4233551025391,15.5914640426636 168.899993896484 0 5482445		

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 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 "Internal Analysis" 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.

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 could 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

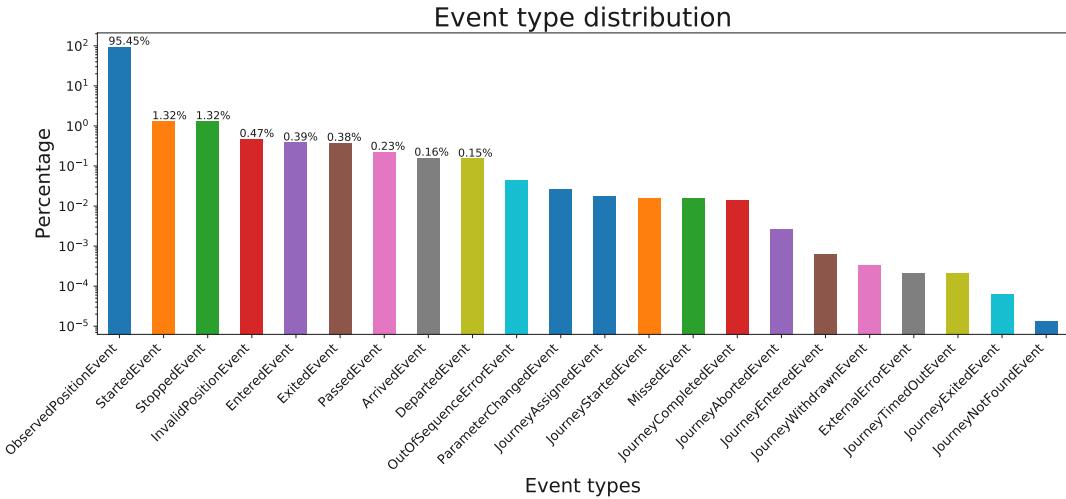


Figure 3.4: The distribution of event types for an arbitrary day in the dataset.

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 used 12. The events to use were chosen by analysing the log for a single day in great detail. Event types which occurred rarely and seemingly at random were discarded, as no pattern could be determined for them. 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 had the same coordinates. In this thesis project the `InvalidPositionEvent` type was discarded, due to the missing parameters and static GPS position.

The 12 event types used in this thesis project were:

- `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 "Internal Analysis" 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 could, 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 arrived at or departed 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 "Internal Analysis" 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.

3.3 Dataset Pre-Study

The dataset pre-study was conducted after the basic characteristics were identified. It analysed the dataset in greater depth, identifying interesting scenarios that occurred in the data. The analysis was conducted by manually parsing the data. This manual task used EDA methods to simplify the process; the focus was put on visualising the data in order to detect patterns and outliers. All events during an arbitrary day were initially used for this task. Anomalies that occurred were isolated and categorised by applying the rationale behind the binary search algorithm. The log was divided in two equal halves in order to see which half inhibited the anomalies. The procedure was repeated as long as the anomalies were present. This virtually halved the search space at each iteration. If an anomaly was missing from both halves it was an indication that the anomaly occurred by combining the events around the split.

The search was more complex than one simple binary search algorithm run, as anomalies could occur in both splits. If anomalies were present in both splits, the split intervals were saved and then the search proceeded in one of the splits. When the search exhausted the arbitrarily chosen split, it started 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.

Human Error: Early Stopping

Figures 3.5 and 3.6 illustrate two scenarios when the `JourneyCompletedEvent` type for a started journey would be missing. In Figure 3.5, the bus driver is supposed to visit the three markers in order to complete the journey for a particular line. In Figure 3.6, the dashed line highlights the route of the bus line, while the blue line is the actual route the bus drove. In both these real-world examples, the bus drivers ignore the final stops of the journeys.

The "Internal Analysis" 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 could be detected early. However, it would be 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 would have to be introduced to the system in order to separate these two cases. In the case of a wrong turn, the bus would eventually converge to the average journey, while in the early-stopping scenario the journey would most likely never converge.



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

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 "Internal Analysis" 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 there for a few minutes, before it drives to the first bus stop for the new bus line number which was assigned.

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.

Self-Overlap

Self-overlap causes issues when used as training data for GP models [18]. 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 done in the thesis project. In [17], 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 [17]. Generally, f is unknown and needs to be estimated using a GP model trained on the full trajectory [17]. If the pairwise distance between non-subsequent interpolated points is under a threshold, then the self-overlap is detected.

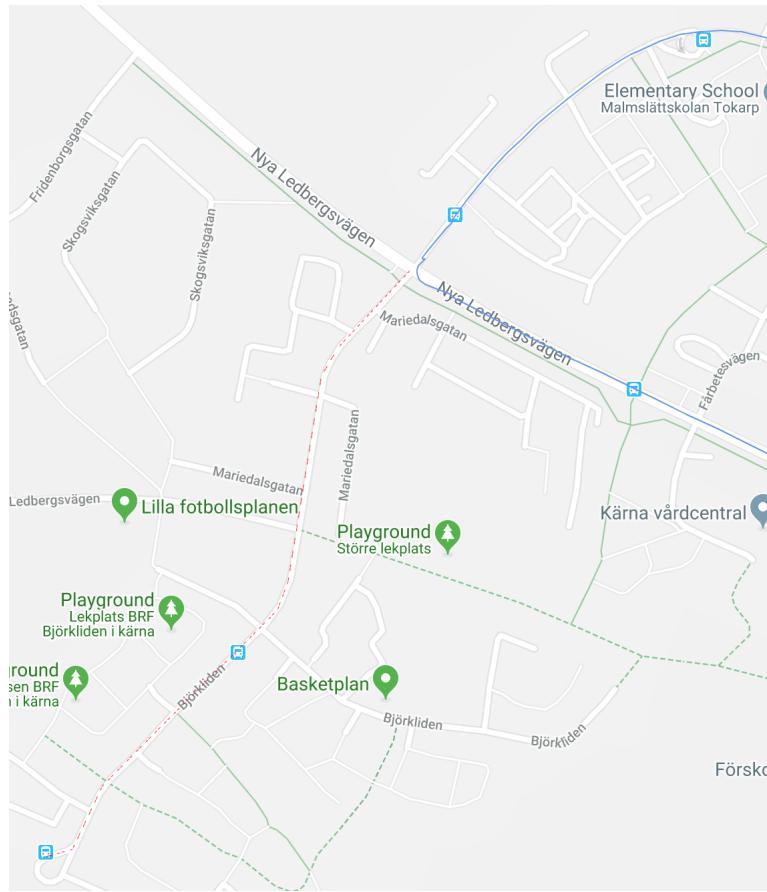


Figure 3.6: Another example of early stopping in a journey. The red dashed line is the planned route of the bus line. The blue line at "Nya Ledbergsvägen" is the actual route the bus drove. This results in the journey never being deemed completed, creating an erroneous ordering of contextual events (JourneyCompletedEvent never received).

3.4 Pre-Processing Events

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

During this pre-process step all the events from a vehicle type other than "Bus" were ignored. A *geo-fence* was 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 was to provide context to `ObservedPositionEvents`. Only using this one contextless event type greatly reduces the span of potential problems one could solve with the provided dataset. Context was 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. How-

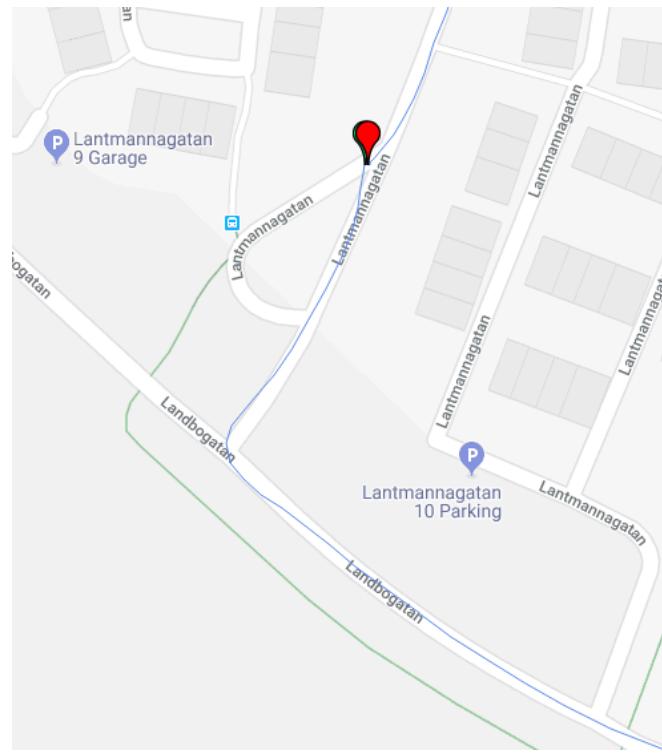


Figure 3.7: Real-world example of a bus driver stopping roughly 45 meters before reaching the final bus stop. The red marker is the GPS position where the bus stopped and the bus icon on the map is the pre-determined position of the bus stop. The bus stops there for a few minutes before it drives off to the first bus stop for a new journey. The bus stop detection algorithm systematically does not identify these cases.

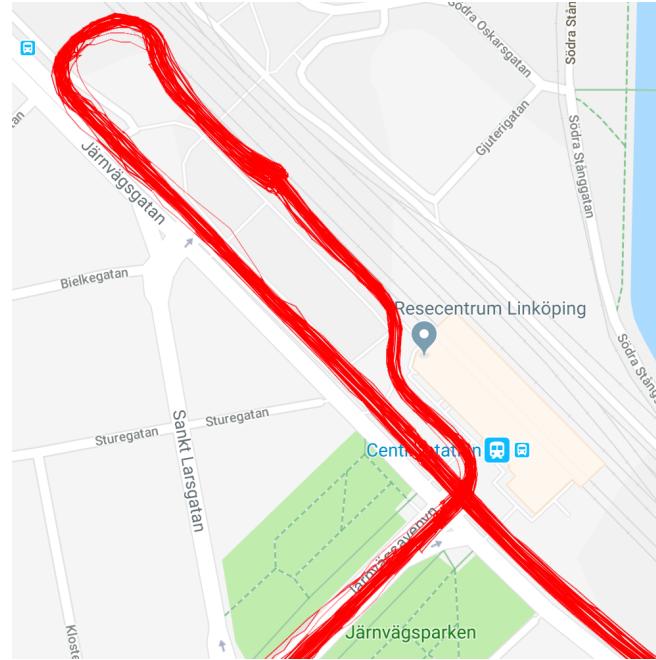


Figure 3.8: Example of self-overlap in a bus journey. The red lines show the trajectories of the buses driving for the same bus line.

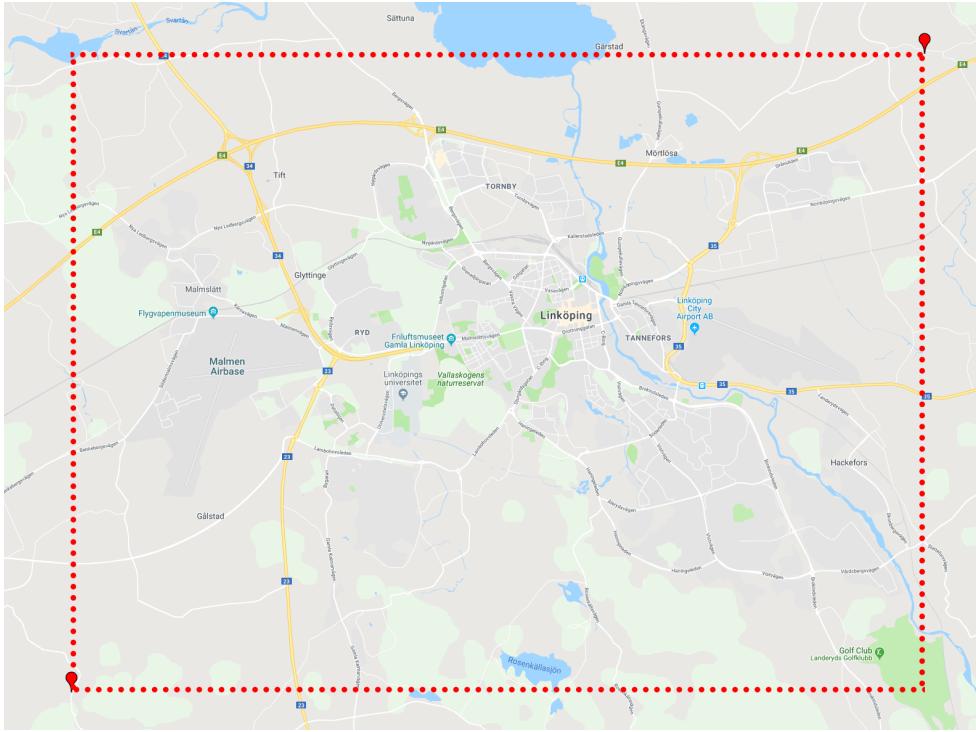


Figure 3.9: A geo-fence is constructed to filter out events occurring outside the virtual perimeter. The two red markers create a rectangular boundary, which is illustrated with the red-dotted line. The geographical area is the city of Linköping.

ever, this is not always the case when working with real-world data, as shown in Figure ???. 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 would not work when dealing with streamed real-time data, as the ordering cannot be changed without introducing a buffer. The context-providing FSM was slightly altered in order to support the processing of streamed real-time data. Another motivation behind the change was to remove the assumption that the sequence of *Event IDs* is always correct. The ObservedPositionEvents 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 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 ObservedPositionEvents received from the starting bus stop to the final bus stop. However, it is not always the case that the JourneyCompletedEvent type is in an erroneous ordering. Occasionally the type is missing from the sequence of events due to either human errors or imprecise algorithms in the "Internal Analysis" component. These scenarios can easily be detected by utilising the context-providing FSM with the added buffer extension.

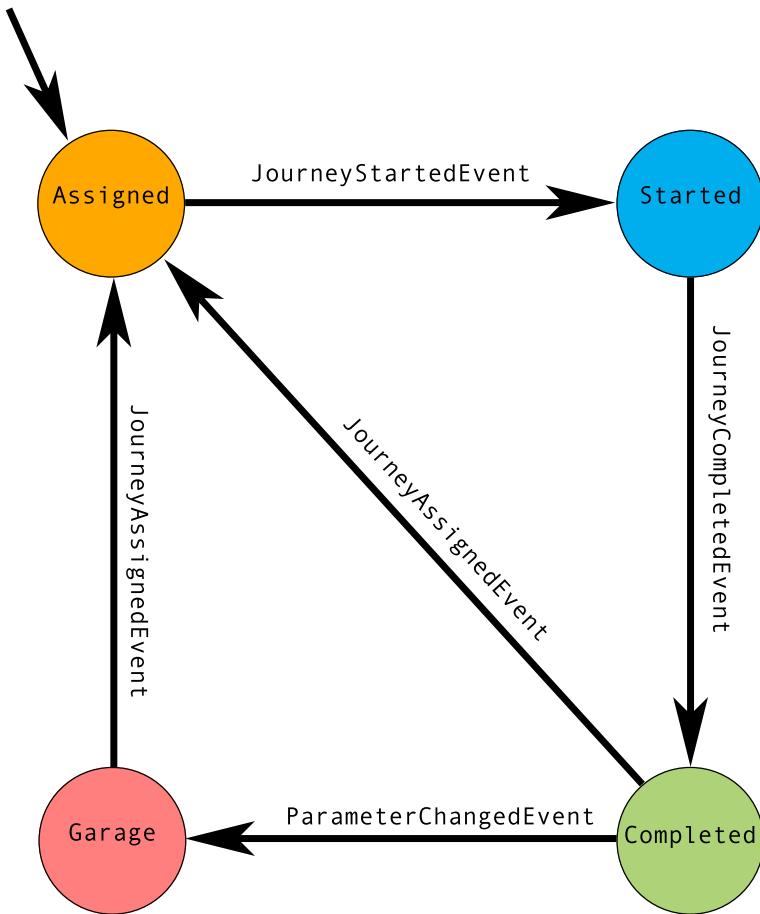


Figure 3.10: Finite-state machine providing context to `ObservedPositionEvents`. The constructed FSM is simplified to illustrate the best-case scenario. The "Assigned" state is the starting state. `ObservedPositionEvents` are assigned to the current FSM state.

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 could be extended to not only include `ObservedPositionEvent` types, but also `ArrivedEvent`, `DepartedEvent`, and `PassedEvent` types. A missed final bus stop could 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 would cause the complexity of the FSM to increase.

An alternative approach was implemented instead, as the existing context-providing FSM could easily be verified to produce correct output. The approach extracted all events of the types `StartedEvent`, `StoppedEvent`, `PassedEvent`, `ArrivedEvent`, and `DepartedEvent` for each specific *Vehicle ID*. These events were 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 were discarded. Events of the type `PassedEvent`, `ArrivedEvent` or `DepartedEvent` were included in the journey if



Figure 3.11: Real-world scenario illustrating when events occur in a logical ordering. The blue line to the left is `ObservedPositionEvents` in the "Started" state. Upon reaching the final bus stop for the line the state changes to "Completed". 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. 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 (not shown in the image), 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".

they occurred within a specific period of time of the first `ObservedPositionEvent`. When all `ObservedPositionEvents` of a journey were processed, the procedure continued with the next journey. The set of extracted events thus naturally reduced in size for each journey processed. Events occurring in-between journeys were discarded. When adding a bus stop the journey the name of the bus stop was also added to a separate array, in order to simplify the process of detecting faulty journeys.

Results

The result of the pre-processing step is a collection of journeys for each bus line. A journey consists of all the `ObservedPositionEvents` sent by the bus while the FSM was 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 was solved by sorting the events based on *Event ID* and using a buffer to retroactively add `ObservedPositionEvent` in the "Assigned" state to the "Started" state in the case

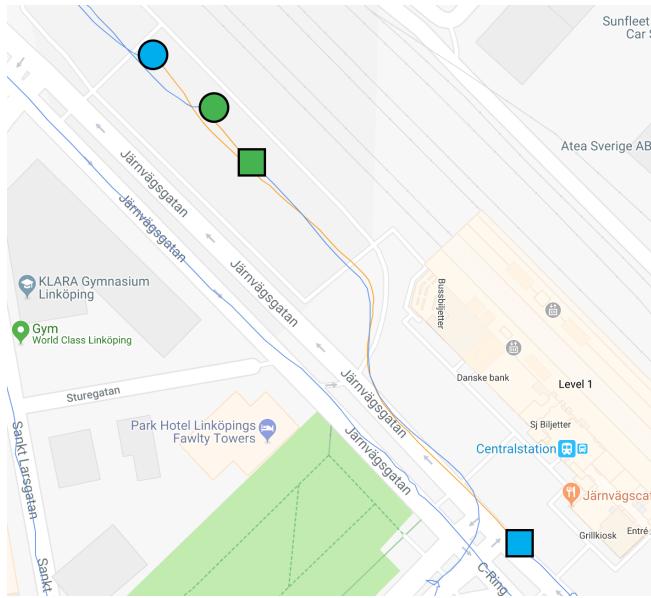


Figure 3.13: Another example of early stopping in a journey. The two circles (green and blue) denote the final bus stops for the respective bus lines. One of the buses is assigned a new bus line at the blue rectangle in the bottom-right corner, long before reaching the final bus stop. The other bus is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

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 faulty journeys will only be 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 faulty journeys are only marked as anomalies and discarded in this thesis. The fraction of faulty journeys can be calculated and used as a baseline when comparing an improved bus stop detection method with the existing one. Faulty journeys can be categorised based on the error in the journey, e.g. a missed starting bus stop is a different error compared with a missed final bus stop. The categorisation of faults could provide deeper insights into the dataset. For example, the insights could 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 should be investigated in-depth with regard to the whole dataset and not only for an arbitrary day. Some of the events could perhaps be used to easier detect or handle the faulty journeys. During the initial manual processing of the data, no reasonable pattern was discovered for any of the excluded event types. This may be an indication that the rarer event types are wrongly produced by the "Internal Analysis" component. A more probable explanation would be 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 would probably have yielded better results and pattern recognition.

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



4 Bus Stop Detection

The first high-level problem is the problem with inaccurate bus stop detection. This causes the system to occasionally miss bus stops, which results in faulty or incomplete journeys. Incomplete journeys require more pre-processing to be done in order to use the data. The pre-processing might also cause inaccuracies to occur in order to use the data. An example of this 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 stop coordinates are pre-determined, meaning that they do not always represent the actual bus stop. This introduces an overhead to the system, where bus stop coordinates need to be updated in order to continue to provide accurate detection of bus stops.

Figure 4.2 shows the results from the existing bus stop detection algorithm of the "Internal Analysis" 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).

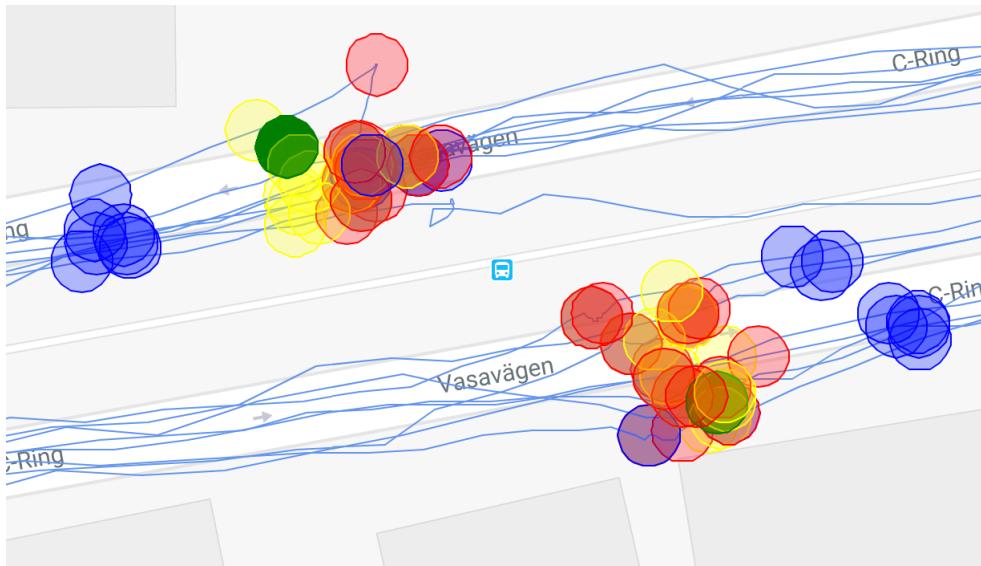


Figure 4.2: Output from the bus stop detection algorithm from the "Internal Analysis" component. The green circles are the pre-determined coordinates for the bus stop. The bus stop shown has two platforms, depending on if the bus travels west (top road) or east (bottom road). The red and yellow clusters 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 will merely act as a hypothesis on how the bus stop detection can be improved. The steps will not be implemented and the method will thus not be 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 will also be extended with `EnteredEvent` and `ExitedEvent` types.

Dynamic Radius

One of possible improvements to make is to introduce a dynamic radius to bus stops. The radius could be based on analysis of the positional data from buses passing by the bus stop.

Instead of using a pre-determined radius, the radius could be decided by 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 would then be required, in a pre-study step, to categorise the errors. Some errors, like the one in Figure 3.7, could 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 could adapt to this and increase the 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 would be to look at the stopped positions of buses at every bus stop. The positional data would create 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 would be 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 would be for the possibility of analysing the design of bus stops. The system would produce 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 would imply that the design of the bus stop could benefit by being changed. A wide multimodal distribution could, respectively, imply the need for certain changes to the bus schedules. For example, at specific times it could be the case that multiple buses concurrently arrive at the same bus stop, due to the nature of the time schedule. This would cause a queue at the bus stop, or potentially unsafe loading and unloading of commuters. A slight alteration in the schedule for a few of these buses could prevent this from occurring as often. Arrival time prediction could be used as an assisting tool to predict bus stop queues.

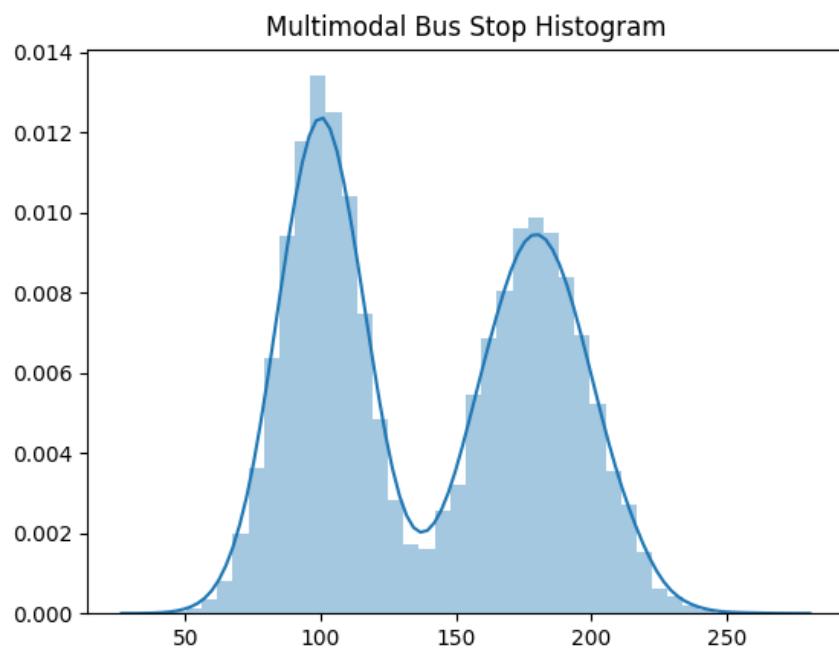


Figure 4.3: Simulated example of a multimodal bus stop. The buses mainly stop at two different positions of the platform (the two modes).



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. 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 could 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 would be 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 would exhibit a, potentially, larger difference in variance.

Road segments could 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 would most likely get a higher average variance, as the environment could change within the segment. The contextual segmentation would look at sequences of points in the journey, and group the sequential points which exhibit similar variance. The result would potentially be 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 would be major if the geographical segmentation was done between bus stops. The number of road segments would be 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 / Local Minima:* Each bus has its own GPS sensor, where different buses could have different models with their own calibrations. In the general case the different GPS sensors show small variance due to different calibrations. 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 "Internal Analysis" component. No bus stops are detected in this particular scenario. The GPS position determination is done by the sensor through an optimisation process. The optimisation process could get stuck in a local minima, which would affect the accuracy of the GPS position. A local minima could also be the explanation to the constant bias observed in the extreme scenarios. If it was due to a local minima, the accuracy of the GPS position would improve over time. The extreme situations can fortunately be detected by combining the positional events with the bus stop events. However, differences in calibrations or local minima which yield smaller offsets cannot be detected with this combining approach.
3. *Time:* The timestamp of the position events affects 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 timestamp. 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.

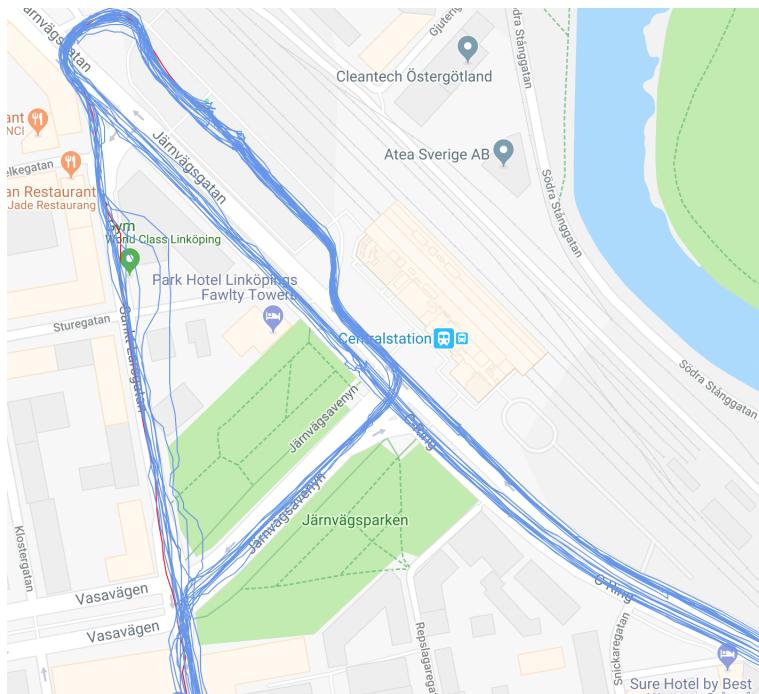


Figure 5.1: Visualisation of multiple journeys by different buses in the same area. Each blue line is a journey in the "Started" state. The variance differs between different road segments, as shown around "World Class Linköping" compared to "Järnvägsgatan".

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.

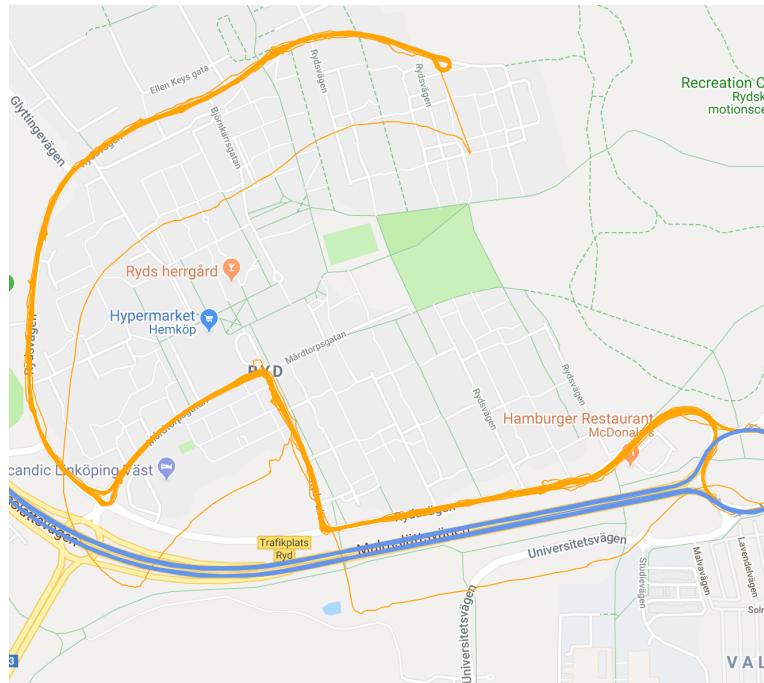


Figure 5.2: Illustrates the case of a poorly calibrated GPS sensor on an individual bus. The orange lines creating the thicker line are all the other journeys for the same bus line, but done by other buses. The thin orange line is one bus with an extremely poorly calibrated GPS sensor. It seems to have 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 were discarded. The result is a journey (trajectory) consisting of a sequence of coordinates where stops are invisible time-wise.

Examples of the procedure is shown in Figures 5.4-5.6. All of the figures show the same trajectory. Figure 5.4 shows the trajectory without stop compression, where the stops are highlighted by drawn lines. Figure 5.5 shows the stop compressed speeds and coordinates of the trajectory. Figure 5.6 zooms in on an arbitrary segment of the stop compressed 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

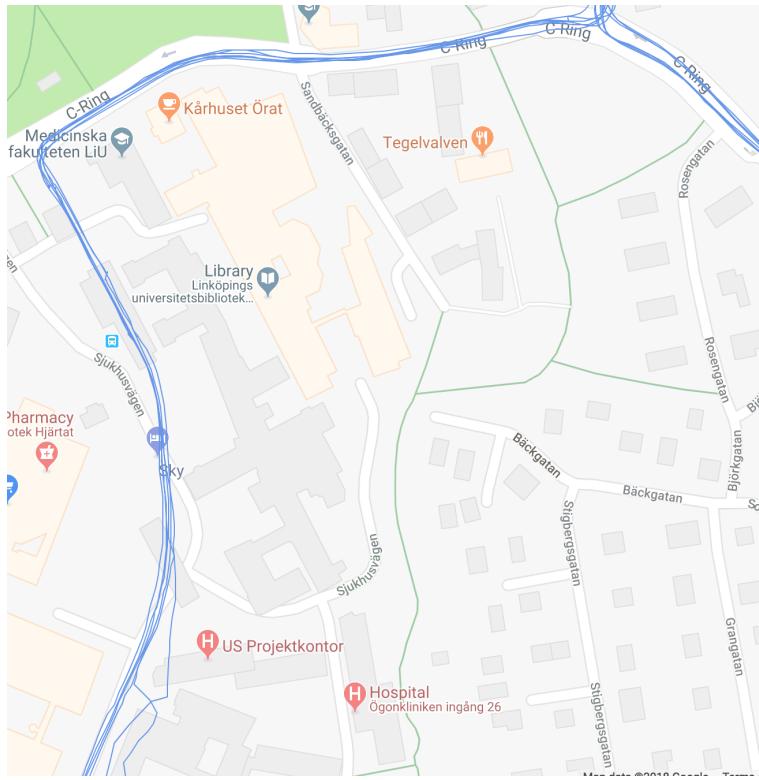


Figure 5.3: Real-world example of multiple buses driving on a road not correctly modelled by Google Maps. For example, at "Sjukhusvägen" the buses are driving through a building according to Google Maps. In this particular scenario the area has received major road re-structuring and renovation. The Google Maps representation is thus likely outdated.

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

$$f_1 : (x, y) \longmapsto \tau, \quad (5.1)$$

where (x, y) is the (longitude, latitude)-pair and τ the synchronised *time* parameter. 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 were tested and implemented:

1. **Speed value of 3:** Instead of discarding `ObservedPositionEvents` 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 was 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 `ObservedPositionEvents` were included, regardless of their respective *Speed* values. Instead, the filtering was done by looking at the distance between each `ObservedPositionEvent`. The fifth decimal roughly corresponds to changes in metres¹. The distance filter is set to remove a point with a distance smaller than 6×10^{-5} to the previous, accepted point.

¹World Geodetic System: <http://epsg.io/7030-ellipsoid>

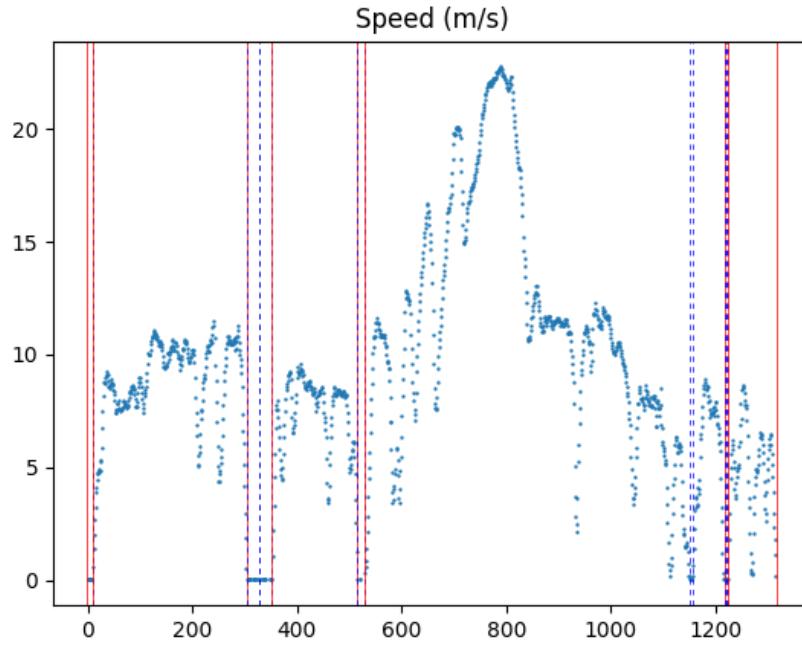


Figure 5.4: Example of a trajectory with uncompressed speeds. The red lines are `ArrivedEvents` and `DepartedEvents`. The dashed blue lines are `StartedEvents` and `StoppedEvents`. The red and blue lines usually coincide.

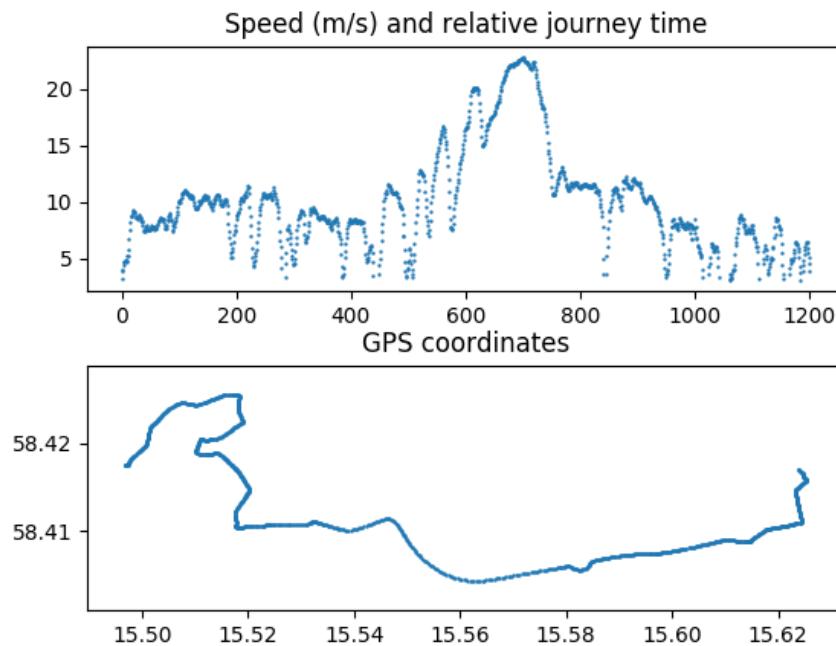


Figure 5.5: Example of a stop compressed trajectory. The *Speed* threshold is set to 0.1 m/s. The various speeds are shown in the top graph, while the trajectory is drawn in the bottom graph.

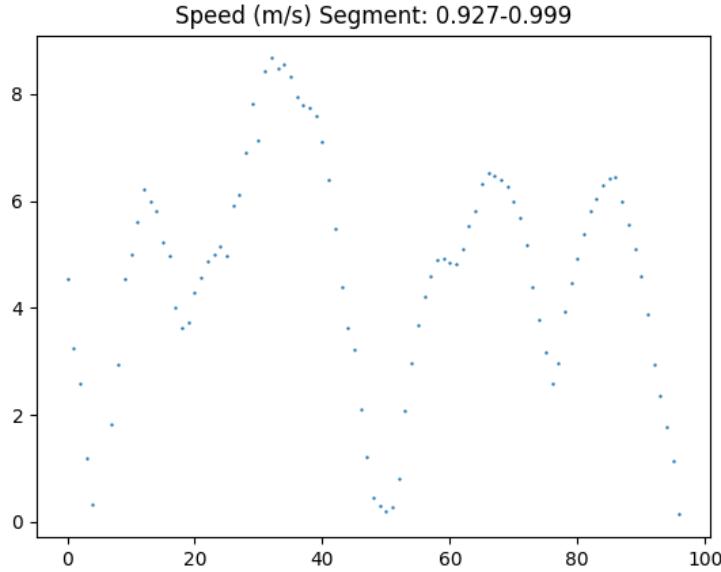


Figure 5.6: Another example of a stop compressed trajectory segment. The duration of a stop causes no delay in the relative time of the trajectory.

The first approach achieved poor results, see Figure 5.7, as points in the beginning of the trajectory were 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 `ObservedPositionEvents` from trajectories. The regression field is shown in the gradient background; the colour maps to a τ value. The second approach achieves better results. Figures 5.8a and 5.8b 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 5.8b.

Figure 5.8c 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 5.8a and 5.8b. 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} in an attempt to prevent overfitting, as many points in the trajectory are close to each other. This causes the model to underestimate the variance of observations. Similar results would be achieved if the trajectory was downsampled to only use every n data point. The fixed variance is too large for the model, which causes extremely poor regression.

Figure 5.8d 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 assume zero mean for the input data, which in this thesis project is achieved by fitting a global scalar model to the input data. The same trajectory used for f_1 is used to fit this scalar. All trajectory coordinates are centralised and normalised by the same scalar. The GPflow framework [21] is used to implement all GP functionality in this thesis work. The f_1 GPs were trained with a RBF kernel and the hyperparameters were automatically optimised using the GPflow framework. Automatic Relevance Determination [6] was used with the RBF kernel to allow for different length-scales in different dimensions.

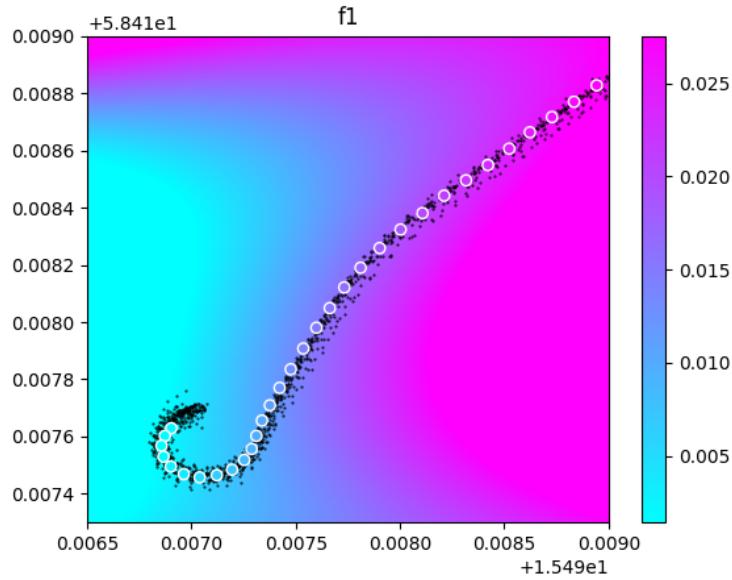


Figure 5.7: f_1 GP from the first approach. The coloured, outlined circles are training data points. The black data points are observed `ObservedPositionEvents` from all trajectories. The regression field is visualised by the gradient background.

5.3 Segment Self-Overlapping Journeys

After the input space synchronisation, each trajectory is 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

$$f_2 : \tau \mapsto x \quad (5.2)$$

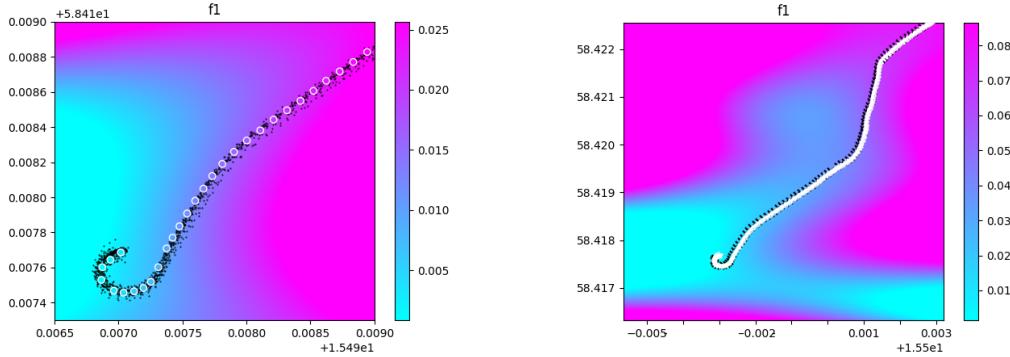
$$f_3 : \tau \mapsto y \quad (5.3)$$

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 `ObservedPositionEvents`, 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} , separate from the data noise used in GP f_1 .

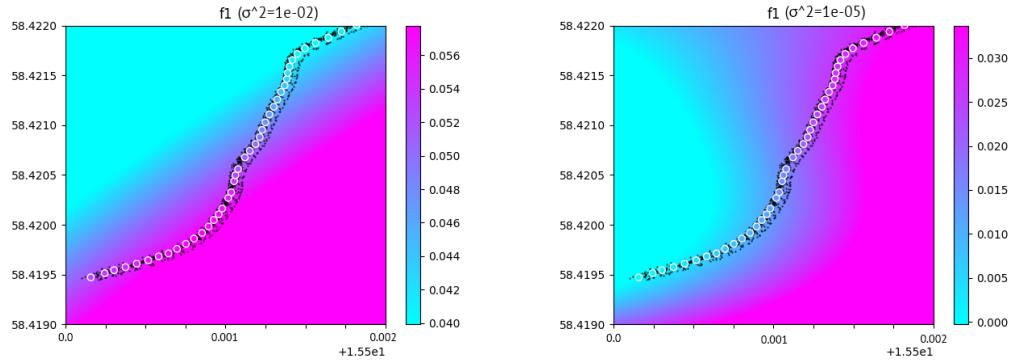
5.4 Probability Models

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 $p((x_*, y_*))$ is of interest in this thesis project, which can be calculated with two different approaches.



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

(b) Same f_1 GP as in 5.8a, 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 5.8: 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.

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

$$\mu_k = \begin{bmatrix} \mu_{k_x} \\ \mu_{k_y} \end{bmatrix} \quad (5.4)$$

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

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

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

where K is the number of trajectory models.

Unimodal Mixture of Gaussian Process Model by Combining

In order to create a unimodal MoGP model, f_{2_k} and f_{3_k} are point-wise combined for each trajectory k . The aggregation is done by using the Combining equations 2.11 and 2.12 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):

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

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

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

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

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 was trained on its own f_{2_k} and $f_{3_k} \Rightarrow M_k = 1$ for all $k = 1, \dots, K$ 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

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

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

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

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

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

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

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

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

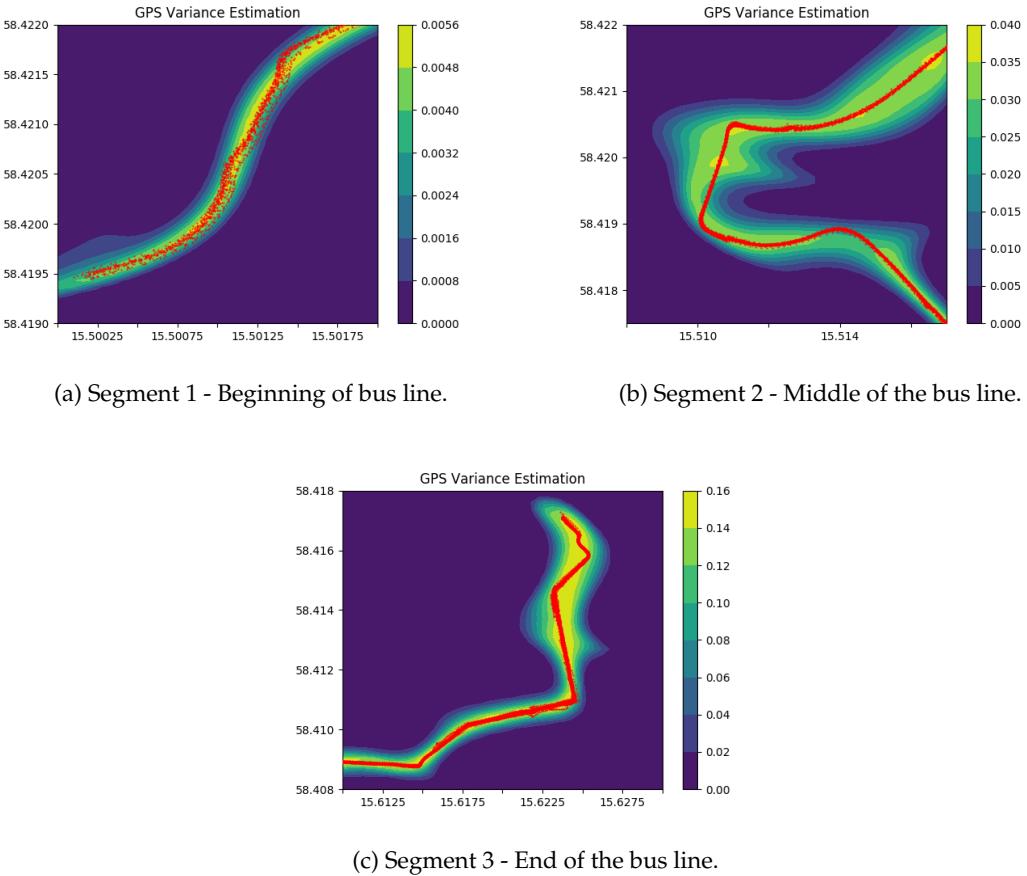


Figure 5.9: GPS variance variation estimation. Shows the probability that a GPS position is observed for a given bus line. The model uses a fixed data noise of 10^{-3} . The red dots are ObservedPositionEvents from the data set. The Y-axis is the latitude coordinates and the X-axis is the longitude coordinates.

5.5 GPS Variance Estimation Results

The unimodal MoGP model is visualised by evaluating the MoGP model on a grid of longitude and latitude values. 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 is integrated with the grid resolution in order to create a probability distribution. Figure 5.9 shows the GPS variation for three segments of a bus line.

5.6 Discussion

The results presented in Figure 5.9 all show low probabilities of observing certain GPS positions, even if the grid position is surrounded by 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 5.9c, 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. The reason for the increased variation might also be a poor generalisation from the f_1 GP, which perhaps handles the sharp turn poorly. The f_1 GP might have performed better if it was trained on segments of the trajectory, instead of trying to learn the full trajectory. The removal of the first segment from all trajectories were a necessary step in order to try and 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 variation was evaluated visually. Ideally, other metrics would be applied, such as determining locations where the variation was 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 were evaluated both visually and by looking at the resulting hyperparameters after the optimisation step. If the likelihood of the data were optimised to be small (10^{-6}), it could be a result of overfitting to the linear trend of the data.

In this thesis project, the data was 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 were fixed to 10^{-3} . They were tested with a data noise of 10^{-5} , as in the case of f_1 , but this resulted in barely any variation visible. The variation shown in Figure 5.9 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 were in this thesis project optimised using the L-BFGS-B algorithm implemented in open-source software Scipy². It would perhaps have been better to optimise the hyperparameters through Markov chain Monte Carlo (MCMC) using Hamiltonian Monte Carlo (HMC) sampling, implemented in the GPflow framework. The use of sampling to optimise hyperparameters 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 were restricted to 37. The trajectories were all collected from the same day. It took, on average, ca. 8 hours on the to retrain all the models, which caused changes to be expensive time-wise. It was not always immediately clear if a change would generalise well or not. For example, some configurations showed extremely poor results on only a small number of trajectories, while the results were average on the others. The training time proved to be a huge drain on the effectivity of the feature extraction and model optimisation.

²<https://docs.scipy.org/doc/scipy/reference/index.html>

The aim of the thesis project was 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 were this analysis is possible in a future work.

Only trajectories from buses were 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.

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, or which satellites were available, would give information about the potential periodicity of the variation.

Road Segments

Another area not investigated in this thesis project is the use of the Google Maps Roads API³. 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 scenarios with discrepancies between roads in Google Maps and actual roads the buses drove. The Google Maps roads are in these scenarios usually incorrectly drawn or has missing roads. Figure 5.3 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 5.3. 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.

³<https://developers.google.com/maps/documentation/roads/intro>



6

Arrival Time Prediction

Trajectory forecasting is the process of predetermining when a bus will arrive 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 will use 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?*
- *What is the probability of the bus arriving at the earliest/latest possible time?*
- *Which stop impacts the arrival time of the bus the most?*

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. The reason behind this is to indirectly model traffic and red lights as a latent variable. 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

$$f_1 : (x, y) \longmapsto \tau, \quad (6.1)$$

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 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 the problems described in Section 3.3. Each segment will have a small model overlap, which means that neighbouring segments will 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 5 ObservedPositionEvents, which is roughly 6.5% of an average-length segment (ca. 77 ObservedPositionEvents).
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 majorly 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 ObservedPositionEvents 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 date 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 ObservedPositionEvents on the Y-axis.

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

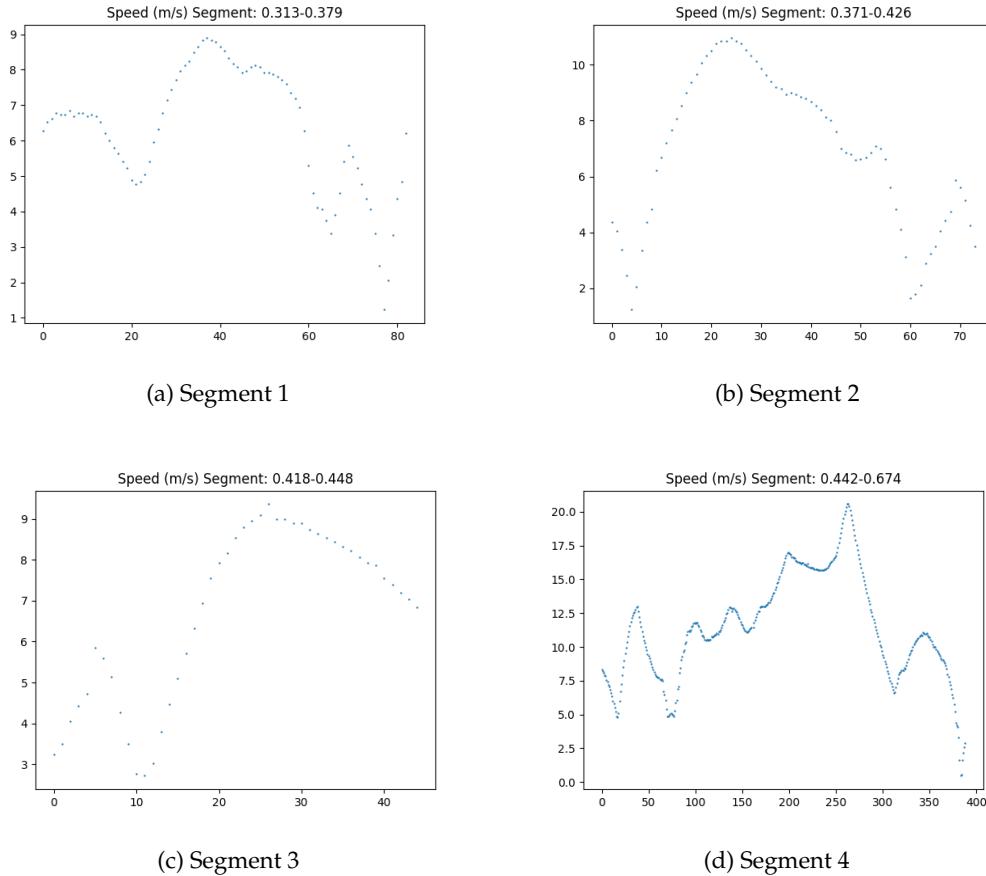


Figure 6.1: Example of four trajectory segments with segment overlap. The X-axis is the relative time (in seconds) of the trajectory segment, the Y-axis is the speed of each `ObservedPositionEvent` point.

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

$$f_A : \tau \mapsto t_s \quad (6.2)$$

$$f_B : (\tau, v) \mapsto t_s \quad (6.3)$$

- 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 data produced by the particular bus the GP models are evaluated with. GP model $f_{B_0}^{(2)}$ is trained on data produced by another bus. 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.

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

However, GP model $f_{B_0}^{(2)}$ performs extremely poorly on unseen data; which is an indication that the model does not generalise well.

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 will be the focus on 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:

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.

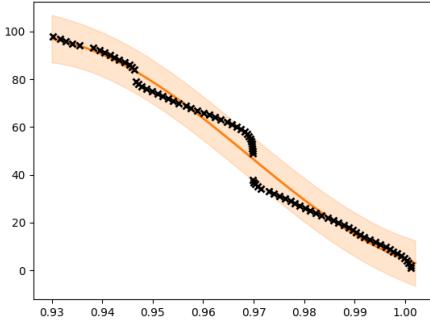
Model Evaluation f_A

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. However, in some segments the different choice of kernels greatly affected the robustness of the GP model. Figure 6.2 shows a segment for two different trajectories, were either only the RBF kernel performs well or only the RBF+Linear kernel. While the two kernels work equally well in the general case, there are segments were neither work, depending on the trajectory. Manual analysis of the different segments show that most segments follow a linear trend with occasional curvature. This suggests that a RBF+Linear kernel could be more suitable in the general case. While GPflow optimises the hyperparameters of the model, it seems like the models occasionally get stuck in a local minima, as shown in Figure 6.2b and 6.2c. The problem with poor performance could thus be solved by changing the hyperparameter initialisation.

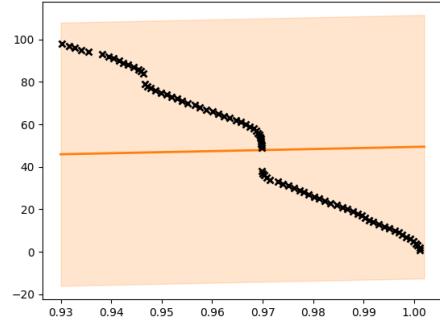
Figure 6.3a shows how the model in 6.2b performs better if the natural logarithm is applied to the input data. The improved performance indicates that the RBF+Linear model performs better if the input values are closer, or around zero. In Figure 6.3b, only the Linear kernel component from the RBF+Linear model shown in Figure 6.2b is visualised. The figure

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.

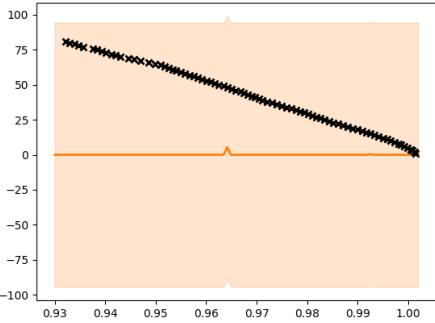
Kernel	RMSE $f_{A_0}^{(1)}$	RMSE $f_{A_0}^{(2)}$	MAE $f_{A_0}^{(1)}$	MAE $f_{A_0}^{(2)}$
Linear	24.85	25.26	20.56	21.49
White	26.80	26.80	22.38	22.38
RBF	1.56	13.53	0.84	11.69
RBF+Linear	1.56	13.60	0.84	11.84
Matern32	11.19	13.56	9.34	11.75



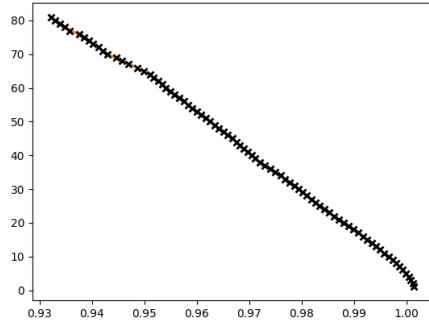
(a) Model of trajectory 3, segment 18, using the RBF kernel.



(b) Model of trajectory 3, segment 18, using the RBF+Linear kernel.

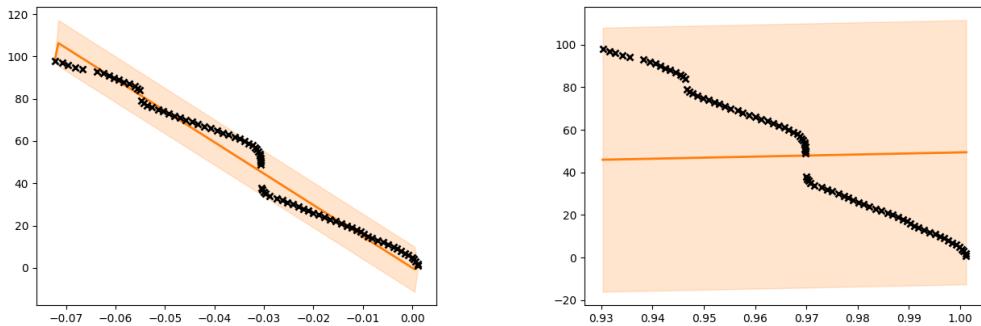


(c) Model of trajectory 5, segment 18, using the RBF kernel.



(d) Model of trajectory 5, segment 18, using the RBF+Linear kernel.

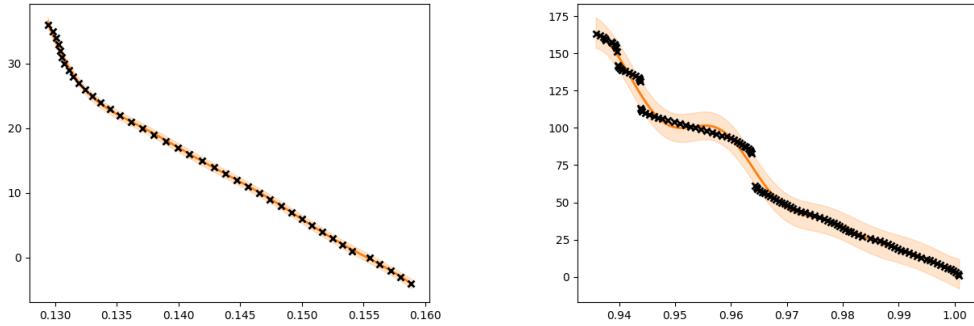
Figure 6.2: Visualisation of f_A models with different kernels. Each figure contains the predictive mean and variance of the model, alongside the training data points. The X-axis is the τ values of the trajectory segment and the Y-axis is the arrival time to the end of the segment, in seconds.



(a) RBF+Linear kernel with logarithmised input. The model is trained on data from trajectory 3, segment 18.

(b) Linear kernel component of the RBF+Linear model. The model is trained on data from trajectory 3, segment 18.

Figure 6.3: 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.



(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.

indicates that the Linear kernel component takes over the combined kernel. In order to try and resolve the problems, the initial hyperparameters of the RBF+Linear model are varied manually, until a stable model is acquired.

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. $\sigma_{Linear}^2 = 500$, where σ_{Linear}^2 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:

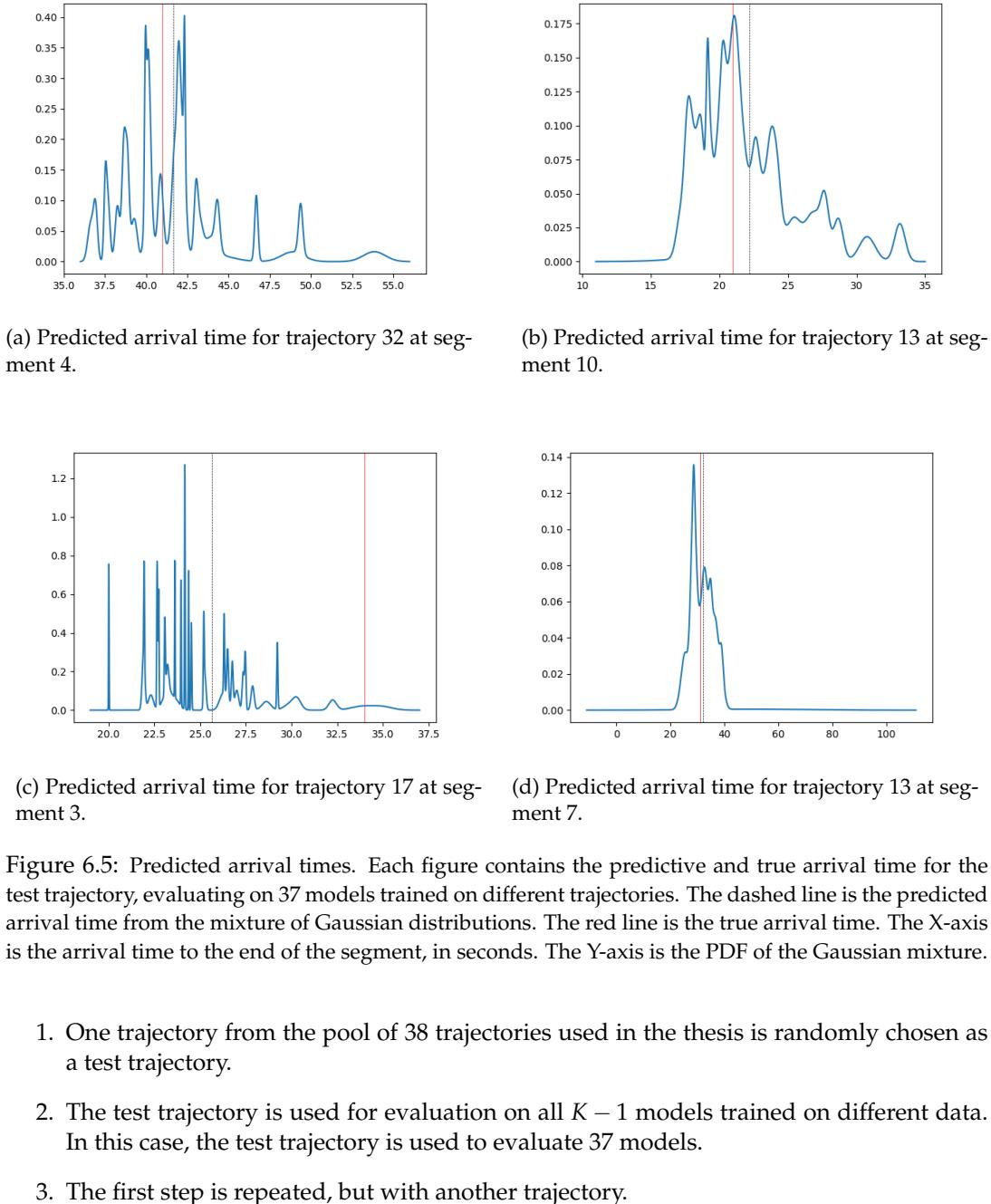


Figure 6.5: Predicted arrival times. Each figure contains the predictive and true arrival time for the test trajectory, evaluating on 37 models trained on different trajectories. The dashed line is the predicted arrival time from the mixture of Gaussian distributions. The red line is the true arrival 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.

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. 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 scores for the model is presented in Table 6.4. The RMSE and MAE is calculated for each 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

$$SE = \frac{\sigma}{\sqrt{n}}, \quad (6.4)$$

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

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

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.

The RMSE and MAE scores for each trajectory, without considering individual segments, is presented in Table 6.5

Table 6.4: Evaluation of arrival time prediction model. The RMSE and MAE is calculated for each segment in each trajectory. The average score is shown, together with the standard error using Bessel's correction.

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

Jag är inte helt säker på vilken score som är bäst att använda. En kollar på medelscores från alla segment, medan den andre kolla på en trajektorie nivå.

Table 6.5: Evaluation of arrival time prediction model. The RMSE and MAE is calculated for each 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

6.5 Discussion

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 will be 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 would be to calculate the *data log likelihood* of a set of observed data points, given a GP model. The data log likelihood would 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 would increase the weights of the predictions from the few models and reduce the weight of the predictions from the majority of the models. The result would be a better prediction of the arrival time. This method could 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 were to be 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 was 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.

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

Kernels

Due to time limitations in the thesis project it was necessary to narrow down the model search space as development continued. The choice of kernel is a valid discussion to be had. The performance between the RBF kernel and the RBF+Linear kernel were so similar that the hyperparameter initialisation might have caused the RBF+Linear kernel to slightly perform better. The linear trend in the data could have been removed using 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 could also have been optimised using HMC sampling, instead of the L-BFGS-B algorithm. This could have avoided the local minima problems with the hyperparameters. It was a non-trivial problem to find a good initialisation for the hyperparameters, as they were 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, ca. 7 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 would also have been interesting to experiment with, as the model might have captured new patterns.

Common Stops

In this thesis project only bus stops were regarded as a common stop, due to time limitations. The following stop compression approaches are available:

1. *Bus Stops*: This is the approach used in the thesis project. Only bus stops were considered as common stops, as this was the simplest approach possible (beyond no segmentation at all). The approach would result in a probability distribution of arrival time predictions for the next bus stop. Red lights and traffic would be a latent variable not being modelled by the approach, which could have an impact on the probability distribution. However, with the improved arrival time aggregation, this approach would could model red lights and traffic as latent variables. The simplicity of the model is maintained, which makes the model easy to understand, while increasing the complexity in the arrival time aggregation.
2. *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 OpenStreetMap¹, as they contain information about crossings with red lights. A crossing would then be more accurate labelled as having a red light or not, which could be modelled by the system. Nevertheless, traffic would still be a latent variable not modelled by the extended approach.

3. *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 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 6.6 illustrates a scenario where 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 should 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.

¹<https://www.openstreetmap.org/>

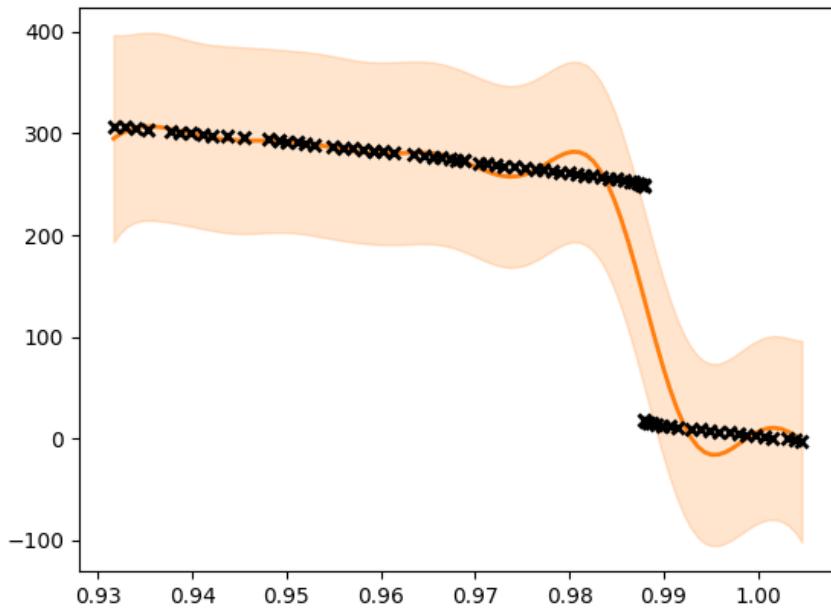


Figure 6.6: Example of poor bus stop detection for a bus stop with multiple modes. The bus arrives to a platform dedicated to the drop-off of passenger. This platform is not registered as a part of the bus stop by the system, which causes a large offset in the predicted arrival times. The system registers the bus at the bus stop once the bus drives past the main mode (the main platform) of the bus stop.

6.6 Future Work

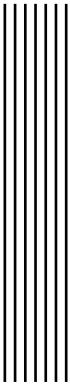
Future work on arrival time prediction (trajectory forecasting) should be focused on improving the arrival time prediction aggregation. The weighted mixture method should be implemented and tested as a next step. One could also experiment with the different approaches to model common stops, by adding 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 one day worth of data. The model should also be tested on 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 needing 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 FSM. 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 GPR, creating a MoGP. One 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. The errors range from 5 to ca. 19 seconds. 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](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.