Scalable Gaussian Process Assisted Positioning Using Cell Fingerprinting

David Bergh

Master's thesis
2024:E12

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

Scalable Gaussian Process Assisted Positioning Using Cell Fingerprinting

Author
David Bergh
da7700be-s@student.lu.se

Supervisor
Gabrielle Flood
gabrielle.flood@math.lth.se

Assistant Supervisor
Magnus Oskarsson
magnus.oskarsson@math.lth.se

Assistant Supervisor
Martin Larsson
martin.larsson@math.lth.se

Examiner
Alexandros Sopasakis
alexandros.sopasakis@math.lth.se

Abstract

GNSS is currently the most prominent method for outdoor positioning, providing an accuracy of approximately 5 metres in good conditions. However, for environments such as dense city areas or indoors, a GNSS signal reception might not be possible. An alternative method of positioning is the fingerprint-based method, which has been widely adopted for indoor positioning by sampling the signal strength of WiFi beacons. This method can also be deployed for outdoor positioning by using the signal strength of mobile network cells.

This thesis presents several fingerprint-based positioning models using Gaussian Processes (GPs) and Sparse Gaussian Processes (SGPs) to model the signal strength propagation. A combination of global and local optimisation is used to find the position of highest likelihood, which yields our predicted position.

Two baseline positioning methods were used to compare with our SGP models, namely the Weighted K-Nearest Neighbours (WKNN) algorithm and the Weighted Centroid Localisation (WCL) algorithm. When validating against the most extensive and realistic dataset, the WKNN achieved an approximately 20% smaller mean error than that of our proposed model. When validating with data not suited for WKNN our proposed model performed with approximately 33% better accuracy than that of WCL.

Keywords— positioning, fingerprint, cell, sparse Gaussian process

Acknowledgements

I would like to thank the team at Combain for offering me this thesis opportunity and for assisting with their knowledge within the field of positioning. Additionally, I would like to thank my supervisors Gabrielle Flood, Magnus Oskarsson and Martin Larsson for providing invaluable guidance and encouragement throughout this thesis. Lastly, I would like to thank my family and my girlfriend for their love and support.

Contents

Acronyms

CDF	Cumulative Distribution Function
CID	Cell Identity
GNSS	Global Navigational Satellite Systems
GP	Gaussian Process
KL	Kullback-Leibler
L-BFGS-B	Limited-memory Broyden-Fletcher-Goldfarb-Shanno Bounded
LAC	Location Area Code
LDPL	Log-Distance Path Loss
LOS	Line-of-Sight
LTE	Long Term Evolution
MCC	Mobile Country Code
MSE	Mean Square Error
MNC	Mobile Network Code
MPG	Mean Potential Gain
NLML	Negative Log Marginal Likelihood
NLOS	Non-Line-of-Sight
NLRL	Negative Log Relative Likelihood
PDF	Probability Density Function
PSC	Primary Scrambling Code
RSS	Received Signal Strength
SGP	Sparse Gaussian Process
WCL	Weighted Centroid Localisation
WKNN	Weighted K-Nearest Neighbours

Symbols and Notation

$\triangleq$	equals by definition
∇	gradient (with respect to $\mathbf{x}_*$)
$\lvert K_y \rvert$	determinant of matrix K_y
$\mathrm{cov}(\mathbf{x})$	Gaussian process covariance function
$\mathrm{cov}_q(\mathbf{x})$	sparse Gaussian process covariance function
Δd	distance between signal provider and receiver
d_0	reference distance
d	number of dimensions
$\mathrm{diag}(\cdot)$	diagonal vector of a matrix
$\mathbb{E}(\cdot)$	expected value of a random variable
$\mathbb{E}[\mathbf{f}_*]$	posterior predictive mean
F_V	variational lower bound
$f(\mathbf{x})$, $\mathbf{f}$	vector of latent function values
$\mathbf{f}_*$	posterior prediction function values
$\mathbf{f}_m$	inducing function values
$\mathbf{f}_r$	posterior function values from s unique cells at a given position $\mathbf{x}_*$
$\mathcal{GP}$	Gaussian process defined by mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$
h	height of cell regarding log-distance path loss model
I	identity matrix
$k(\mathbf{x}, \mathbf{x}')$	covariance (kernel) function evaluated for positions $\mathbf{x}$ and $\mathbf{x}'$
$\mathbf{k}_*$	covariance vector evaluated between positions X and a single position $\mathbf{x}_*$
K, $K(X, X)$	$n \times n$ input positions covariance matrix
K_*, $K(X, X_*)$	$n \times n_*$ covariance matrix between input and test positions
$K_{**}, K(X_*, X_*)$	$n_* \times n_*$ test positions covariance matrix
$K_{nm}, K(X, X_m)$	$n \times m$ covariance matrix between input and inducing positions
$K_{mn}, K(X_m, X)$	$m \times n$ covariance matrix between inducing positions and input
$K_{mm}, K(X_m, X_m)$	$m \times m$ inducing positions covariance matrix

$K_{*m}, K(X_*, X_m)$	$n_* \times m$ covariance matrix between test and inducing positions
K_y	$n \times n$ noisy input covariance matrix where $K_y = K + \sigma_n^2 I$
ℓ	length scale, hyperparameter for Gaussian covariance function
$\mathcal{L}(\mathbf{y}'_*, \mathbf{f}'_*)$	loss function over s observed y_* and predicted f_* values
L, L_B	Cholesky factor
L_{total}	path loss (dBm)
$\log(a)$	natural logarithm of a
$\boldsymbol{\mu}$	Gaussian process mean vector
m	number of inducing positions
$m(\mathbf{x})$	Gaussian process mean function
$m_q(\mathbf{x})$	sparse Gaussian process mean function
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	normal (Gaussian) distribution of mean $\boldsymbol{\mu}$ and covariance Σ
N	path loss exponent regarding log-distance path loss model
n	number of input positions
n_*	number of test positions
$\mathcal{O}(\cdot)$	big Oh, the limiting behaviour of a function $g(n)$ as $n \to \infty$
$\phi(\mathbf{f}_m)$	variational Gaussian distribution
$\phi^*(\mathbf{f}_m)$	variational Gaussian distribution which optimally approximates $p(\mathbf{f}_m \vert \mathbf{y})$
$PL(d_0)$	path loss at d_0 metres from the signal provider
$p(a\vert b)$	conditional probability density of stochastic variable a given b
$q(\mathbf{f}, \mathbf{f}_m)$	variational posterior over $\mathbf{f}$ and $\mathbf{f}_m$
σ_f	signal variance, hyperparameter for Gaussian covariance function
σ_n	noise variance (hyperparameter)
Σ	Gaussian process covariance matrix
s	number of observed RSS values for a scan
$\mathrm{Tr}(K)$	trace (sum of diagonal elements) of square matrix K
$\mathbb{V}(\cdot)$	variance of a random variable
$\mathbb{V}[\mathbf{f}_*]$	posterior predictive variance
X	$d \times n$ input positions
X_*	$d \times n_*$ test positions
X_m	$d \times m$ inducing positions
$\mathbf{x}_{cell}$	estimated cell position
$\mathbf{x}$	input position, vector of d coordinates
$\mathbf{x}'$	arbitrary input position, vector of d coordinates
$\mathbf{x}_*$	test input position, vector of d coordinates
$\mathbf{x}'_*$	initial optimal position (predicted)
$\mathbf{x}''_*$	final optimal position (predicted)
$\hat{\mathbf{x}}_*$	global minimum over loss function $\mathcal{L}(\cdot, \cdot)$
$\mathbf{x}_r$	reported position for measurements $\mathbf{y}_r$
$\mathbf{y}$	observed target vector for input positions X
$\mathbf{y}_r$	vector of s reported RSS values to s unique cells during a scan

1 Introduction

1.1 Background

Although Global Navigational Satellite Systems (GNSS) offer reliable and accurate positioning for open space areas, one potential problem is the fact that the received signal is prone to disruptions. For accurate positioning, a direct Line-Of-Sight (LOS) to four or more satellites is required [1], which is not always possible to achieve. Some environments, such as dense city areas, provide Non-Line-Of-Sight (NLOS) signals at best, meaning that the received signals have been distorted [1]. Other environments may exhibit no signal reception and therefore no positioning. Examples of such environments are indoor, underground, urban canyons and dense forests [2]. For the environments that provide a precise GNSS positioning the issue arises of GNSS receivers exhibiting a large power consumption [3].

An alternative to GNSS positioning is fingerprint-based positioning. By recording the particular characteristics at several locations we can construct a model that predicts an estimated position when given a set of characteristics. One such characteristic that has been widely used for fingerprint positioning refers to the signals sent from cellular network providers to be received by mobile devices. Specifically, a unit of measurement called Received Signal Strength (RSS), which is measured in dBm. Cell towers, referred to as cells, transmit and receive signals from nearby mobile devices. As such a mobile device can record a position with the help of GNSS and also include the RSS from nearby cells. When sufficient amounts of data have been gathered it is possible to create a predicted RSS distribution over an area that can be used for fingerprint-based positioning.

A common method for modelling RSS distributions involves representing each distribution by a Gaussian Process (GP) [4]–[6]. A process that is similar to the GP is the Sparse Gaussian Process (SGP), which approximates the GP but is more efficient regarding memory demands and computational complexity. Given an RSS for a certain cell, we can use a GP to determine the likelihood of measuring that RSS at any given position. If provided with a set of multiple RSS values to different cells, the combined likelihood can also be determined for any given position. The problem then remains to find the position that offers the maximum likelihood of assuming the given set of RSS values, which for example can be done iteratively or through other approximations. If it is possible to create a method that consistently and effectively finds the optimal position according to the model, we have a solid foundation for providing an alternative to GNSS positioning. This model can then be compared with other methods of fingerprint positioning for comparison. Common examples of such algorithms are the Weighted Centroid Localisation (WCL) and the Weighted K-Nearest Neighbours (WKNN).

Fingerprint positioning solves both of the issues arising from using GNSS positioning because of the reliability of the cellular network and the low power consumption of signalling. Another advantage of using a fingerprinting model based on RSS values is that it is adaptable. The signal provider does not have to be a cell, and the receiver does not have to be any specific mobile device. The training phase simply requires the collection of RSS, cell ID and positional data to be able to predict the position of future similar devices.

While the accuracy of a fingerprint positioning model certainly is of interest, another relevant metric is scalability. It is expected of the model to be able to provide positioning quickly and scale appropriately as the amount of collected data grows. In this respect, scalability specifically refers to the scaling of the computational load as the amount of training data increases.

1.2 Goal

The goal of this thesis is to develop a positioning model using cell fingerprinting with the help of GPs. The primary metrics of importance for the developed model will be accuracy and scalability. Accuracy will be measured by the mean error when predicting positions for a validation data set and scalability will be measured as computational complexity and memory usage. From these definitions the following questions are to be answered:

- Which GP-based model offers the highest accuracy?
- How does the optimal model perform compared to common methods of positioning such as WCL or WKNN?
- What is the tradeoff between accuracy and scalability?

1.3 Contribution

The work is expected to contribute with the following:

- A new method, or modification of existing methods, providing scalable fingerprint-based positioning using GPs.
- Performance comparisons of multiple variants of GP-based positioning.

2 Theory

2.1 Cell Fingerprinting

Cell fingerprinting consists of two phases: the offline training phase and the online phase. The offline training phase consists of devices providing their current position and characteristics associated with that position. The collected data is then used to build a model capable of predicting new positions given a set of characteristics. The characteristic that traditionally is recorded for a cell-based fingerprinting model is the RSS value from a cell. The online phase uses the previously built model to compute an estimated position given a new set of reported characteristics, in our case a set of RSS measurements. We will now continue by defining a GP and how it can be used to provide a predicted position.

2.2 Gaussian Processes

In this thesis an uppercase letter will be used to signify a matrix and a letter with bold type will signify a vector. For any further confusion regarding notation, we refer to the list of symbols and notations on page vi.

A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [7]. It defines a function value $f(\mathbf{x})$ that has a Gaussian (normal) distribution for any variable $\mathbf{x} \in \mathbb{R}^d$, where d defines the number of dimensions. Let $\mathbf{x}'$ define a secondary arbitrary position of the same shape as $\mathbf{x}$. By defining the mean and covariance function over positions $\mathbf{x}$ and $\mathbf{x}'$ of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{2.1}$$

the GP is expressed as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{2.2}$$

We will initially assume GPs with a zero mean function, but non-fixed mean functions will also be considered further on. Covariance functions are positively definite and describe how closely correlated two variables are. The covariance function that will be used throughout this thesis is the Gaussian kernel, defined by

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(\mathbf{x} - \mathbf{x}')^\top(\mathbf{x} - \mathbf{x}')\right). \tag{2.3}$$

This is a widely used kernel, where the two parameters ℓ and σ_f affect the smoothness and vertical variation of a posterior distribution, respectively. An important feature of the Gaussian kernel is that it has a convenient gradient with respect to either $\mathbf{x}$ or $\mathbf{x}'$, which will be useful further on.

2.2.1 Posterior Distribution

Let $X \in \mathbb{R}^{d \times n}$ denote the matrix of training positions and let $\mathbf{x}_i \in \mathbb{R}^d$ denote the ith training input. Thus, X consists of n columns of training positions. For positioning along a 2D plane, X is of size $\mathbb{R}^{2 \times n}$, consisting of latitudinal and longitudinal values. Furthermore, K defines a matrix of covariances according to $K_{ij} = k(\mathbf{x}_i, \mathbf{x}'_j)$ from (2.3). To realistically model for the noisy data a normally distributed noise ϵ is added such that the target value y is defined as $y = f(\mathbf{x}) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_n^2 I)$. The noisy prior is therefore $\text{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I$. We introduce a new matrix X_* of size $\mathbb{R}^{d \times n_*}$, denoting n_* test positions, and we denote the posterior predictions at positions X_* as $\mathbf{f}_*$. The joint distribution of the observed target values $\mathbf{y}$ and function values $\mathbf{f}_*$ is given by

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \tag{2.4}$$

For notational simplicity, let $K_y = K(X, X) + \sigma_n^2 I$ of size $\mathbb{R}^{n \times n}$, $K_* = K(X, X_*)$ of size $\mathbb{R}^{n \times n_*}$, $K_*^\top = K(X_*, X)$ of size $\mathbb{R}^{n_* \times n}$ and $K_{**} = K(X_*, X_*)$ of size $\mathbb{R}^{n_* \times n_*}$. The joint distribution from (2.4) is reformulated as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_y & K_* \\ K_*^\top & K_{**} \end{bmatrix}\right). \tag{2.5}$$

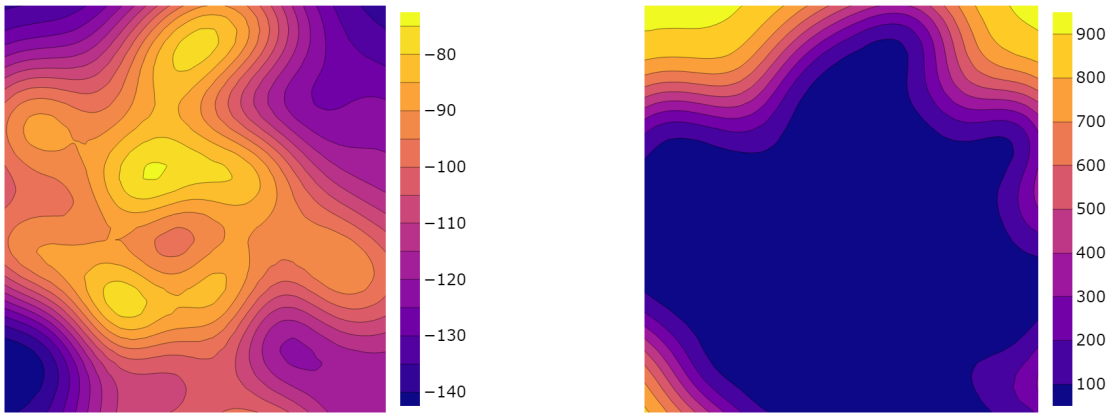
Let $\mathbf{f}_* \triangleq \mathbf{f}_* | X, \mathbf{y}, X_*$ unless stated otherwise. The predictive distribution yields

$$\mathbf{f}_* \sim \mathcal{N}(\mathbb{E}[\mathbf{f}_*], \text{cov}(\mathbf{f}_*)), \tag{2.6}$$
$$\mathbb{E}[\mathbf{f}_*] = K_*^\top K_y^{-1} \mathbf{y}, \tag{2.7}$$
$$\text{cov}(\mathbf{f}_*) = K_{**} - K_*^\top K_y^{-1} K_*, \tag{2.8}$$

where $\mathbb{E}[\mathbf{f}_*]$ denotes the posterior mean and $\text{cov}(\mathbf{f}_*)$ denotes the posterior covariance. Note that for the predictive distribution, the noise parameter needs to be added as the measured RSS values $\mathbf{y}_*$ at unknown positions are also assumed to be noisy. Also, note that the diagonal elements of the covariance matrix are the pointwise variances at positions X_*. The variance for a predictive distribution is as such $\mathbb{V}[\mathbf{f}_*] = \text{diag}(\text{cov}(\mathbf{f}_*) + \sigma_n^2 I)$. This predictive distribution uses a zero mean function. For the use case of predicting RSS values which range from $[-140, -44]$, a zero mean function would indicate that positions far away from our training data X assume a mean RSS of 0. This modelling contradicts how the signal strength generally fades the further away the receiver is from the cell. A more realistic modelling would be to use a fixed mean function $m(\mathbf{x})$ that is less than our lower bound for RSS values. The expected RSS value for far-away positions would then be modelled to converge to the mean function value. An even more realistic mean function would be to base the mean function on the Log-Distance Path Loss (LDPL) model [8, p. 214], or some other distribution that has the highest RSS value at a certain position and a decrease in RSS the further away the mobile device is. Whichever mean function $m(\mathbf{x})$ is used, the predictive mean can be incorporated into (2.7). Including an explicit mean function and the noise variance σ_n^2 results in the predictive posterior

$$\mathbb{E}[\mathbf{f}_*] = \mathbf{m}(X_*) + K_*^\top K_y^{-1}(\mathbf{y} - \mathbf{m}(X)), \tag{2.9}$$
$$\mathbb{V}[\mathbf{f}_*] = \text{diag}(K_{**} - K_*^\top K_y^{-1} K_* + \sigma_n^2 I). \tag{2.10}$$

An example of the resulting posterior distribution is presented in Figure 2.1.



(a) Posterior mean.

(b) Posterior variance.

Figure 2.1: Two contour plots displaying the mean RSS and the variance for a single cell over an area of 1000×1000 metres.

2.2.2 Hyperparameter Optimisation

The goal of hyperparameter optimisation is to find the set of hyperparameters ℓ, σ_f and σ_n which model the training data optimally. In other words, we want to maximise the marginal likelihood, defined as the probability of the observed data given our current model. The marginal likelihood $p(\mathbf{y}|X)$ is the integral of the likelihood $p(\mathbf{y}|\mathbf{f}, X)$ times the prior $p(\mathbf{f}|X)$, expressed as

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}. \tag{2.11}$$

Rasmussen [7] further derives the log marginal likelihood as

$$\log(p(\mathbf{y}|X)) = -\frac{1}{2}(\mathbf{y} - \mathbf{m}(X))^\top K_y^{-1}(\mathbf{y} - \mathbf{m}(X)) - \frac{1}{2}\log|K_y| - \frac{n}{2}\log(2\pi). \tag{2.12}$$

The goal is now to find the hyperparameters ℓ, σ_f and σ_n which maximise the marginal likelihood. As it is common for optimisers to optimise for the minimum instead of the maximum we reformulate the problem to minimise the Negative Log Marginal Likelihood (NLML). The problem to solve is therefore formulated

$$\underset{\ell,\sigma_f,\sigma_n}{\arg\min} -\log(p(\mathbf{y}|X)). \tag{2.13}$$

This minimisation is very demanding as it requires computing the matrix inversion K_y^{-1} and the determinants $|K_y|$. As the NLML will be computed frequently it is crucial that it is done quickly. Algorithm 2.1 defined in [7] displays an efficient and numerically more stable approach for computing the log marginal likelihood, which can simply be converted to the NLML. Note that for each iteration of running the algorithm the Cholesky decomposition of K_y is computed. This operation dominates the total computational complexity as it is $n^3/6$ [7].

2.3 Sparse Gaussian Processes

Computation of the inverse of K_y scales as $\mathcal{O}(n^3)$ while the memory required scales as $\mathcal{O}(n^2)$, which becomes intractable for larger datasets. The purpose of sparse GPs is to reduce the computational complexity as well as memory usage. The type of SGPs that will be used in this thesis is based on selecting m inducing data points (where $m < n$) which model the function space well enough. Using such a method reduces the time complexity to $\mathcal{O}(m^2 n)$ [7]. Note that the following demonstrated formulas and derivations are either directly from the work by Titsias [9], or from other sources based on the work by Titsias [9].

2.3.1 Parameter Optimisation

We assume $\mathbf{f}_m$ to be m inducing function values at positions X_m. These positions are variable, as are the hyperparameters ℓ, σ_f and σ_n in the pursuit of modelling the function space optimally. The goal is to select these variables such that they are as similar as possible to the posterior GP. We achieve this by minimising the Kullback-Leibler (KL) divergence between the true posterior distribution $p(\mathbf{f}, \mathbf{f}_m|\mathbf{y})$ and the approximate distribution $q(\mathbf{f}, \mathbf{f}_m)$. For clarification, the KL divergence defines the relative entropy, or distance, between two distributions [7].

We define $K_{mm} = K(X_m, X_m)$ as the covariance matrix for the inducing positions X_m. We then define $K_{nm} = K(X, X_m)$ as the covariance between the training positions X and inducing positions X_m. Expectedly, the transpose is defined as $K_{mn} = K(X_m, X) = K_{nm}^\top$.

The minimisation of the KL divergence is equivalent to the maximisation of the variational lower bound F_V, defined by

$$F_V(X_m, \phi) = \int p(\mathbf{f}|\mathbf{f}_m)\phi(\mathbf{f}_m) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}_m)}{\phi(\mathbf{f}_m)} d\mathbf{f} d\mathbf{f}_m. \tag{2.14}$$

Here, $\phi(\mathbf{f}_m)$ denotes a Gaussian distribution dependent on the mean vector $\boldsymbol{\mu}$ and covariance matrix Σ, which are chosen to approximate $p(\mathbf{f}_m|\mathbf{y})$. Following [9], by differentiating (2.14) with regard to $\phi(\mathbf{f}_m)$, the optimal maximising distribution ϕ^* is derived as

$$\phi^*(\mathbf{f}_m) = \mathcal{N}(\mathbf{f}_m|\boldsymbol{\mu}, \Sigma), \tag{2.15}$$

where

$$\boldsymbol{\mu} = \sigma^{-2} K_{mm} \Lambda K_{mn} \mathbf{y}, \tag{2.16}$$
$$\Sigma = K_{mm} \Lambda K_{mm}, \tag{2.17}$$
$$\Lambda = (K_{mm} + \sigma^{-2} K_{mn} K_{nm})^{-1}. \tag{2.18}$$

With an expression for $\phi(\mathbf{f}_m)$ defined by $\boldsymbol{\mu}$ and Σ, we focus on selecting X_m as to maximise F_V. The variational lower bound (dependent on just X_m) is then

$$F_V(X_m) = \log(\mathcal{N}(\mathbf{y}|\mathbf{0}, Q + \sigma_n^2 I)) - \frac{1}{2\sigma_n^2} \operatorname{Tr}(\widetilde{K}), \tag{2.19}$$

where $Q = K_{nm} K_{mm}^{-1} K_{mn}$ approximates the true covariance matrix K and $\widetilde{K} = K - Q$ [9]. We define $A \triangleq L^{-1} K_{mn} \sigma_n^{-1}$ and $B \triangleq I + AA^\top$ to further simplify the expression, where $LL^\top = K_{mm}$ is the Cholesky decomposition for K_{mm}. Let $\mathbf{c} = L_B^{-1} A \mathbf{y} \sigma_n^{-1}$ where $L_B L_B^\top = B$ is the Cholesky

decomposition of B. Using these definitions and following the derivations by [10], the variational lower bound can be expressed as

$$F_V(X_m) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|B| - \frac{n}{2}\log(\sigma_n^2) - \frac{1}{2\sigma_n^2}\mathbf{y}^\top\mathbf{y} + \frac{1}{2}\mathbf{c}^\top\mathbf{c} - \frac{1}{2\sigma_n^2}\operatorname{Tr}(K) + \frac{1}{2}\operatorname{Tr}(AA^\top). \quad (2.20)$$

The computational complexity of (2.20) is $\mathcal{O}(m^2 n)$ for each iteration, from finding the Cholesky factor L of K_{mm}. We now have all the means necessary to compute the optimal parameters which maximises the variational lower bound F_V. To be clear, the parameters which initially will be optimised for are ℓ, σ_f, σ_n and X_m. It is also possible to instead optimise for a subset of these hyperparameters. After they have been selected, we can compute $\boldsymbol{\mu}$ and Σ using (2.16) and (2.17). Since we are using a Gaussian kernel the covariance matrices are differentiable, meaning that a gradient-based minimiser can be used for quicker convergence toward the optimal values.

2.3.2 Posterior Distribution

We now want to express the posterior distribution. The distribution of function values at the inducing positions X_m can be written as

$$q(\mathbf{f}_*) = \int p(\mathbf{f}_*|\mathbf{f}_m)\phi(\mathbf{f}_m)d\mathbf{f}_m. \tag{2.21}$$

Following this expression the predictive posterior distribution $p(\mathbf{f}_*)$ is given by the mean $m_q(\mathbf{f}_*)$ and covariance $\text{cov}_q(\mathbf{f}_*)$ as

$$p(\mathbf{f}_*) = \mathcal{N}(\mathbf{f}_*|m_q(\mathbf{f}_*), \text{cov}_q(\mathbf{f}_*)) \tag{2.22}$$
$$m_q(\mathbf{f}_*) = K_{*m}K_{mm}^{-1}\boldsymbol{\mu} \tag{2.23}$$
$$\text{cov}_q(\mathbf{f}_*) = K_{**} - K_{*m}K_{mm}^{-1}K_{*m}^\top + K_{*m}K_{mm}^{-1}\Sigma K_{mm}^{-1}K_{*m}^\top, \tag{2.24}$$

where $K_{*m} = K(X_*, X_m)$ defines the covariance between the test positions and the induced positions. We can reuse the previously mentioned definitions B, $\mathbf{c}$, L and L_B to reduce the number of computations. By adding the noise variance σ_n^2 and reusing the previous definitions, we get the following expressions for the predictive posterior expected value and variance:

$$\mathbb{E}[\mathbf{f}_*|\mathbf{f}_m] = K_{*m}L^{-\top}L_B^{-\top}\mathbf{c}, \tag{2.25}$$
$$\mathbb{V}[\mathbf{f}_*|\mathbf{f}_m] = \text{diag}(K_{**} - K_{*m}L^{-\top}(I - B^{-1})L^{-1}K_{*m}^\top + \sigma_n^2 I). \tag{2.26}$$

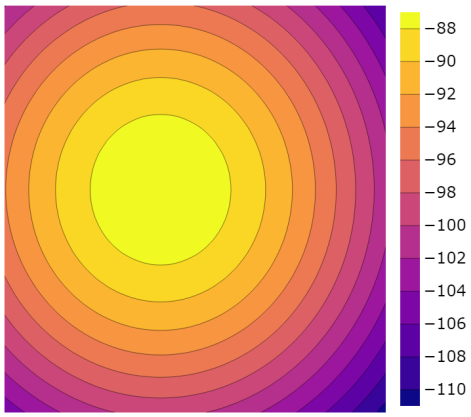
The precomputable matrices are defined

$$D_1 = L^{-\top}L_B^{-\top}\mathbf{c}, \tag{2.27}$$
$$D_2 = L^{-\top}(I - B^{-1})L^{-1}, \tag{2.28}$$

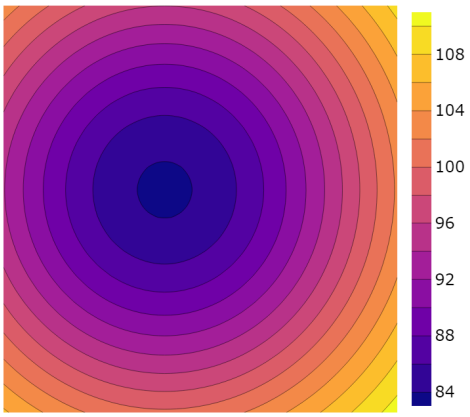
where $D_1 \in \mathbb{R}^{m\times 1}$ and $D_2 \in \mathbb{R}^{m\times m}$. For every unique cell, the variables which need to be stored for an SGP to offer quick posterior predictions are therefore the optimised hyperparameters (ℓ, σ_f, σ_n) and the matrices D_1, D_2 and X_m. Using the new notation, (2.25) and (2.26) can be expressed as

$$\mathbb{E}[\mathbf{f}_*|\mathbf{f}_m] = K_{*m}D_1 \tag{2.29}$$
$$\mathbb{V}[\mathbf{f}_*|\mathbf{f}_m] = \text{diag}((\sigma_f^2 + \sigma_n^2)I) - \text{diag}(K_{*m}D_2K_{*m}^\top). \tag{2.30}$$

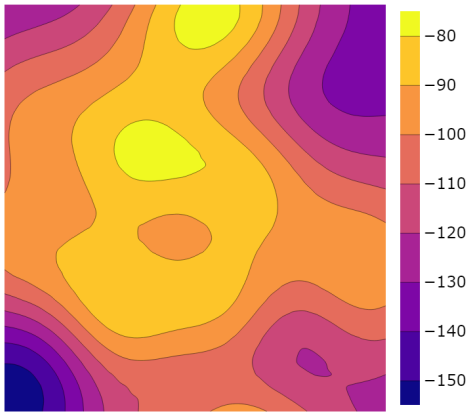
Using these definitions, the SGP was computed for the same cell as was used previously in Figure 2.1. The resulting posterior when using 1, 32 and 128 inducing points is presented in Figure 2.2. This figure gives an intuition for the effect of m.



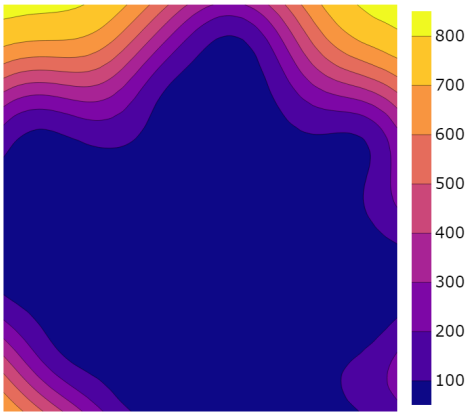
(a) Posterior mean for $m = 1$.



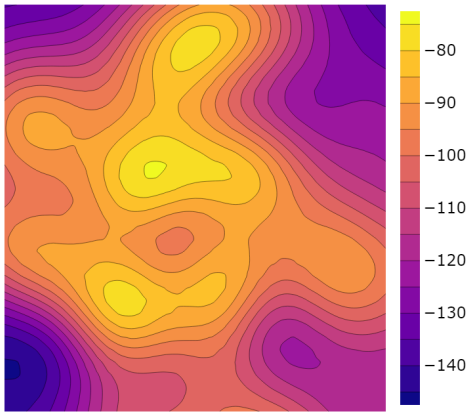
(b) Posterior variance for $m = 1$.



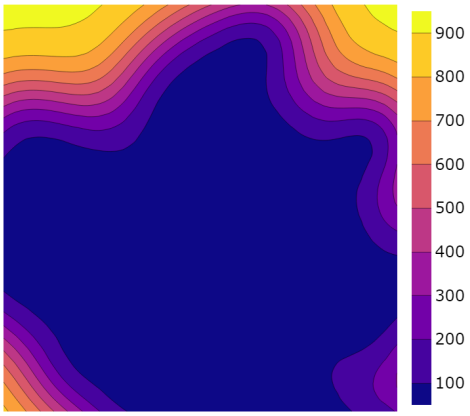
(c) Posterior mean for $m = 32$.



(d) Posterior variance for $m = 32$.



(e) Posterior mean for $m = 128$.



(f) Posterior variance for $m = 128$.

Figure 2.2: Six contour plots displaying the RSS distribution for a single cell over an area of 1000×1000 metres. The left column displays the mean RSS and the right column displays the variance. The top row uses 1 inducing point, the middle row uses 32 inducing points and the bottom row uses 128 inducing points.

2.4 Negative Log Relative Likelihood

Let $\mathbf{y}_r = [y_r^1, ..., y_r^s]$ denote a set of s reported target RSS values to s different cells from a single mobile device. Assume that for each of the s cells, a GP has been pre-trained such that a posterior can quickly be determined for any test position $\mathbf{x}_*$. Given an arbitrary test position $\mathbf{x}_*$ and the set of reported target values $\mathbf{y}_r$ we would like a total likelihood measure of each GP assuming the target values at the given position. For each GP, the posterior f_* at position $\mathbf{x}_*$ can be determined. We denote the set of posteriors for s unique cells at position $\mathbf{x}_*$ as $\mathbf{f}_r = [f_r^1, ..., f_r^s]$, as they relate to the reported values $\mathbf{y}_r$. By using the Probability Density Function (PDF) of the normal distribution the likelihood of the values $\mathbf{y}_r$ can be determined. Inserting the Gaussian distributions from our posterior into the PDF gives

$$f_Y(y_r, f_r) = \frac{1}{\sqrt{2\pi\,\mathbb{V}[f_r]}} \exp\left(-\frac{1}{2}\frac{(y_r - \mathbb{E}[f_r])^2}{\mathbb{V}[f_r]}\right). \tag{2.31}$$

Note that the PDF determines the relative likelihood for the distribution to assume the value y_*, not the probability. This relative likelihood might be useful when compared to other relative likelihoods. The PDF for the ith GP is as such $f_Y(y_r^i, f_r^i)$. Using Bayes' law we define the joint relative likelihood as the product of the relative likelihood of each cell assuming the RSS value at a given position $\mathbf{x}_*$. As the expression could become unstable we would like to use the log relative likelihood $\log(\prod_{i=1}^{s} f_Y(y_r^i, f_r^i))$ instead. By using the log-likelihood the expression can be reformulated into a summation. As the NLML uses the negative we would like to do so for the relative log-likelihood as well for consistency. We therefore define the Negative Log Relative Likelihood (NLRL) as

$$-\sum_{i=1}^{s} \log(f_Y(y_r^i, f_r^i)) = \sum_{i=1}^{s} \frac{1}{2}\left(\log(2\pi\,\mathbb{V}[f_r^i]) + \frac{(y_r^i - \mathbb{E}[f_r^i])^2}{\mathbb{V}[f_r^i]}\right). \tag{2.32}$$

We now have an expression for how well the model fits the validation data relative to other models. A model that has a greater likelihood of assuming the given RSS values at the true position will give a smaller NLRL value. Note that the NLRL does not necessarily imply better modelling of the propagation or better conditions for finding a good prediction position.

2.5 Loss Function

We will now move on to the predictive properties given a posterior prediction, be it from a sparse or a regular GP. The goal of this section is to provide a loss function that describes the measure of likelihood given a position $\mathbf{x}_*$. In other words, by finding the minimum of the loss function we find the position $\hat{\mathbf{x}}_*$ which has the highest likelihood of assuming the measured values $\mathbf{y}_r$ according to our model. Finding the position $\hat{\mathbf{x}}_*$ will be done in Section 2.6, we must first describe the loss function.

As stated in Section 2.4, the NLRL describes the relative likelihood of a set of posteriors $\mathbf{f}_r$ assuming RSS values $\mathbf{y}_r$ for s cells. The position $\hat{\mathbf{x}}_*$ of maximum relative likelihood is then found by minimising the NLRL and can be defined as

$$\hat{\mathbf{x}}_* = \arg\min_{\mathbf{x}_*} \sum_{i=1}^{s} \frac{1}{2}\left(\log(2\pi\,\mathbb{V}[f_r^i]) + \frac{(y_r^i - \mathbb{E}[f_r^i])^2}{\mathbb{V}[f_r^i]}\right). \tag{2.33}$$

Note that the constants $\frac{1}{2}$ and 2π can be dismissed when searching for the minimum or maximum. We therefore define the loss function $\mathcal{L}(\mathbf{y}_r, \mathbf{f}_r)$ as

$$\mathcal{L}(\mathbf{y}_r, \mathbf{f}_r) \triangleq \sum_{i=1}^{s} \left(\log(\mathbb{V}[f_r^i]) + \frac{(y_r^i - \mathbb{E}[f_r^i])^2}{\mathbb{V}[f_r^i]} \right). \tag{2.34}$$

The solution $\hat{\mathbf{x}}_*$ is then

$$\hat{\mathbf{x}}_* = \arg\min_{\mathbf{x}_*} \mathcal{L}(\mathbf{y}_r, \mathbf{f}_r). \tag{2.35}$$

Figure 2.3 displays an example of the loss function values that were computed using (2.34) over an area of approximately 1100×1300 metres. This contour plot gives an intuition for what the search space might look like.

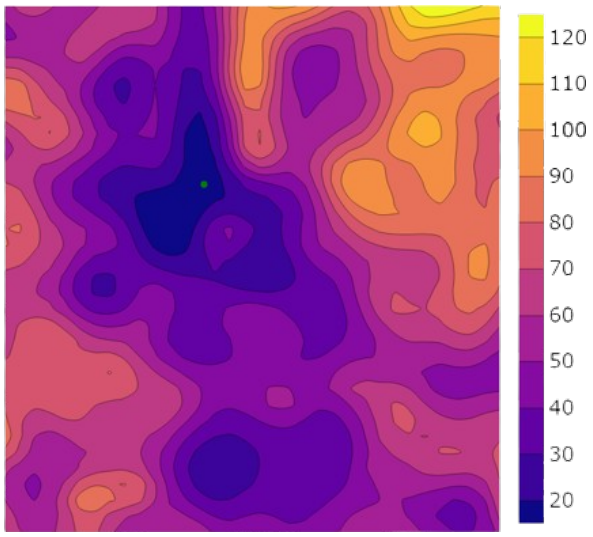


Figure 2.3: Displays the loss function evaluated over an area of 1100×1300 metres for three cells where the colour indicates the loss function value. The measured RSS values were $\mathbf{y}_r = [-70, -78, -73]$ and the green dot represents the true position from which the measurements were made.

2.6 Maximum Likelihood Estimation

The purpose of the loss function minimiser is to find a global minimum $\hat{\mathbf{x}}_*$ of the loss function $\mathcal{L}(\mathbf{y}_r, \mathbf{f}_r)$ according to (2.35). There are several ways to do so, either through iterative methods or by approximating the loss function such that the positions where the gradient $\nabla\mathcal{L}(\mathbf{y}_r, \mathbf{f}_r) = 0$ are analytically solvable. The method used for this thesis uses both a global minimiser and a local minimiser. The global minimiser used throughout this thesis is a grid search, partly because of its simplicity and partly because of its parallelizability. Grid search requires a specified precision and bounds to search in, we therefore start by defining how the initial bounds are selected.

2.6.1 Bounds Selection

For the initial bounds, we use a very simplistic but effective approach. For each set of training positions X for a certain cell, the 80% confidence interval is computed and selected as bounds for that cell. That is, the latitudinal bounds are selected as the 10th percentile and the 90th percentile of the sorted latitudinal values. The same is done for the longitudinal values. Given s cells to predict for the grid search, we select the bounds of the search space as the minimum and maximum bounds

of all of the affected cells. Note that the selected percentile can be varied and examined further but is arbitrarily chosen as 80%.

For each cell with input data X, there exists a median center, selected as the median of the latitudinal values and the median of the longitudinal values. For the case where the true position is surrounded by the median positions of each cell, we only need the median position as bound for each cell to cover the true position. Another viable solution for bounds selection could therefore be relative to the number of RSS values included in a scan, but this will not be examined further in this thesis.

2.6.2 Global Minimiser

Provided the previously defined bounds, the grid search algorithm can now be defined. Let $a_1 \leq x_1 \leq b_1$ and $a_2 \leq x_2 \leq b_2$ define the bounds for horizontal and vertical positions. Note that the bounds will be defined for positions in metres rather than coordinates to provide an equidistant grid. Given a precision of p metres, new sets $\mathbf{u}$ and $\mathbf{v}$ are constructed which will consist of the possible horizontal and vertical values, respectively. The set $\mathbf{u}$ is constructed such that $u_1 = a_1$ and any $u_{i+1} = u_i + p$ where $u_i \leq b_1$ for all $u_i \in \mathbf{u}$. Similarly, $v_1 = a_2$ and any $v_{i+1} = v_i + p$ where $v_i \leq b_2$ for all $v_i \in \mathbf{v}$. Let the horizontal and vertical values for a test position $\mathbf{x}_*$ be represented by x_*^1 and x_*^2, respectively, such that $\mathbf{x}_* = [x_*^1,\ x_*^2]$. Given a loss function $\mathcal{L}$, a set of observed RSS values $\mathbf{y}_r$ and Gaussian distributions $\mathbf{f}_r$ at position $\mathbf{x}_*$, grid search chooses an initial solution $\mathbf{x}'_*$ as

$$\mathbf{x}'_* = \underset{x_*^1 \in \mathbf{u}, x_*^2 \in \mathbf{v}}{\arg\min}\ \mathcal{L}(\mathbf{y}_r, \mathbf{f}_r | \mathbf{x}_*). \tag{2.36}$$

2.6.3 Local Minimiser

Given an initial solution $\mathbf{x'}_*$ we want to find a local minimum $\mathbf{x}''_*$ which is closer to the global minimum $\hat{\mathbf{x}}_*$. The goal is simply to converge to a local minimum with the help of a faster, but local minimiser. The local minimiser used is a bounded version of the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm denoted as L-BFGS-B [11]. The first reason for choosing L-BFGS-B is that it uses the Jacobian and an approximation of the Hessian for faster convergence. The second reason for using it is that it is included in the `scipy.optimize` package. We will simply pass the appropriate arguments regarding the minimisation objective into the minimisation function provided by `scipy`. L-BFGS-B works best when provided with a definition of the gradient, which is used to effectively descend towards a local minimum. This section will therefore derive an expression for the gradient with respect to $\mathbf{x}_*$ of the loss function. Consequently, the required expressions for the gradients of posteriors provided by GPs will be derived.

The gradient with respect to $\mathbf{x}_*$ of the sum of loss functions for s posteriors and measured RSS values is

$$\nabla \mathcal{L}(\mathbf{y}_r, \mathbf{f}_r) = \sum_{i=1}^{s} \nabla \left(\log(\mathbb{V}[f_r^i]) + \frac{(y_r^i - \mathbb{E}[f_r^i])^2}{\mathbb{V}[f_r^i]} \right). \tag{2.37}$$

Given a predictive posterior f_*^i and a function value y_*^i the resulting loss function gradient for a single GP is expressed as

$$\nabla \left(\log(\mathbb{V}[f_r^i]) + \frac{(y_r^i - \mathbb{E}[f_r^i])^2}{\mathbb{V}[f_r^i]} \right) = \frac{\nabla\, \mathbb{V}[f_r^i]}{\mathbb{V}[f_r^i]} + \frac{\mathbb{E}[f_r^i] - y_r^i}{\mathbb{V}[f_r^i]^2} \left(2\, \mathbb{V}[f_r^i] \nabla [\mathbb{E}[f_r^i]] - (\mathbb{E}[f_r^i] - y_r^i) \nabla\, \mathbb{V}[f_r^i] \right). \tag{2.38}$$

2.6.4 Gaussian Process Gradient

The goal of this section is to derive expressions for the gradient of the posterior mean and covariance of a regular GP. These expressions can then be inserted into (2.38) to get the gradient of the loss function.

We initially derive an expression for the gradient of the covariance function between an input position $\mathbf{x}$ and a test position $\mathbf{x}_*$. The gradient with regard to $\mathbf{x}_*$ of the covariance is then defined as

$$\nabla k(\mathbf{x}, \mathbf{x}_*) = \frac{\mathbf{x} - \mathbf{x}_*}{\ell^2} k(\mathbf{x}, \mathbf{x}_*). \tag{2.39}$$

From this expression, the gradient of the covariance vector for a single test position $\mathbf{x}_*$ is described by $\nabla \mathbf{k}_* = \nabla \mathbf{k}(X, \mathbf{x}_*)$. Let $\boldsymbol{\alpha} = K_y^{-1}(\mathbf{y} - \mathbf{m}(X))$. Assuming the use of a mean function m that is independent of $\mathbf{x}_*$, the gradient with respect to $\mathbf{x}_*$ can be derived from (2.9). This gives

$$\nabla \mathbb{E}[f_*] = \nabla[\mathbf{k}_*^\top]\boldsymbol{\alpha}, \tag{2.40}$$

where $\nabla \mathbb{E}[f_*]$ is of size $\mathbb{R}^d$. Note that the f_* used in this expression denotes a general posterior prediction and can be substituted with the posterior prediction f_r as well. The gradient with respect to $\mathbf{x}_*$ of the variance is derived from (2.10) as

$$\nabla \mathbb{V}[f_*] = -\operatorname{diag}(\nabla[\mathbf{k}_*^\top] K_y^{-1} \mathbf{k}_* + \mathbf{k}_*^\top K_y^{-1} \nabla[\mathbf{k}_*]), \tag{2.41}$$

where $\nabla \mathbb{V}[f_*]$ is also of size $\mathbb{R}^d$. Equations (2.40) and (2.41) can then be inserted into (2.38) to express the gradient of the loss function $\nabla \mathcal{L}(\mathbf{y}_r, \mathbf{f}_r)$.

2.6.5 Sparse Gaussian Process Gradient

This subsection will derive an expression for the gradient of the SGP. The expression is quite similar to that of the GP and uses the definition for the gradient of the covariance function described in (2.39). Remember the definition of the SGP posterior from (2.29) and (2.30)

$$\begin{aligned}
\mathbb{E}[\mathbf{f}_*|\mathbf{f}_m] &= K_{*m} D_1, \\
\mathbb{V}[\mathbf{f}_*|\mathbf{f}_m] &= \operatorname{diag}((\sigma_f^2 + \sigma_n^2) I) - \operatorname{diag}(K_{*m} D_2 K_{*m}^\top).
\end{aligned}$$

Let $\nabla \mathbf{k}_{m*} = \nabla \mathbf{k}(X_m, \mathbf{x}_*)$ denote the gradient of the covariance vector between the inducing positions and the test position $\mathbf{x}_*$. The gradient for the posterior consisting of a single position is then derived from (2.29) and (2.30) as

$$\nabla \mathbb{E}[f_*|\mathbf{f}_m] = \nabla[\mathbf{k}_{*m}] D_1, \tag{2.42}$$

$$\nabla \mathbb{V}[f_*|\mathbf{f}_m] = -\operatorname{diag}(\nabla[\mathbf{k}_{*m}] D_2 \mathbf{k}_{*m}^\top + \mathbf{k}_{*m} D_2 \nabla[\mathbf{k}_{*m}^\top]). \tag{2.43}$$

Inserting (2.42) and (2.43) into (2.38) gives the gradient of the loss function when using SGPs.

2.7 Mean Potential Gain

To evaluate the solution found by grid search followed by a local minimisation, we compare it with the solution found by local minimisation at the true position. If the latter achieves a solution of greater likelihood (smaller loss function value), there exists a better solution than the originally found solution. We denote the improvement by selecting the better solution as the potential gain, measured in metres. As the potential gain is calculated for several predictions during validation, we denote the mean potential gain as MPG. Note that the MPG does not signify an explicit potential for improvement, as this depends on how the loss function is modelled. For example, assume that a new and better solution was found by the local minimiser at the true position. There could still exist a position of even greater likelihood not yet discovered, which could either be much closer or further away from the true position. The MPG simply gives an estimate for the most general cases.

Algorithm 1 describes how the MPG is determined for k true positions X_{true} and predicted positions X_*. The other inputs are functions, where `loss()` denotes the loss function $\mathcal{L}()$. The distance function is denoted `distance()` and computes the distance between two positions. The `L-BFGS-B()` function returns the local minimum over the loss function when starting at a given position.

Algorithm 1: Psuedocode for determining the MPG.

input: X_{true}, X_*, loss(), distance(), L-BFGS-B()
total_potential $\leftarrow 0$
$k \leftarrow$ length(X_{true})
for $i \leftarrow 1 \rightarrow k$ **do**
  $\mathbf{x}_* \leftarrow X_*^i$
  $\mathbf{x}_{true} \leftarrow X_{true}^i$
  $\mathbf{x}_{true\_min} \leftarrow$ L-BFGS-B($\mathbf{x}_{true}$)
  if loss($\mathbf{x}_{true\_min}$) < loss($\mathbf{x}_*$) **then**
    error $\leftarrow$ distance($\mathbf{x}_*$, $\mathbf{x}_{true}$)
    error$_{new}$ $\leftarrow$ distance($\mathbf{x}_{true\_min}$, $\mathbf{x}_{true}$)
    total_potential $\leftarrow$ total_potential + error - error$_{new}$
return total_potential

2.8 Log-Distance Path Loss

The Log-Distance Path Loss (LDPL) model expresses the path loss L_{total} from the signal provider, defined by [8, p. 214] as

$$L_{total} = PL(d_0) + 10N \log_{10}(\Delta d/d_0) + \sigma_n^2, \tag{2.44}$$

where the variables are denoted as follows:

- L_{total}: path loss (dBm)
- $PL(d_0)$: path loss at d_0 metres from the signal provider (dBm)
- N: path loss distance exponent
- Δd: distance between signal provider and signal receiver (m)
- d_0: reference distance (m)
- σ_n^2: Gaussian random noise variable

For a simplified version we assume $d_0 = 1$ and ignore the noise σ_n^2. For free space areas, a typical value for the path loss distance exponent is 2.0, we will therefore assume $N = 2.0$ [12]. By making the previously mentioned changes and rearranging the expression we get the path loss

$$L_{total} = PL(1) + 20\log_{10}(\Delta d). \tag{2.45}$$

An expression is needed for the distance between the assumed cell position $\mathbf{x}_{cell}$ and the reported device position $\mathbf{x}$. For a more stable (and somewhat more realistic) modelling a height term h (metres) is introduced for the cell position while the mobile device is still assumed to be at ground level. Using simple geometry an expression for the distance Δd can be derived as

$$\Delta d = \sqrt{h^2 + (\mathbf{x}_{cell} - \mathbf{x})^\top (\mathbf{x}_{cell} - \mathbf{x})}. \tag{2.46}$$

Expanding (2.45) with (2.46) gives

$$L_{total} = PL(1) - 10\log_{10}(h^2 + (\mathbf{x}_{cell} - \mathbf{x})^\top (\mathbf{x}_{cell} - \mathbf{x})). \tag{2.47}$$

Given a set of training data, we can define a loss function as the square error of the predicted RSS over all training data X and $\mathbf{y}$ as

$$\mathcal{L}_{ldpl}(PL(1), \mathbf{x}_{cell}, h) = \sum_{i=1}^{n} \left(t_0 - 10\log_{10}(h^2 + (\mathbf{x}_{cell} - \mathbf{x}_i)^\top (\mathbf{x}_{cell} - \mathbf{x}_i)) - y_i\right)^2. \tag{2.48}$$

The optimising parameters are then found by the minimisation

$$\underset{PL(1), \mathbf{x}_{cell}, h}{\arg\min}\ \mathcal{L}_{ldpl}(PL(1), \mathbf{x}_{cell}, h). \tag{2.49}$$

The parameters to optimise for are as stated $PL(1)$, $\mathbf{x}_{cell}$ and h. Given a set of well-defined bounds and initial guesses, this should be no problem for a minimiser such as L-BFGS-B. A reasonable guess is that $\mathbf{x}_{cell}$ is within the bounds of X. It is also reasonable to assume the initial position $\mathbf{x}_{cell}$ as $\mathbf{x}_i$, for which the corresponding measured RSS y_i is the maximum measured RSS of $\mathbf{y}$. The true optimal location $\mathbf{x}_{cell}$ is then probably within 100 metres of the reported maximum $\mathbf{x}_i$. An acceptable initial value for $PL(1)$ would be y_i. A decent estimate of the height bounds of the cell would be $1 \leq h \leq 150$.

With a set of optimal parameters, the approximate propagation model could then function as a mean function for a Gaussian process. As this model is not constant for $\mathbf{x}_*$, either the gradient of the LDPL will have to be included or the minimisation for the optimal position $\hat{\mathbf{x}}$ will have to be done with approximative gradients if LDPL is used as a mean function.

2.9 Weighted Centroid Localisation

WCL is a positioning algorithm that is easy to implement and offers fast predictions. The method will be used to compare precision with other positioning models and is described as follows. Assume that we want to predict the position of a mobile device that has reported the RSS values $\mathbf{y}_r = [y_r^1, ..., y_r^s]$ to s unique cells. Let $X_{cell} \in \mathbb{R}^{d \times s}$ denote the position of the cells such that the position of the ith cell is $\mathbf{x}_{cell}^i$. The cell position $\mathbf{x}_{cell}^i$ is naively selected as the input training position $\mathbf{x}$ which corresponds to the highest reported RSS value y. With the cell positions selected, a simple centroid localisation method defines the predicted position $\mathbf{x}_*$ as the centroid of the cells according to

$$\mathbf{x}_* = \frac{1}{s}\sum_{i=1}^{s} \mathbf{x}_{cell}^i. \tag{2.50}$$

The assumption is made that from a position of maximum RSS the signal strength decreases somewhat uniformly. We can therefore improve the accuracy of the above method by taking RSS into account. In other words, greater RSS values should weigh more when computing the centroid. We therefore redefine the algorithm as

$$\mathbf{x}_* = \frac{1}{s}\sum_{i=1}^{s} w_i \mathbf{x}_{cell}^i, \tag{2.51}$$

where w_i denotes a weight for the ith cell. We select the weights such that greater RSS values imply a greater weight according to

$$w_i = \alpha \exp(y_r^i), \tag{2.52}$$

$$\alpha = 1/\sum_{i=1}^{s} \exp(y_r^i), \tag{2.53}$$

where y_r^i denotes the measured RSS value from cell i and α is a normalising constant such that the sum of weights is 1. Given a set of RSS values to unique cells we can now compute an estimated position using WCL.

2.10 WKNN

WKNN is a non-parametric method which can be used for positioning. It is based on the simpler k-nearest neighbours method but adds weights to the k-nearest neighbours based on how similar they are to the input. The similarity is determined by a covariance function, which for this thesis is the inverse of the Euclidean distance. For this thesis the SKlearn package is used to compute the predictions, specifically the `neighbors.KNeighborsRegressor` regressor [13]. The coordinate positions are still converted to a metre representation to scale the distances proportionately.

The problem with WKNN is that it requires training data in the form of scans, where several RSS measurements are related to a single position. During testing, we will be using data acquired from two databases, one of which will not be applicable to use for training of WKNN. For the data that works for WKNN, each training scan is converted into a vector where the ith element refers to the measured RSS to the ith cell. Since every scan does not necessarily contain the same cells, a joined vector is created with the cell IDs of all of the unique cells in the training data. Each scan is then merged into the joined vector of IDs, where cell IDs that are not included in a scan are assigned a default RSS value of -140.0. The input data of the regressor is a feature vector, whereas the target data is the metre representation of the coordinates.

Regarding complexity, during training of the WKNN `auto` will be used as a parameter for the `algorithm` option for all simulations. A choice can then be made regarding which algorithm the regressor will use. For $D > 15$ the brute force algorithm is used, where D denotes the number of features of the input. For our case, D is equal to the number of unique cells in the training data. For all our datasets $D > 15$ applies, we will therefore assume that the brute force algorithm is used for every application of the WKNN method. The computational complexity of brute force is $\mathcal{O}(Dn^2)$, as it involves computing the distance to all n training positions for each training position.

3 Methodology

3.1 Programming Language and Libraries

The programming language used throughout the thesis is Python. It is simple, flexible and can utilise several helpful libraries. Following are the imported non-standard libraries and what they are used for.

- `JAX` - Automatic differentiation of mathematical expressions.
- `NumPy` - Linear algebra computations.
- `Pandas` - Filtering, sorting and other handling of large datasets.
- `Plotly` - Producing graphs.
- `SciPy` - Minimisation algorithms.
- `SKlearn` - WKNN method.

3.2 Data

The training data is required to contain degrees latitude ($-90 < x_1 < 90$), degrees longitude ($-180 < x_2 < 180$), RSS ($-140 \leq y \leq -44$ dBm) and the ID of the cell which the signal was from. However, for validation of the models an additional attribute `scanId` is needed to link several RSS entries to a single scan. The entries that contain a scan ID can both be used for testing the models and also for constructing propagation models for each cell. The data used will be limited to Long Term Evolution (LTE) cells for simplicity, but the models are transferable to other connection types as long as they contain the previously mentioned data attributes. Two different databases will be used to provide data, with the first database being called Learning. From Learning the attributes latitude, longitude, RSS and cell ID can be extracted for each entry. The second database is called Slam. It contains the same attributes as Learning but additionally contains a scan ID attribute, meaning that all entries with the same scan ID were collected from the same mobile device at the same position and time. For example, there might be 8 entries with the same scan ID of 101. They therefore contain RSS values to 8 different cells but were collected by the same mobile device at the same time and place.

One problem with the measured RSS-ID pairs is that a unique cell ID is not included in the original data. Rather, every entry contains a Mobile Country Code (MCC), Mobile Network Code (MNC), Location Area Code (LAC) and Primary Scrambling Code (PSC). The MCC identifies a country with a 3-digit code. The MNC identifies a mobile network carrier such that given an MCC and MNC the mobile network carrier can be uniquely identified. Countries are divided into smaller areas defined by their LAC, which is a 16-bit integer. For each location area, there exists one or more cells. The PSC identifies cells within a location area, but not uniquely. A code collision occurs when two neighbouring cells use the same PSC. Network providers therefore plan the PSC assignment for each cell to minimise the number of code collisions. Generally, cells sharing the same PSC will therefore be at a distance from each other.

This can be used by clustering measured RSS values to create a unique cell ID. By clustering the positions (latitude, longitude) of a certain set of IDs (MCC, MNC, LAC, PSC), a unique cell ID can be given to this set of measurements. An estimated position can then be determined for each cell such that new entries can be grouped to the correct cell ID based on which cell is nearest. Currently, there already exists a database of recorded cells and their corresponding estimated positions which will be used to match new measurements with a cell ID.

3.2.1 Learning

In the dataset Learning, there are roughly 8 billion RSS and position pairs for LTE connections spanning all across the globe. The data has been collected by the use of both mobile phones and other tracking devices by using GNSS. As Learning data does not contain scan IDs, it will exclusively be used for training purposes. The data from Learning is generally more unreliable but covers a greater area than Slam.

3.2.2 Slam

Slam contains approximately 5.8 million GNSS positions and 25 million RSS entries that are related to these positions. The data is more recent and, most importantly, includes a `scan_id` attribute. This is the reason for the increased number of RSS entries compared to the number of scans, as for the 5.8 million scans the mean number of RSS connections is 25/5.8. Slam also contains positional data for every street in the city of Lund, Sweden, as provided by Guggenheimer in [14]. Slam is not as large as Learning, nevertheless, because of its coverage within Lund it is very valuable for both validation and constructing accurate RSS propagation models.

3.2.3 Data Preprocessing

The data from both Learning and Slam have outliers which provide inaccurate data. Usually, these are found and filtered out as bad data, either because they are not recent enough or because the devices that reported the data are not reliable. However, some devices tend to report pre-defined RSS values. An example of such behaviour is how there were > 2500 reports of RSS -140 while there were less than 25 reports of RSS -139 for the Bus validation set. It seems as though a signal was received from the cell, but the RSS could not be determined and a default value was instead reported. When examined further some RSS values were found to disproportionately be used as default. The choice was therefore made to include a filter that removes certain reported RSS values from "unreasonable" distributions.

The filter dismisses data of a certain RSS value y if the number of reported RSS values for either $y+1$ or $y-1$ are less than 25% of the number of reports for y. This is only considered if there are more than 10 reports for a certain RSS. RSS values that are commonly removed are -140, -100 and -44. Filtering the data results in more consistent distributions and is therefore applied to all data used for this thesis.

Another modification that is made to the data is normalisation. Originally the data is bounded by $-90.0 < x_1 < 90.0$, $-180.0 < x_2 < 180.0$. Instead, we would like to convert the input data to metres and normalise the input such that it is centered close to the origin. This makes computations more stable and offers comprehensible units of distance. Assume that two different positions $\mathbf{x}$ and $\mathbf{x}'$ are of the same latitude ($x_1 = x_1'$) but have a longitudinal difference of $\phi = |x_2 - x_2'|$ degrees. Note that the distance between them decreases the further away from the equator they are (assuming they maintain the longitudinal difference ϕ). This has to be taken into account when creating the normalised positions to get the correct distance in metres. The haversine formula is used for this purpose [15]. The RSS values of $\mathbf{y}$ and $\mathbf{y}_r$ are not normalised, but might be modelled separately by the use of custom mean functions depending on which model is used.

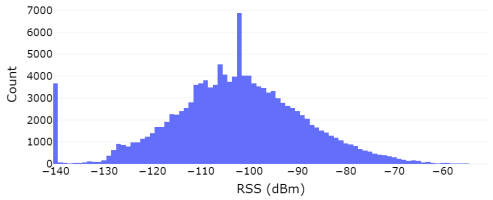
3.3 Dataset Groups

Two datasets will be required for each experiment: one training dataset and one validation dataset. Some models are intractable for training on larger datasets. However, larger datasets offer more reliable results. We will therefore divide the testing into three different groups of data which have different purposes.

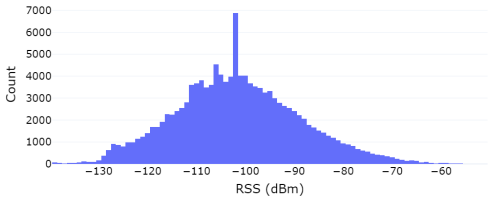
All of the datasets share one constraint, that they only contain positions within the bounds of Lund. These bounds are defined by the latitudinal limits [55.734, 55.681] and the longitudinal limits [13.144, 13.256]. These bounds will be shown as a black square in geographical scatter plots for clarification.

3.3.1 Group 72

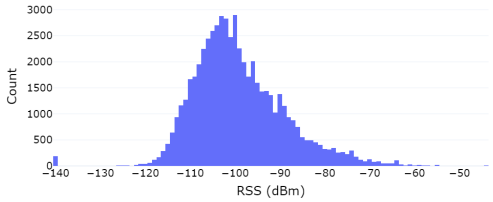
The first group of datasets is used to quickly validate approximately how well a model performs and is referred to as group 72. An initial dataset is extracted from Slam containing 868 678 entries. A secondary dataset is extracted from Learning of 500 000 entries. As only the Learning dataset has a unique ID for every cell these IDs are mapped onto the Slam dataset by comparing the reported MCC, MNC, LAC, and PSC. If there are two or more matches for these four identifiers the geographically closest cell is selected as the cell which provided the signal. Old data and entries of bad RSS values are then filtered out as described in Section 3.2.3. The results from applying the filtering of the disproportionate RSS distributions are shown in Figure 3.1.



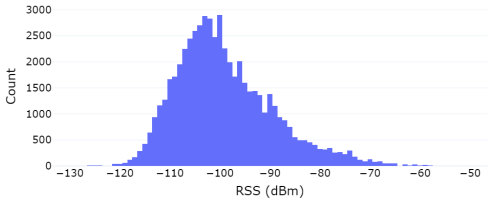
(a) RSS distribution of Slam data of group 72 before filtering.



(b) RSS distribution of Slam data of group 72 after filtering.



(c) RSS distribution of Learning data of group 72 before filtering.



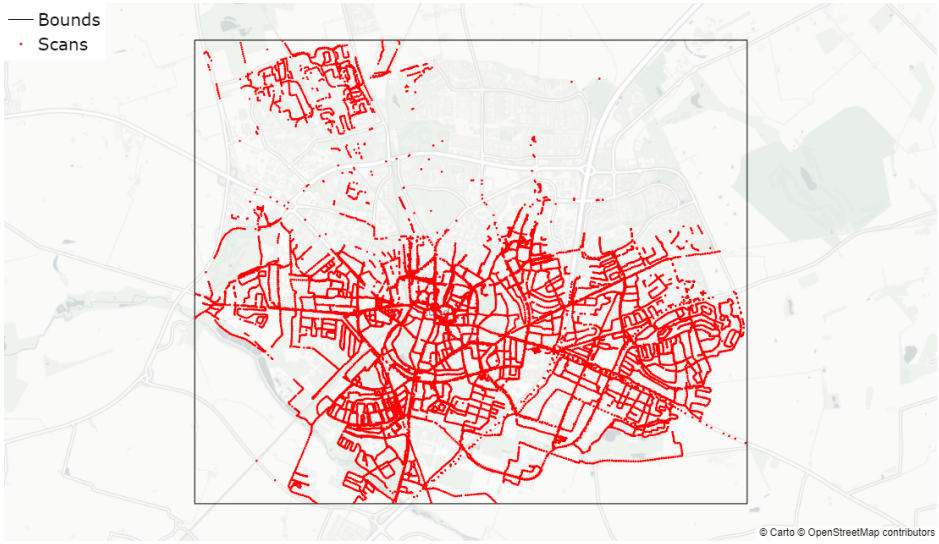
(d) RSS distribution of Learning data of group 72 after filtering.

Figure 3.1: The left column displays the original distribution of reported RSS values, whereas the right column displays the distribution after removing unreliable values. The top row shows the distribution for Slam data while the bottom shows for Learning data.

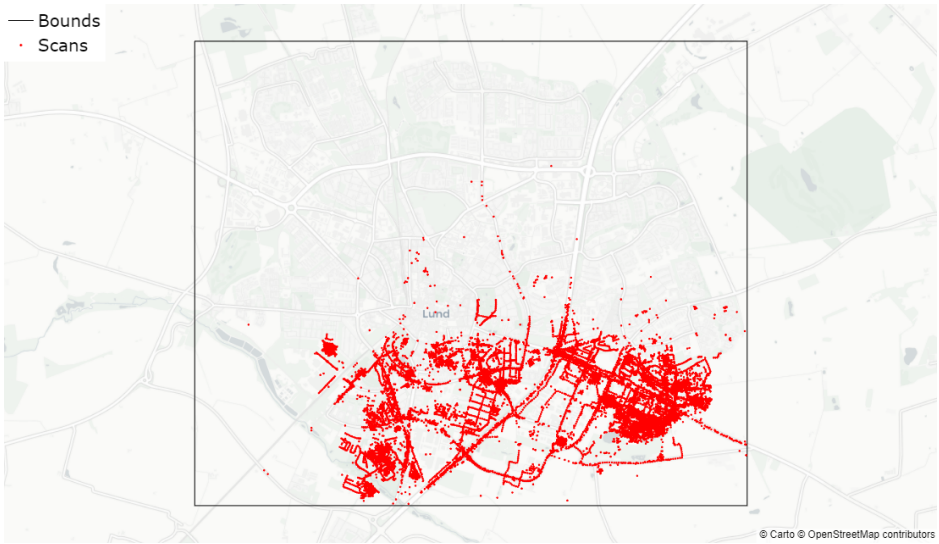
From these two datasets (Learning and Slam), we only keep the entries that relate to cells existing in both of the datasets. This is done to constrict the validation dataset to cells which are both in Slam and Learning. From the Slam dataset 72, scans are selected which contain the highest number of RSS entries per scan. These scans are removed from the Slam training data and made into a validation dataset. We will refer to the training datasets as training Slam 72 and training Learning 72. Specific information regarding group 72 is shown in Table 3.1, whereas the position of every scan can be seen in Figure 3.2.

Table 3.1: Displays number of entries, unique cells and scans for each dataset in group 72.

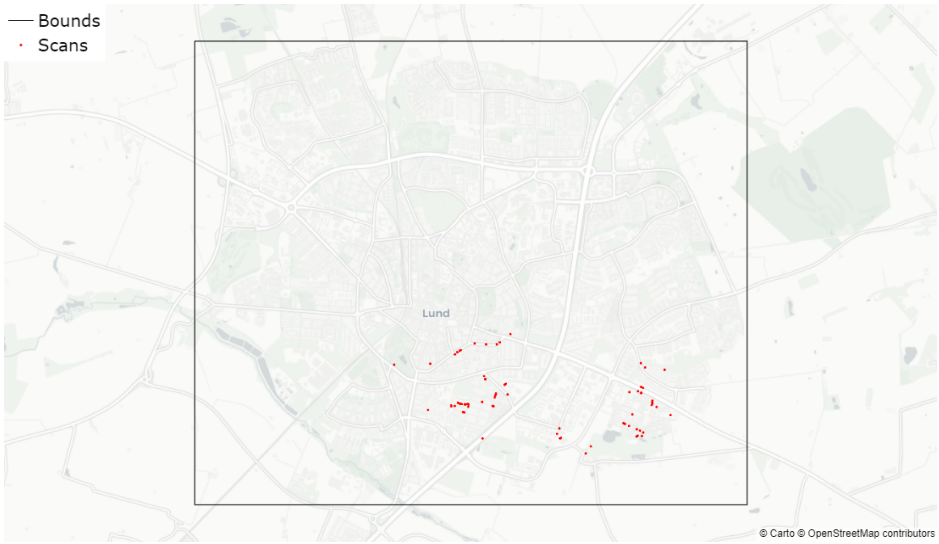
Dataset	Entries	Unique cells	Scans
Training Slam 72	130 669	110	37 359
Training Learning 72	57 086	73	57 086
Validation 72	1040	110	72



(a) Location of every scan for the Slam training dataset of group 72.



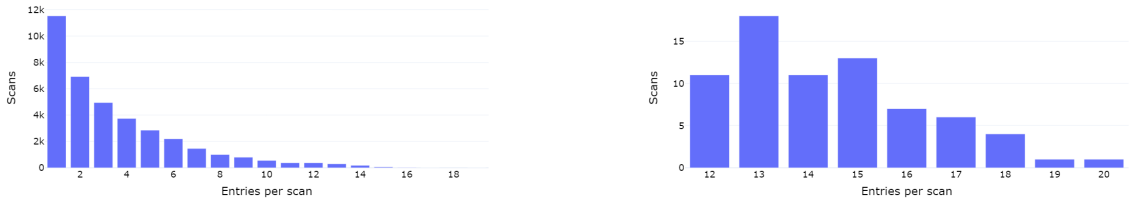
(b) Location of every entry for the Learning training dataset of group 72.



(c) Location of every scan for the validation dataset of group 72.

Figure 3.2: Geographical scatter plots where every dot signifies a scan with at least one RSS-id pair. For the Slam and validation dataset, these scans can contain several pairs.

The bar plots in Figure 3.3 display the distributions of the number of RSS-ID pairs per scan. Note that the validation dataset was selected to include the highest number of RSS-ID pairs.



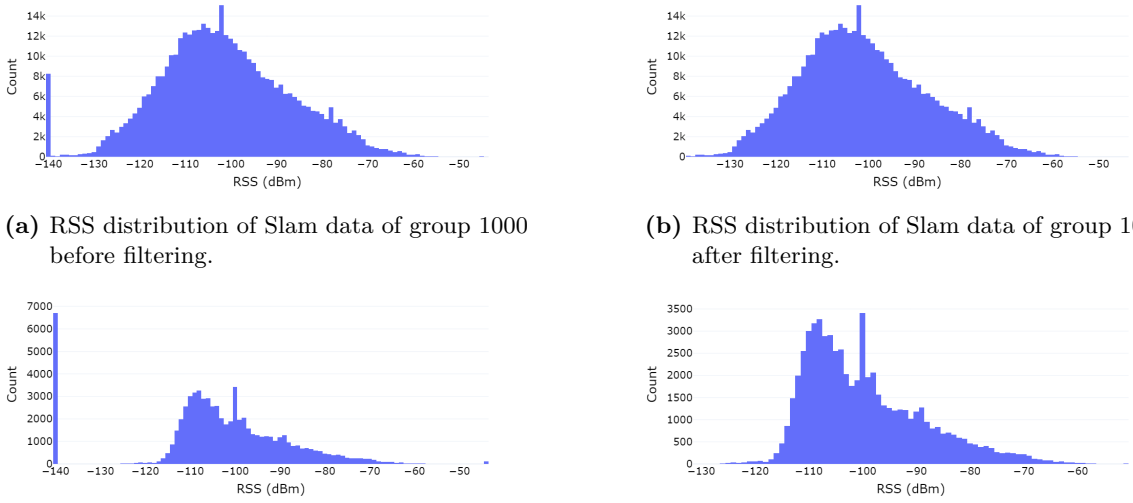
(a) Distribution of the number of RSS-ID pairs for the Slam training dataset of group 72

(b) Distribution of the number of RSS-ID pairs for the validation dataset of group 72.

Figure 3.3: Bar plots displaying the number of entries per scan for group 72. The x-axis represents the number of RSS-ID pairs, whereas the y-axis represents the number of scans with a certain number of RSS-ID pairs.

3.3.2 Group 1000

The validation dataset of the second group contains 1000 scans which are randomly selected from the preprocessed Slam dataset mentioned previously. A new set of 236 790 entries is then retrieved from Learning. These entries are selected by requesting all entries with a cell ID related to the scans from the validation dataset. The Learning data was also filtered to have been measured between the dates 1/1 2014 and 1/3 2023. The reason for the specific end date is because after that date some of the Slam data is inserted into Learning, and we would like to keep validation data from entering the training dataset. The filtering of abnormal RSS distributions is shown in Figure 3.4. The mean number of entries for each scan in the validation set is significantly decreased since the scans are not selected based on the number of entries per scan. This group will give some insight into how well the positioning would work for an average use case where there might not be as many entries per scan. This validation set is also larger, making the results more reliable for the general use case. Table 3.2 shows the specifics regarding the 3 datasets of group 1000.



(a) RSS distribution of Slam data of group 1000 before filtering.

(b) RSS distribution of Slam data of group 1000 after filtering.

(c) RSS distribution of Learning data of group 1000 before filtering.

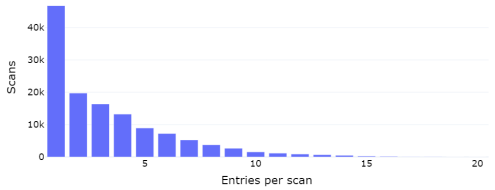
(d) RSS distribution of Learning data of group 1000 after filtering.

Figure 3.4: Displays the distribution of RSS values before and after filtering. The top row of plots displays the distribution for Slam data while the bottom displays for Learning data.

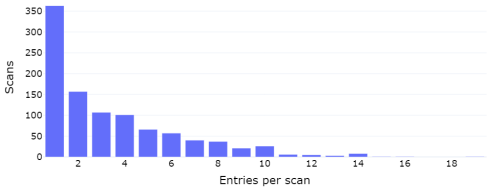
Table 3.2: Displays number of entries, unique cells and scans for each dataset in group 1000.

Dataset	Entries	Unique cells	Scans
Training Slam 1000	427 288	323	128 963
Training Learning 1000	62 979	138	62 979
Validation 1000	3426	323	1000

We include bar plots in Figure 3.6, which display the distribution of the number of RSS-ID pairs per scan. The geographical locations of the scans for the 3 datasets are shown in Figure 3.5.



(a) Distribution of the number of RSS-ID pairs for the Slam training dataset of group 1000

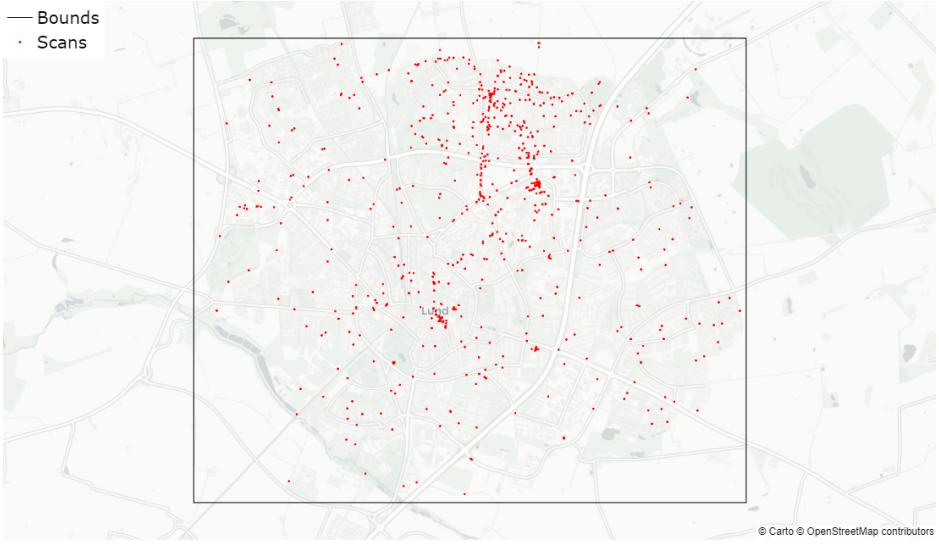


(b) Distribution of the number of RSS-ID pairs for the validation dataset of group 1000.

Figure 3.6: Bar plots displaying the number of entries per scan for group 1000. The x-axis represents the number of RSS-ID pairs, whereas the y-axis represents the number of scans with a certain number of RSS-ID pairs.

(a) Location of every scan for the Slam training dataset of group 1000.

(b) Location of every entry for the Learning training dataset of group 1000.

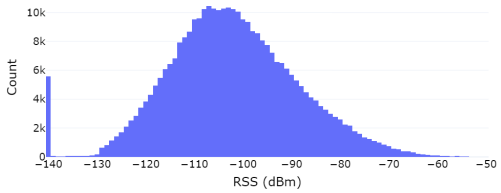


(c) Location of every scan for the validation dataset of group 1000.

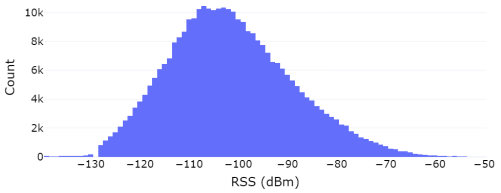
Figure 3.5: Geographical scatter plots where every dot signifies a scan with at least one RSS-id pair. For the Slam and validation dataset, these scans can contain several pairs.

3.3.3 Group Bus

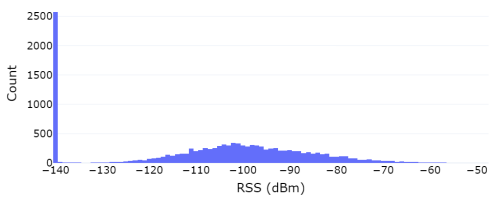
The third and final group is acquired from the work of Guggenheimer [14]. It contains one training dataset and one validation dataset which have both originally been retrieved from the Slam database. We will refer to this group as group Bus. The validation dataset is named the Bus dataset as it was collected while travelling by bus. The RSS distributions before and after the filtering of these datasets are shown in Figure 3.7. The sizes of the datasets are listed in Table 3.3. Maps displaying the locations of the scans are displayed in Figure 3.8.



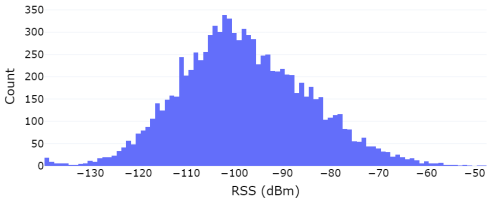
(a) RSS distribution of training data of group Bus before filtering.



(b) RSS distribution of training data of group Bus after filtering.



(c) RSS distribution of validation data of group Bus before filtering.



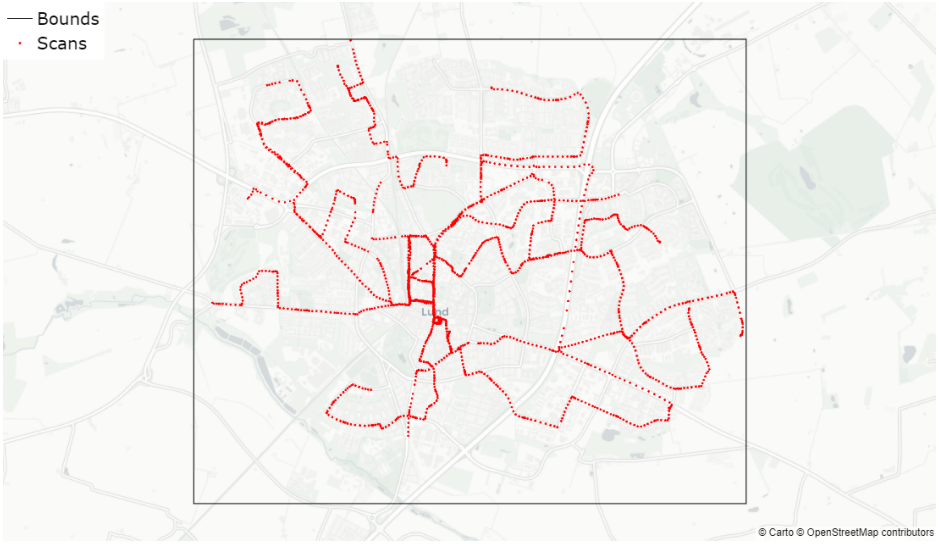
(d) RSS distribution of validation data of group Bus after filtering.

Figure 3.7: Displays the distribution of RSS values before and after filtering. The top row of plots displays the distribution for Slam training data while the bottom row displays for validation data.

Table 3.3: Displays number of entries, unique cells and scans for each dataset in group 1000.

Dataset	Entries	Unique cells	Scans
Training Bus	319 797	266	74 909
Validation Bus	9753	266	2431

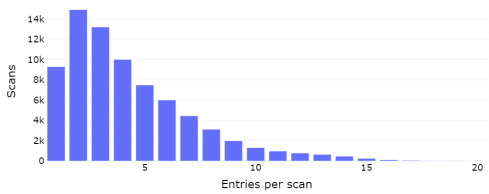
(a) Location of every scan from the training dataset of group Bus.



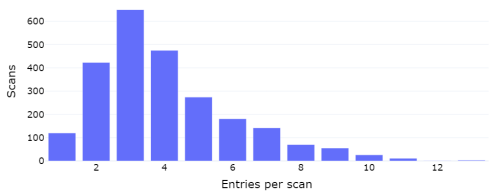
(b) Location of every scan from the validation dataset of group Bus.

Figure 3.8: Geographical scatter plots where every dot signifies a scan with at least one RSS-ID pair.

We also include Figure 3.9, displaying the distribution of the number of RSS-ID pairs per scan.



(a) Distribution of the number of RSS-ID pairs for the training dataset of group Bus



(b) Distribution of the number of RSS-ID pairs for the validation dataset of group Bus.

Figure 3.9: Bar plots displaying the number of entries per scan for group Bus. The x-axis represents the number of RSS-ID pairs, whereas the y-axis represents the number of scans with a certain number of RSS-ID pairs.

3.4 Gaussian Process Posterior

We are now going to describe the implementation of a process capable of producing a posterior distribution. The inputs given are the training data for a specific cell $(X, \mathbf{y})$, a set of hyperparameters $(\ell, \sigma_f, \sigma_n)$ and the test positions X_*. The output is a data structure containing the expected value $\mathbb{E}[\mathbf{f}_*]$ and variance $\mathbb{V}[\mathbf{f}_*]$ for the set of test positions.

A process is initially created which can find the optimal hyperparameters $(\ell, \sigma_f, \sigma_n)$ through minimisation of the NLML described in (2.12). The minimisation is done numerically with the `minimise` method from the `scipy.optimize` package. The solver which is used is the L-BFGS-B algorithm. We also save the number of iterations run when minimising the NLML and a measure of the computational complexity which will be named log complexity. The log complexity measure is determined depending on the computational complexity of calculating the NLML for a GP. For a regular GP that operation scales as $\mathcal{O}(n^3)$ for each iteration. The resulting measure of complexity is therefore the number of iterations times the size of the training data (n) cubed. However, computing the NLML for an SGP scales as $\mathcal{O}(m^2 n)$. The complexity of an SGP will therefore be calculated as the number of iterations times the number of inducing points m squared times the size of the training data n. For simplicity, the mean will be used for the NLML and the number of NLML iterations as well. These values are saved into a log file once the training is completed.

The next option which is added to the process is whether or not to produce a sparse GP. An option m is therefore added which denotes the number of inducing positions. If m is provided and $m < n_*$ an SGP will be trained for m inducing positions along with the other specified options. The matrices that are precomputed from the training data are saved and ready to use when computing a posterior according to 2.21.

The final significant choice to be made is which mean function $m(\mathbf{x})$ to use. The option is added to use a fixed mean value z such that $m(\mathbf{x}) = z$ for any $\mathbf{x}$. Another option is to let the mean value be modelled according to the LDPL model. An LDPL process is therefore built with the same functionality as the GP modelling process. That implies a new set of possible hyperparameters for the LDPL model which will be possible to optimise for or fix to certain values. This model is not meant to be exact but to provide an approximate distribution for the signal propagation. For areas with few available data points the mean function will then give a better indication of the posterior mean when compared to a fixed mean function. The same optimising method is used for finding the optimal LDPL as was used for finding the optimal GP hyperparameters. With the help of the new LDPL model, we can train optimal LDPL parameters (given X and $\mathbf{y}$) by maximising the marginal likelihood. We can then use the prediction of the LDPL model (given a position $\mathbf{x}_*$) as the mean function for the GP.

It might be the case that the positioning accuracy does not correlate to how well the GP is modelled to its training data, but to other factors. In other words, a model that has a greater marginal likelihood might not necessarily mean that it contributes to improving the accuracy of the resulting prediction. If however the prediction accuracy correlates to how well the GP is modelled it is important to save the log marginal likelihood and the configurations related to the result. As the different configurations are tested we can provide an optimal configuration that hopefully improves the accuracy of positioning.

3.5 Positioning

With the previously described posterior distribution, we are now able to compute the loss function value $\mathcal{L}(\mathbf{f}_r, \mathbf{y}_r)$ according to 2.35. The input to the function is a set of posterior distributions $\mathbf{f}_r$ for s different cells at a position $\mathbf{x}_*$ and a set of observed RSS values $\mathbf{y}'_*$ for these cells. The output is the loss function value to minimise for.

As the global minimisation used is a grid search the search field needs to be defined. The purpose of the search field is simply to set the bounds for latitude and longitude coordinates. A simple function is created which when given training input for a GP provides the bounds for the latitudes

and longitudes according to Section 2.6.1. The search space is then bounded by the minimum and maximum latitude and longitude of these individual bounds. It is therefore unlikely, but possible, that the true position lies outside the search field. In the case that the true position lies outside of the search field the predicted position will hopefully be close enough to the limiting bounds, providing a decent prediction anyway.

With the search field defined a function is created which produces a grid of test positions X_* given the search field and the length in metres between any two neighbouring positions. A loss function mapping can now be computed using X_* as input matrix resulting in an approximate solution $\mathbf{x}'_*$ as defined by (2.36).

The local minimisation of the loss function from starting position $\mathbf{x}'_*$ is made with the help of the `minimise` method. The minimisation is improved if provided with the loss function gradient described in (2.38). This requires a definition of the gradients $\nabla\,\mathbb{E}[f_*]$ and $\nabla\,\mathbb{V}[f_*]$ which have been defined for GPs without a custom mean. These gradients are provided for faster convergence, while an approximated gradient can be used for GPs with a custom mean function dependent on $\mathbf{x}_*$. By the end of the local minimisation we are given a position $\mathbf{x}''_*$ which hopefully is close to the global minimum $\hat{\mathbf{x}}_*$. This marks the end of the positioning process.

3.6 Baseline Methods

To get a perspective of the results from the previous models two baseline models are used to compare with, namely WCL described in Section 2.9 and WKNN described in 2.10. The WCL model uses the naive approach of assuming the centroid for a cell to be the position of the highest measured RSS. The positioning method is then implemented using `NumPy`. WKNN is implemented with the help of SKlearn. The Euclidean distance will be used as a metric for the weights and the number of neighbours. The number of labels is equal to the number of unique cells from the training dataset. Given a training dataset of scans, for each scan, there will be several cells which there was no reception from. These fields are filled in as the lowest connection strength possible, -140.0. The RSS values were then normalised between $[0, 1.0]$ and passed to the `KNeighborsRegressor` as input. The target values are the metre representations of the coordinates.

3.7 Positioning Validation

With the use of the positioning, we want to validate the accuracy of several unique model configurations. A validator class is therefore constructed with the purpose of testing models on validation sets. The input required is a training dataset, a validation data set containing the entries for several scans and a prediction model that has been configured appropriately. The model is then trained on the given training data. When training is done the validator can get the predicted position for all of the validation scans and find the error in metres between the predicted position and the true position. Although there are other measurements of accuracy such as Mean Square Error (MSE), the mean error is computed because of its simplicity.

4 Results

This section is divided into three parts based on the three groups of datasets described in Section 3.3. Group 72 was used to demonstrate the general accuracy of GP models, group 1000 was used to compare the best-performing SGP models and group bus was used to simulate a newly collected dataset. Note that group 1000 and group bus are larger datasets, which is why only SGP models will be tested for those groups, as regular GPs would not be able to complete the training and validations in a reasonable time.

Firstly, results relating to the training of the models will be presented in sections named *Distribution Modelling*. Hyperparameter distributions will be presented along with metrics such as the mean NLML, the mean NLRL, the mean number of NLML iterations and the mean complexity. The NLML and NLRL are described in Sections 2.2.2 and 2.4, respectively. These metrics report how well the training and validation data is modelled. The number of NLML iterations and complexity is described in Section 3.4. These metrics describe the computational complexity of the training phase. Note that for Learning datasets we will skip this section as we are only interested in the prediction performance when compared to the baseline models for Learning.

Secondly, starting with a basic positioning method more features will be added to the positioning method to assess how the accuracy improves. These results will be presented in sections named *Positioning*. Each of the tests will result in a mean error which ultimately is used to express accuracy. We will also report the MPG (described in Section 2.7) and the number of points evaluated during the positioning phase. The best-performing model will then be compared to the baseline models, namely the WCL method described in Section 2.9 and the WKNN method described in Section 2.10.

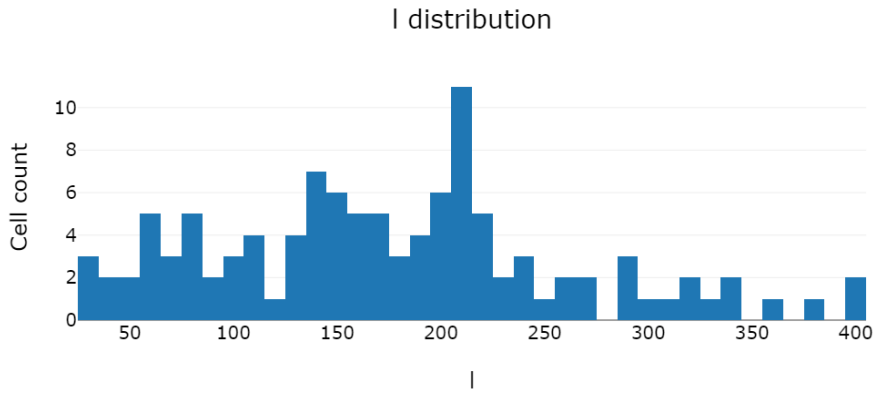
4.1 Group 72

To be as thorough as possible a very simple model was created which could be improved by stepwise adding features. Slam data was used while improving the model as it was more reliable. Learning was therefore only tested for the optimal model.

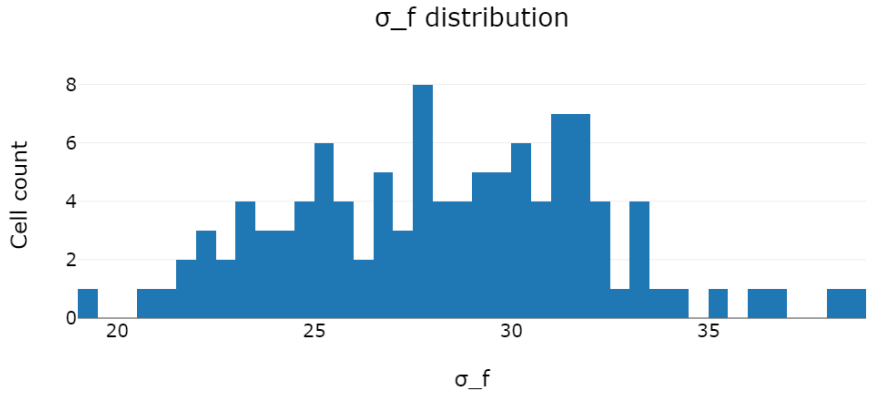
4.1.1 Slam 72

Distribution Modelling

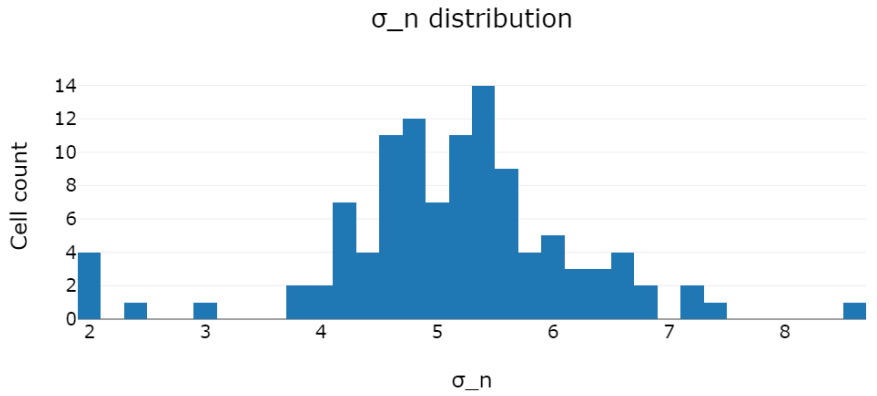
The initial model was based on a regular GP and was therefore simply named GP. For every cell, the hyperparameters were optimised separately. Every regular GP started the minimisation of the NLML with the initial values $\ell = 202.5$, $\sigma_f = 20.5$ and $\sigma_y = 5.5$. The custom mean function was selected to be a constant -140.0 RSS (dBm). The resulting training metrics are listed in Table 4.1, whereas the resulting distributions of optimised hyperparameters are shown in Figure 4.1.



(a) Distribution of hyperparameter ℓ.



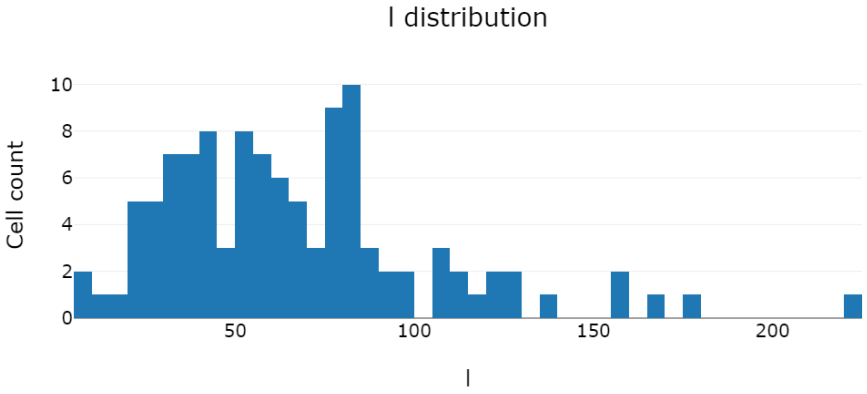
(b) Distribution of hyperparameter σ_f.



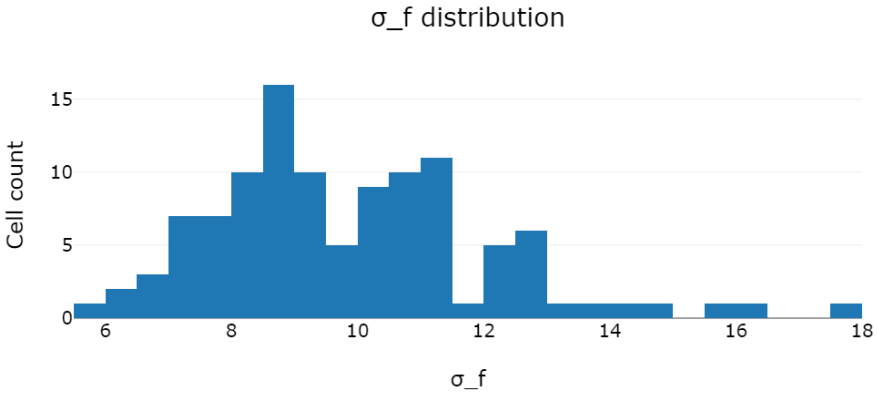
(c) Distribution of hyperparameter σ_n.

Figure 4.1: Distribution of hyperparameters ℓ, σ_f and σ_n for 110 GPs when optimised over dataset 72. A constant mean function of -140.0 (dBm) was used.

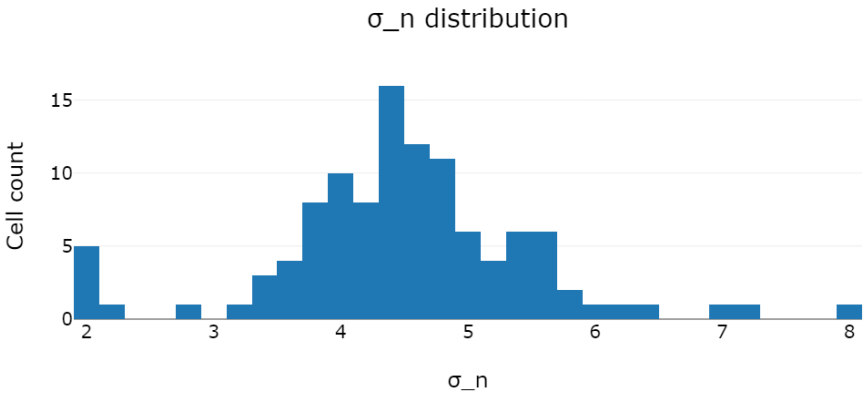
The next set of models introduced custom mean functions. The first mean function was the LDPL and was named GP-LDPL. Given a set of training data X and $\mathbf{y}$, the optimal hyperparameters for the LDPL model were found as described in Section 2.8. The only change from our previous test was therefore the mean function. The resulting hyperparameter distributions are shown in Figure 4.2, whereas the resulting training metrics are shown in Table 4.1.



(a) Distribution of hyperparameter ℓ.



(b) Distribution of hyperparameter σ_f.



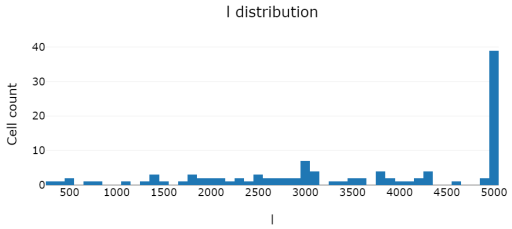
(c) Distribution of hyperparameter σ_n.

Figure 4.2: Distribution of hyperparameters ℓ, σ_f and σ_n for 110 GPs when optimised over dataset 72. The mean function used was the LDPL propagation model.

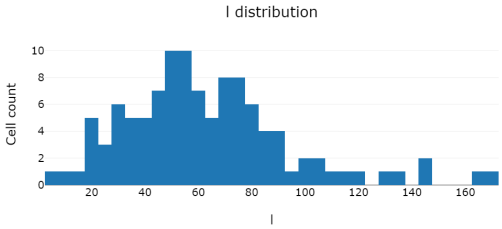
Table 4.1: Presents the mean NLML, mean NLRL, mean number of NLML iterations and mean complexity for multiple GP models trained on the Slam 72 dataset.

Model	NLML	NLRL	NLML iterations	Complexity
GP	3835.9	45.8	20.9	1.4e+11
GP-LDPL	3718.7	44.3	24.4	1.5e+11
GP-SGP(1)	3710.7	44.6	24.6	1.5e+11
GP-SGP(8)	3687.6	44.7	24.9	1.4e+11
GP-SGP(32)	3657.4	46.6	25.3	1.6e+11
GP-SGP(128)	3671.5	48.3	22.9	1.3e+11

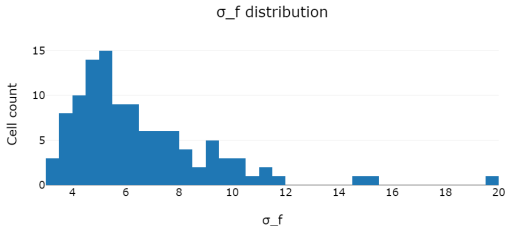
The third set of models used SGPs as mean functions. The reason for testing custom mean functions was to evaluate if the prediction accuracy could be improved by evening the input such that there was less variation of RSS. As the purpose of SGPs is to approximate the distribution optimally they were used for that purpose by using an SGP with a low number of inducing points m. Given a set of training data X, $\mathbf{y}$ and a specified number of inducing points m, an SGP was trained to model the propagation as described in Section 2.3. The SGP was then used as the mean function for evaluating $\mathbf{m}(X)$ or $\mathbf{m}(X_*)$ for a regular GP. To display the effect of m, several different numbers of inducing points were tested (1, 8, 32, 128). The models were named GP-SGP(m), where m was replaced with the number of inducing points. The hyperparameter distribution for SGPs and GPs of the model where $m = 1$ is shown in Figure 4.3. The NLML and NLRL are shown in Table 4.2.



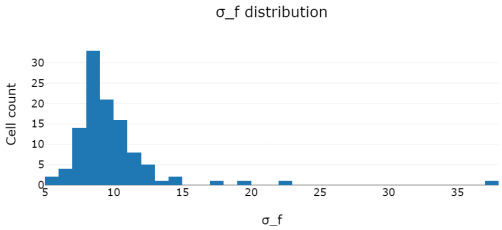
(a) Distribution of hyperparameter ℓ for SGPs.



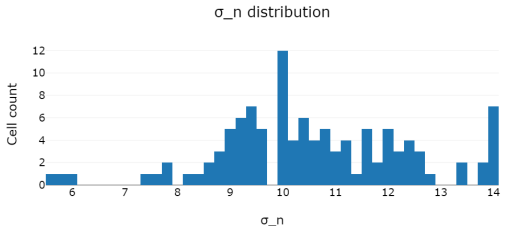
(b) Distribution of hyperparameter ℓ for GPs.



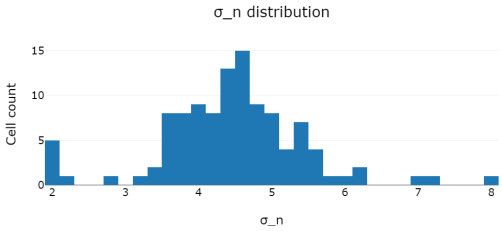
(c) Distribution of hyperparameter σ_f for SGPs.



(d) Distribution of hyperparameter σ_f for GPs.



(e) Distribution of hyperparameter σ_n for SGPs.



(f) Distribution of hyperparameter σ_n for GPs.

Figure 4.3: Distribution of hyperparameters ℓ, σ_f and σ_n for 110 SGPs and GPs, where the SGPs were used as mean functions to the GPs. The left column contains the SGPs whereas the right contains the GPs. The rows display the hyperparameters ℓ, σ_f and σ_n in that order.

The fourth set of models to be validated used SGPs with a constant mean function. The SGPs were trained with varying numbers of inducing positions m on the training data and used a constant mean function of -140.0. Similarly to previous experiments, several values for m were tested (1, 8, 32, 128, 256) and the models were named SGP(m). The reason for not continuing for greater m is that the computation time increased quadratically. We therefore set an arbitrary limit of $m \leq 256$ for the SGP-based models. The resulting training metrics of the SGP-based models are shown in Table 4.2.

The fifth set of models tested using LDPL as a mean function, whereas the last set of models to be validated used an underlying SGP as the mean function. As the goal was for the mean function to simply produce an approximate propagation, the SGP mean function used samples of fewer m. The purpose of the original SGP was to approximate the regular GP as accurately as possible, which is why greater values of m were tested. These tests were named SGP(m1)-SGP(m2), where m1 and m2 denoted the number of inducing points in the original SGP and the SGP representing the mean function, respectively.

Table 4.2: Presents the mean NLML, mean NLRL, mean number of NLML iterations and mean complexity for multiple models trained on the Slam 72 dataset.

Model	NLML	NLRL	NLML iterations	Complexity
SGP(1)	4447.6	53.7	49.2	5.8e+4
SGP(8)	4229.3	50.3	691.1	5.7e+7
SGP(32)	4046.9	47.7	1492.7	1.9e+9
SGP(128)	3951.1	46.7	1334.6	3.3e+10
SGP(256)	3922.5	46.3	1014.4	1.1e+11
SGP(128)-LDPL	3870.5	45.6	1539.7	3.5e+10
SGP(128)-SGP(1)	3865.6	46.0	1601.8	3.7e+10
SGP(128)-SGP(8)	3918.3	47.1	1048.6	2.9e+10
SGP(256)-LDPL	3834.7	45.2	1112.3	1.1e+11
SGP(256)-SGP(1)	3838.0	45.4	1041.5	1.0e+11
SGP(256)-SGP(8)	3876.8	46.3	832.9	8.8e+10

Positioning

The initial positioning method was the simplest possible. When parsing the training data the initial bounds were selected according to Section 2.6.1. Instead of using a minimiser, the centroid (middle position) of the bounds was selected as the predicted position. Since all of the models used the same bounding technique the resulting mean error was the same for all models, specifically **1598.8** metres.

The next positioning method tested was the global minimiser (grid search) described in Section 2.6.2. Since the precision played a large role in the results multiple values for the precision were tested, starting from 200.0 metres and going down to 80.0 metres in decrements of 40. The results regarding the mean error are presented in Figure 4.4, which shows how the accuracy increases for all models as the grid precision decreases. Note that the number of evaluated positions for the grid search is the same for every model since the bounds are identical. The mean number of evaluations for each grid search precision is shown in Table 4.3.

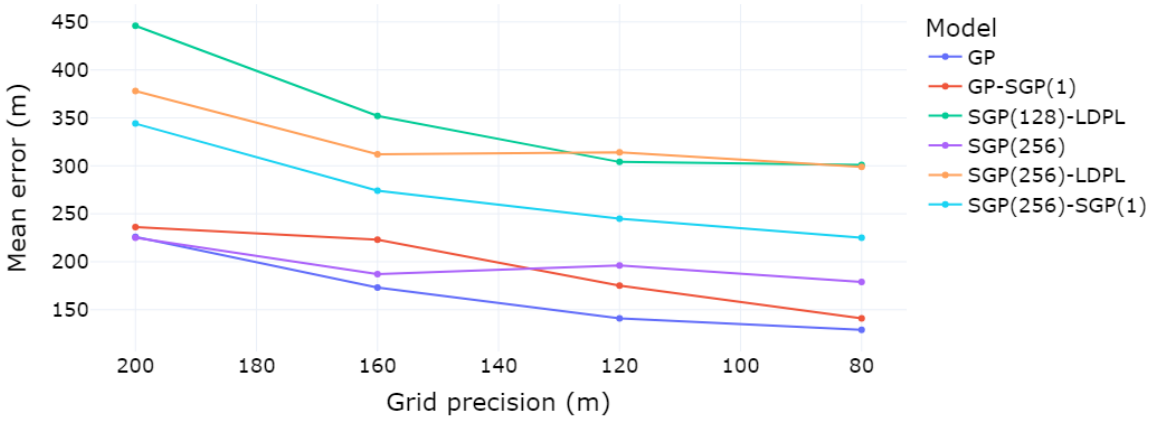


Figure 4.4: Mean error when using the grid search minimisation method for different precisions.

Table 4.3: Displays the average number of positions that were computed during grid search for models trained on the Slam 72 dataset.

	Grid search precision (m)			
	200	**160**	**120**	**80**
Mean evaluations	1104	1733	3049	6791

To further optimise for the position of the highest likelihood the local minimiser described in Section 2.6.3 was implemented. The results from using the global and the local minimiser are shown in Figure 4.5. To get a sense of how much a better minimiser could potentially improve the result the MPG was computed for each of the tested models. These results are shown in Figure 4.6. For each of the models, the mean number of position evaluations during the local minimisation is presented in Figure 4.7. From these results, we can see a general decrease in the number of evaluations as the grid search distance decreases. It is also apparent that the mean number of evaluations for the local minimiser is generally less than 10 for all models.

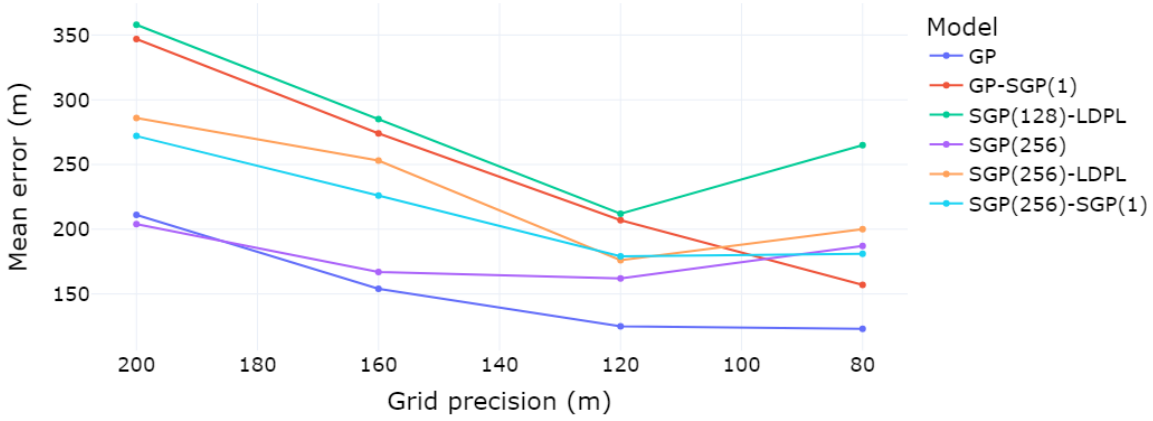


Figure 4.5: Mean error when using the grid search minimisation method followed by a local L-BFGS-B minimiser for different grid search precisions.

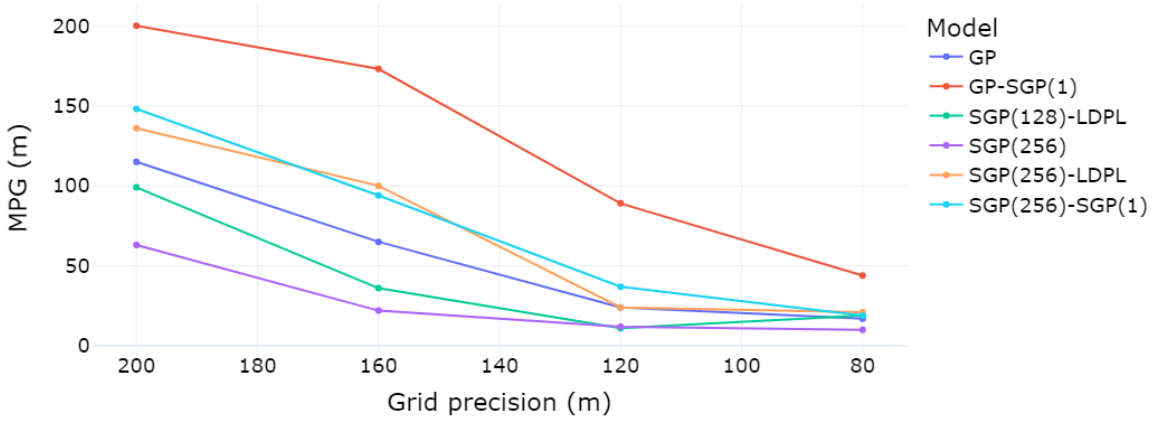


Figure 4.6: Displays the mean potential gain (MPG) of our tested models for group 72.

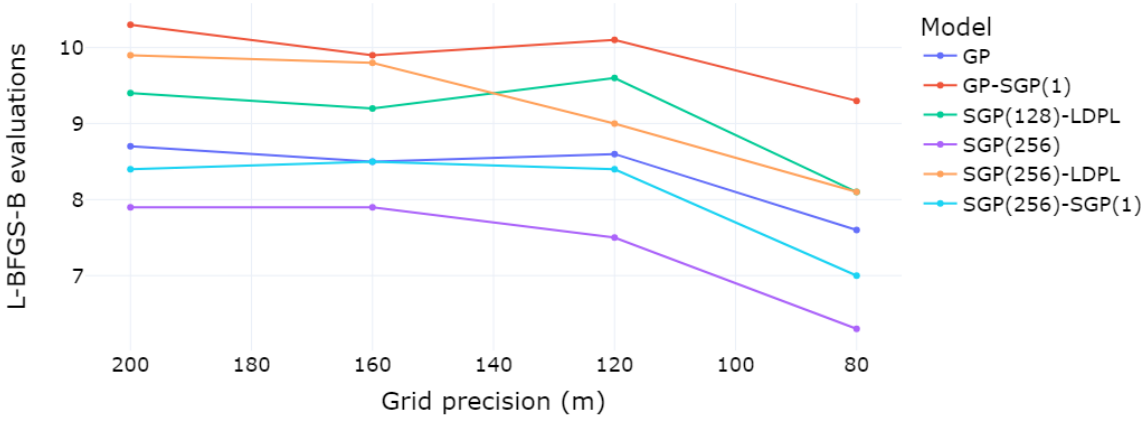
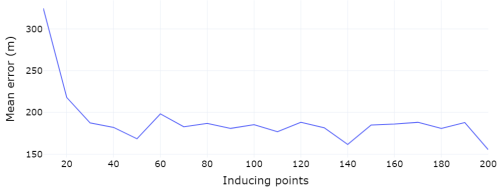
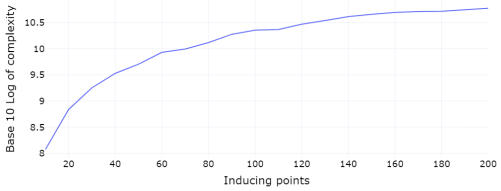


Figure 4.7: Displays the average number of positions that were evaluated by the local minimiser for models trained on the Slam 72 dataset.

The best-performing SGP model was determined to be SGP(256), further analysis was therefore conducted regarding this model. We initially analysed the effect of m on the mean error and the complexity of the hyperparameter training. We did so by gradually increasing m from 10 to 200 and computing the complexity measure and the mean error with a grid search precision of 80 metres when training on Slam data. The results are shown in Figure 4.8. From these observations the relationship between the mean error and log complexity was established, as presented in Figure 4.9. A second-order polynomial trendline was yielded by minimising the square error. The resulting trendline was $y = 3380.07 - 628.91x + 30.85x^2$, where y represents the mean error in metres and x represents the logarithm of the computational complexity. The second-order polynomial was deemed to describe the relationship between the mean error and the log complexity good enough within the range $10 \leq m \leq 200$. This resulted in a range for log complexities between 8 and 10.8. For log complexities outside of this range, the trendline does not hold.



(a) Displays the mean error over the number of inducing points, m.



(b) Displays the base 10 logarithm of the mean complexity over m.

Figure 4.8: Two graphs displaying how varying the number of inducing points m for the SGP from 10 to 200 affects the mean error and complexity for group 72.

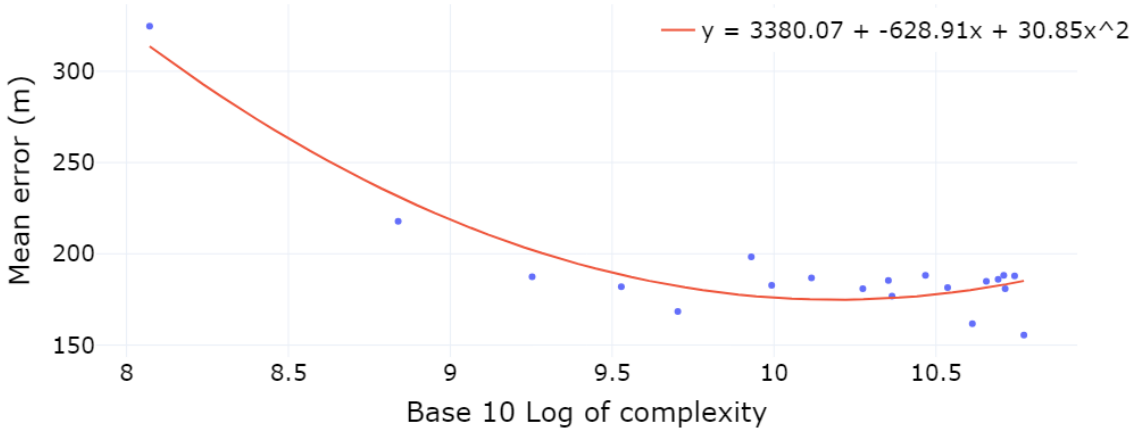


Figure 4.9: A scatter plot displaying the observed values for the mean error over the complexity for group 72. The trendline was calculated by minimising the mean square error, which yielded the function $y = 3380.07 - 628.91x + 30.85x^2$, where y represents the mean error in metres and x represents the logarithm of the computational complexity.

The SGP(256) model was then compared to the baseline methods based on WKNN and WCL. The results are shown in Table 4.4.

Table 4.4: Mean error in metres when validating over dataset 72 with the selected optimal positioning technique. Note that SGP(256) used a grid search precision of 80 metres.

Model	Mean error (m)
SGP(256)	162
WCL	710
WKNN	116

4.1.2 Learning 72

Table 4.5 shows the resulting mean error when training on the Learning dataset of group 72. As WKNN was only applicable to datasets with scans, it was not included for Learning.

Table 4.5: Mean error in metres when validating over validation dataset 72 after training on Learning 72 with the selected optimal positioning technique.

Model	Mean error (m)
SGP(256)	429
WCL	642

4.2 Group 1000

The next group that was evaluated was the group with a validation set consisting of 1000 random scans. There existed two training datasets, one from Slam and one from Learning.

4.2.1 Slam 1000

Distribution Modelling

The metrics regarding the distribution modelling are displayed in Table 4.6.

Table 4.6: Displays the mean NLML, mean NLRL, mean number of NLML iterations and mean complexity for multiple models trained on the Slam 1000 dataset.

Model	NLML	NLRL	NLML iterations	Complexity
SGP(128)-LDPL	4376.2	10.9	1211.3	4.1e+10
SGP(256)	4420.8	10.9	873.1	1.7e+11
SGP(256)-LDPL	4342.1	10.8	823.3	1.3e+11
SGP(256)-SGP(1)	4350.1	10.8	823.0	1.3e+11

Positioning

The number of evaluations made for each precision of grid search is presented in Table 4.7. The mean error when using well-performing SGP models is presented in Figure 4.10, whereas the MPG is presented in Figure 4.11. The mean number of positions to evaluate when using the L-BFGS-B minimiser is presented in Figure 4.12.

Table 4.7: Displays the average number of positions that were computed during grid search for models trained on the Slam 1000 dataset.

	Grid search precision (m)			
	200	**160**	**120**	**80**
Mean evaluations	293	452	791	1756

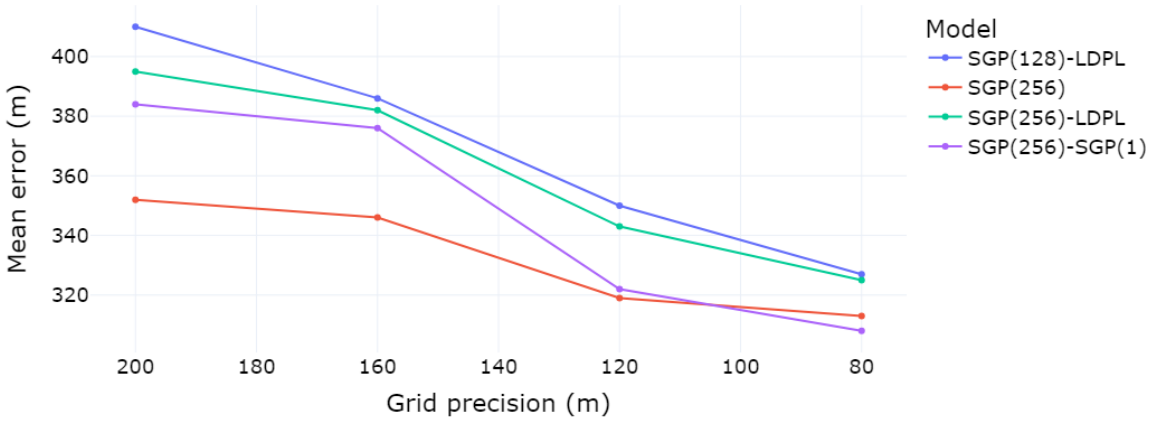


Figure 4.10: Mean error when using the grid search minimisation method followed by a local L-BFGS-B minimiser for different grid search precisions.

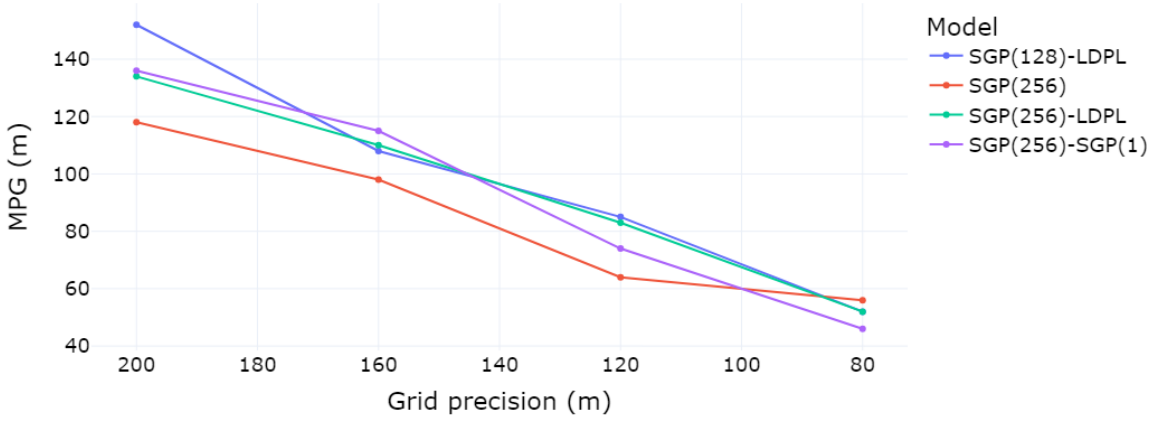


Figure 4.11: Displays the mean potential gain (MPG) of our tested models for group 1000.

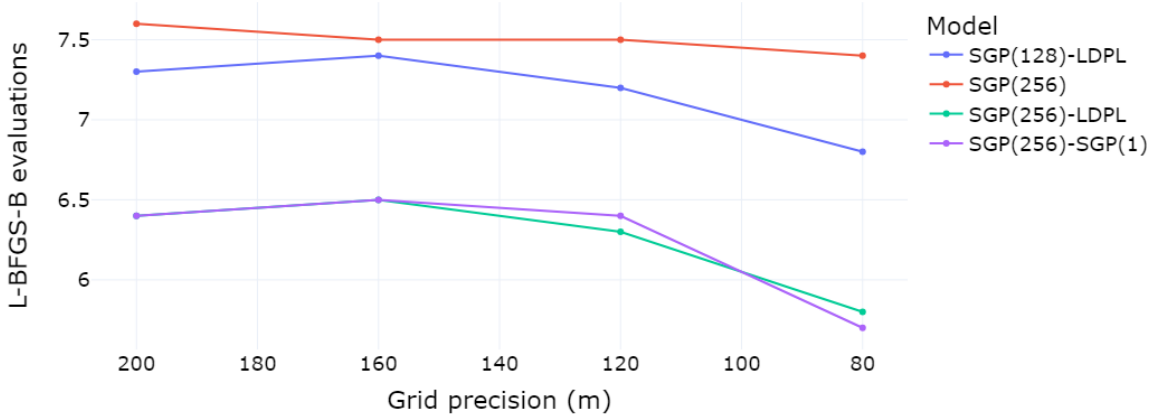
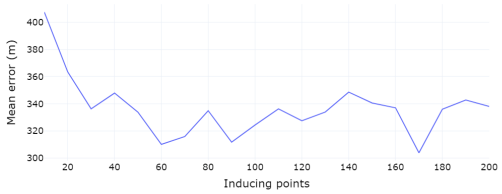
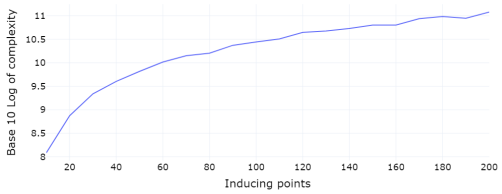


Figure 4.12: Displays the average number of positions that were evaluated by the local minimiser.

We chose to examine the SGP(m) model without a custom mean function as it was one of the models that had performed best for the best grid search precision (80 metres), while also consistently beating the other models for worse grid search precisions (120, 160, 200 metres). Regarding the SGP(m) model, the mean error over m and the complexity over m can be seen in Figure 4.13. From these results, a second-order polynomial relationship was established between the mean error and the log complexity, as presented in Figure 4.14. This yielded a trendline $y = 2160.54 - 357.48x + 17.42x^2$ by minimising the square error, where y denotes the mean error in metres and x denotes the logarithm of the computational complexity.



(a) Displays the mean error over the number of inducing points, m.



(b) Displays the base 10 logarithm of the mean complexity over m.

Figure 4.13: Displays the relationships between the number of inducing points, mean error and complexity for group 1000. The number of inducing points was selected between 10 and 200.

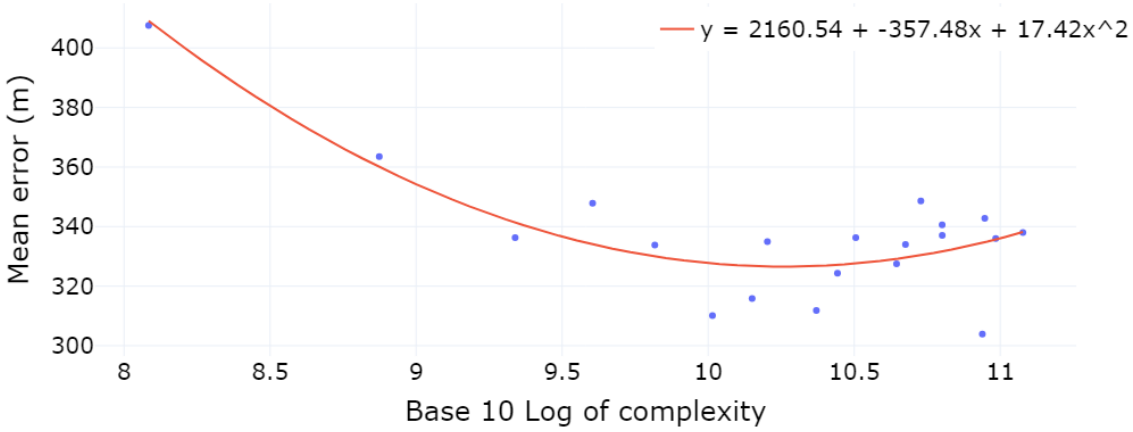


Figure 4.14: A scatter plot displaying the observed values for the mean error over the complexity for group 1000. The trendline was calculated by minimising the mean square error, which yielded the function $y = 2160.54 - 357.48x + 17.42x^2$, where y represents the mean error in metres and x represents the logarithm of the computational complexity.

When comparing with the baseline for this dataset we chose the model that had achieved the lowest mean error result for the dataset, namely SGP(256)-SGP(1). The results of the comparison with the baseline models are presented in Table 4.8.

Table 4.8: Mean error in metres when validating over dataset 1000 with the selected optimal positioning technique.

Model	Mean error (m)
SGP(256)-SGP(1)	308
WCL	676
WKNN	168

4.2.2 Learning 1000

We evaluated the SGP(256) model and the WCL baseline model against the Learning 1000 dataset. The results are presented in Table 4.9.

Table 4.9: Mean error in metres when validating over dataset 1000 with the selected optimal positioning technique.

Model	Mean error (m)
SGP(256)	1638
WCL	1650

4.3 Group Bus

This section will evaluate the performance of our models when training and validating models on the Bus datasets.

Distribution Modelling

The resulting metrics related to the distribution modelling are presented in Table 4.10.

Table 4.10: Displays the mean NLML, mean NLRL, mean number of NLML iterations and mean complexity for multiple models trained on the Slam 1000 dataset.

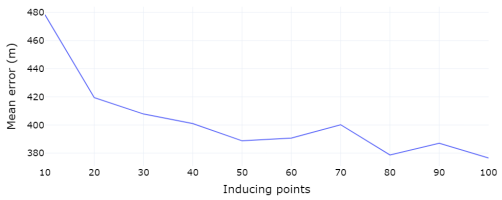
Model	NLML	NLRL	NLML iterations	Complexity
SGP(128)	2489.9	14.9	693.8	2.0e+10

Positioning

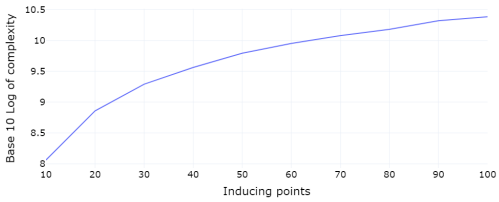
The comparison between the mean error for SGP(128) model and the baseline models is presented in Table 4.11. The relationships between the number of inducing points, the mean error and the complexity for group Bus are presented in Figure 4.15. The resulting second-order polynomial trendline representing the relationship between the mean error and the log complexity is presented in Figure 4.16. The trendline was found to be $y = 2039.32 - 313.67x + 14.85x^2$, where y represents the mean error in metres and x represents the logarithm of the computational complexity.

Table 4.11: Mean error when validating over the Bus validation dataset.

Model	Mean error (m)
SGP(128)	387
WCL	655
WKNN	321



(a) Displays the mean error over the number of inducing points, m.



(b) Displays the base 10 logarithm of the mean complexity over m.

Figure 4.15: Displays the relationships between the number of inducing points, mean error and complexity for group Bus. The number of inducing points was selected between 10 and 100.

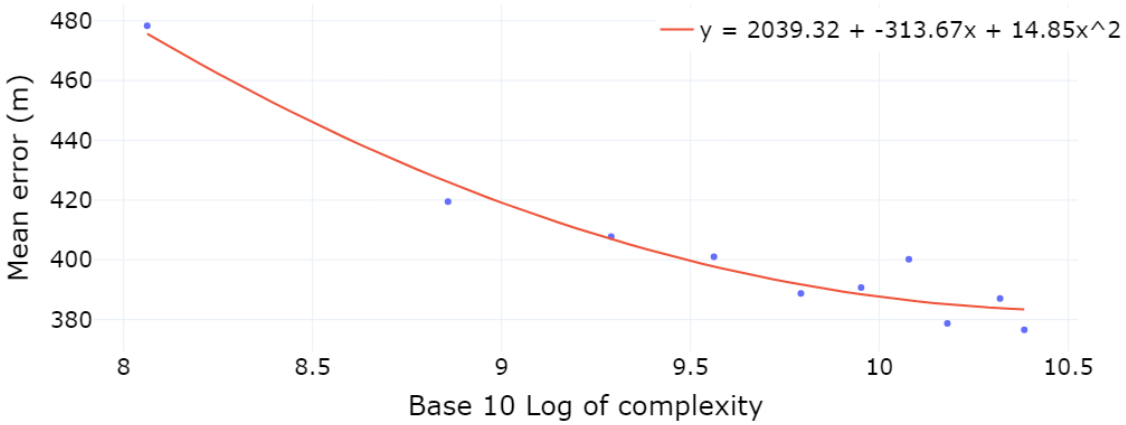


Figure 4.16: A scatter plot displaying the observed values for the mean error over the complexity for group Bus. The second-order polynomial trendline was calculated by minimising the mean square error, which yielded the function $y = 2039.32 - 313.67x + 14.85x^2$, where y represents the mean error in metres and x represents the logarithm of the computational complexity.

5 Discussion

The discussion will be separated into a few topics which will be discussed. Firstly, we will discuss the results related to distribution modelling. Secondly, the positioning results will be discussed. Thirdly, the positioning results will be analysed both for the baseline models and the GP-based models. Finally, we will discuss future work.

5.1 Distribution Modelling

For this section, we will analyse the results regarding the metrics used in the distribution modelling, presented in Section 4.1.1.

5.1.1 Hyperparameters

As stated in the results section, the hyperparameters were optimised freely with a local minimiser. The resulting distributions of the hyperparameters for regular GPs, shown in Figure 4.1, seem to be concentrated around a mean value for σ_f and σ_n but noisier for ℓ. These hyperparameter distributions differ somewhat from the distributions shown for a GP with a custom mean function. As the custom mean function models the RSS propagation the variation for the training data is reduced, as described in (2.9). Another prominent trend is the decreased mean of ℓ. When comparing the hyperparameter distributions of the GP with the custom mean SGP(1) in Figure 4.3, the same trend is repeated where the SGP achieves a better marginal likelihood with larger values for ℓ. Further, the resulting optimised hyperparameters ℓ and σ_f are reduced for the basis GP when compared to a GP with a constant mean function.

5.1.2 Metrics

For this section, the metrics NLML, NLRL, NLML iterations and complexity will be analysed.

The NLML and NLRL presented in Table 4.1 and 4.2 show how well the distributions fit the training and validation data. Note that we want to maximise the marginal likelihood and therefore we want to minimise the NLML, the same applies to the NLRL. One general trend that can be seen when comparing the SGP(m) models is that the number of inducing points negatively correlates with both the NLML and the NLRL.

The results show that the model that achieves the minimal NLRL is the GP with an SGP mean of 1 inducing point. As the number of inducing points increased for the SGP custom mean function, the resulting NLML decreased, but the NLRL increased. This means that the model improved the modelling of the training data, but the modelling of the validation data suffered as a result. For the case where $m = 1$ of the custom mean SGP the propagation was modelled by a single inducing point. The posterior distributions were then decided only by the covariance to that inducing point according to the Gaussian kernel defined by (2.3). The mean function was therefore similar to the LDPL model in the sense that one position had the maximum RSS and the propagation declined purely dependent on the Euclidean distance to that position according to some function. Regarding the SGP models presented in Table 4.2, the model with the minimal NLRL was the SGP(256)-LDPL model. Interestingly, this model also achieved the lowest NLML. Other models that also achieved low scores for the NLRL measure are SGP(128)-LDPL, SGP(256) and SGP(256)-SGP(1).

Regarding Table 4.2, no significant trend was found regarding the number of NLML iterations other

than that SGPs with very small m did not require as many NLML iterations to find a minimising set of hyperparameters. However, the complexity is important to consider when increasing m. As m increases from 1 to 8 and the number of NLML iterations increases approximately 14-fold, we can expect a mean increased computational load of about $8^2 * 14 \approx 900$, which seems to agree with the resulting complexity of SGP(8) as $(5.8 * 10^7)/(5.7 * 10^4) \approx 1000$.

5.2 Positioning

5.2.1 Grid Search

Starting with the results from group 72 using just a grid search, the results of Figure 4.4 display how much of a difference the precision makes. The number of points that need to be evaluated scales quadratically, making the number of evaluations with a precision of less than 80.0 metres very large for normal distances between cells. One factor that can be used to combat this is by reducing the initial grid size. For predicting positions where the RSS of 3 or more cells are collected it is reasonable to assume that the expected position is somewhere in between these 3 cells. It would therefore be unnecessary to examine the furthest reach of any cell. Even if the true position is outside of the initial grid for some positions it might enable higher grid search precision which then grants greater accuracy for other predictions.

Note how the mean error fluctuates heavily for different grid search precisions. Since there are only 72 validation scans minor changes to the grid precision can greatly affect which position has the greatest likelihood. This could be counteracted by examining the grid search positions further. For some of the models the gradient can be computed, it might therefore have been beneficial to use a local minimiser for several of the grid search positions instead of just one.

5.2.2 L-BFGS-B

Figure 4.5 introduces the local minimiser named L-BFGS-B. The results show little improvement when comparing the results from Figure 4.4 to Figure 4.5. The cause of this is more than likely that in many cases the function space is not perfect, and there exists many local minimiums. This leads to L-BFGS-B being very sensitive to its initialisation. For most cases, the prediction is improved and the local minimum is closer to the true position, but there are cases where finding the local minimum worsens the prediction. As presented in Figures 4.7 and 4.12, the mean number of evaluations required for the L-BFGS-B minimiser to find the local minimum is minuscule when compared to the number of grid evaluations required for both group 72 and group 1000, shown in Table 4.3 and 4.7. However, one important difference between the grid search minimisation and the local minimisation is the vectorization which is made possible when evaluating multiple positions at once.

5.2.3 MPG

The MPGs for Slam data of group 72 and group 1000 are displayed in Figures 4.6 and 4.11, respectively. As the grid search precision increases the MPG converges toward zero, meaning that the solution of the highest likelihood (according to the model) is found more often. This can be pretty useful for evaluating the potential of a model without using an extensive global search algorithm.

5.3 Baseline Comparison

We will now compare the results of the best-performing models for the different datasets.

5.3.1 Group 72

The sparse model which performed best for predicting the group 72 dataset was selected from Figure 4.5 as the SGP(256) model with a mean error of 162 metres when the grid search precision was 120 metres. However, several models performed similarly and the results were very noisy for this dataset. Other models that performed similarly include SGP(256)-SGP(1) and SGP(LDPL). It is worth noting that the SGP(256) performed best for the lowest grid search precision (200 metres), even better than the regular GP model. The optimal GP-based model when disregarding computational complexity was the regular GP model, with a mean error of 123 metres.

For the Slam dataset of group 72, when comparing the result of the best-performing GP model (162 metres) to the results of the baseline models shown in Table 4.4 the WCL had the highest mean error (709 metres), whereas the WKNN had the lowest mean error (116 metres). The difference is not as overwhelming though, as the regular GP achieved a mean error of 123 metres.

It is important to regard how the number of inducing points affects the positioning accuracy and complexity when selecting m, as presented in Figure 4.8. From the graph, it is apparent that the prediction accuracy flattens out somewhat after $m = 50$ and the mean error seems to fluctuate between 160 and 200 metres. The same trend of convergence is seen for the complexity as m increases in Figure 4.8b.

The relationship between the log complexity and the mean error is presented in Figure 4.9. The trendline approximately describes the relationship between the mean error and the computational complexity. As can be seen in the figure, the trendline has a minimum mean error at a log complexity of approximately 10.2. As the trendline is described by $y = 3380.07 - 628.91x + 30.85x^2$, we can also calculate the derivative with respect to the computational complexity. For example, the derivative at $x = 8$ is -135.31. As the complexity approaches 10.2 the mean error reaches a minimum. Although this measure is approximate and assumes a second-order polynomial relationship, it offers insight into the cost of decreasing the mean error. Note that the relationship is specific to this dataset.

The results of the Learning dataset for group 72 (presented in Table 4.5) show that SGP(256) achieved a mean error of 429 metres, whereas WCL achieved a mean error of 642 metres. The mean error of SGP(256) is therefore approximately 1/3 less than that of WCL.

5.3.2 Group 1000

We will now examine the results of group 1000, shown in Figure 4.8. The best-performing GP-based model was SGP(256)-SGP(1), with a mean error of 308 metres for a grid precision of 80 metres. It is worth noting that the SGP(256) performed similarly for higher grid search precisions (80 m) but for all other tested grid search distances > 80, the SGP(256) model maintained a lower mean error when compared to SGP(256)-SGP(1). The WKNN baseline method had the greatest accuracy of all the tested methods, with a mean error of 168 metres.

We would like to refer to Figure 4.13 for further information regarding how the accuracy and complexity are affected by the number of inducing points m. A similar trend, as was seen for group 72 in Figure 4.8, can be observed for group 1000 as well. The relationship between the log complexity and the mean error is presented in Figure 4.14. The trendline created for the observed points is described by the function $y = 2160.54 - 357.48x + 17.42x^2$. Comparing this to the corresponding trendline of group 72 shows a similar trend, where a minimum mean error is achieved at a log complexity of around 10.2. We can determine the derivative of the mean error with regard to the complexity for $x = 8$, yielding -78.76. There is a clear difference between the derivative of the mean error at $x = 8$ for Group 72 compared to Group 1000. It is therefore more costly to decrease the mean error for group 1000 when compared to group 72.

Regarding the Learning dataset, the results shown in Table 4.9 display a mean error of 1638 metres for SGP(256) and 1650 metres for the WCL method. These are both terrible results and can be explained by the small amounts of training data regarding very few cells shown in Table 3.2. The table shows how less than half of the cells used in the validation data exist in the training data. The

models therefore have very little to work with when training on that Learning dataset. The reason for this small amount of shared cells is that a large amount of the Slam data was collected after the Learning data was. When selecting 1000 random scans many of these will therefore derive from the newer Slam data, for which many entries contain newer established cells that are not in the Learning dataset. We consider the results from this Learning dataset as unreliable.

5.3.3 Group Bus

Group Bus is arguably the most realistic dataset regarding training and validation of real-world positioning scenarios. When positioning data is collected the scans are often done within intervals of a few seconds, creating a trail of very similar scans. The problem with extracting random scans from such trails of data and using them as validation is that the training data still contains scans that are largely similar to the validation data. However, when the Bus dataset was collected no such data was leaked into the training data, which is a more realistic scenario for positioning validation. We therefore believe the results from group Bus offer the most realistic results for the mean error of the evaluated models.

The resulting mean error from group Bus is presented in Table 4.11, where the best model was the WKNN method at a mean error of 321 metres, whereas the SGP(128) model achieved a mean error of 387 metres. The WKNN method therefore had a mean error that was approximately 20% less than that of SGP(128). In turn, the SGP(128) had a mean error that was approximately 40% less than that of WCL.

The reason for using SGP(128) instead of SGP(256) was simply to save time. SGP(128) already had a calculated complexity of $2.0 * 10^{10}$, as shown in Table 4.10. By increasing it further the complexity would increase by a factor of 4, while still maintaining roughly the same prediction accuracy.

The trendline which describes the relationship between the mean error and the complexity is presented in Figure 4.16, and is described by the function $y = 2039.32 - 313.67x + 14.85x^2$. The derivative of the trendline at $x = 8$ is -76.07, and $x = 10.56$ defines the global minimum of the trendline. What can be gathered by the trendlines of Group 72, 1000 and Bus is that they all seem to converge toward a local minimum around $x = 10.5$, and increasing the complexity further does not decrease the mean error. For all three of the groups, this complexity is achieved when using approximately 110 inducing points. It is important to note that this assumes a simple polynomial relationship, which does not accurately model the mean error as the number of inducing points increases above 200. For $m > 200$, we would require further analysis to model the mean error correctly.

5.4 Future Work

For this section, we will present some ideas which might be of interest for further work on the topic.

During this thesis, the only input parameters that have been considered were the measured RSS and the ID of the cell. For a more extensive prediction model, one might consider involving other parameters besides the ones used for RSS propagation. One example would be the ID of the device that reported the measured RSS, as certain devices could perhaps have unique noise distributions. It is also possible that some devices might show a trend of dampening. For this thesis, the measured values were assumed to have a Gaussian noise ϵ, but it might be possible to relate unique devices to unique non-Gaussian noise distributions.

Another potential improvement involves the minimisation algorithm used for the NLML minimisation. The used minimiser (L-BFGS-B) is a local minimiser and might not find the optimal set of the hyperparameters ℓ, σ_f and σ_n. It could therefore be valuable to use a global minimiser. One potential such minimiser is the firefly algorithm, used in [4] both to minimise the NLML and to find the predicted position by minimising the loss function. This algorithm could potentially offer better results for both problems, which would be interesting to compare with the method used in this thesis.

We would also like to suggest optimising every individual GP differently. Throughout this thesis, all trained SGPs have used the same number of inducing points and limits regarding their hyperparameters. Remember that during the training of an SGP, maximising the variational lower bound F_V leads to improving the approximation of $p(\mathbf{f}, \mathbf{f}_m|\mathbf{y})$, as described in Section 2.3.1. It would be reasonable to assume that SGPs which represent cells with noisy RSS propagations could improve their approximation a lot more compared to training data with smooth propagations when provided more inducing points. There might therefore exist an exact or approximating measure of which SGPs would improve their variational lower bound the most when provided an extra inducing point. This would lead to a more efficient training phase.

6 Conclusion

This thesis presents several models for fingerprint-based positioning using SGPs. The models that used custom mean functions did not offer a significant increase in positioning compared to the SGP model with a constant mean function. The models using a custom mean function were therefore disregarded. Our most stable model used an SGP to model the RSS propagation, and depending on the required precision the number of inducing points could be selected to fit this requirement.

The two baseline models that were used to compare with the GP-based models were WKNN and WCL. We found that the WKNN model offered the greatest accuracy when applicable to the dataset. For the dataset which was deemed the most realistic for a real-world positioning problem (group Bus), WKNN achieved a mean error of 321 metres whereas the proposed SGP-based model achieved a mean error of 387 metres. For datasets that did not contain scans, but rather one RSS-ID entry per position, the SGP model and the WCL method were compared. For evaluation with the datasets of group 72 the proposed SGP model achieved a mean error of 429 metres, whereas WCL achieved 642 metres. For evaluation with the datasets of group 1000 SGP achieved a mean error of 1638, compared to a mean error of 1650 metres for WCL.

We will now answer the questions presented in the goals Section 1.2.

- Which GP-based model offers the highest accuracy?

The regular GP offered the greatest prediction accuracy for a majority of the cases where it could be tested. This was expected as all the other models are SGPs which optimise their hyperparameters to approximate the regular GP. Regarding the SGPs, using a custom mean function did not have much of an effect and could be considered redundant for the employed tests of positioning accuracy. The optimal scalable model therefore used SGPs to model the RSS propagation of each cell.

- How does the optimal model perform compared to common methods of positioning such as WCL or WKNN?

The optimal SGP model achieved a mean error that was approximately 20% greater than that of the WKNN method. For the datasets where WKNN was not applicable, the SGP model offered more accurate predictions than the WCL method. How much more accurate was dependent on how extensive the provided training data was. However, for the Bus dataset, the SGP model had a mean error that was approximately 40% lower than that of WCL.

- What is the tradeoff between accuracy and scalability?

Assuming a simple relationship between the complexity and the resulting mean error we can determine how much the mean error is expected to decrease as we increase the number of inducing points. When using 10 inducing points for the groups 72, 1000 and Bus, the mean error had an expected decrease of 135.31, 78.76 and 76.07 metres for each factor 10 increase in complexity respectively. The trendlines describe that the minimum mean error was achieved when using between 100 and 120 inducing points for our tested datasets.

Bibliography

[1] L. Wang, P. Groves, and M. Ziebart, "Multi-constellation gnss performance evaluation for urban canyons using large virtual reality city models.", *Journal of Navigation*, vol. 65, no. 3, pp. 459-476 –476, 2012, ISSN: 03734633.

[2] M. Aldibaja, N. Suganuma, K. Yoneda, and R. Yanase, "Challenging environments for precise mapping using gnss/ins-rtk systems: Reasons and analysis.", *Remote Sensing*, vol. 14, no. 16, p. 4058, 2022, ISSN: 20724292.

[3] A. Grenier, E. Lohan, A. Ometov, and J. Nurmi, "A survey on low-power gnss.", *IEEE Communications Surveys & Tutorials, Communications Surveys & Tutorials, IEEE, IEEE Commun. Surv. Tutorials*, vol. 25, no. 3, pp. 1482–1509, 2023, ISSN: 1553-877X.

[4] S. Yiu and Y. Kai, "Gaussian process assisted fingerprinting localization.", *IEEE Internet of Things Journal, Internet of Things Journal, IEEE, IEEE Internet Things J*, vol. 3, no. 5, pp. 683–690, 2016, ISSN: 2327-4662.

[5] S. S. Moosavi and P. Fortier, "Fingerprinting positioning in distributed massive mimo systems using affinity propagation clustering and gaussian process regression.", *Wireless Personal Communications: An International Journal*, vol. 121, no. 3, pp. 1835–1855, 2021, ISSN: 0929-6212.

[6] X. Wang, S. Mao, J. Zhang, S. Periaswamy, and J. Patton, "Indoor radio map construction and localization with deep gaussian processes.", *IEEE Internet of Things Journal, Internet of Things Journal, IEEE, IEEE Internet Things J*, vol. 7, no. 11, pp. 11 238–11 249, 2020, ISSN: 2327-4662.

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006, ISBN: 9780262182539.

[8] *Introduction to RF Propagation*. Wiley, 2005, ISBN: 9780471655961.

[9] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes.", *Journal of Machine Learning Research*, vol. 5, no. Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, AISTATS 2009, pp. 567-574 –574, 2009, ISSN: 15324435.

[10] J. Hensman. "Derivation of sgpr equations". Accessed: 2024-01-08. (2016), [Online]. Available: `https://gpflow.github.io/GPflow/develop/notebooks/theory/SGPR_notes.html`.

[11] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization.", *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[12] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd. USA: Prentice Hall PTR, 2001, ISBN: 0130422320.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011, Accessed: 2024-02-5.

[14] D. Guggenheimer, "Global cell-id positioning using artificial neural network", M.S. thesis, Lund University - Centre for Mathematical Sciences, 2024.

[15] F. A. Hizham and R. V. H. Ginardi, "Development of information retrieval method and haversine formula to determine clinic recommendation in jember.", *2021 International Conference on ICT for Smart Society (ICISS), ICT for Smart Society (ICISS), 2021 International Conference on*, pp. 1–6, 2021, ISSN: 978-1-6654-1697-9.

Master's Theses in Mathematical Sciences 2024:E12
ISSN 1404-6342

LUTFMA-3528-2024

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/