Learned Multi-Sensor Indoor Positioning of Mobile Devices

Christoffer Kjellson

Master's thesis
2021:E31

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Learned Multi-Sensor Indoor Positioning of Mobile Devices

Student:
Christoffer Kjellson
c.kjellson@gmail.com

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

Deputy supervisor:
Kalle Åström
karl.astrom@math.lth.se

Deputy supervisor:
Martin Larsson
martin.larsson@math.lth.se

Deputy supervisor:
Björn Lindquist
bjorn@combain.com

Examiner:
Carl Olsson
carl.olsson@math.lth.se

Abstract

Attenuation of the microwave signal used in the Global Positioning System (GPS) due to interaction with building structures complicates the task of accurate indoor positioning. For this reason, research on alternative approaches is being performed at universities and companies worldwide. The Global Indoor Navigation (GIN) research project in Lund is one such initiative, where the main goal is to perform indoor positioning in a vast number of buildings globally rather than in specific buildings locally. This thesis proposes and investigates several methods and algorithms for indoor positioning with applications in the GIN project that are adaptive to different buildings, users and devices. The main contribution is a novel multi-purpose deep learning architecture and training procedure which leverages received signal strength indications (RSSI) from Wi-Fi and Bluetooth beacons. The approach achieves a mean positioning error that is comparable with state of the art WKNN methods, offering improved inference speed. During one training session, three multi-layer perceptrons are generated, which all can be used separately in different applications. In addition to the development of this model, it is also investigated how the resulting position estimates can be combined with estimated displacements in position that are based on inertial sensors in mobile devices. This part also shows promising results, as the investigated approach decreases the mean positioning error from 7.2 m to 5.4 m for the largest dataset used. The models and algorithms were evaluated on two fingerprinting datasets, and in a Kaggle competition where these contributed to the second place entry.

Keywords: *Indoor Positioning, Deep Learning, Fingerprinting, RSSI, Machine Learning, Pedestrian Dead Reckoning, Kaggle, Sensor Fusion, SDAE, LSTM*

Acknowledgements

I would first like to thank my supervisors Magnus Oskarsson, Björn Lindquist, Kalle Åström and Martin Larsson for valuable input and discussions during this thesis, and also for enabling the project. Secondly, I would like to thank everyone else involved at Combain Mobile for help with equipment and hardware. Joining a team in the related Kaggle competition during the last month of this thesis was also a great learning experience, which I have my four great collaborators from around the world to thank for. Finally, I would like to thank my fellow students, friends and family for the great support during this thesis, and also during the last five years of my studies.

Contents

List of abbreviations

- **bssid** - basic service set identifiers
- **dBm** - decibel-milliwatts
- **GIN** - Global Indoor Navigation
- **GPS** - Global Positioning System
- **IMU** - Inertial Measurement Unit
- **KNN** - K-Nearest-Neighbours
- **LS-VCE** - Least Squares Variance Component Estimation
- **LSTM** - Long Short-Term Memory
- **MLP** - Multi-Layer Perceptron
- **OLS** - Ordinary Least Squares
- **PCC** - Pearson Correlation Coefficient
- **PDR** - Pedestrian Dead Reckoning
- **RNN** - Recurrent Neural Network
- **RSSI** - Received Signal Strength Indication
- **SDAE** - Stacked Denoising Autoencoder
- **VCE** - Variance Component Estimation
- **WKNN** - Weighted K-Nearest-Neighbours
- **WLS** - Weighted Least Squares

List of Figures

List of Tables

Chapter 1

Introduction

Navigation is a fundamental part of modern society, but also of human history. The ability to navigate efficiently has always been integral for finding resources, and reducing the time and effort induced by traveling between two locations. Today, there are many tools that help us navigate the world, mainly the global navigation satellite system (GNSS) that enables the GPS technology, but also maps, signs etc.

In many indoor spaces the GPS signal is lost or is of low quality due to attenuation from the interaction between the building structures and the microwave signal [25]. This causes difficulties in indoor navigation and positioning, especially since indoor spaces are growing larger and more complex every year. Accurate indoor positioning can provide a number of valuable services, for example tools for indoor navigation, locating people in need and locating assets and equipment. An effective solution could also result in new ways of improving accessibility and removing bottlenecks in indoor spaces. Due to the impaired GPS accuracy indoors there has recently been a surge in research on alternative ways of performing accurate indoor positioning. One such approach is based on using the received signal strength indication (RSSI) that a device receives from multiple Wi-Fi and Bluetooth beacons located inside a specific building.

The RSSI measurements can be used to perform indoor positioning in various ways. One approach is to use geometrical methods. With known or estimated beacon positions inside buildings it is possible to perform trilateration of a device since the distance to a beacon is inversely proportional to the RSSI. Examples of such approaches are described in [16, 4, 18]. A different approach is to gather large amounts of RSSI-measurements from known positions inside a building and then use these as a reference for assigning positions to new measurements, which is classically called fingerprinting. There are drawbacks to both approaches. For example fingerprinting often depends on manually collected data at known positions inside buildings, which usually requires extensive manual labor. Trilateration on the other hand, requires knowledge of the beacon positions, which can for example be found using self-calibration methods [3].

The above methods are often used to get an estimate of the absolute position in a building. Other sensors, such as sensors in the inertial measurement unit (IMU) in a mobile device (e.g. accelerometer and gyroscope), can be used in multiple ways to refine the positioning estimates. This is often called sensor fusion.

1.1 The Global Indoor Navigation research project

This thesis work was part of the Global Indoor Navigation (GIN) research project[1]. The project was initiated by Combain and is as of June 2021 performed in cooperation with the Centre for Mathematical Sciences at Lund University until September 2021. The main goal of the project is to automatically detect building characteristics that can be used for positioning and other navigation services on a global scale. The precursors to the GIN project were the "Indoor 3D" research project and the "Indoor GPS" research project.

1.2 Goal of the thesis

The aim of this thesis was to explore and improve methods of using RSSI to position correspondences to perform indoor positioning, with a focus on applicability within the GIN research project and Combain's positioning services. This means that a solution should be able to adapt to multiple buildings without manual tuning of parameters and be efficient for use in online positioning. A sub-goal was to investigate how measurements from other sensors, such as IMU-sensors, in a mobile device can be used in combination with the RSSI-based positioning to improve the positioning accuracy. Since the thesis was part of the GIN project, outcomes that are not directly related to positioning, but are relevant for navigation, such as estimation of walls were also considered.

1.3 Contributions

The main contributions of this thesis work are:

- A deep learning architecture and training procedure for combined accurate indoor positioning, floor-classification and RSSI-map generation.
- A customized pedestrian dead reckoning (PDR) approach for improving estimates of relative positions.
- An optimization procedure for combining estimates of relative and absolute positions.
- Two approaches for estimating locations of walls within buildings.
- Algorithms for utilizing building floor-plans, and/or known reference points to improve positioning accuracy.

[1]GIN URL: https://combain.com/research/

Chapter 2

Background

2.1 Theory

In this section the fundamental theory needed to understand this thesis is introduced. The work spans several areas within machine learning, optimization, positioning, and statistics, and for this reason only the theory of the main components of this thesis are covered here. No concepts are covered in detail, which means that readers familiar with the basic theory can skip this section.

2.1.1 Received signal strength indication

All Wi-Fi and Bluetooth beacons radiate a signal with a certain signal strength. The signal strength decreases exponentially with the distance from the beacon to the receiver, and this decrease is called the path loss. The dependence on distance of the signal strength can be described by the log-distance path loss model

$$L = L_0 + 10\mu \log_{10} \frac{d}{d_0}, \tag{2.1}$$

where d is the distance to the receiver, d_0 is the reference distance, L is the total path loss, L_0 is the path loss at the reference distance, and μ is the path loss exponent factor [13]. The path loss exponent factor is in vacuum and infinite space $\mu = 2$, while it can vary in other materials and for different signal frequencies [13]. The path loss exponent is for example heavily affected by walls, humans, and other objects inside buildings. This means that the signal strength from Wi-Fi and Bluetooth beacons does not decrease with the same rate in all directions from the beacon position in most indoor spaces. The signal strength is also dependent on other factors, such as multi-path effects, standing waves, and interaction with the environment [34].

The received signal strength indication (RSSI) is an indicator of the actual received signal strength. The RSSI is measured in various ways by different devices [17], which means it has no unit. Nevertheless, the RSSI is in literature often equated with the received signal

strength RSS measured in dBm. A typical interval used for RSSI, which will be assumed in this thesis is $[-100, 0]$ dBm, where a larger value indicates a higher signal strength.

Each Wi-Fi and Bluetooth beacon can send out multiple signals and at different frequencies that all can be detected by a device. The information of which signal is received is identified by the basic service set identifiers (bssid), which is a unique identifier for the signal. Two signals that originate from the same beacon can both be used for improving positioning accuracy. All detected signals with a unique bssid will therefore in this thesis be treated as an individual beacon.

Since the RSSI from each beacon inside a building is heavily attenuated by interaction with walls [13], the pairwise correlation between RSSI from beacons in the same rooms is high compared to the correlation with beacons in other parts of the building. A measure of pairwise correlation between stochastic variables is the Pearson correlation coefficient (PCC) [20, p. 45]. It is for two arrays $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ each containing N observations of the stochastic variables x and y defined as

$$\mathrm{PCC}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \frac{\mathrm{cov}(\hat{\mathbf{x}}, \hat{\mathbf{y}})}{\sqrt{\mathrm{var}(\hat{\mathbf{x}})\mathrm{var}(\hat{\mathbf{y}})}}, \tag{2.2}$$

where $\mathrm{cov}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is the estimated covariance between x and y, and $\mathrm{var}(\hat{\mathbf{x}})$ is the estimated variance.

2.1.2 Fingerprinting

As mentioned in Section 1, one way to deal with RSSI-based positioning is to use fingerprinting techniques. This means that a positioning model for a specific building can be generated by recording RSSI-measurements and the corresponding locations in all areas of the building that is to be mapped. Positioning can then be performed in multiple ways by making use of these measurements. To deal with the difficulties posed by the unreliable nature of RSSI measurements described above, several machine learning approaches have been applied to the problem of fingerprinting [34].

2.1.3 Machine learning for indoor positioning

Indoor positioning is often considered a regression problem, since the goal is to predict two continuous x- and y-coordinates. Some approaches discretize the building space and perform multi-class classification [28]. In this thesis the focus was on regression, and therefore only machine learning methods relevant for regression are considered here.

The most commonly used approach for fingerprinting is the weighted k-nearest neighbour algorithm (WKNN) [34]. By defining a distance measure of the RSSI-space and a weighting scheme, the positions of the k-nearest neighbors in the fingerprinting dataset can be used for positioning. When using KNN-approaches the inference time is often affected due to the need of sorting distance measures to all the reference RSSI measurements in the fingerprinting database. By design, it also requires storing all the collected data for use in online positioning.

Other common techniques used for regression in fingerprinting are regression trees and random forests, support vector machines and various probabilistic approaches. These will not be covered further here, but the interested reader is referred to [34] for further explanations on how these methods have previously been used in fingerprinting. For this thesis the focus was on applying artificial neural networks for the problem of positioning based on RSSI to position correspondences. This decision is motivated further in chapter 3.

2.1.4 Artificial neural networks

Artificial neural networks (ANN) have shown a great capability of generalizing and learning complex functions [10, p. 5]. An ANN is a network of nodes and connections that maps an input to an output through a usually predetermined and fixed structure. The network structure is often divided into layers. This can be done in several ways, but in most applications it is required that the technique of back-propagation can be used to train the network. Back-propagation is the technique for updating weights in a neural network by starting at the output and then propagating the gradient backwards through the network [10, p. 211].

Each node in the network computes the function

$$y = \phi(b + \sum_{i=1}^{N} w_i x_i), \tag{2.3}$$

where y is the output, ϕ is the activation function, b is the bias and w_i are the N weights which are multiplied with the output values x_i from each of the input nodes. The choice of activation function is arbitrary but can have a great effect on the performance of the network. Three examples of activation functions used in this thesis are the *Sigmoid*, *Tanh*, and *LeakyReLU* functions. These are given by

$$\begin{cases} Sigmoid(x) = \frac{1}{1+e^x}, \\ Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \\ LeakyReLU(x) = max(\alpha x, x), \quad 0 < \alpha < 1, \end{cases} \tag{2.4}$$

where α is a constant ($\alpha = 0.01$ is the default value in Pytorch and is used in this thesis).

A simple version of a neural network is the multi-layer perceptron (MLP) which consists of an input layer, a number of hidden layers each containing a fixed number of nodes, and an output layer. The connections in an MLP are such that all adjacent layers are fully connected, i.e., a node in one layer receives an input from all other nodes in the previous layer. An example of an MLP is visualized in Figure 2.1 with input dimension three, output dimension one, and two hidden layers both with dimension two, where the arrows indicate the direction of the connections.

Common practice when training ANNs is to use three different sets of data, a training dataset, a validation dataset, and a test dataset [10, p. 118-119]. The training data is used for training the actual network by updating the weights using backpropagation. The validation dataset is used to determine hyperparameters for the network, such as layer dimensions and activation

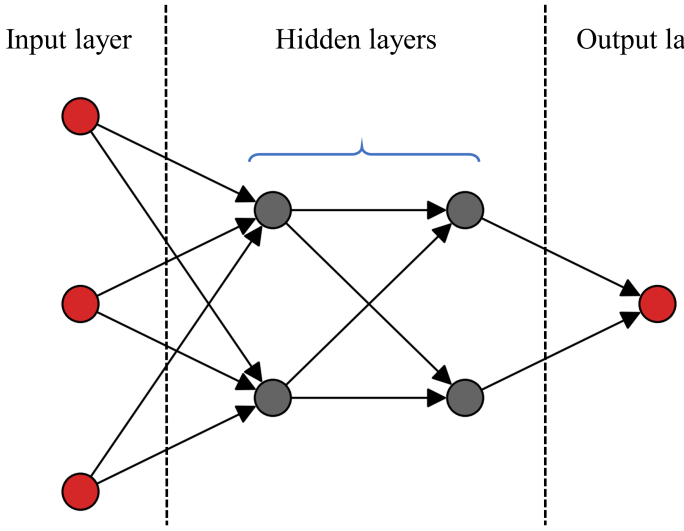


Figure 2.1: *Visualization of an example of a multi-layer-perceptron with input dimension three, two hidden layers both of dimension two, and a single output.*

functions. Finally the network is evaluated on a separate test-set, and it is the accuracy of this result that ideally should be representative of the network performance.

There are several techniques that can be applied during the training process to improve the performance of an ANN. Techniques used in this thesis are dropout, batch-normalization, early stopping, and the RMSProp optimizer. The science of optimizing the training procedure of ANNs is vast, and therefore these concepts will not be covered in further detail here. The interested reader is referred to [10] for further explanations of these concepts.

2.1.5 Augmentations

Data augmentation is a regularization technique that can be used to improve the generalizability of neural networks [10, p. 236-238]. The approach is based on manipulating the input data in such a way that it is in some way different to the original data, but also so that it should in theory result in the same output as the unobstructed data. An example of this could be a collection of RSSI measurements that are used as input to a positioning neural network. If 20 beacons are detected with a certain RSSI, then detecting 19 of these with the same values, but with one of the beacons missing, should still result in the same position estimate. Data augmentations are used extensively in this thesis, and the specifics are introduced in Section 3.

2.1.6 Autoencoders

Autoencoders are neural networks that are commonly used for dimensionality reduction, and extracting meaningful features from unlabeled data [10, p. 499-501]. A typical autoencoder is a mapping described by the function

$$\mathcal{F} : X \to X' \qquad \mathcal{F} = \mathcal{D} \circ \mathcal{E}. \tag{2.5}$$

$\mathcal{F}$ is a composition of two MLP's, $\mathcal{E}$, which is called the encoder, and $\mathcal{D}$, which is called the decoder. The output from the encoder $\mathbf{z} = \mathcal{E}(\mathbf{x})$ when applied to an input array $\mathbf{x}$ is called the latent space. The learning objective is commonly to minimize the reconstruction error

$$\mathcal{L}(\mathbf{x}) = ||\mathbf{x} - \mathcal{D} \circ \mathcal{E}(\mathbf{x})||^2. \tag{2.6}$$

A denoising autoencoder is an autoencoder where the input is in some way corrupted, but trained to reconstruct the original input. Letting ϕ be the corrupting function the new learning objective becomes

$$\mathcal{L}(\mathbf{x}) = ||\mathbf{x} - \mathcal{D} \circ \mathcal{E}(\phi(\mathbf{x}))||^2. \tag{2.7}$$

Denoising autoencoders can also be stacked and trained in a layer-wise greedy fashion. These are then called stacked denoising autoencoders (SDAE) [33]. SDAEs have in previous work been used for fingerprinting [1], [35], motivated by the noisy nature of the RSSI measurements. These types of models originate from the theoretical work on the model called $M1$ in [15], where the latent space of an autoencoder is used as input to an MLP to perform semi-supervised learning. Unlabeled data can then be used to train the autoencoder, which is shown to improve performance for the classical MNIST dataset [15].

2.1.7 Recurrent neural networks

A different type of ANN is the recurrent neural network (RNN) which is designed to handle sequential data. It does so by preserving a hidden state between each timestep. A typical architecture used when applying RNNs is visualized in Figure 2.2, where a sequence of arrays is used as input to an RNN, which preserves a hidden state between timesteps. In this case the figure shows a many-to-one RNN which means that the RNN only returns a single output, but several other setups are possible.

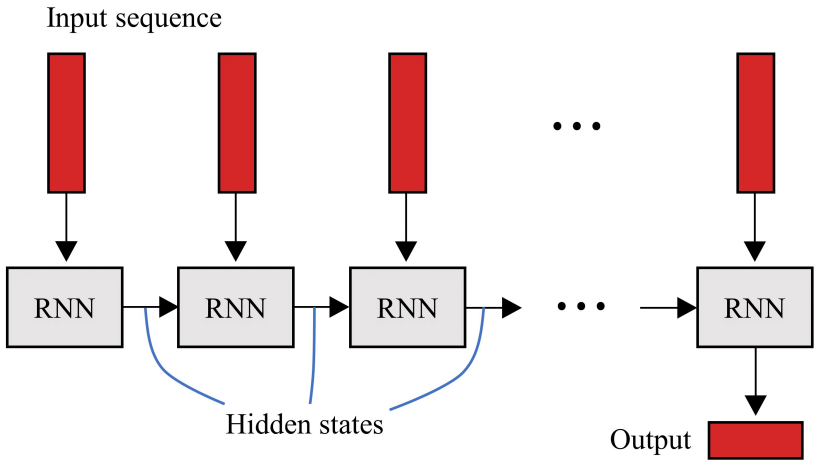


Figure 2.2: *Visualization of an example of a many-to-one recurrent neural network architecture.*

The LSTM-network (Long Short-term Memory) was introduced in 1997 and is a commonly used RNN architecture for data that consists of long sequences [10, p. 404-407]. The structure reduces the vanishing gradient problem, with the cost of including more trainable parameters than a classical RNN. The structure is more complex and will not be covered in detail here.

2.1.8 Ordinal classification

Classification is the task of assigning labels. In machine learning problems this is usually done by assigning a new datapoint a probability for each class, and then assigning it the label belonging to the class with the highest probability. In many applications the classes do not have any intrinsic order. However, when doing classification of floors, as in this thesis, the classes have a known relationship. When such order is available this information can be used to perform ordinal classification. Further details of the relevant implementation is provided in Section 3.

2.1.9 Ensembling

Ensembling is a method used in machine learning that involves combining the results from separate models to improve the overall performance. As described in [11, p. 605], there are multiple ways of performing ensembling, but in this thesis the concept is used as a way of combining resulting positioning estimates from different models.

2.1.10 Pedestrian dead reckoning

Pedestrian dead reckoning (PDR) involves estimating a relative position and orientation in relation to some frame of reference. Starting at a known position one can for example use sensor measurements from inertial sensors to estimate this relative displacement. Sensors used in common mobile devices are rarely of high quality, which means it is not feasible to describe the movement of the device using Newtonian laws of motion [7].

Instead, when performing positioning of a mobile device, a straight-forward PDR approach is to use the accelerometer, magnetometer and gyrometer to detect steps, and headings for each step. This task is made more complex if the algorithm is used in applications where it has to generalize for different users, walking styles and orientations.

Another factor that is relevant for PDR in indoor positioning is the fact that the magnetic field is influenced by metal structures in buildings and other metal objects. Recent approaches for indoor positioning [36] have due to this observation been considered using solely readings from the magnetometer. Since buildings can influence the magnetic field, the magnetometer readings that are used to determine the heading of a device can therefore be affected and lead to inaccurate PDR results.

2.1.11 Kalman filter and smoother

The Kalman filter and smoother are methods commonly used in tracking applications [6, p. 637]. The filter relies on forward recursion of a linear dynamical system while the smoother relies on backward recursion. For this reason the Kalman filter is mainly used when predicting future states, while the smoother is used to optimize a fixed set of known states. Depending on the underlying physical model of a system the Kalman filter and smoother can be set up in different ways, but with the main assumption that the system is linear and Gaussian. Furthermore, Kalman filtering is a generalization of least squares, also allowing states to change with time [29].

2.1.12 Optimization and weighted least squares

Ordinary least squares (OLS) is an approach for estimating solutions to overdetermined problems that are optimal in the least squares sense. An overdetermined linear system can be described by

$$A\mathbf{x} = \mathbf{b}, \qquad A \in \mathcal{R}^{m \times n}, \qquad m > n, \tag{2.8}$$

where A and $\mathbf{b}$ are known matrices of size $m \times n$ and $m \times 1$, and $\mathbf{x}$ is an array of unknown values [20, p. 219]. The OLS solution can then be derived from the normal equations and is given by

$$\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}. \tag{2.9}$$

The OLS solution is based on the assumption that the elements of the residual vector $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ are independent and homoscedastic, i.e., $\mathbf{r} \in \mathcal{N}(0, \sigma^2 I)$ [23]. When the residuals do not have the same standard deviation the problem is instead called heteroscedastic. The weighted least squares introduces weighting to the OLS formulation to compensate

for heteroscedasticity. It can be derived from the Gauss-Markov theorem that the optimal weighting matrix W for a heteroscedastic least squares problem is given by a diagonal matrix containing the reciprocal of the variances of the residuals [23]. This means that the weighted least squares solution is given by

$$\hat{\mathbf{x}} = (A^T W^T W A)^{-1} A^T W^T W \mathbf{b}, \qquad W = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \dots \\ 0 & \frac{1}{\sigma_2^2} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.10}$$

2.1.13 Variance component estimation

Methods of variance component estimation (VCE) provide a way to estimate the variance of dependent stochastic variables from observations. One way of estimating the variance components of a linear system is to use least squares variance component estimation (LS-VCE) [31]. The linear system can be described by

$$A\mathbf{x} = \mathbf{b} + \epsilon, \tag{2.11}$$

where it is assumed that the noise vector ϵ has the covariance matrix

$$Q = Q_0 + \sum_{i=1}^{k} \sigma_i^2 Q_i. \tag{2.12}$$

Here $Q_j, \quad j = 0, ..., k$ are known parts of the covariance matrix, while σ_i^2 are the unknown variances. The LS-VCE [31] gives that the estimate of standard deviations is

$$\begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \end{bmatrix} = (A_{vh}^T A_{vh})^{-1} A_{vh}^T \mathbf{y}_{vh}, \tag{2.13}$$

where

$$\begin{cases} A_{vh} = \begin{bmatrix} vh(B^T \begin{bmatrix} Q_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} B) & vh(B^T \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q_2 \end{bmatrix} B) \end{bmatrix}, \\ \mathbf{y}_{vh} = vh(B^T \mathbf{b}\mathbf{b}^T B). \end{cases} \tag{2.14}$$

Also, B is a matrix of which the column space spans the null-space of A^T and vh denotes the operation of stacking all elements in the upper triangular part of a matrix into a single column matrix.

2.2 Related work

A detailed overview of the subject of RSSI-based fingerprinting as of 2017 is provided in [34]. The authors describe methods using versions of k-nearest-neighbours (KNN), support vector machines, and neural networks. Due to the noisy nature of the RSSI measurements it is also noted that some type of filtering is usually applied to the output of the positioning algorithm, usually Bayesian filtering, Kalman filtering, or particle filtering.

Since 2017 several new methods have been proposed, such as recurrent neural networks (RNN) in [12], which use sequential measurements, and can therefore act as a denoising filter. Stacked denoising autoencoders have been used as a way to extract better features, and reduce the impact of noise and missing values [1], [35]. A generative model improving positioning at different times of the day was proposed by [5] using generative adversarial networks (GAN). A semi-supervised approach using a variational autoecoder combined with an MLP is described in [24].

A neural network for estimating relative positions, heading, and uncertainty of these values is described in [7], where it is referred to as IONet. This network makes use of the LSTM architecture. A common approach for then combining such estimates for displacements and orientations by sensor fusion is to use the Kalman filter [2], [30]. The Kalman filter was also combined with variance component estimation in [22] for outdoor positioning.

2.3 Kaggle

Kaggle is an online community which, among other things, hosts competitions in data science and machine learning. There are competitions both for learning purposes and competitions hosted by companies and corporations with the goal of solving a specific problem using a dataset that is made publicly available.

Chapter 3

Methodology

3.1 Strategy

To demonstrate previous knowledge of the unreliability of RSSI measurements an initial investigation of these measurements was carried out. This confirmed the signal attenuation from building walls, the fact that there is inherent noise in the RSSI measurements, and that beacons are sometimes not detected, even if the signal strength in theory should be high at the specific location. Previous knowledge and results of this investigation were then taken into consideration when designing a system for indoor positioning based on RSSI measurements.

The next step was to develop an approach that could alleviate the above problems specialized for the task of positioning, using known RSSI to position correspondences. Previous approaches and other regression algorithms were considered and initial tests and evaluations were performed. From these evaluations and previous work it was determined that a deep learning model was the most promising approach for applications within the GIN project.

Several ANN architectures and training procedures were then investigated to achieve a high positioning accuracy while maintaining simplicity for being applicable in real world applications, for a large number of buildings, and for online positioning. In previous work a common approach is to try to get an improved representation of the building environment using autoencoders, such as stacked denoising autoencoders [1], [35], and variational autoencoders [24]. Several such approaches were investigated, but results indicated that augmentations could achieve similar or better results. Therefore, the best developed model did not rely on this idea. This final model was evaluated on three different datasets.

After this model was developed it was investigated how the estimated positions could be improved by making use of sensor data from the inertial measurement unit (IMU) in a mobile device. Several ideas and methods were tested both for estimating relative displacements, and combining these with estimates of absolute position by using optimization.

The above steps will be described in further detail below, along with descriptions of additional algorithms and outcomes.

3.2 Datasets

Before going into detail about the positioning algorithms the datasets used will be presented. Three datasets were used for evaluating the algorithms described in this thesis. These were meant to be representative of different types of buildings where indoor positioning might be useful. Therefore, the datasets originated from three university buildings, one office environment, and 24 shopping malls respectively. A dataset can consist of single separate measurements or sequences of measurements from a user walking between known locations. Such sequences will be called tracks.

3.2.1 The UJIIndoorLoc dataset

A publicly available dataset for fingerprinting is called UJIIndoorLoc [32], and it has been frequently used in positioning research. The training dataset contains 19,937 measurements from three buildings, with 4–5 floors, collected with 25 unique mobile devices. For every position there are corresponding RSSI measurements with a dimension of 520, which means that in total 520 beacons were detected during collection of the training dataset. The UJIIndoorLoc dataset was used in the 2015 EvAAL competition[1]. The participants in the competition had access to a training dataset and a validation dataset. The proposed solutions were then tested on a separate test set to determine the outcome of the competition.

In this thesis part of the training set was used as validation data, and the official validation set was used as test data due to the fact that the competition test-set is not publicly available. The validation set of 1,111 points was collected a few months after the collection of training data, by different users, to represent a real-world positioning scenario. The partitioning of this dataset used in this thesis is visualized in Table 3.2.1, and details of the dataset are shown in Table 3.2.1. The percentage of measurements in each building in the training data is approximately B0: 25%, B1: 25%, B2: 50%, while in the test data it is on the other hand B0: 50%, B1: 25%, B2: 25%. This means that the accuracy achieved in building B0 has the highest impact on the total mean floor classification accuracy and positioning error. Figure 3.1 shows a visualization of the positions in the original UJIIndoorLoc dataset from the training and validation dataset.

Table 3.1: *Partitioning of the UJIIndoorLoc dataset used in this thesis in comparison with the original partitioning in the EvAAL competition for evaluation of the proposed positioning model.*

Original	Training data		Validation data	Test data
Used	Training data (90%)	Validation data (10%)	Test data	-

[1]EvAAL competition URL: http://evaal.aaloa.org/2015/competition-home

Table 3.2: *Properties of UJIIndoorLoc dataset.*

	Training data	**Validation data**	**Test data**
#measurements	17,943	1,994	1,111
#unique positions	692	604	1,068
collection period	May-June 2013	May-June 2013	Sep-Oct 2013

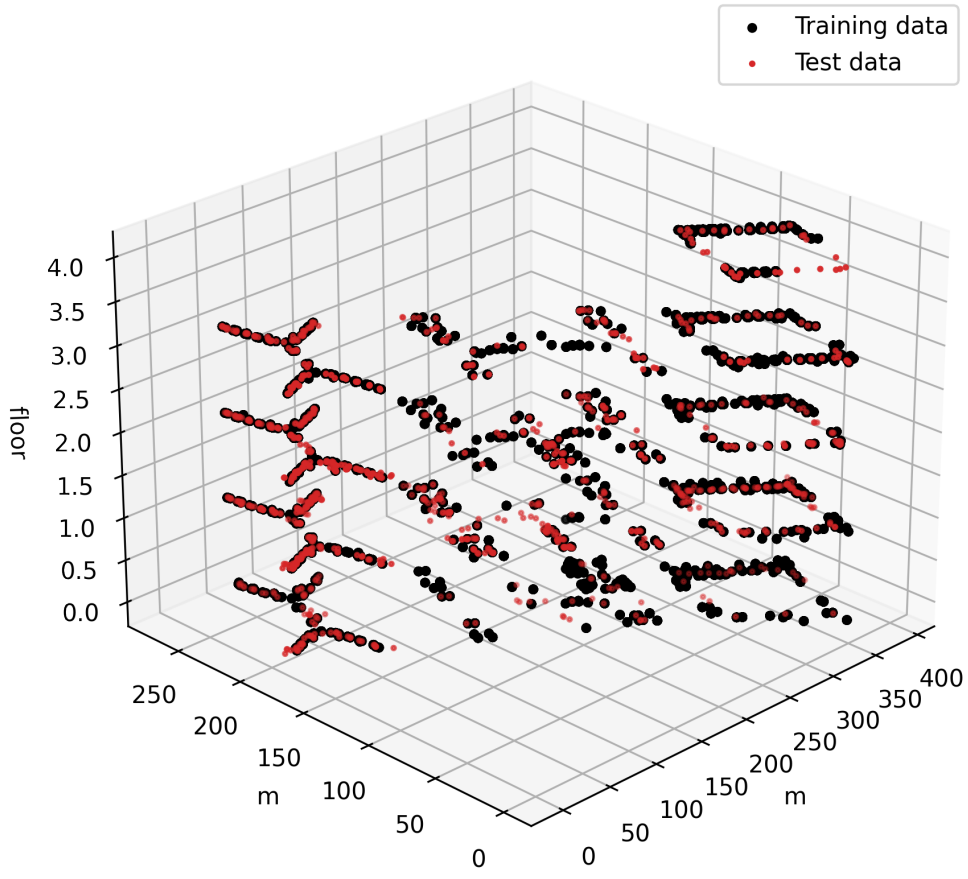


Figure 3.1: *Positions in the UJIIndoorLoc for the original partitioning into a training and validation dataset.*

3.2.2 The office dataset

The office environment, where the second dataset was collected, is a common office space containing different types of rooms, such as conference rooms, work areas, a kitchen, etc. The space is of dimensions 17×30 m. During the autumn of 2020 a number of tracks were collected manually using the app Traxmate, implemented by Combain, by multiple users and devices. The number of detected bssids was 1,341, and it was known that 64 of these bssids originated from beacons within the office space. The tracks were split into a dataset with training, validation and test tracks. Figure 3.2 shows the positions in the office dataset in the training, validation and test sets. Table 3.2.2 shows the number of measurements and positions in the different partitions of the dataset. The beacon positions and main inner walls had also been mapped in the office environment, and are shown in Figure 3.3. This information was mainly used to evaluate the generative model, and to get an understanding of where signal attenuation would occur.

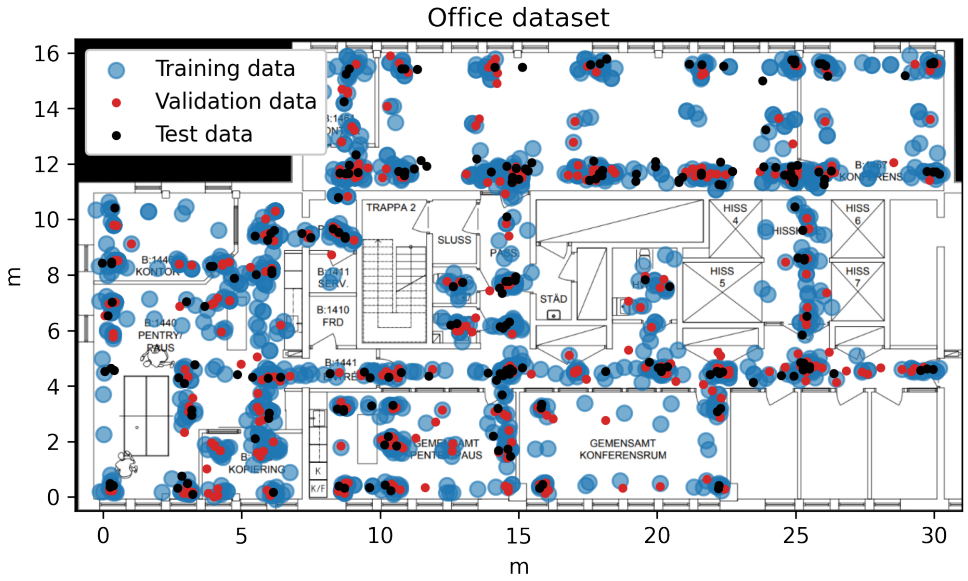


Figure 3.2: *Positions in the office dataset for training, validation and testing in the office environment.*

Table 3.3: *Properties of the office dataset.*

	Training data	Validation data	Test data
#measurements	1,680	421	297
#unique positions	1,680	421	297

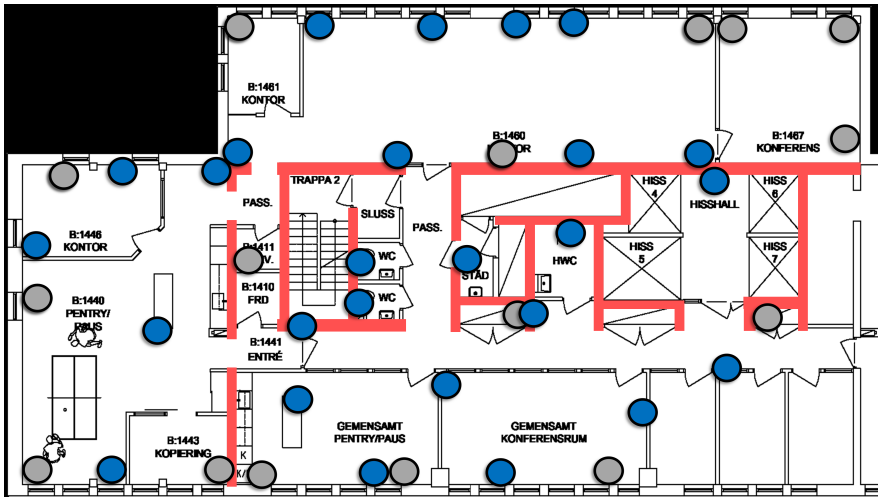


Figure 3.3: *Positions of Wi-Fi beacons (gray), Bluetooth beacons (blue) and main inner walls (red) in the office environment.*

3.2.3 The Kaggle dataset and competition

A few weeks into this master thesis a competition named "Indoor Location and Navigation" was announced on the Kaggle website. The competition was hosted by Microsoft, and the goal of the competition was to do indoor positioning. The dataset provided by the host contained data collected in 204 different shopping malls in China. Of these shopping malls 24 were relevant for the evaluation part of the competition. The evaluation metric was given by

$$S = \frac{1}{N}\sum_{i=1}^{N}(\sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2} + 15|\hat{f}_i - f_i|), \tag{3.1}$$

where $\mathbf{x} = [x_i, y_i, f_i]$ contains the true position for point i and $\hat{\mathbf{x}} = [\hat{x}_i, \hat{y}_i, \hat{f}_i]$ contains the predicted position. In other words, the score is based on the mean positioning error of all points in the test set, with the addition of a heavy penalty if the floor prediction is wrong. The timely publication of the Kaggle dataset enabled possibilities of further evaluations of the positioning methods developed in this thesis, and therefore it was decided to join the competition. The final month of the competition was executed in collaboration with a team, but the methods for absolute positioning, pedestrian dead reckoning and optimization presented in this thesis were developed independently of the team. An additional outcome of the competition was a collection of algorithms for using building floorplans and/or known reference points within the buildings, and these were on the other hand developed in collaboration with the team.

The dataset provided in the Kaggle competition consisted of data from 204 different shopping malls in China, which each contained approximately 5,000–20,000 Wi-Fi measurements with a corresponding ground truth position. The buildings were of approximate dimensions 100–700 m and had up to eight different floors. The number of unique detected beacons in the 24 buildings ranged from 490 to 7,030.

Since only 24 of the buildings were relevant in the competition, only measurements from those buildings were used for evaluation in this thesis. Each track in the dataset included, among other things, raw IMU-data, RSSI measurements, and the corresponding ground truth positions. The ground truth positions were specific locations in the buildings between which a user could walk, located in a grid-like pattern. This meant that the difference in time between ground truth positions was variable. The RSSI to position correspondences were extracted by interpolating in time between these known points to the time of each RSSI measurement. The data in one building in both the test- and training sets were collected by the same users, devices, and at the same time. This is different from the two above datasets.

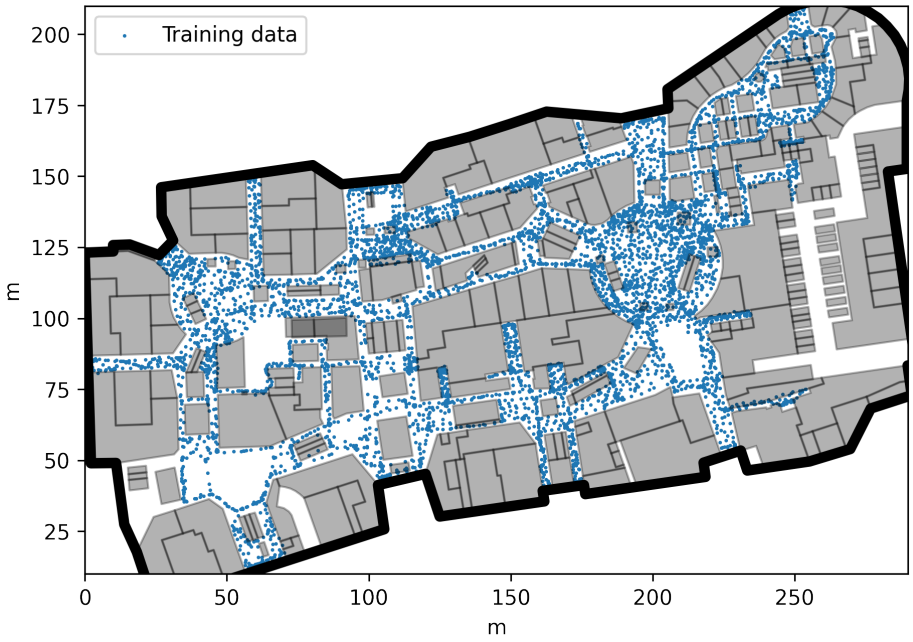


Figure 3.4: *Positions in the Kaggle training dataset from the ground floor of one building. Test data positions are not shown which were by design hidden by the competition host.*

The official Kaggle test-set contained 626 tracks with a total of 10,133 points taken from 24 different buildings. This partitioning was used for evaluating the estimates of absolute positioning models in this thesis. For evaluating the PDR-model on the other hand the tracks previously used for validation were instead used for testing, and the training set was further split into a new training and validation set. The two partitions of the Kaggle dataset for the two evaluations are shown in Table 3.2.3.

Table 3.4: *Partitioning of the Kaggle dataset used in this thesis in comparison with the original partitioning for evaluation of the fingerprinting and PDR model.*

Original	Training data			Test data
Fingerprinting	Training data (80%)		Validation data (20%)	Test data
PDR	Training (70%)	Validation (30%)	Test data	-

3.3 Investigation of RSSI characteristics

To provide an idea of the behavior of RSSI measurements a brief demonstration of important characteristics was performed. First, a mobile device was placed for one hour in line of sight of a chosen Wi-Fi beacon, and the RSSI was measured each five seconds. This was performed in an empty environment, free from disturbances at a distance of approximately 3 m from the beacon, and on a rotating plate to avoid standing waves. The experiment shows that the RSSI measured by the device contains noise despite the absence of environmental disturbances. Figure 3.5 shows a plot of the measurements with respect to time, and the approximate distribution of the same measurements together with the mean value.

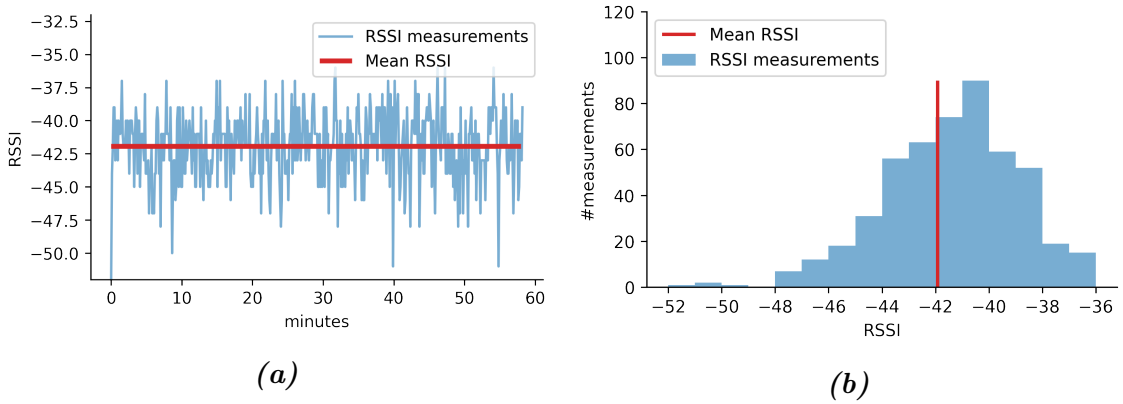


Figure 3.5: *Plot a) and histogram b) of RSSI measurements taken from one beacon during one hour each 5 seconds and the mean RSSI.*

The mean of the measurements was –41.9 dBm which corresponds to a distance of 3.1 m using the log distance path loss model in equation (2.1). Adding and subtracting the estimated standard deviation of 2.5 dBm from the mean value instead results in distances of 1.7 m and 5.6 m respectively, again by using equation (2.1). This shows that even without interaction with other devices, walls, people etc. an error corresponding to more than one meter can still be expected when considering a single beacon. Note that the test was run in an undisturbed environment, and the noise is expected to increase even more from interaction with people and other devices [34].

The influence of walls was also investigated. This was done by placing the mobile device at a distance of 2.5 m from a beacon with and without a wall between the device and the beacon for 15 minutes each, again measuring the RSSI each 5 seconds. Figure 3.6 shows the RSSI distributions for the two measurements. The mean values of the distributions were –41.0dBm and –51.9 dBm respectively, which corresponds to distances of 2.5 m and 31.1 m according to the log distance path loss model. The signal attenuation due to interaction with the thick wall had in this case introduced a corresponding error in distance that is higher than 25 m.

The number of beacons and their positions and ID's in Combain Office had been previously mapped. This motivated an investigation into the correlation between pairs of Wi-Fi and

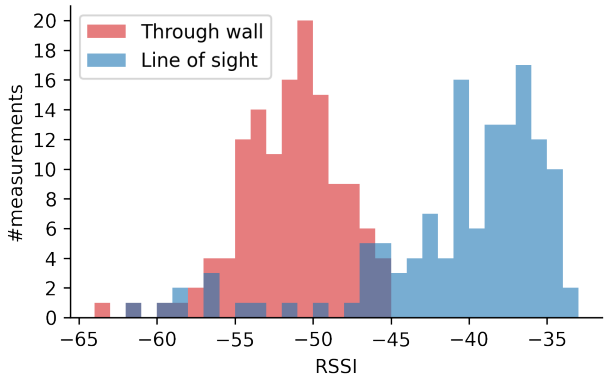
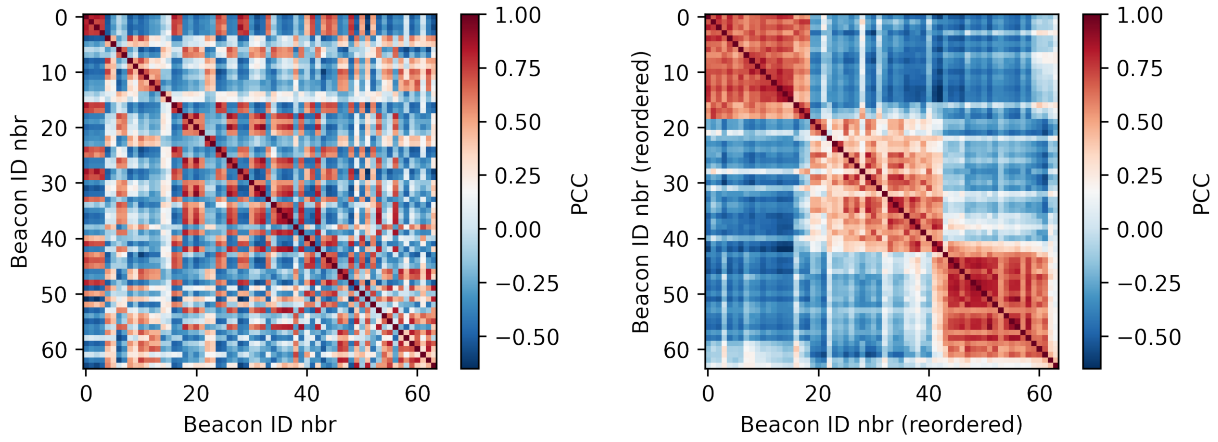


Figure 3.6: *Distributions of RSSI measurements collected during 15 minutes at a distance of 2.5 m from the beacon, with and without a wall in between.*

Bluetooth beacons. By making use of the office dataset the pairwise PCCs were estimated for the 64 known bssids to provide further insight into the behaviour of the RSSI measurements. Figure 3.7a shows a visualization of the pairwise PCCs for these 64 beacons. Clustering the beacons with regard to the PCC assuming three clusters, and reordering the rows and columns in the matrix according to the clustering, resulted in the matrix shown in Figure 3.7b. Visual examination of this matrix confirms the three-cluster assumption, which is consistent with the fact that the office environment has three large open spaces separated by the main walls, as can be seen in Figure 3.3.



(a) ***(b)***

Figure 3.7: *Visualizations of beacon pairwise PCC matrices for the 64 known bssid's in the office environment, a) before clustering, and b) after clustering and reordering rows and columns.*

One other complication of RSSI measurements, in addition to the noise, is that beacons are not always detected. Figure 3.8 shows a visualization of RSSI's for data points in the office dataset for three different beacons. When the beacon has not been detected the point is marked by a small red dot. As can be seen in several measurements of RSSI the beacon

has not been detected, even if it in some instances is less than a few meters away from the measuring device.

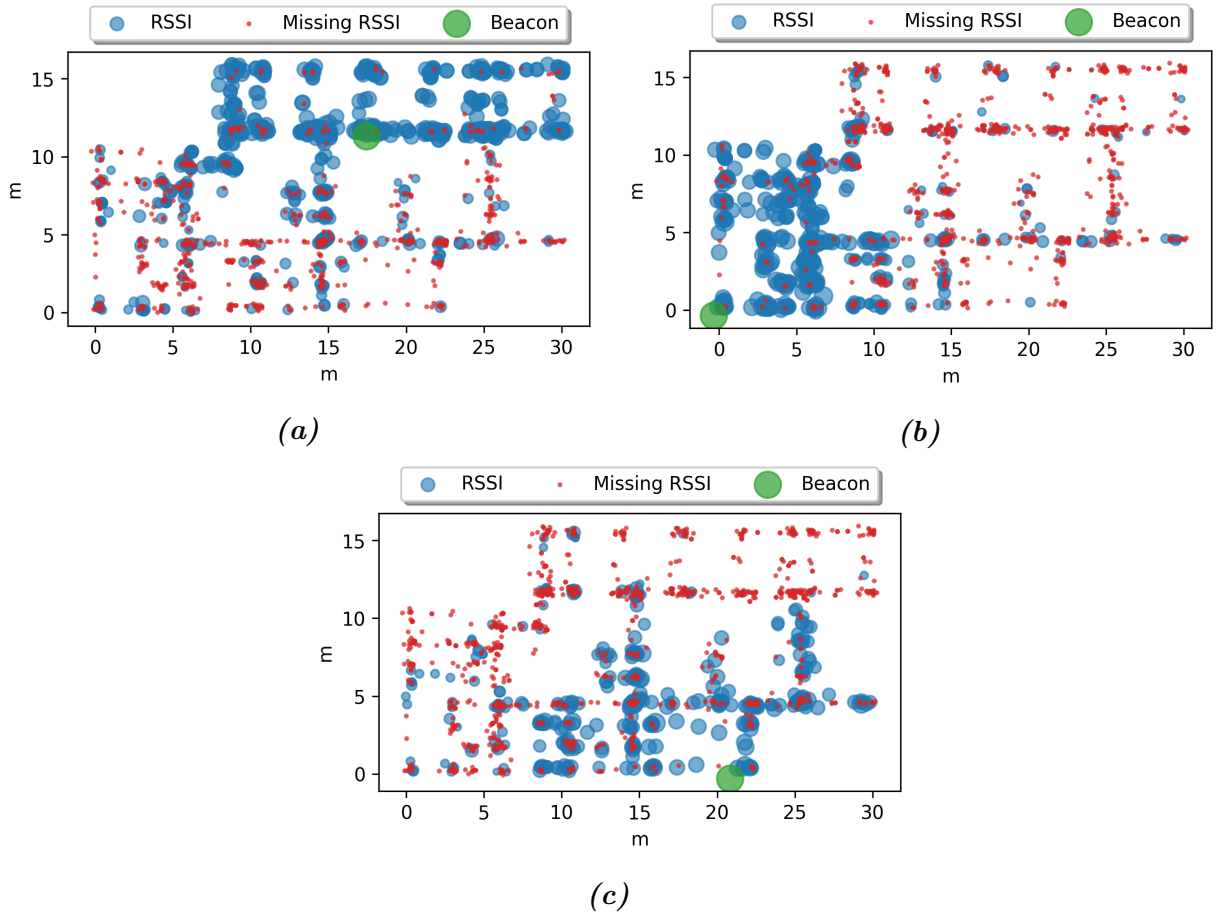


Figure 3.8: *RSSI for three different beacons at all positions in the office dataset. The size of blue indicates the magnitude of RSSI for the beacon, red indicates that the beacon has not been detected, and the green dot shows the true beacon position.*

3.4 RSSI-based positioning model

With the above investigations and datasets in mind, a custom neural network architecture and training procedure was constructed. The main features of the structure of this model are visualized in Figure 3.9.

The model was designed to be trained on RSSI to position correspondences in a classical fingerprinting dataset, such as the UJIIndoorLoc dataset. Each value in the input corresponds to the RSSI from a beacon (normalized from the interval $[-100, 0]$ to $[0, 1]$). All beacons that have been detected in the building dataset contribute with one input value, which means that the input dimension is the same as the number of detected beacons in the training data. When performing positioning for a new measurement only the RSSI's from these beacons

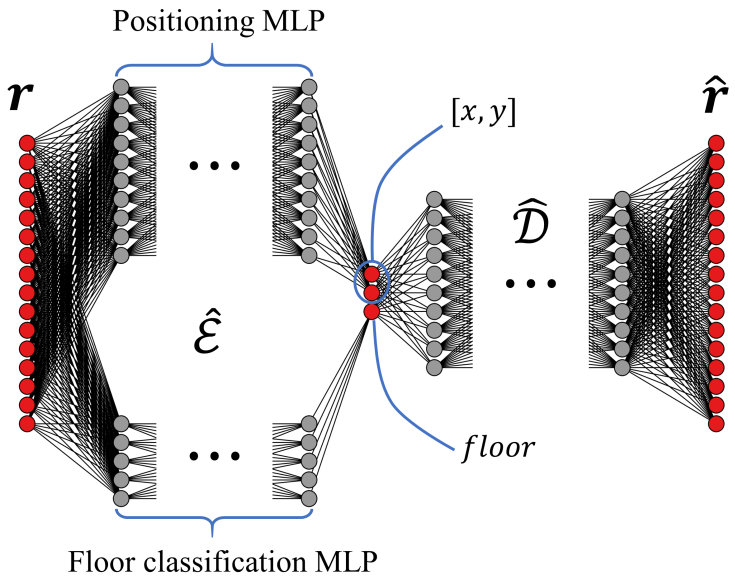


Figure 3.9: *Visualization of the custom neural network for positioning floor classification and RSSI-map generation. The model maps input RSSI to x, y and floor coordinates in the array* $\hat{\mathbf{x}}$*, and then reconstructs the input from the position estimate.*

are considered. A beacon that is not detected is assigned the value 0, which means it is missing. Below follows a mathematical motivation for the structure of the model.

Assume that

$$\tilde{R}(\mathbf{x}_i) = R(\mathbf{x}_i) + e(\mathbf{x}_i), \tag{3.2}$$

is a vector-valued function that describes the RSSI at point $\mathbf{x}_i \in \mathcal{R}^3$ at timestamp i inside a building for all detectable beacons, and for a single device. Assume that $e(\mathbf{x}_i) \in \mathcal{N}(0, I\sigma^2(\mathbf{x}))$ is the inherent noise of each beacon at position $\mathbf{x}_i$, with covariance matrix $I\sigma^2(\mathbf{x}_i)$, where $\sigma^2(\mathbf{x}_i)$ is the variance of the signal from each beacon at position $\mathbf{x}_i$. This means that $E(\tilde{R}(\mathbf{x}_i)) = R(\mathbf{x}_i)$, where $E(\cdot)$ denotes expected value.

The method of positioning based on RSSI can then be described as a function $\mathcal{E}$ that computes the mapping

$$\mathcal{E} : R(\mathbf{x}_i) \to \mathbf{x}_i. \tag{3.3}$$

Now assume that

$$R(\mathbf{x}_1) \neq R(\mathbf{x}_2), \quad \forall \mathbf{x}_1 \neq \mathbf{x}_2, \tag{3.4}$$

i.e., that the expected RSSI-array detected at position $\mathbf{x}_i$ is unique for all possible positions inside the building. This would mean that the mapping $\mathcal{E}$ is invertible. Let $\mathcal{D}$ represent the inverse mapping of $\mathcal{E}$ according to

$$\mathcal{D} : \mathbf{x}_i \to R(\mathbf{x}_i). \tag{3.5}$$

The mappings $\mathcal{E}$ and $\mathcal{D}$ can then be estimated with neural networks. Let $\hat{\mathcal{E}}$ be an MLP approximating $\mathcal{E}$, and $\hat{\mathcal{D}}$ a different MLP approximating $\mathcal{D}$. A neural network $\mathcal{F}$ can then

be constructed implementing the function

$$\mathcal{F}(\mathbf{r}_i) = \hat{\mathcal{D}} \circ \hat{\mathcal{E}}(\mathbf{r}_i) = \hat{\mathbf{r}}_i, \tag{3.6}$$

where $\hat{\mathbf{r}}_i$ is an approximation of $\mathbf{r}_i$. The neural network, which has similarities with a classical autoencoder, is trained to minimize a loss function which for each sample i is

$$L(\hat{\mathbf{x}}_i, \hat{\mathbf{r}}_i) = C\frac{1}{n_R}||\hat{\mathbf{r}}_i - \mathbf{r}_i||^2 + \frac{1}{n_x}||\hat{\mathbf{x}}_i - \mathbf{x}_i||^2, \tag{3.7}$$

where n_R is the input dimension, n_x is the dimension of the position, and C is a hyper-parameter that determines how much the reconstruction error should affect the loss in relation to the positioning error.

3.4.1 Ordinal floor-classification

In cases where the altitude is not known, but only the floor, it is in that case suggested that the third variable in $\hat{\mathbf{x}}_i$ is considered as a continuous variable for performing ordinal classification. The neural network is designed such that the variables in $\hat{\mathbf{x}}_i$ are fixed between 0 and 1 by use of the sigmoid activation function. This means that the internal representation of the building is scaled by width, length and height (or number of floors). In turn, this means that it is possible to fix a probability function for the continuous variable f for use in ordinal classification. For this purpose the cumulative logistic link function is used, defined for n_f number of floors as

$$P_{floor}(f|k) = \begin{cases} \sigma(q_0 - f) & \text{if } k = 0, \\ \sigma(q_k - f) - \sigma(q_{k-1} - f) & \text{if } 0 < k < n_f, \\ 1 - \sigma(q_{n_f - 1} - f) & \text{if } k = n_f, \end{cases} \tag{3.8}$$

where $\sigma(x)$ is the sigmoid function (see Equation (2.4)), and $q_k = k/n_f \quad k \in [1, n_f - 1]$ is the breakpoint separating the classes for floor k and floor $k+1$. The resulting loss function for one sample when using ordinal floor classification instead of three-dimensional fingerprinting is

$$L(\hat{\mathbf{x}}_i, \hat{\mathbf{r}}_i) \;=\; C\frac{1}{n_R}||\hat{\mathbf{r}}_i \,-\, \mathbf{r}_i||_2^2 \,+\, \frac{1}{n_x - 1}||\hat{\mathbf{x}}_i^{\{1-2\}} \,-\, \mathbf{x}_i^{\{1-2\}}||_2^2 \,-\, D \log P_{floor}(\hat{\mathbf{x}}_i^{\{3\}}|\mathbf{x}_i^{\{3\}}), \tag{3.9}$$

where D is an additional weighting parameter for the floor loss, and $\mathbf{x}^{\{i\}}$ denotes the ith entry of $\mathbf{x}$.

In this thesis the main focus was on improving positioning accuracy. Floor classification was also considered a less complex task for a neural network to perform than regression of a continuous altitude, which could in theory have been done. For these reasons the positioning section of the network $\hat{\mathcal{E}}$ was split into two parts, a floor-classification MLP, and a positioning MLP. This version of the model is what is used in the experimental part of this thesis, and is visualized in Figure 3.9. The trained model can be split into three parts that can be used separately. The upper part of $\hat{\mathcal{E}}$ in Figure 3.9 estimates the position, the lower part estimates the floor, and the right part of the network $\hat{\mathcal{D}}$ estimates an expected RSSI array at a specific location inside the building.

3.4.2 Training procedure

To improve the performance of the positioning model several known techniques were applied during the training procedure. The main components of this procedure were batch normalization, dropout, the RMSProp optimizer, the choice of activation functions and augmentations.

Batch normalization was applied between each layer in all three MLPs, using the default parameters in Pytorch. To avoid overfitting and to improve the model's ability to generalize dropout at a rate of 0.1 between all layers in the network was also used. The RMSProp optimizer was applied for updating the network weights. A learning rate of 0.0003 was chosen for this optimizer, which was otherwise used with the default parameters. The activation functions used between all hidden layers was the LeakyReLU function. It was chosen both on the basis of best performance, but it also has a low computational complexity during inference due to piece-wise linearity (see Equation (2.4)).

Finally, the used augmentative scheme consisted of three obstructions of the input RSSI. The proposed augmentations are based on previous knowledge of useful augmentations in similar scenarios [26], but also on the observation that the RSSI strength can vary depending on device properties [19].

- **Augmentation 1 - Gaussian noise:** The first augmentation was based on adding Gaussian noise to the original RSSI measurements. Since the original measurements in the training data already contains noise it does not follow that the standard deviation of the added noise should be as large as the standard deviation of the inherent noise of the original signal. Experimentally it was determined that a standard deviation of 1 dBm of this augmentation was the most effective level.
- **Augmentation 2 - Additive offset:** The second augmentation involved adding an offset to the input RSSI. The main idea behind this augmentation was that the relationship between the RSSI from different beacons should be the main influence for determining the position, and not the independent levels of RSSI. This augmentation can therefore help the network to generalize for new devices. The chosen limits of this augmentation was ± 5 dBm. This means that a random offset in this interval is added to all values in the input array. Measurements in the input array that through this augmentation end up larger than 0 or smaller than -100 were subsequently snapped back into this range.
- **Augmentation 3 - Beacon-dropping:** Finally dropping of beacons was applied to the input array. In this implementation beacons were dropped with probability 0.2. This means that on average 80% of the original RSSI values are used in each measurement when training the network. This is a known technique when for example considering denoising autoencoders which learn to reconstruct an improved representation of a noisy input [10, p. 504-505]. Another reason for this augmentation was that two RSSI arrays that only differ in that some of the beacons have not been detected, but all other RSSI values are the same, should still result in the same position. The value of the beacon dropping probability 0.2 was also determined experimentally.

The main settings, hyper-parameters and dimensions used during training of the network are presented in Table 3.5.

Table 3.5: *Settings and architecture of the proposed model when trained on the UJIIndoorLoc, Office and Kaggle datasets.*

Dimensions of $\hat{\mathcal{E}}$ (positioning)	500-400-300-200-100-50
Dimensions of $\hat{\mathcal{E}}$ (floor)	100-50
Dimensions of $\hat{\mathcal{D}}$	50-100-200-300-400-500
Batch size	300
Optimizer	RMSProp
Training stabilization	Batchnormalization
Activation function	LeakyReLU
Learning rate	0.0003
Noise augmentation std.	1 dBm
Offset augmentation limits	±5 dBm
Missing input augmentation	20%
Dropout	0.1
Reconstruction loss weight C	0.1
Floor classification loss weight D	0.0001

3.4.3 Evaluation

The positioning neural network was evaluated on the three separate datasets presented above. In addition to these evaluations further tests were carried out using only the UJIIndoorLoc dataset to investigate various properties of the architecture and training procedure.

To test the inference time the network was moved from Pytorch to Numpy, and each of the 1,111 test measurements were run sequentially through the network of the corresponding building. This was done to avoid bias from using the GPU during inference, which is not always available. The result was obtained by averaging over 100 runs.

The manual collection of ground truth positions can be time consuming, as mentioned above. Therefore a test of the feasibility of using the method with less available ground truth data was performed. This was done by randomly excluding varying percentages of data from the training set. For each building the training dataset was split into subsets of 50%, 10%, 5% and 1% of the original dataset. This corresponds to approximately 2500, 500, 250 and 50 points respectively for building 0 and 1, and 4500, 900, 450 and 90 points for building 2. One positioning model was trained for each building, percentage and subset, and the positioning accuracies were computed on the test set of 1,111 measurements. The same settings of parameters and model dimensions were used as presented in Table 3.5, except for the batch

size which was decreased according to 300, 200, 100 and 20 for the four different percentages of used data. For consistency the same validation dataset as in the previous experiment was used, which means that a model trained on for example 50 ground truth data points also uses 238-487 additional points for validation in early stopping of the training procedure, depending on which building was considered. For comparison both the baseline WKNN and the M1_SDAE models (described below) were tested in this approach as well. The parameter k in the baseline WKNN was decreased according to 7, 6, 5, 4 and 3 to compensate for the decrease in data. The first four layers of the M1_SDAE was trained using all data, while the MLP was trained with the same decrease in batch-size as given above.

The total loss that is minimized during training of the network is a sum of three loss components, see equation (3.9). Therefore, it is of interest to make sure that all three components behave as expected during a training session, and not just the sum. For this reason the three components of the loss were recorded for both the training and validation data during one training session at each epoch.

Then the effect of the reconstruction weight C on the positioning accuracy of the network was investigated. This was done for building B1 in the UJIIndoorLoc dataset. The original idea of including the generative model $\hat{D}$ in the network was to enable the use of unlabeled data during training by only considering the reconstruction error for those samples. This approach did not improve the accuracy of the model. However, the addition of the reconstruction loss can still affect the final positioning accuracy when using only labeled data. The values of the reconstruction weights tested were $C \in [0.001, 0.01, 0.1, 1, 10]$, and ten models were trained and evaluated on the test data for all five values of the reconstruction weight. A value of $C = 0.001$ in this context corresponds to almost training the two MLP's for positioning and floor-classification on their own, and not taking the reconstruction into consideration. On the other hand a value of $C = 10$ corresponds to training an autoencoder, with less consideration of the values of the latent space.

Finally it was investigated how the augmentations affected the network performance. This was done by simply training the network for all buildings in the UJIIndoorLoc dataset, but excluding some, or all, augmentations of the input.

3.4.4 Probabilistic version

When estimating the absolute position of a device it is often useful to know the accuracy of the resulting position estimate. The original version of the neural network does not provide any such information of the certainty of an estimated position. A probabilistic version was therefore implemented to provide additional information about the resulting position estimates.

The approach was to let the output of $\hat{\mathcal{E}}$ be two Gaussian distributions instead of two coordinates. This means that the new output array of $\hat{\mathcal{E}}$, losing the point index i for convenience, is given by

$$\hat{\mathbf{x}} = \hat{\mathcal{E}}(R) = \begin{bmatrix} \hat{\mu_1} \\ \hat{\sigma_1} \\ \hat{\mu_2} \\ \hat{\sigma_2} \\ \hat{f} \end{bmatrix}, \tag{3.10}$$

where the position coordinates are now assumed to belong to Gaussian distributions $\hat{x} \in \mathcal{N}(\mu_1, \sigma_1)$ and $\hat{y} \in \mathcal{N}(\mu_2, \sigma_2)$. To train the network the loss function for one data point is altered from the form in equation (3.9) to

$$L(\hat{\mathbf{x}}, \hat{\mathbf{r}}) = C\frac{1}{n_R}||\hat{\mathbf{r}} - \mathbf{r}||_2^2 - \log(P_{coord}(\mathbf{x}^{\{1-2\}}|\hat{\mathbf{x}}^{\{1-2\}})) - D\log(P_{floor}(\mathbf{x}^{\{3\}}|\hat{\mathbf{x}}^{\{3\}})), \tag{3.11}$$

where

$$P_{coord}(\hat{\mathbf{x}}|\mathbf{x}) = \frac{1}{2\pi\hat{\sigma}_1\hat{\sigma}_2} exp\left[\frac{(\mathbf{x}^{\{1\}}-\hat{\mu}_1)^2}{\hat{\sigma}_1^2} + \frac{(\mathbf{x}^{\{2\}}-\hat{\mu}_2)^2}{\hat{\sigma}_2^2}\right]. \tag{3.12}$$

The network can then be trained in the same way as described above, with the exception of the need to change the weights C and D due to the different nature of the positioning loss.

The estimated standard deviations in the final model of such a neural network are commonly known to be biased, and needs calibration [27]. Due to time constraints this calibration was not performed. However, to get an idea of whether the network is able to learn an estimate of the magnitude of the error, the PCC was computed for the true errors and estimated standard deviations when performing inference on the test set.

3.4.5 Models for comparison

Three of the other models implemented for positioning are used for comparison with the above proposed model. These are a weighted k-nearest neighbours (WKNN) model, a baseline random forest regressor (RFR) and a model called M1_SDAE based on the M1-model [15] and the stacked denoising autoencoder (SDAE). To determine the optimal parameters for these models the same training/validation split as above was used.

The WKNN was implemented in the same way as the WKNN model described in [8]. This means that the model considers the k-nearest neighbours in the RSSI-space according to the Sorensen distance metric, and weights the corresponding positions according to the reciprocal of these distances. The Sorensen distance measure between two preprocessed measurement arrays of RSSI $\mathbf{p}_1$ and $\mathbf{p}_2$ is given by

$$D(\mathbf{p}_1, \mathbf{p}_2) = \frac{\sum_{i=1}^{N} |p_{1i} - p_{2i}|}{\sum_{i=1}^{N} |p_{1i} + p_{2i}|}. \tag{3.13}$$

See [8] for details of the preprocessing steps. The inference time for the baseline WKNN was computed on the UJIIndoorLoc test set, building-wise, by running each of the 1,111 test

measurements sequentially through the regression algorithm using the `BallTree` algorithm in Sklearn. The result was obtained by averaging over 100 runs.

The baseline RFR-model is based on the `RandomForestRegressor` in the python library Sklearn. It uses 100 randomly generated decision trees to perform regression of the RSSI-input. Random forest approaches were not developed further due to the fact that other approaches were considered more promising for dealing with the RSSI-variation problem.

Several types of autoencoders, such as ordinary autoencoders, variational autoencoders and stacked denoising autoencoders were considered for use in the M1 model [15]. Since it was observed that these exhibited similar behaviour when performing semi-supervised learning only the best performing model is considered here, called M1_SDAE. This model has a similar structure to the proposed model above, with the difference that the first four hidden layers of dimensions 500, 400, 300 and 200 are trained separately as denoising autoencoders which are then stacked as described in [33]. The weights in these layers are then fixated, and the output is used as an input to a subsequent MLP of dimensions 100-50-2. Improved performance was observed when training the final MLP with the same augmentations as used above, so these were once again used during the second part of the M1 training procedure. As mentioned above, when trained on less labeled data all data was still used to train the initial layers of the SDAE.

3.5 IMU-based PDR model

The mobile application Traxmate, implemented by Combain, includes a system for pedestrian dead reckoning that is based on step detection and estimation of headings between timestamps of interest. Since the heading estimates rely on the magnetic field it was also of relevance for Combain and the GIN research project to investigate alternative ways of performing pedestrian dead reckoning that could solve the problem of magnetic field disturbances from building structures. With access to the Kaggle dataset containing raw IMU data in addition to known positions inside buildings it was deemed feasible to test if estimated positions could be used to improve the PDR accuracy.

To address the task of multiple devices and walking styles a similar structure to the neural network IONet [7] was implemented. The main idea as described in the article is a sequence based physical model, which from a known initial position and orientation estimates a displacement length and a new orientation at each timestep. The IONet leverages raw measurements from the accelerometer and gyrometer, which amounts to six independent variables that are used as input to the network.

Now consider a situation where the initial orientation is not known, but instead it is possible at all times to get an estimate of the true orientation in a global frame of reference. Two of the outputs obtained from tracks in the Kaggle dataset were from the Android sensor types TYPE_ROTATION_VECTOR and TYPE_ACCELEROMETER. The output from the sensor TYPE_ROTATION_VECTOR contains three values that can be recomputed to the azimuth, pitch and roll $z, p, r \in [0, 2\pi]$. The azimuth is the rotation of the device in the horizontal plane, which means this is the most relevant feature for pedestrian dead reckoning,

as the problem at hand is mainly two-dimensional. The pitch and roll were still used as input to the neural network, since these can be an indication of speed, walking style etc.

The azimuth changes from 0 to 2π instantly when the heading changes from north-west to north-east. To deal with this discontinuous behavior, the first input to the neural network is not the raw azimuth, but instead $\sin(z)$ and $\cos(z)$ to achieve a continuous representation of the heading. Since only the problem of a user walking with the phone laying flat in the hand the pitch and roll were not altered since these values do not significantly change over time. The input to the LSTM network was then a sequence of M measurements $\mathbf{x}_{LSTM}^{\{i\}}, i \in 1, ..., M$, while the output is a single array y_{LSTM} given by

$$\mathbf{x}_{LSTM}^{\{i\}} = \begin{bmatrix} \sin(z_i) \\ \cos(z_i) \\ p_i \\ r_i \\ a_x i \\ a_y i \\ a_z i \end{bmatrix}, \qquad \mathbf{y}_{LSTM} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \tag{3.14}$$

where $\mathbf{a}_i = [a_{xi}, a_{yi}, a_{zi}]$ is the accelerometer reading at timestamp i. For training stability all inputs were normalized based on the distribution of values observed in the training data, or based on the interval $[0, 2\pi]$ for the angular inputs.

The main difference between the implemented model in this thesis and IONet is the input and output of the network. The main assumptions are still valid, but the model is trained to estimate a displacement between two timestamps as a many-to-one model instead of continuously estimating a new position and orientation. The main characteristics of the customized LSTM-network are presented in Table 3.6, and the training procedure was implemented in Pytorch.

Table 3.6: *Settings and architecture of the modified version of IONet [7] for the many-to-one model in this paper.*

Recurrent unit	2-layer LSTM, 50 hidden nodes
LSTM to output MLP dimensions	100-100
Dropout	0.25
Sliding window width	80
Stride	40

The evaluation of the PDR-model was done in a slightly different fashion than the fingerprinting model since the ground truth positions of the test-set were hidden in the Kaggle dataset. Therefore, the model was evaluated on the 548 tracks chosen as validation set for the absolute positioning model, and trained on the remaining data. This partitioning is visualized in Table 3.2.3.

To incorporate estimated positions in the model to compensate for building specific disruptions to the magnetic field it was tested to alter the training procedure. The original

LSTM was trained as before, but one MLP mapping the internal state of the LSTM to a displacement was subsequently trained for each building. The absolute position was then also used as input to the MLP. The position coordinates were again normalized between 0 and 1. This approach did not improve the positioning result when trained on the Kaggle dataset, and possible reasons for this are discussed in section 5. However, the results from using the base version of the model showed a clear improvement compared to the classical step-counting approach, which is why it is presented in the results, and was subsequently used for the optimization algorithm.

3.6 Sensor fusion by optimization

The two above described models result in estimates of absolute position and relative position. This leads to the problem of sensor fusion. The approach for combining these two estimates used here is based on weighted least squares (WLS). Consider a track with N points of interest at which RSSI measurements have been made. This means that the fingerprinting algorithm will provide N estimates of absolute positions and the relative positioning will be used to generate $N-1$ displacement vectors.

Let $X = [\mathbf{x}^T, \mathbf{y}^T]^T$ contain the true x- and y-coordinates for the N points, and $\Delta X = [\Delta\mathbf{x}^T, \Delta\mathbf{y}^T]^T$ contain the true $N-1$ displacement vectors. It then holds that

$$\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix} = \begin{bmatrix}E & \mathbf{0}\\ \mathbf{0} & E\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}, \qquad E = \begin{bmatrix}-1 & 1 & 0 & \dots\\ 0 & -1 & 1 & \dots\\ \vdots & \vdots & \vdots & \ddots\end{bmatrix}. \tag{3.15}$$

Now let $\hat{X}$ contain the N point estimates from fingerprinting and $\hat{\Delta X}$ the $N-1$ displacement estimates from the LSTM. Then consider the residual arrays

$$\begin{cases}\mathbf{v}_1 = \hat{X} - X,\\ \mathbf{v}_2 = (\widehat{\Delta X} - \Delta X) = (\widehat{\Delta X} - EX).\end{cases} \tag{3.16}$$

Assume that the residual vectors are normally distributed according to

$$\begin{cases}\mathbf{v}_1 \in \mathcal{N}(0, \sigma_1^2 Q_1),\\ \mathbf{v}_2 \in \mathcal{N}(0, \sigma_2^2 Q_2),\end{cases} \tag{3.17}$$

where $\sigma_2^1 Q_1$ is the covariance matrix for $\mathbf{v}_1$ and $\sigma_2^2 Q$ is the covariance matrix for $\mathbf{v}_2$. If the residuals are independent it holds that $Q_1 = I$ and $Q_2 = I$.

To obtain an optimal estimate, X^*, of the track positions the weighted least squares minimization problem is given by

$$X^* = \arg\min_X w_1||\mathbf{v}_1(X)||_2^2 + w_2||\mathbf{v}_2(X)||_2^2, \tag{3.18}$$

where w_1 and w_2 are weights. The best linear unbiased estimate X^* (BLUE) is according to [23] given when the least squares is weighted according to

$$\begin{cases} w_1 = \frac{1}{\sigma_1^2}, \\ w_2 = \frac{1}{\sigma_2^2}. \end{cases} \tag{3.19}$$

When training a neural network, which both the absolute and relative positioning is based on, there is access to ground truth data. Due to having access to a test set it is possible to estimate a standard deviation of the positioning error for a trained positioning model. Assuming that residuals v_{1i} in $\mathbf{v}_1$ for x- and y-coordinates are normally distributed and independent, the positioning error is Rayleigh distributed. If the mean positioning error for the fingerprinting model is μ_1 and the relative positioning model is μ_2, then this corresponds to a standard deviation for the two models given by

$$\begin{cases} \sigma_1 = \sqrt{\frac{2}{\pi}}\mu_1, \\ \sigma_2 = \sqrt{\frac{2}{\pi}}\mu_2. \end{cases} \tag{3.20}$$

Since a relative positioning model that only uses inertial sensors can be used in any environment, this standard deviation is assumed to be constant. However, the fingerprinting model in different buildings will have different accuracy depending on the number of beacons, building structure, etc. This means that the weight for residuals v_{1i} when optimizing tracks in different buildings will vary. Let the matrices A and b be given by

$$A = \begin{bmatrix} I \\ E \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \hat{X} \\ \hat{\Delta X} \end{bmatrix}. \tag{3.21}$$

The least squares problem in (3.18) can then be rewritten in matrix form as

$$\arg\min_X ||WAX - W\mathbf{b}||_2^2, \qquad W = \begin{bmatrix} w_1 I & \mathbf{0} \\ \mathbf{0} & w_2 I \end{bmatrix}. \tag{3.22}$$

The optimized x-coordinates are then given by equation (2.10).

Since raw IMU-data could only be obtained in the Kaggle dataset, this is the only dataset the WLS approach was evaluated on. The estimates of absolute and relative positions in the official test set consisting of 626 tracks and 10,133 points were combined using the least squares approach. The mean validation errors for the 24 positioning models were in the range 4.3–10.9 m, which corresponds to weights w_1 in the range 0.013–0.085. For the relative predictions the expected error was assumed to be consistently 1.83 m which corresponds to a weight w_2 of 0.47. This means that the second type of residual v_{2i} was weighted higher in all buildings considered.

The WLS optimization problem can be derived by starting from a linear dynamical system and applying the Kalman smoother equations. An brief overview of this procedure will be provided here to give an intuition of the relationship between the WLS approach and the

more general Kalman filter and smoother. A linear dynamical system, assuming Gaussian measurement errors, describing the problem is given by

$$\begin{cases} \hat{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{u}_i, & \mathbf{u}_i \in \mathcal{N}(0, \Sigma_i), \\ \widehat{\Delta\mathbf{x}}_i = \mathbf{x}_{i+1} - \mathbf{x}_i + \mathbf{v}_i, & \mathbf{v}_i \in \mathcal{N}(0, \Gamma_i), \end{cases} \tag{3.23}$$

where $\hat{\mathbf{x}}_i$ are estimates of absolute positions, $\widehat{\Delta\mathbf{x}}_i$ are estimates of relative positions, $\mathbf{x}_i$ are the true positions, $\mathbf{u}_i$ and $\mathbf{v}_i$ are the errors with covariance matrices Γ_i and Σ respectively, at timestamp i. Using the Kalman smoother equations the joint probability distribution of all $\mathbf{x}_i$ can be computed. Since it is assumed that the errors are Gaussian, this joint distribution is a multivariate Gaussian distribution. The maximum-likelihood estimator of such a distribution is given by a least squares problem, and in this case a weighted least squares problem due to assuming variable variances of errors in estimated positions and displacements.

3.6.1 Variance component estimation

Instead of assuming that the error of a positioning model is constant for all users and devices in one building, it was investigated if the variance (squared standard deviation) of estimated positions could be estimated automatically during online positioning. The positioning accuracy of a device carried by a user might be an outlier that has not been observed in the training data, for example due to sensor differences or walking styles. There are also many applications where the standard deviations of the residuals to be minimized in equation (3.18) are not known. Therefore, the aim of this investigation was to find a way to estimate these standard deviations without any previous knowledge of the two underlying models.

Assume that the structure of the covariance matrices are known down to the scale factors σ_1 and σ_2 in equation (3.17). Then σ_1 and σ_2 can be estimated by least squares variance component estimation (LS-VCE), as described in section 2. Again, once the optimal σ_1 and σ_2 have been estimated the optimized x-coordinates are given by equation (2.10).

The LS-VCE algorithm for estimating standard deviations was initially evaluated on simulated tracks. 5,000 tracks with a mean distance of 10 m between each point were generated assuming the distributions $v_{1i} \in \mathcal{N}(0, 4)$ and $v_{2i} \in \mathcal{N}(0, 1)$. This was done for tracks consisting of 10, 20 and 50 points. The chosen standard deviations of the errors mean that, as in common positioning applications, the standard deviation of the error in absolute position v_{1i} was assumed to be larger than the one for the relative positioning error v_{2i}. After the tests on simulated data the method was also tested on tracks from the Kaggle dataset.

3.7 Wall-estimation

Knowledge of wall locations can for example provide information about where open spaces and rooms are located inside buildings, and where it is possible to walk. This can provide information useful in navigation applications. Two ways of estimating walls were explored in this thesis, based on observations and results from the previous sections.

The first approach is based on knowledge of beacon positions and an estimate of the number of open spaces inside a building. Clustering of the correlation coefficients can then be performed, as done above. Each point inside a building is then assigned the cluster label for which the nearest beacon belongs to. The boundaries where the indoor space is separated by different clusters of beacons can then be interpreted as building walls.

The second approach uses the generative model represented by the mapping $\hat{\mathcal{D}}$ which is obtained from each training session on data from a specific building, but is not used for positioning. Instead it is possible to use it for generating expected RSSI values for all beacons detected in the training data, for all locations inside a building. This can be done for each floor, or even continuously in all three dimensions since the variable used for floor classification is continuous. This enables an investigation into the position of beacons, and the coverage of each beacon signal strength.

Another application of the generative model, which is of interest in the GIN project, is the estimation of walls. As was shown in the investigation of RSSI measurements, the signal strength from a Wi-Fi beacon is heavily attenuated when passing through a wall. It can therefore be argued that such decrease in signal strength should also manifest itself in the generated RSSI-maps. This leads to a possible approach for estimation walls.

Assume that the decoder $\hat{\mathcal{D}}$ has generated one RSSI-map for each beacon in a building at a certain floor and with a specific resolution by feeding it all possible positions on that floor. To be able to detect walls it is proposed that the resolution of the grid is approximately 0.1–1 m. For n beacons this results in n two dimensional RSSI images, containing the expected signal strength for each beacon across the whole floor. Algorithms for image processing can then be used to locate walls. The proposed algorithm for extracting walls from such a collection of RSSI images is presented in Algorithm 1. In the process the Sobel filter is used which is an image filter that estimates the derivative in the x- and y-direction. Filtering an image H along the x- and y-direction using a Sobel filter gives the outputs

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * H, \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * H, \tag{3.24}$$

where $*$ denotes the convolution operation.

Algorithm 1: Wall estimation

Result: $h \times w$ wall probability image G_{prob}

Choose threshold thr for the range below the max RSSI to consider for gradients (20 dBm used here) **for** *i=1:N* **do**

- Generate $h \times w$ RSSI image H_i
- Set $H_i[H_i < max(H_i) - thr] = max(H_i) - thr$ Get G_{ix}, G_{iy} by applying Sobel filter to H_i in x- and y-direction
- Compute magnitude of filtered images $G_i = \sqrt{G_{ix}^2 + G_{iy}^2}$

Compute the sum $G = \sum_{i=1}^{N} G_i$

Normalize $G_{prob} = G/max(G)$

3.8 Additional algorithms used in the Kaggle competition

In the Kaggle competition some constraints were relevant to positioning, which increased the positioning accuracy drastically for all participants. First and foremost it was known that both training and test data was collected in the corridors of the shopping malls, as can be seen in Figure 3.4. Maps were also made available, which made it possible to determine the exact areas where data could be found. Secondly it was also known that the surveyors collecting the data had been walking between a finite number of known points within the buildings. In addition to those constraints it is in such a competitions also feasible to ensemble the results from several different models to improve the positioning accuracy. Since the above constraints rarely apply for buildings in general the further techniques developed in the Kaggle competition will only be described briefly.

Before the team merger, i.e., before a team was joined, the main solution relied on the neural networks described above combined with the WLS optimization strategy. The maps were then utilized by iterating between moving points that were outside the corridors to the inside of the corridor geometry, and performing WLS optimization. This was done successively increasing the weight w_1 in Equation (3.18), which means that it is assumed that the absolute position estimates become more and more accurate. The iteration was run until convergence, and finally the points that were closer than 7 m to a known collection point were moved to this location. The essence of this approach is to fit each track into the corridor geometry of the buildings, where it was known that a surveyor could walk.

When merging with the team, the solution was essentially the same, but ensembles of absolute position estimates and relative position estimates were used. Ensembling was performed by optimizing the weights for each model based on the validation dataset. Also instead of moving points to the nearest known collection points after the iterative corridor scheme, an optimization strategy was used to fit the track to the best possible grid of known collection points. Overviews of the two steps for making use of maps and known collection points are shown in Algorithm 2 and 3. The cost used for each track window grid in Algorithm 3 was a weighted sum of the changes in position occurring from moving the estimate from the

points achieved in step 1 to the grid points. The cost was then a weighted sum of changes in absolute positions, relative positions, directions, total traveled distance (computed by using Dijkstra's algorithm between points by using the maps), and the number of times one point was used twice in the track.

Algorithm 2: Step 1 - Iteratively move points into corridor geometry

Result: Tracks adapted to corridor geometry
Get estimates of absolute and relative position from neural networks or ensemble
Estimate weights for WLS based on validation accuracy
while *not converged* **do**
    for *each test track* **do**
        Run WLS with weights w1, w2
        Move all points to the nearest point inside the corridor geometry
    decrease w1

Algorithm 3: Step 2 - Compute optimal tracks using known collection points

Result: Optimized tracks
for *each test track* **do**
    Split track into windows of width $n_{window} = 7$
    for *each window in track* **do**
        Locate the 7 nearest collection points to each point in the window
        Compute total cost for all 8^8 possible grids (also considering the original position)
        Choose window with lowest cost
    Append windows into full track

Chapter 4

Results

In this section the results of the developed models are presented, and this is done in four parts. In the first part the results of the main contribution in this thesis is presented, namely the positioning neural network. Secondly, the result of the LSTM-network for PDR is presented. In the third part the estimates from the fingerprinting and PDR models are combined to investigate the improvement from applying weighted least squares optimization. Finally results from additional outcomes of the thesis are presented, such as wall estimation, online estimation of error variances, and the Kaggle competition.

4.1 RSSI-based positioning

The first part of this section focuses on the main contribution of this thesis, namely the RSSI-based positioning model. To evaluate to what degree the model can adapt to different datasets and building environments the model was tested for the totality of 28 buildings in the three datasets, trained using identical settings as presented in Table 3.5. The settings were optimized for the UJIIndoorLoc dataset, which was considered the most representative fingerprinting dataset in this thesis, and for that reason this result is presented first. The main positioning results are presented in Table 4.1.

Table 4.1: *Main 2D positioning results achieved for the three datasets. The datasets consist of 3, 1 and 24 separate buildings respectively. The result for the Kaggle dataset is shown before and after applying sensor fusion by WLS.*

Dataset	**Mean error (m)**
UJIIndoorLoc dataset	6.69
Office dataset	2.32
Kaggle dataset (before WLS)	7.23
Kaggle dataset (after WLS)	5.44

4.1.1 Tests on the UJIIndoorLoc dataset

Several tests were carried out using the UJIndoorLoc dataset. Firstly, the goal was to achieve as high accuracy as possible of positioning and floor classification. Then several investigations of the properties and performance of the model were performed as described in Section 3.

Table 4.1.1 shows the best positioning and floor classification accuracies achieved on all three buildings of the UJIIndoorLoc dataset. This shows that the best accuracy on both floor classification and positioning was achieved for building B0. Since approximately 50% of the test data is from this building it also has the highest influence on the total mean positioning error and accuracy.

Table 4.2: *Positioning results using one separate model for each building in the UJIndoorLoc dataset and the total mean of these results.*

	B0	**B1**	**B2**	**Mean**
Floor accuracy	96.08%	85.99%	94.40%	92.89%
Median error (m)	3.93	5.24	5.04	4.56
Mean error (m)	5.49	7.63	7.93	6.67

Since the UJIIndoorLoc test set is not publicly available the UJIIndoorLoc dataset has been used in different ways in research. For comparing the proposed neural network with previous solutions only algorithms that have been evaluated on the same original validation set are considered. Several approaches using neural networks have been applied to the UJIIndoorLoc dataset, but only the CNNLoc algorithm [28] and Scalable DNN [14] were found that had been evaluated on this known subset of the data, which is why other neural network approaches are not considered in this comparison. Table 4.3 shows the result of the proposed model and the result from a number of different approaches for fingerprinting. The Error corresponds to the mean positioning error for all points in the testing dataset, irrespective of the accuracy of floor and building classification.

The proposed model is compared favourably to all approaches except the WKNN approach when considering positioning accuracy, but it is also comparable with AFARLS [8]. The training time for the proposed model was in the order of 600 seconds, while the inference time was 0.22 seconds. The implemented WKNN had an inference time of 2.34 seconds.

Table 4.3: *Comparison of results of various approaches that have been tested on the UJI-IndoorLoc validation dataset containing 1,111 instances evaluated by the mean positioning error.*

	Floor accuracy	**Error (m)**
Baseline RFR	-	10.13
M1_SDAE	-	7.92
Probabilistic [21]	-	9.03
Scalable DNN [14]	91.27	9.29
WKNN [8]	95.32	**6.58**
AFARLS [8]	95.41%	6.78
CNNLoc [28]	**96.03%**	11.38
Proposed model	92.89%	6.67

Next, the test of using less ground truth data was considered. The means over the three buildings and all generated models for each percentage of used training data is presented in Table 4.4 together with the result when using all data. For reference the corresponding results when using the baseline WKNN and M1_SDAE are also presented. Since these are means of positioning results based on all subsets, the distributions of mean and median errors for the 5% and 1% case are shown in Figure 4.1 and 4.2. This shows that when using 5% of the training data for training the mean error for the proposed model is reliably below 11 m, while for the 1% case it is reliably below 15 m for all three buildings.

Table 4.4: *Mean positioning errors for using subsets of the training data for WKNN, M1_SDAE and proposed model (averaged over 1, 2, 10, 20 and 100 generated models with independent training data for each percentage).*

% used data	**WKNN**	**M1_SDAE**	**Proposed model**
100%	**6.58**[8]	7.92	6.67
50%	**6.85**	8.15	7.12
10%	8.15	8.52	**8.03**
5%	9.25	8.99	**8.55**
1%	13.75	12.31	**11.81**

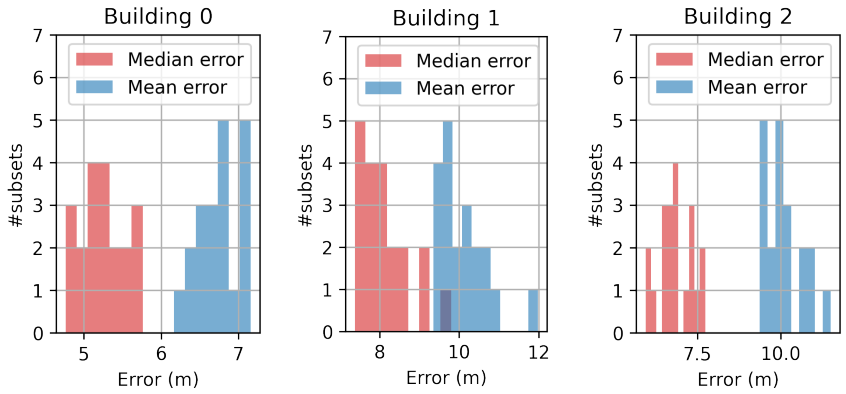


Figure 4.1: *Distributions of mean errors for each model trained on 20 independent 5% subsets of the training data for the three buildings in the UJIIndoorLoc dataset.*

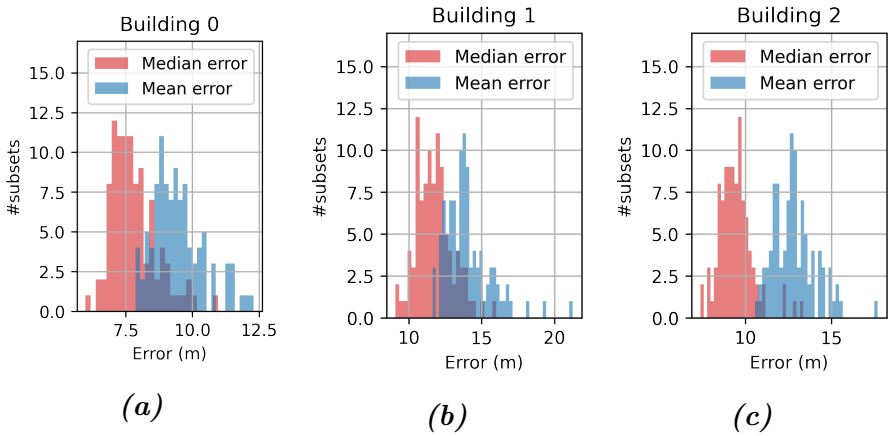
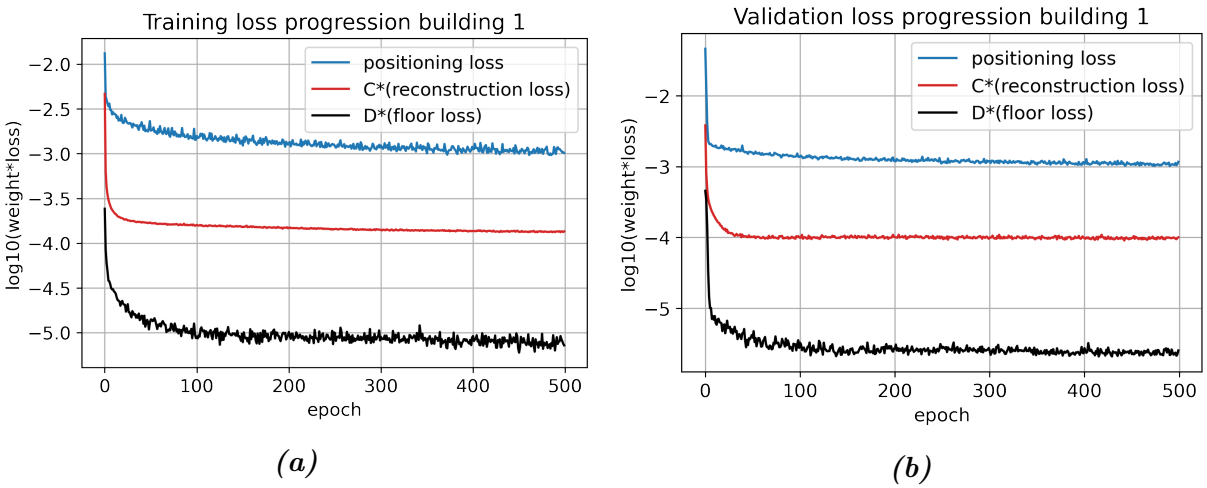


Figure 4.2: *Distributions of mean errors for each model trained on 100 independent 1% subsets of the training data for the three buildings in the UJIIndoorLoc dataset.*

The three loss components in Equation (3.9) were recorded separately for each epoch during one training session for building B1. The losses are shown in Figure 4.3, which shows that all training and validation loss components decrease for the entirety of this training session.



(a) *(b)*

Figure 4.3: *Progression of the three losses presented in Equation* (3.9) *a) for the training data and b) for the validation data during one training session for building B1 in the UJIIndoorLoc dataset.*

Another test was performed to investigate how the positioning accuracy depended on the reconstruction weight C. This dependency is shown in Figure 4.4. A slight decrease in positioning error is observed as the reconstruction weight is increased until the value $C = 10$ when the error increases drastically.

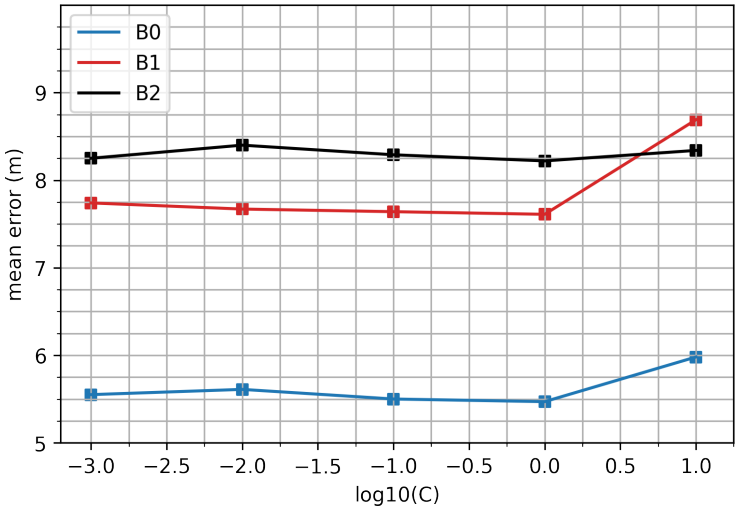


Figure 4.4: *Dependency of mean positioning error on the reconstruction weight C, taking the mean of ten model results trained on building B1 in the UJIIndoorLoc dataset.*

Finally the influence of augmentations was tested. Table 4.1.1 shows the resulting accuracy with and without using the augmentative scheme for each of the three buildings in the UJIIndoorLoc dataset from one trained model. The results show that the difference from using augmentations is more than 1.5 m for buildings B1 and B2, while it is negligible for building B0. The result of augmentations results in an improvement of more than 1.2 m for

the overall mean positioning error. As can be seen in Table 4.1.1, the effect on positioning accuracy from excluding augmentation 3 (beacon-dropping) is the highest, while the effect of excluding augmentation 1 and 2 is slightly lower. The results indicate that all augmentations used were involved in improving the positioning accuracy.

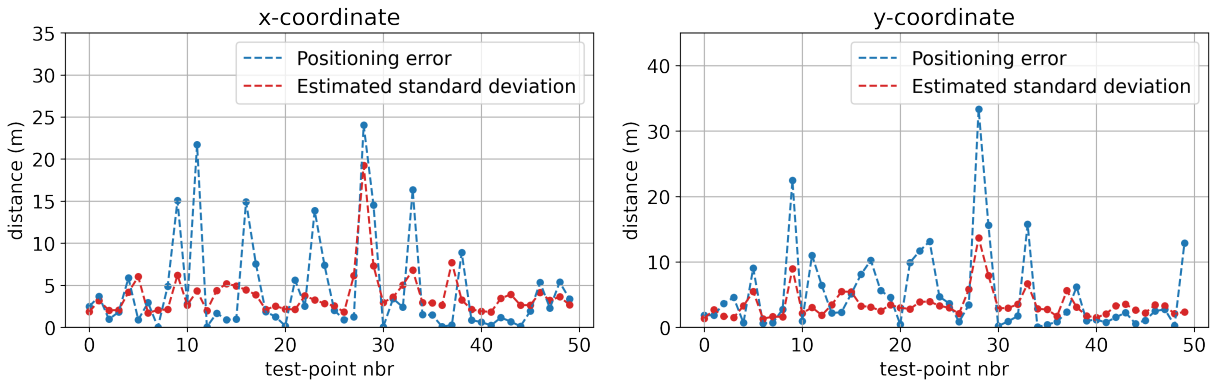
Table 4.5: *Positioning results on the UJIIndoorLoc dataset with all augmentations, combinations of two augmentations, or no augmentations for each of the three buildings, and the combined mean error.*

	B0	**B1**	**B2**	**Mean**
All augmentations	5.49	7.63	**7.93**	**6.67**
No augmentations	5.55	10.53	9.59	7.90
Augmentation 1 & 2	**5.23**	8.92	9.10	7.18
Augmentation 1 & 3	5.46	7.96	8.78	6.95
Augmentation 2 & 3	5.51	**7.52**	8.73	6.84

4.1.2 Probabilistic version

The probabilistic version of the proposed neural network was first trained on the UJIIndoorLoc dataset. The achieved accuracy was similar to the accuracy of the base version of the custom neural network. However, when trained on the Kaggle dataset, a significant degree of overfitting was observed. Due to time constraints the reason for this overfitting could not be investigated further. However, the results from training on the UJIIndorLoc dataset still indicates that it is possible to estimate a standard deviation for the output, which is why the approach is still presented in this report.

When estimating positions in the test set the new RSSI measurements have not previously been seen by the positioning network. The main question is then if the estimated standard deviations made by the probabilistic version are correlated with the error in positioning. Figure 4.5 shows the errors in positioning and the estimated standard deviations for 50 points in the test set. This indicates that for some of the measurements the model is able to detect points of low accuracy, for example point 28. The mean PCC between all $2\times1{,}111$ position estimates of x- and y-coordinates and the corresponding estimated standard deviations was computed to be 0.26. This shows that the actual magnitudes of the positioning errors and the estimated standard deviations are positively correlated, although not perfectly correlated.



(a) *(b)*

Figure 4.5: *Positioning errors for the x-coordinate a) and y-coordinate b) for 50 points in the UJIIndoorLoc test set, and the corresponding estimated standard deviations $\hat{\sigma}_1$ and $\hat{\sigma}_2$.*

4.1.3 Tests on the office dataset

After the tests on the UJIIndoorLoc dataset the model was tested on the office dataset. As mentioned in chapter 3 identical settings were used, see Table 3.5, to give a sense of the generalizability of the model. The results on the dataset for the WKNN and the proposed neural network are shown in Table 4.1.3. The results show that the accuracy in the office between the neural network model and the WKNN are again similar.

Table 4.6: *Results of four models trained and evaluated on the office dataset.*

	Median error (m)	**Mean error (m)**
WKNN	2.18	2.44
Proposed model	**2.14**	**2.32**

4.1.4 Tests on the Kaggle dataset

The positioning model was as with the office dataset trained with the same settings as in 3.5, but now using the Kaggle dataset. The resulting Kaggle score on the test-set consisting of 10,133 points from 24 buildings was 7.23 m. Since the floor classification is taken into account in the score, this is the maximal value of the actual positioning error in two dimensions, see Equation (3.1). However, from investigations of floor classification between members of the Kaggle team it is believed that the accuracy was 100%. This is possible due to the fact that it was known that each track was on one single floor, so the prediction of floor could be set by the majority decision. Using this optimal floor classification obtained from the proposed positioning model together with positions based on the WKNN model resulted in a score of 6.12 m.

From joining a team in the Kaggle competition, it was possible to compare the proposed model for positioning to the other approaches developed in the team. Four of the five absolute

positioning models in the team relied solely on RSSI measurements, and the positioning errors of these approaches on the validation set of 548 tracks are shown in Table 4.1.4. These are the results of a single model on the validation set (or an ensemble of five models for the RNN). The GPR_WKNN model is based on the common WKNN algorithm, but filling areas where data is missing with expected measurements using Gaussian processes. The RNN is based on processing a sequence of RSSI measurements and estimating a position for each timestamp of interest. This result is not shown for a single model, but for a 5 model ensemble due to the fact that the validation error was not computed for a single model. The Disc_MLP is a model based on discretizing the space into squares of 2×2 m and predicting a probability of position for all possible squares. This leads to a multi-class problem with up to 10,000 classes.

Table 4.7: *Results on the chosen validation set from the Kaggle dataset for four of the approaches implemented by the team.*

	Mean error (m)
Disc_MLP	8.08
RNN (5 model ensemble)	7.81
GPR_WKNN	**6.30**
Proposed model	7.84

As can be seen in Table 4.1.4 the validation errors for the neural network models are similar. It should be noted that despite the complexity of the RNN and Disc_MLP models, these have a very similar accuracy compared to the proposed model which only uses a single MLP for positioning. The GPR_WKNN on the other hand has in this case a validation error of more than 1.5 m lower than the other models. Possible reasons for this will be discussed in chapter 5. In the competition ensemble the proposed model was not used. The ensemble instead relied mainly on the GPR_WKNN and the Disc_MLP, despite the Disc_MLP achieving the highest validation error. The reasons for this will also be discussed in chapter 5.

4.2 Estimates of relative positions

The LSTM-network for PDR was evaluated on a dataset consisting of 548 tracks from the Kaggle dataset, independent from the data used to train the model as shown in Table 3.2.3. The resulting mean positioning error was 1.83m.

Code was provided in the Kaggle competition for basic pedestrian dead reckoning. This is used as reference for comparing the performance of the relative positioning model. The provided code resulted in a mean error of 2.55 m on the same 548 test tracks, which is more than 0.7 m higher than the result of the LSTM-network. The results from this section are summarized in Table 4.8.

Table 4.8: *Results of the LSTM model for relative positioning, and the Microsoft PDR algorithm for reference.*

	Mean error (m)
Baseline PDR	2.55
LSTM	**1.83**

4.3 Combining absolute and relative position estimates

The WLS optimization was applied to the estimates of absolute and relative positions in the Kaggle test set, consisting of 626 tracks and 10,133 positions. The approach decreased the mean positioning error from 7.23 m to 5.44 m. This is an improvement in positioning accuracy of almost 1.8 m, or 25%. The result is shown in Table 4.9 together with the original accuracy, and also after applying OLS for reference.

Table 4.9: *Positioning accuracy on the Kaggle test-set before and after applying optimization. The OLS result is also included for reference.*

	Mean error (m)
Before optimization	7.23
After OLS	6.36
After WLS	**5.44**

To visualize the full procedure of absolute positioning, relative positioning and optimization one track was chosen from the Kaggle validation set (since the ground truth positions were known for test tracks). Figure 4.6 shows the estimates of absolute position, estimated PDR track (visualized by starting from the known ground truth position), and the optimized track together with the ground truth track. The mean positioning error for this specific track was decreased from 5.97 m to 2.89 m.

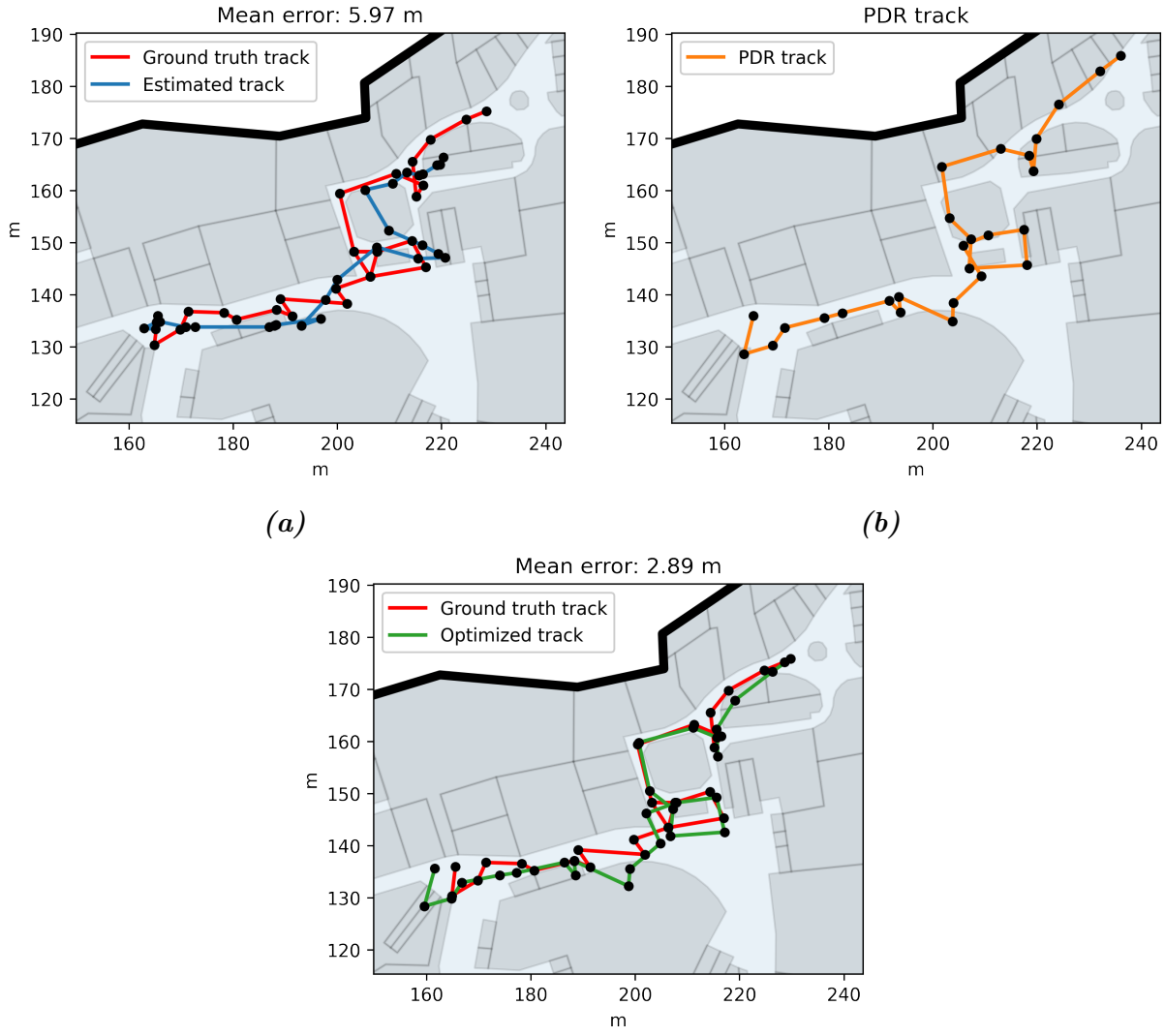


Figure 4.6: *a) Positioning result and ground truth track. b) PDR estimate of relative positions visualized by starting from the known ground truth position. c) Optimized result and ground truth track. The plots show the tracks in the indoor environment where the track data was collected.*

4.4 Online estimation of error variances

Table 4.4 shows the results from applying LS-VCE to the simulated tracks. This shows the positioning accuracy for four different weightings of the least squares optimization, and the original positioning error. The WLS ground truth uses weights based on the ground truth standard deviations, and the WLS optimal uses the optimal weights as determined by line-search. The positioning accuracy is improved compared to the OLS approach with 0.5–0.7 m by using standard deviations from the LS-VCE, depending on the number of observations (i.e. the track length). The result is on the other hand 0.2–0.3 m higher than the result from using optimal weights.

When applied to tracks from the Kaggle dataset the estimated standard deviations did not improve the positioning accuracy when used as weights in the WLS optimization. It was consistently observed that the estimated standard deviation σ_1 by LS-VCE was lower than σ_2, which would indicate that the positioning accuracy of the absolute positions are higher than for the relative estimates. Since this does not match with the observed error variances, the results from using the LS-VCE method for WLS on the Kaggle dataset are worse than when performing OLS. Reasons for this are discussed in chapter 5.

Table 4.10: *Mean positioning errors (m) for simulated data of lengths 10, 20 and 50 using: no optimization, OLS, WLS with weights based on estimated standard deviations, WLS with weigths based on ground truth standard deviations, and WLS with optimal weights.*

Optimization	**N=10**	**N=20**	**N=50**
None	5.019	5.004	5.008
OLS	2.848	2.743	2.689
WLS LS-VCE	2.341	2.079	1.962
WLS ground truth	2.097	1.910	1.827
WLS optimal	2.005	1.826	1.785

4.5 RSSI-map generation

The proposed neural network results in three MLPs, a positioning MLP, a floor classification MLP, and a generative MLP from one training session. The results of the two former MLP's have been presented above, but not for the generative model. A generative model could, as previously mentioned, for example be useful for investigating and visualizing beacon positions and signal coverage in buildings.

Since the true expected RSSI at each point inside a building was not known (and can not be theoretically computed due to the complex and ever changing indoor environment), only a visual evaluation of the generated RSSI maps was performed in this thesis. Figure 4.7 shows the generated RSSI-map for one of the beacons detected in the office dataset, visualized together with the floorplan and the known beacon position. This shows that the RSSI-map

is maximal near the beacon position, and that it decreases at different rates in different directions. Figure 4.8 shows the generated RSSI maps from the ground and first floor in one of the Kaggle dataset buildings for one of the beacons.

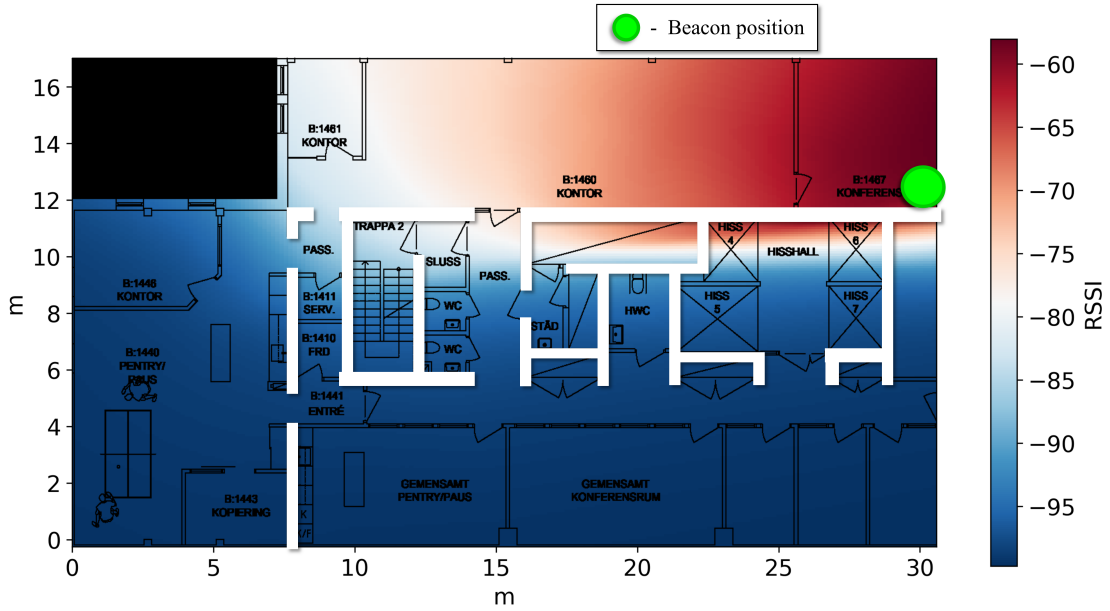
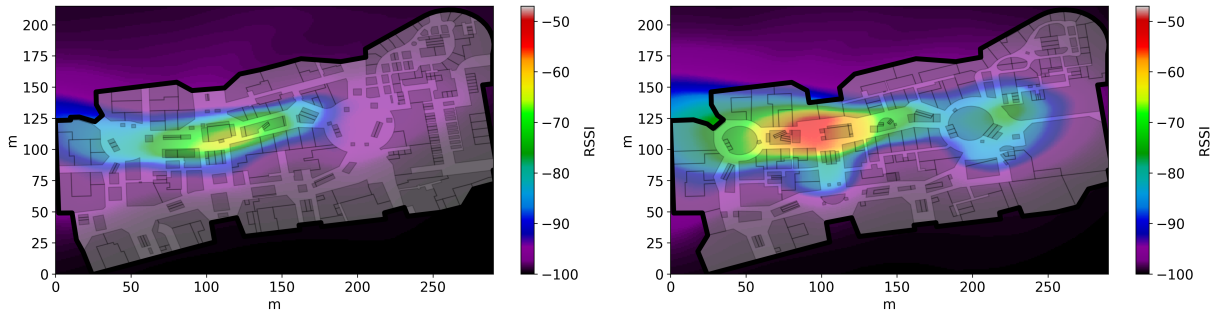


Figure 4.7: *Expected signal strength for one of the detected beacons in the office dataset, visualized with the floorplan and known beacon position. Actual main inner walls are marked with white.*



(a) ***(b)***

Figure 4.8: *Generated RSSI-maps at a) floor 0 and b) floor 1 for one Wi-Fi beacon in one example building in the Kaggle dataset.*

4.6 Wall estimation

One goal of the GIN project is to find the location of walls. The methods developed in this thesis provide two ways of doing wall estimation. The first way is applicable when the location of the beacons are known or can be estimated. Then the clustering of the PCCs as described in Section 3 can be used to locate walls. Figure 4.9 shows the result of this algorithm run in the office environment, assuming three clusters.

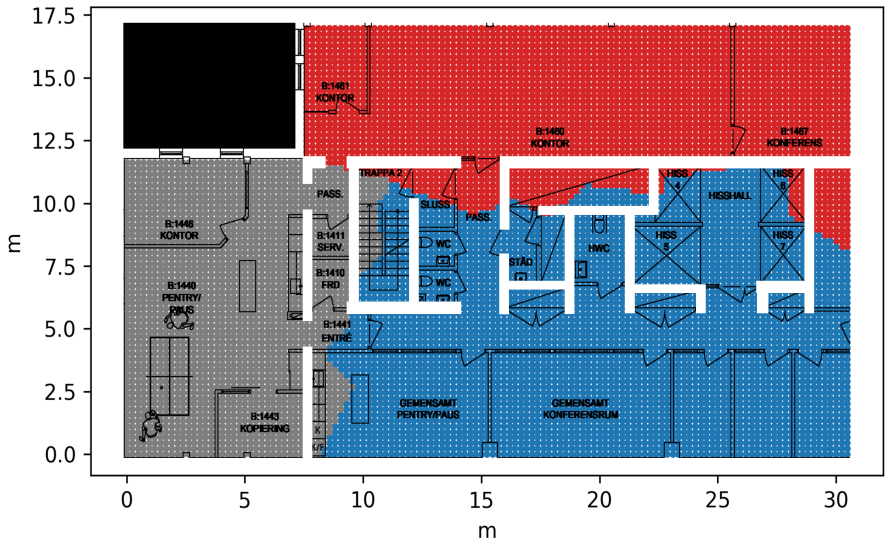


Figure 4.9: *Visualization of wall estimation based on clustering of the beacon correlation coefficients. Main inner walls are marked with white.*

The second algorithm for estimating walls does not require knowledge of beacon positions and the number of open spaces. Instead it makes use of the gradients in generated RSSI maps at a specific floor, as described in chapter 3. Figure 4.10 shows the estimated wall probability map in the office space. Figure 4.11 shows the corresponding wall probability map generated on one floor in one building in the Kaggle dataset.

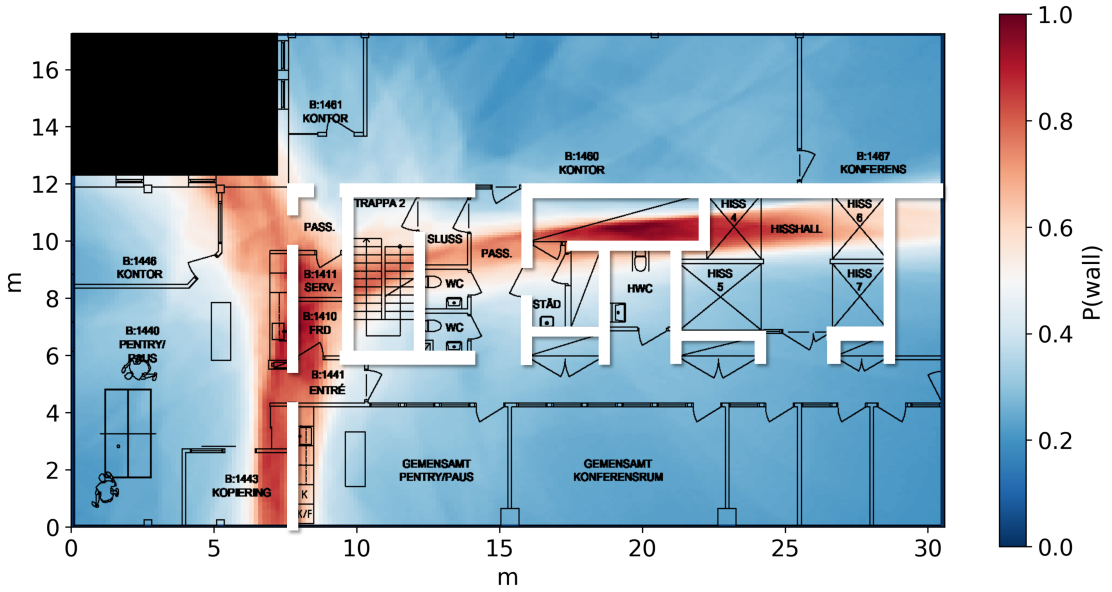


Figure 4.10: *Visualization of the estimated wall probability map in the office environment. The main inner walls are marked with white.*

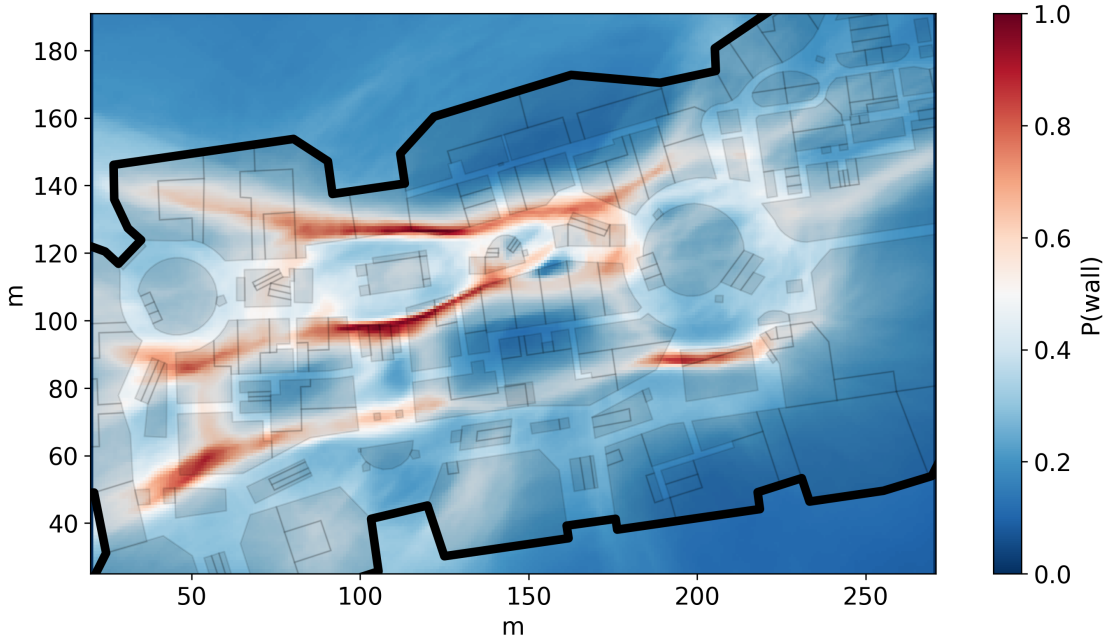


Figure 4.11: *Visualization of the estimated wall probability map on one of the floors in the Kaggle dataset.*

4.7 Results in the Kaggle competition

As mentioned above, the absolute positioning model did not contribute to the ensemble in the team submission despite lower validation error than for example the Disc_MLP, for reasons that will be discussed. However, the relative positioning LSTM was the main model for estimating displacements between points (47% of the ensemble). Adding the two described postprocessing steps, using maps and the known gridpoints, resulted in a final score of 2.20 m and the second place in the overall standings.

Chapter 5

Discussion

Since this thesis covers multiple areas within indoor positioning, deep learning and other areas, there are also many aspects to consider when evaluating the methods and results. This section will focus on discussing the choices of methods, and how to interpret the results by again dividing the section into the main components of the thesis.

5.1 RSSI-based model

5.1.1 Positioning

The proposed neural network achieves a mean positioning error that is among the lowest of compared methods for all three datasets when only considering a single input RSSI array. It was also trained using the same settings for all three datasets. This could be an indication that the model and training procedure has a great ability to generalize between buildings and datasets since no tuning of parameters specific to various datasets was performed. In addition to this, the outcome is three MLPs which all can be used in different applications.

It was noted that at a total training time of 600 seconds for the three UJIIndoorLoc buildings the proposed model has a significantly longer training time than for example the WKNN (only the ball-tree for improved search speed has to be computed). The training time is increased further without access to a GPU. On the other hand the inference time is reduced compared to KNN-based approaches. The inference time for the 1,111 test points was 0.22 seconds when running building-wise inference one point at a time in numpy, without utilizing GPU acceleration. This can be compared to the WKNN approach which had an inference time of 2.34 seconds. Since the neural network dimensions are the same independent of the amount of data, this inference time does not change significantly with access to more or less data. The inference time of the WKNN approach on the other hand increases with the amount of available data.

When trained on less data the proposed neural network shows resiliency of positioning error, since it is almost one meter lower than for the WKNN when using 5% of the data, and almost two meters lower when using 1% of the data. Using such small amounts of ground truth

data when training a neural network commonly leads to serious overfitting, and a bad model performance, but when training this model with very few datapoints clear overfitting was not observed. It is argued that it is the augmentative scheme applied in the training procedure that reduces this problem, and results in a better generalizability. One might also expect that when reducing the amount of training data the degree of augmentation and dropout would have to be increased. This was not observed either, as training the model with the same settings as with the full dataset always resulted in the lowest positioning error. One might therefore draw the conclusion that the used augmentations actually result in measurements that represents the training data well. This is reasonable since the augmentations were derived from the properties of RSSI, and not from the amount of data available.

When comparing the proposed neural network to the M1_SDAE model, the increase in positioning error when using less data is very similar, only with the difference that the M1_SDAE has a slightly higher overall error. This was the case despite that the SDAE was trained with all the data to obtain the first four layers. These results could indicate that it is not actually the unlabeled data that helps the network learn, but instead the denoising nature of the SDAE algorithm. From the results this seems to be achieved at least equally well by augmentations.

Training the three models on the UJIIndoorLoc dataset without any augmentations did, when using all data, increase the positioning error with more than 1.2 meters. This is a clear indication that the augmentative scheme is an important addition to the model. An interesting observation is that the mean positioning error is improved much more for building B1 and B2 when applying augmentations than for B0. Since there are many factors at play such as different devices and users, different indoor environments, distribution of data, etc., it is difficult to speculate in the exact reason for this difference. What can be said is that for all buildings considered the augmentative strategy never appears to disrupt the training procedure and positioning accuracy, but only results in an improvement. It can also be said that the most important augmentation seems to be the beacon-dropping, since excluding it from the training procedure increases the positioning error with 0.5 m, compared to 0.2 m and 0.3 m for the two other augmentations. On the other hand, it can by this observation also be concluded that all three augmentations seem to improve the performance of the model.

The three components of the loss function, plotted for validation and training data in Figure 4.3, indicate that neither of the learning objectives are prone to overfitting under the proposed framework. This is clear from the observation that all parts of the validation loss continue to decrease during the entire training progression, while also staying below the training loss. It is possible for the validation loss to be lower than the training loss due to the fact that when computing the validation loss neither augmentations or dropout is used. The spiky appearance of the loss progression curves is also due to the augmentations, which introduce noise into the training, and thus resulting in a fluctuating loss curve.

When varying the reconstruction weight C as shown in Figure 4.4 a slight decrease in positioning accuracy was observed until it reached a value of $C = 10$ (note the log-scale in the plot). Even though the results are averaged over ten model estimates the difference is still within the margin of uncertainty. In other words, the indication is that for reasonable values

of C, the reconstruction error does not increase the positioning error, and might in fact result in a slight improvement. A decrease in positioning error from using the reconstruction error would not be unreasonable, since the network is then not only trained to accurately perform positioning, but also so that this position should reconstruct the input. The reconstructed output should in theory vary drastically for points on different sides of a wall. Therefore, if the network predicts a point that is on the wrong side of this wall, it would produce a high reconstruction error, which would then correct the network prediction during the training process. Since walls are not very wide this effect can not be expected to be very large. The results indicate that this hypothesis might hold, but since the effect is expected to be small further investigations would have to be performed to be able to draw a solid conclusion.

The probabilistic model was equally accurate on the UJIIndoorLoc dataset, but when trained on the Kaggle dataset heavy overfitting was observed. Due to time constraints the reason for this could not be further investigated. Instead the model is proposed as an indication that it might be possible to alter the neural network in this way to estimate the accuracy of each estimate along with the actual position. It was shown that the estimated standard deviations of each position estimate in the UJIIndoorLoc dataset had a positive correlation with the actual error. As mentioned in Section 3, to retrieve the actual standard deviation estimate, calibration has to be performed due to the bias inherent in the training procedure. This is also left for future work, since the first step of avoiding overfitting for all datasets should be investigated first.

In the office environment the mean positioning error is much lower than for the UJIIndoorLoc and Kaggle datasets at 2.3 m compared to 6.7 m and 7.2 m. The reason for this is argued to mainly be the differences in the density of beacons. In the office environment there are 64 detected bssid's that are within the walls of this specific space of size 17×30 m (in total 1,341 bssid's from other parts of the building as well). This can be compared to each building in the UJIIndoorLoc dataset which has approximately 200 detected bssids, but the building sizes are in the order of 100×100 m.

An outlier when testing on the Kaggle validation dataset is the GPR_WKNN method which achieves more than 1.5 m lower positioning error than the neural network models. The Kaggle dataset contained tracks that rarely overlapped. This meant that both validation and test tracks were often located in areas not well covered by the training data. It was in this thesis observed that in these situations neither basic WKNN algorithms or neural networks were able to extrapolate well to areas that are not covered with training data, although the WKNN seemed to do this slightly better as the result on the test set was approximately one meter lower for the standard WKNN. It is therefore believed that using Gaussian processes to fill the coverage holes with expected RSSI, as in the GPR_WKNN, is what improved the positioning accuracy of this model even further. An example of such approaches is described in [9]. With this in mind it might be possible to also use Gaussian processes together with augmentations to train the neural network in areas that are not covered by data as well. Due to time restrictions the idea was not investigated in this thesis, but is proposed for future research.

A reason for why the Disc_MLP was a main component in the ensemble to the best competition entry despite the highest validation error could be that it provided diversity into

the position ensemble. If the errors made by the four other models were highly correlated to each other, but not to the disc_MLP, it is reasonable that the ensemble should consist of estimates from the best single model of these four models, and the Disc_MLP.

5.1.2 Classification

The floor classification was not a main focus in this thesis, since Combain already has efficient methods for determining the floor label based on pressure sensors, and that it is a more trivial problem. Nevertheless, it was implemented for consistency with the motivation of a positioning model in three dimensions, and the ability of generating RSSI-maps on different floors. The accuracy of 92.89% on the UJIIndoorLoc is not among the top results of floor classification compared to previous work, but is still an improvement from for example the baseline KNN [32] of 89.92%, and would have a significantly lower inference speed, due to consisting of a single two-layer MLP, based on the above tests of inference time.

The approach of doing ordinal classification enabled generation of continuous three dimensional RSSI-maps. Using the ordinal classification method was reasoned to be appropriate since the positioning is based on RSSI, which is inversely proportional to the distance between the beacon and the receiver. Therefore, measurements from floors closer to each other should also have a higher correlation in RSSI.

For the continuous variable f in Figure 3.9 to represent the elevation of a device inside the building one could argue that the breakpoints q_k in Equation (3.8) should be spaced in proportion to the height of each floor. This is a valid point, however the floors in the vast majority of buildings are equally spaced. Combined with the fact that floor heights are rarely known, this point was not taken into consideration in this thesis.

5.1.3 Generation

When considering autoencoders, the decoder is often discarded due to the fact that the latent space is an arbitrary representation of the input space. In the proposed model the latent space of the autoencoder represents the actual 3D position (or 2D if the building has one floor). This means that the decoder can be used to generate RSSI-maps in the building for each beacon, also for areas not covered by the data.

The generated RSSI maps were not expected to perfectly reflect the actual signal strength in the building, since these also have to take into account cases when the signal is missing during training (i.e. RSSI -100). Therefore, the RSSI value for a beacon that results from the generative model was expected to be lower than the true expected strength at a specific distance according to the path loss model, which was also observed (see Figure 4.7). This is not an issue since the reference path loss is usually known, which means that the map can be calibrated after generation. What is often of higher interest is not the actual signal strength, but how the signal propagates in the indoor environment.

The generated RSSI maps in the office environment where the beacon locations were known showed a clear correlation between the maximum expected RSSI and the known beacon position, as indicated in Figure 4.7. The office dataset was collected by walking along

straight lines, which can be seen in Figure 3.2, as the points are located along straight lines not only in corridors but also in the open spaces. Despite this, the linear structure of the data was not visible in the generated RSSI maps, see Figure 4.7. Instead the output map appears to interpolate well between collection points, as the generated map decreases further away from the beacon position, and decreases faster when the signal reaches one of the main walls. It can be noted that just below the top main wall in Figure 4.7 the model still predicts a high RSSI for the chosen beacon. The reason for this is that there are no measurements taken right on the other side of that wall, which means that the model is free to assume any signal strength in that area.

When generating RSSI maps in one of the Kaggle buildings, the beacon and wall locations were not known. Nevertheless it is shown that the model is able to generate different and reasonable maps for separate floors. These can be said to be reasonable by visual evaluation, since The generated maps also indicate that the model has not overfitted to specific points, and that the generated heatmaps follow the indoor geometry where the data was collected.

5.2 Wall estimation

Discussing the generative model leads directly to the topic of estimating walls, since the second method is based on generated RSSI-maps. From the resulting wall probability map shown in Figure 4.10 it is clear that the estimated walls overlap with the known locations of main inner walls in the building, as shown in Figure 3.3. The reason that the algorithm does not detect all walls separately is that there are few or no measurements taken between these walls. This means the the generative model is free to interpolate the signal strength in these areas. In other words, to be able to estimate walls accurately, it is suggested that there has to exist measurements on both sides of the wall which result in a high magnitude of the gradient in the generated RSSI-maps. When instead considering the plot in Figure 4.11 data was only collected in the white areas of the building (the corridors). This results in wall estimations that simply separate the spaces where data has been collected, and are not necessarily located where the walls actually lie.

5.3 PDR - relative position estimation

The ordinary PDR algorithms are known to occasionally suffer from difficulties of adapting to different devices with different sensors, and to different walking styles. For example the stride length that is estimated based on the accelerometer might vary between users that have the same stride-length but who are holding the device with a varying degree of stability. With a sufficiently large dataset, as was provided in the Kaggle competition, it was shown that it is possible to improve the estimate of relative position drastically compared to the baseline approach provided by the competition sponsors.

Some indications of disturbances to the magnetic field were observed in the Kaggle dataset, but this could also be due to device properties, incorrect calibrations and/or walking styles. The accuracy of relative positioning did not improve when incorporating this data in the

LSTM-network. It is suggested that in this case either the number of training examples was too small for each building to reliably improve the relative positioning accuracy or the disturbances of the magnetic field were not significant.

5.4 Optimization

It was shown that performing WLS reliably improved the positioning accuracy, since the mean error for all 10,133 points in the Kaggle test set was reduced from 7.23 m to 5.44 m. The operation involves computing the inverse of a tridiagonal matrix with a size that is twice the number of points in the track to be optimized. Due to the extensive methods existing for computing such inverses, the approach is computationally inexpensive, and it is therefore also feasible for use in online positioning.

The assumption that the residuals v_{1i} and v_{2i} all have the same standard deviations is not generally true. The positioning accuracy can be expected to vary drastically between different users and devices despite performing positioning in the same building. Using WLS showed a clear improvement compared to the standard OLS method, which indicates that the assumption can be effective for decreasing the positioning error even though it does not always hold in theory.

It was the apparent shortcomings of this assumption that motivated the investigation of online estimation of the variance for each track using LS-VCE. It was shown that the LS-VCE could for simulated tracks improve the positioning accuracy drastically without any assumptions on the magnitude of errors being made by the positioning models. The approach did however not work as expected for the Kaggle dataset. An observation that was made of the errors of positioning estimates in the Kaggle dataset was that these were highly correlated with respect to time. This violates the assumption of Q_1 and Q_2 being identity matrices. In some applications this violation could be negligible, but in this specific case the conclusion was that the violation of the assumption was too strong. To remedy this problem it might be possible to use knowledge of the behavior of positioning models to model the structure of Q_1 and Q_2, but no successful approach was found in this thesis. However, since the approach has been shown to be efficient when errors are independent, as in the simulated data, the LS-VCE approach might still be useful in applications with positioning models that result in errors with a lower degree of correlation in time.

Chapter 6

Conclusion

In this thesis several methods related to indoor positioning, and also how to extract other useful information about buildings from RSSI to position correspondences, were proposed and investigated. The main component of the thesis was the neural network for positioning and RSSI-map generation. From one training session the model and training procedure results in three separate MLPs which all can be used separately in different applications. The main focus when developing this model was on positioning accuracy, which was mostly improved by the use of custom augmentations, but also other known techniques for training neural networks.

By testing on multiple datasets it is shown that the proposed model is able to adapt to multiple types of buildings and achieve a comparable accuray to state of the art WKNN algorithm. The training time of the model was approximately 600 seconds. However, the model had a decreased inference time of one order of magnitude compared to the common WKNN approach with similar accuracy on the UJIIndoorLoc dataset. With access to less labeled data the results also indicated that the model is more robust than the WKNN approach.

A customized deep learning approach for performing pedestrian dead reckoning was also investigated, and the resulting estimates were combined with the absolute position estimates using weighted least squares. Two ways of estimating the weights in the weighted least squares approach were considered, and it was determined that the approach based on previous knowledge of the positioning model errors was most effective for the Kaggle dataset. This approach decreased the mean positioning error with 1.8 m, or 25%.

6.1 Future work

There are several parts of the approaches presented in this thesis that allow for improvements and future investigation. When considering the positioning model there are two main components that are suggested for future work. Firstly the use of Gaussian processes, as used in the GPR_WKNN, might be used in combination with augmentations to make the network generalize better to parts of a building that are not covered by training data. Secondly, the

use of a probabilistic version of the model shows potential as it was able to achieve similar positioning results on the UJIIndoorLoc dataset but also results in estimates of the error magnitude. Further investigations could be performed into why overfitting is observed for some datasets, and how to calibrate the estimated standard deviations to match the errors accurately.

The online estimation of standard deviations using LS-VCE is another area where further work has to be done to make it effective for a dataset such as the Kaggle dataset. Modelling the structure of the covariance matrix components Q_1 and Q_2 might be possible with access to ground truth data, which could make the online estimation feasible, not just for independent errors as in the simulated case, but also in general.

References

[1] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef. Wideep: Wifi-based accurate and robust indoor localization system using deep learning. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, pages 1–10, 2019.

[2] H. Bae, J. Oh, K. Lee, and J.-H. Oh. Low-cost indoor positioning system using ble (bluetooth low energy) based sensor fusion with constrained extended kalman filter. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 939–945, 2016.

[3] K. Batstone, M. Oskarsson, and K. Åström. Robust time-of-arrival self calibration and indoor localization using wi-fi round-trip time measurements. In *2016 IEEE International Conference on Communications Workshops (ICC)*, pages 26–31. IEEE, 2016.

[4] A. Beck, P. Stoica, and J. Li. Exact and approximate solutions of source localization problems. *IEEE Transactions on signal processing*, 56(5):1770–1778, 2008.

[5] A. Belmonte-Hernández, G. Hernández-Peñaloza, D. Martín Gutiérrez, and F. Álvarez. Recurrent model for wireless indoor tracking and positioning recovering using generative networks. *IEEE Sensors Journal*, 20(6):3356–3365, 2020.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, Berlin, Heidelberg, 2006.

[7] C. Chen, C. X. Lu, J. Wahlström, A. Markham, and N. Trigoni. Deep neural network based inertial odometry using low-cost inertial measurement units. *IEEE Transactions on Mobile Computing*, 20(4):1351–1364, 2021.

[8] H. Gan, M. H. B. M. Khir, G. Witjaksono Bin Djaswadi, and N. Ramli. A hybrid model based on constraint oselm, adaptive weighted src and knn for large-scale indoor localization. *IEEE Access*, 7:6971–6989, 2019.

[9] A. A. Golovan, A. A. Panyov, V. V. Kosyanchuk, and A. S. Smirnov. Efficient localization using different mean offset models in gaussian processes. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 365–374, 2014.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] T. Hastie, R. Tibshirani, , and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2009.

[12] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy. Recurrent neural networks for accurate rssi indoor localization, 2019.

[13] S. Jung, C. oh Lee, and D. Han. Wi-fi fingerprint-based approaches following log-distance path loss model for indoor positioning. In *2011 IEEE MTT-S International Microwave Workshop Series on Intelligent Radio for Future Personal Terminals*, pages 1–2, 2011.

[14] K. S. Kim, S. Lee, and K. Huang. A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting. *Big Data Analytics*, 3(1), Apr 2018.

[15] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models, 2014.

[16] M. Larsson, V. Larsson, K. Astrom, and M. Oskarsson. Optimal trilateration is an eigenvalue problem. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[17] G. Lui, T. Gallagher, B. Li, A. G. Dempster, and C. Rizos. Differences in rssi readings made by different wi-fi chipsets: A limitation of wlan localization. In *2011 International Conference on Localization and GNSS (ICL-GNSS)*, pages 53–57, 2011.

[18] D. R. Luke, S. Sabach, M. Teboulle, and K. Zatlawey. A simple globally convergent algorithm for the nonsmooth nonconvex single source localization problem. *Journal of Global Optimization*, 69(4):889–909, 2017.

[19] Z. Ma, S. Poslad, J. Bigham, X. Zhang, and L. Men. A ble rssi ranking based indoor positioning system for generic smartphones. In *2017 Wireless Telecommunications Symposium (WTS)*, pages 1–8, 2017.

[20] K. P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[21] W. Njima, I. Ahriz, R. Zayani, M. Terre, and R. Bouallegue. Smart probabilistic approach with rssi fingerprinting for indoor localization. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2017.

[22] R. Odolinski and P. J. G. Teunissen. Low-cost, 4-system, precise GNSS positioning: a GPS, galileo, BDS and QZSS ionosphere-weighted RTK analysis. *Measurement Science and Technology*, 28(12):125801, nov 2017.

[23] R. L. Plackett. A historical note on the method of least squares. *Biometrika*, 36(3/4):458–460, 1949.

[24] W. Qian, F. Lauri, and F. Gechter. Supervised and semi-supervised deep probabilistic models for indoor positioning problems. *Neurocomputing*, 435:228–238, May 2021.

[25] M. Rainer. Overview of current indoor positioning systems. *Geodesy and Cartography*, 35(1), 2012.

[26] H. Rizk, A. Shokry, and M. Youssef. Effectiveness of data augmentation in cellular-based localization using deep learning. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2019.

[27] R. Roelofs, N. Cain, J. Shlens, and M. C. Mozer. Mitigating bias in calibration error estimation, 2021.

[28] X. Song, X. Fan, C. Xiang, Q. Ye, L. Liu, Z. Wang, X. He, N. Yang, and G. Fang. A novel convolutional neural network based indoor localization framework with wifi fingerprinting. *IEEE Access*, 7:110698–110709, 2019.

[29] H. W. Sorenson. Least-squares estimation: from gauss to kalman. *IEEE Spectrum*, 7(7):63–68, 1970.

[30] M. Sun, Y. Wang, S. Xu, H. Qi, and X. Hu. Indoor positioning tightly coupled wi-fi ftm ranging and pdr based on the extended kalman filter for smartphones. *IEEE Access*, 8:49671–49684, 2020.

[31] P. Teunissen and A. Amiri-Simkooei. Least-squares variance component estimation. *Journal of Geodesy, 82 (2)*, 82, 02 2007.

[32] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 261–270, 2014.

[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12):3371 – 3408, 2010.

[34] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang. Indoor fingerprint positioning based on wi-fi: An overview. *ISPRS International Journal of Geo-Information*, 6:135, 04 2017.

[35] C. Xiao, D. Yang, Z. Chen, and G. Tan. 3-d ble indoor localization based on denoising autoencoder. *IEEE Access*, 5:12751–12760, 2017.

[36] S.-C. Yeh, W.-H. Hsu, W.-Y. Lin, and Y.-F. Wu. Study on an indoor positioning system using earth's magnetic field. *IEEE Transactions on Instrumentation and Measurement*, 69(3):865–872, 2020.

Master's Theses in Mathematical Sciences 2021:E31
ISSN 1404-6342

LUTFMA-3448-2021

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

http://www.maths.lth.se/