UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Henry Teigar

Indoor Localisation Using Received Signal Strength

Bachelor's Thesis (9 ECTS)

Supervisor: Meelis Kull, PhD

Tartu 2018

Indoor Localisation Using Received Signal Strength

Abstract:
Indoor localisation of people has gained a lot of interest during the last decade. Different approaches have been proposed and tested in various environments. This thesis tries to predict a person's location in the SPHERE testing house. SPHERE is a project with an aim to use sensor technology for healthcare, such as early diagnosis of different illnesses by monitoring person's activity in their homes. Accurate localisation of the person can provide useful information for this purpose. We use the received signal strength indicator (RSSI) values between the receivers with fixed positions and one mobile node to perform the localisation. For this we use two machine learning methods: hidden Markov models (HMMs) and k-nearest neighbors algorithm (k-NN). A detailed description of the implementation process of both models used on the SPEHRE dataset is also given. Finally, we provide the results and the comparison of both approaches. We found that after feature pre-processing, the k-NN performed surprisingly well by achieving room-level accuracies around 90%. The initial performance of the HMM was found to be similar to k-NN's but with our modifications to the HMM, we finally achieved accuracies up to 96%.

Keywords: machine learning, localisation, RSSI, HMM, k-NN

CERCS: P170 - Computer science, numerical analysis, systems, control

Vastuvõetud signaalide tugevusel põhinev lokaliseerimine siseruumides

Lühikokkuvõte: Inimeste lokaliseerimine siseruumides on viimase aastakümne jooksul kõvasti populaarsust kogunud. Selleks on välja pakutud ning testitud erinevaid viise. Käesolev töö proovib ennustada inimese asukohta SPHERE testmajas. SPHERE on tervisehoiuga seotud projekt, mille eesmärgiks on kasutada sensortehnoloogiat, et näiteks varakult tuvastada erinevaid haiguseid, jälgides inimese tegevusi tema kodus. Täpne inimese asukoha määramine võib selleks olulist infot anda. Lokaliseerimiseks kasutatakse siin töös vastuvõetud signaali tugevuse (*received signal strength indicator - RSSI*, ingl k) väärtuseid fikseeritud vastuvõtjate ja mobiilse sensori vahel. Selleks kasutatakse kahte masinõppe meetodit: peidetud Markovi ahelaid (*hidden Markov model - HMM*, ingl k) ning k-lähima naabri (*k-nearest neighbor - k-NN*, ingl k) algoritmi. Antakse ka detailne ülevaade mõlema meetodi implementatsiooni protsessist kasutades SPHERE andmestikku. Viimaseks esitame mõlema meetodiga saadud tulemused ning võrdleme

neid. Leiti, et k-NNi võimekus peale tunnuste eeltöötlemist oli oodatust parem, saavutades ruumisiseseid täpsusi 90% ümber. Esialgne HMMi võimekus oli sarnane k-NNi omaga, kuid meie pakutud HMMi muudatustega suudeti viimaks saavutada täpsuseid kuni 96%.

Võtmesõnad: masinõpe, lokaliseerimine, RSSI, HMM, k-NN

Contents

Introduction

The importance of indoor localisation of people has grown significantly during the last decade. As people are spending over 85% of their time in indoor environments [KNO+01], accurate localisation can have a big impact not only by simplifying people's lives, but for example also by helping firefighters, police, soldiers, medical personnel to save lives and perform specific tasks [ZF13].

There are multiple difficulties when it comes to achieving high precision indoor localisation. Standard approaches including Global Positioning System (GPS) that are used for outdoor localisation cannot be easily used due to unreliability and obstacles that are present in indoor environments. Fortunately, with a rapid rise in the popularity of Internet of Things (IoT) and smart-home technologies, an increasing number of homes and other indoor spaces are already equipped with wireless sensor networks (WSNs) which can be used for accurate localisation.

Numerous different algorithms and approaches for localisation in a WSN exist. One of the possibilities is to use received signal strength indication (RSSI) values between a sensor node attached to the person and multiple receivers with fixed locations. However, most of the methods using RSSI require detailed information about the environment. Accurate maps of precise sensor positions and surroundings must be created and that makes the setup process of such methods very time-consuming [Xia11]. The other possibility for indoor localisation is to use different machine learning methods to analyse the RSSI data [STSK16, Xia11].

This thesis focuses on predicting a person's location in a WSN based on the RSSI using the supervised machine learning approach by learning the RSSI patterns and corresponding true location labels for the person. Two methods will be used: hidden Markov model (HMM) and k-nearest neighbors (k-NN). We will use a dataset provided by Sensor Platform for HEalthcare in Residental Environment (SPHERE) challenge [TDK+16a]. A longer description of SPHERE challenge itself and its importance will be covered later in the thesis.

The goal for this thesis is to predict a person's room-level location by the RSSI info from the SPHERE dataset using the HMM and the k-NN. Besides, a detailed comparison of the performance of both approaches and a description of the implementation process will be given. In addition, we will analyse the effect of using different approaches for calculating the HMM parameters.

The thesis has four main sections. In Section 1 we will cover the basic principles of RSSI localisation and give a brief introduction to supervised machine learning. Furthermore, a detailed look at used machine learning approaches will be given. Section 2 will describe the used dataset, including information about the structure of the available data but also explain how and why it was collected. Section 3 describes the implementation

process of previously mentioned machine learning algorithms. Finally, Section 4 presents and analyses the results of the used methods. The repository link to the scripts created for this thesis can be found from Appendix I. Appendix II shows the floorplan of the SPHERE house. A licence is provided at the end of the thesis.

1 Background and related work

This chapter first gives an overview of different approaches for RSSI localisation. Next a brief overview of supervised machine learning will be given. This is followed by a more specific description of the concepts and algorithms used in the problem solution. The chapter ends by covering the related work.

1.1 Localisation using RSSI

Received signal strength indication (RSSI) indicates the total amount of power present in a received radio signal and is typically measured in units of decibel-milliwatts (dBm) which are represented as negative numbers. The closer the values are to 0 dBm, the stronger the received signal is [Sau17].

In order to locate a person in a WSN environment with the RSSI, the RSSI values between the sensor attached to a person and surrounding access points (APs) with fixed locations are measured. The combination of these multiple RSSI values can be used to calculate the approximate position of the person. Typically, at least 3 access points are required for the localisation [Xia11].

The techniques used to convert the RSSI values into the location can be divided into two main categories: range-based and range-free. Range-based methods try to assign a distance value to specific RSSI values based on some function. However, non-line-of-sight (NLOS) scenarios where there is no direct path between the sensor node and receiver nodes, can significantly distort the RSSI and introduce unpredictable measurements. NLOS problems can be avoided by having detailed maps of precise sensor positions and the environment but therefore the setup process of such methods is very time-consuming. In contrast, the range-free approach that is used in this thesis does not require any information about the exact positions of the sensor nodes. Instead, localisation is performed by analysing and processing the patterns found in RSSI data [Xia11, LYABW16]. There are different ways to accomplish this. As mentioned in the introduction, in this thesis we will use a supervised machine learning approach by using hidden Markov model and k-nearest neighbors methods. Even though HMM is not traditionally considered as a supervised machine learning model, the way we calculate and learn its parameters (described in Subsection 3.4.1) can be viewed as supervised machine learning. An explanation of the choice for these methods is covered at the beginning of Section 3, as by then a better overview of both methods and also the used dataset will have been given.

1.2 Supervised machine learning

In supervised machine learning the aim is to learn (i.e. train) a model based on a labelled input dataset and then use the trained model to predict the correct labels of unlabelled data. Based on the type of the labels that need to be predicted, supervised machine learning algorithms are divided into two main groups: regression algorithms and classification algorithms. Regression algorithms try to predict a continuous value (e.g. length of some item), while classification algorithms try to assign a specific class or discrete value to the input (e.g. result of a die roll) [KZP07]. In this thesis, the label we try to predict is the name of the room where a person is located at a given time. As the room label is a discrete value, we need to use classification algorithms.

In general, the learning process and evaluation of the supervised machine learning models are the following:

1. A dataset for training the model is collected (i.e. training data). Each individual data sample (i.e. data point) has a correct label associated with it. In our case, the data sample represents the RSSI values and the label is the name of the room;
2. Model is trained using the training dataset;
3. The performance of the trained model is tested on the testing data where the correct labels exist but are unknown for the model. The correct labels are later compared with the predicted ones to estimate the performance of the model.

A common term in machine learning that is also used frequently later in this thesis is *overfitting*. When the model is overfitted, it does not generalise well on new data, resulting in good results on training data but worse results on test data. In practice, additional steps like cross-validation are often used to reduce overfitting. We will cover this more specifically later in the implementation described in Section 3.

1.3 Hidden Markov model

Hidden Markov Model (HMM) is a probabilistic model that is widely used as a statistical tool for modelling time series data, especially for discrete-valued series. It has gained a lot of success for example in areas focused on signal processing and natural language processing (NLP) [Blu04, ZM09].

HMMs have a visible sequence of observations but the sequence of internal states that generate the observation is unknown or hidden. In our case, the hidden internal states are the names of the rooms and the visible observations are the RSSI values.

When comparing the basic Markov chain and HMM, then in the former, the current state is freely observable and the only parameters are the state transition probabilities. The HMM, however, as the state is hidden and only the output (i.e. emissions) is observable,

has to introduce additional parameters which specify the probability distribution of the emissions in all the different states. HMM also has a parameter that determines the starting state probability [Blu04, ZM09].

The HMM is therefore completely determined by three parameter matrixes: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}$. To better understand the meaning of each parameter, let's define the following notation (similar notation to [RJ86, Blu04] is used):

$$N = \text{number of different states}$$
$$M = \text{number of different observations}$$
$$T = \text{length of observation sequence}$$

S is the state set and V is the observation set. In this thesis the state set consists of all the different rooms and the observation set consists of all the different possible RSSI value combinations.

$$S = \{s_1, s_2, ..., s_N\}$$
$$V = \{v_1, v_2, ..., v_M\}$$

Q is the state sequence and O is the observation sequence. Note that Q and O have the length of T.

$$Q = (q_1, q_2, ..., q_T)$$
$$O = (o_1, o_2, ..., o_T)$$

$\boldsymbol{A}$ is a $N \times N$ transition probability matrix that shows the probability that state i is followed by state j. In our case, it provides the information about the probability of going from one room to another. The probability is found for every combination of two rooms.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix}, a_{ij} = P(q_t = s_j | q_{t-1} = s_i), t = 2, ..., T$$

$\boldsymbol{B}$ is an $N \times M$ observation (i.e. emission) matrix. It provides information about the probability of observation k inspected from state i. The probability is independent of t. In case of our problem, the emission matrix specifies the probability of a specific RSSI value being measured from one specific room. The probability is found for every RSSI value in every room.

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1M} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & b_{N3} & \dots & b_{NM} \end{bmatrix}, b_{ik} = P(v_k | s_i)$$

$\boldsymbol{\pi}$ is a $1 \times N$ matrix that shows the probability of state i being the first state of the state sequence. In our case, it represents the probability of one specific room being the first room of the sequence.

$$\boldsymbol{\pi} = \left[\pi_1, \pi_2, ..., \pi_N\right], \pi_i = P(q_1 = s_i)$$

The compact notation for HMM is:

$$\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$$

1.3.1 Viterbi algorithm

Viterbi algorithm is meant to solve one of the main problems for HMM. The problem is defined as follows: given the model λ and the observation sequence O, how to find the most likely hidden state sequence Q [RJ86].

A naive way to solve this problem would be to calculate the probabilities of all the possible state paths and find the path with the highest probability. The running time for this, however, would be exponential $O(N^T)$, where N is the number of different states and T is the length of the observation sequence.

The Viterbi algorithm solves this recursively by splitting the problem into smaller parts. The complexity for the Viterbi algorithm is $O(N^2 \times T)$, which is much more efficient than using the naive way. The basic principle for Viterbi is that the most likely state path to each specific point in time can be deduced from the most likely state path to the previous point in time. A more detailed explanation can be found from [RJ86].

1.4 K-Nearest Neighbor Classification

As its name suggests, k-NN tries to predict the class of an unknown data sample based on the known classification of its nearest neighbors. In general k-NN first calculates the distances between the new sample and every data point with known classification. Then k nearest neighbors are selected and the most frequent class among the nearest neighbors is assigned to the unclassified sample. That makes k one of the most important configurable parameters of the k-NN. [MPP09].

Figure 1 illustrates the k-NN decision rule when $k = 5$ for data points from two different classes. In this case, the unlabelled sample will be assigned the same class as the data points on the right side, as three out of five closest neighbors belong to that class.

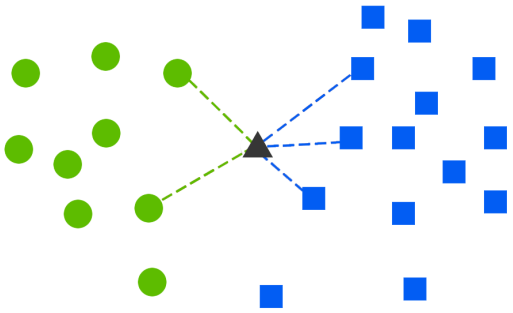

Figure 1. K-NN classification visualisation with two different classes where $k = 5$.

The other important parameter in k-NN besides k is the distance measure. In this thesis Euclidean-based k-NN is used. This means that the distance between two data points is calculated using the Euclidean distance. When taken two data points d_i and d_j with features $(1, 2, .., k)$, then the Euclidean distance is defined as follows [Bar05]:

$$dist(d_i, d_j) = \sqrt{(d_{i1} - d_{j1})^2 + (d_{i2} - d_{j2})^2 + ... + (d_{ik} - d_{jk})^2}$$

where $d_{i1}, ..., d_{ik}$ and $d_{j1}, ..., d_{jk}$ denote the values of k features of d_i and d_j respectively.

1.5 Related work

Indoor localisation is a very popular and actively researched topic. Even though there is no previously published work on the localisation problem on the SPHERE dataset, different approaches and techniques have been proposed for the general use cases of indoor localisation. As this thesis focuses on the localisation using RSSI data, we now give a brief overview of works that also use RSSI information to perform the localisation.

Most of these works focus on range-based RSSI approaches and usually manual mapping of the fixed receiver nodes needs to be done. For example, in one study [JHB+17] it was assumed that the network consists of one mobile node and several other nodes with fixed positions. The position of these fixed sensor nodes in that specific paper was found using a GPS receiver.

There are also multiple published works that similarly to our work use range-free approaches with the implementation of the HMM. However, most of these works use heavily modified approaches. For example, a paper by Y. Ni, J. Liu, S. Liu and Y. Bai [NLLB16] uses the HMM with a fuzzy pattern recognition algorithm and in addition, the technique is tested on a single floor using 13 access points that measured the RSSI

values. As described in the next section, in our work we have a multi-floor environment with only three access points which makes the localisation more challenging in some aspects.

One example of a paper that uses the k-NN for localisation in a WSN is made by Chih-Ning Huang and Chia-Tai Chan [HC11]. They use a custom weighted k-NN approach. To do that, they divide the RSSI values into different groups and multiply the values by predefined ratios for each group. They found that it increases the accuracy of the k-NN. However, they assume that there is no NLOS between sensor nodes. Furthermore, the experiment was done in a single room, and not in a real-life scenario with multiple rooms on multiple floors.

The majority of previously released works that use similar methods to ours, focus on the coordinate-level positioning, which makes the results of these works hard to compare to ours as in this thesis we take the localisation as a classification problem by estimating the room-level location of the person. A work by Lee and Lampe [LL11] is the only example we found that has used classification algorithms for the localisation problem by utilising the k-NN and the HMM among other approaches. They tested the performance of these models in an environment with 9 rooms and 8 access points. All the rooms were on the same floor. With the k-NN they received an accuracy of about 60% and with the HMM around 70%. However, the implementation of the HMM was done very differently from our approach by using it with the Gaussian classifier and therefore cannot be directly compared. There was also no source code of their implementations provided.

These were just a few examples of the related works. In this thesis, we try to explain the actual implementation process of the methods and how it is done specifically on the SPHERE dataset. In addition, a link to the source code of the scripts produced for this thesis is provided in Appendix I.

2 Dataset

This chapter will give a detailed overview of the dataset used in this thesis. We will be using data created for the Sensor Platform for HEalthcare in Residental Environment (SPHERE) challenge [TDK+16b]. First, we will give a brief overview about why and how the data was collected. This is followed by a more specific description about the characteristics of data that are directly relevant for this thesis.

2.1 SPHERE project and data collection

The goal for the SPHERE Interdisciplinary Research Collaboration (IRC[1]) project is to provide a platform that helps to diagnose some of the most common health conditions like obesity, depression, stroke etc. by analysing long-term behavioural patterns that lead to such illnesses. The sensor platform will also provide means to constantly monitor the patients and therefore enable them to live in their homes. The system is currently being deployed to 100 homes in Bristol (UK).

SPHERE uses the following sensors and cameras for data collection in these houses:

1. wrist-worn accelerometer;
2. RGB-D cameras (i.e. video with depth information);
3. passive environmental sensors.

The dataset used in this thesis contains data from a single testing house which is meant for solving the SPHERE challenge. The main aim for this challenge is to predict the current activity and posture of the person using the information from the sensors. Accurate localisation of the occupant can provide useful information for this task as the activities are dependent on the location in the house. For example, if a person is in the kitchen, it is not reasonable to predict that they are walking up the stairs at the same moment.

For the localisation we use the RSSI values between the wrist-worn accelerometer and access points (AP) that are distributed around the testing house. In addition, the true location labels will be used for training and evaluating the machine learning models that are used in this thesis. A more detailed description of the RSSI data and the annotations is provided in Subsections 2.2 and 2.3 respectively.

It is also worth mentioning, that some of the rooms in the SPHERE house were also equipped with passive infrared (PIR) motion sensors that can be used for localisation. However, it is impossible to distinguish different persons individually when using only

[1] IRC - Interdisciplinary Research Collaboration

the motion sensors. Although combining the RSSI values with the information from the PIR sensors can increase the localisation accuracy, we will not be using the PIR sensors in this thesis and this is left for future research.

It is important to note that the data has been produced by recruited participants who had to follow a pre-defined script. The script defined exactly what actions and in which order the participants had to perform. One full execution of this script lasted about 30 minutes. The main reason why the data has been collected by a pre-defined script, was to generate labelled (i.e. annotated) data, that are suitable for machine learning, especially for supervised machine learning algorithms.

The floor plan of the testing house is included in Appendix II.

2.2 Accelerometer data

Participants wore the accelerometer on their dominant wrist. There were four access points around the house which received the wirelessly transmitted signal from the accelerometer. Bluetooth Low Energy (BLE) standard was used to communicate between the accelerometer and the access points. The access points received a continuous numerical stream of acceleration readings. In addition, they also received signal strength indications (RSSI) in units of dBm. These are integer values, that are no lower than -110 dBm and by the definition of RSSI below 0 dBm. The data gets recorded at 20 Hz. As stated earlier, we use the RSSI readings between the wrist-worn accelerometer and the receivers as the input data source for solving this thesis problem.

The accelerometer dataset consists of ten different subsets, each containing one recording of the pre-defined script. For every recording the following information is provided for every data sample (i.e. data from one specific timestamp):

1. **t:** time of the recorded sample in seconds with a precision of 6 digits;
2. **(x,y,z):** acceleration values on the x-/y-/z-axes of the accelerometer. These will be ignored in this thesis;
3. **(Kitchen_AP, Lounge_AP, Upstairs_AP, Study_AP):** these four specify the RSSI readings from each of the four access points in the SPHERE house. Empty values indicate that the access point did not receive any signal.

In each recording there are roughly 35 000 samples (corresponding to about 30 minutes) of accelerometer data, in the order they were gathered. Unfortunately, as a result of technical issues at the time of the recording, only three out of four access points were working. In addition, half the recordings contained data from one subset of four access points and the other half from a different subset of access points. Therefore we will only be using data from 5 recordings, which each have readings from the same three access points named Lounge_AP, Kitchen_AP and Upstairs_AP.

2.3 Annotations

Among other data, also the information about the true location of the person is provided for every recording. These annotations were created by a team of 12 annotators. During the execution of the pre-defined script, a head mounted camera was used to support the annotation process. Eight different occupation labels were used: bath, bed1, bed2, hall, kitchen, living, stairs, toilet. Figure 2 will give a better understanding of the nature of RSSI from the three access points.

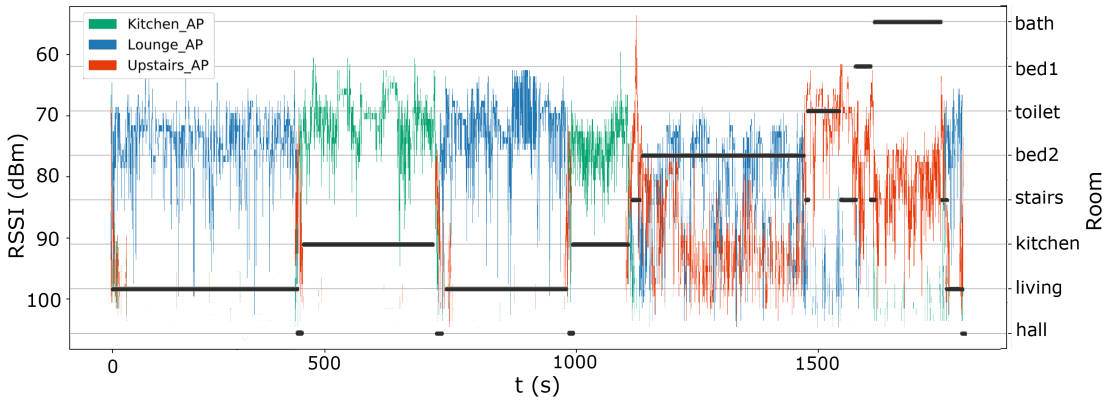

Figure 2. Data visualisation of the first recording. Colours blue, green and red represent RSSI of Lounge_AP, Kitchen_AP and Upstairs_AP correspondingly. The black horizontal lines represent the room (i.e. state) the person is currently in.

The annotation labels for room occupancy are provided in the following form:

1. **start:** a timestamp the participant entered the room in seconds with a precision of 6 digits, relative to the start of the sequence;
2. **end:** a timestamp the participant left the room in seconds with a precision of 6 digits, relative to the start of the sequence;
3. **name:** the name of the room;
4. **index:** the index representation of the room (not used in this thesis).

3 Localisation

This chapter starts by explaining the choice of used machine learning methods and also goes over the libraries used in the scripts written for this thesis. Afterwards a detailed description of the implementation process for both methods will be given. The chapter ends by explaining the evaluation process.

3.1 The choice of methods

As mentioned previously, for the localisation problem we decided to use the hidden Markov models with Viterbi algorithm and also the k-nearest neighbors method.

As the RSSI values coming from the accelerometer are in chronological order, HMM is chosen because it is specifically meant for modelling time-series data. The other aspect is that the RSSI values in the SPHERE dataset are relatively noisy, partly because of the nature of RSSI but also because RSSI values are highly dependent on the exact orientation and activity of the person in the room. As HMM is aware of the surrounding context of the data point, the noise has a lot less impact on the predictions.

K-NN is partly chosen for its popularity for being used as a comparison method to more complex models, mainly due to its simplicity. The other reason for choosing k-NN is that with relatively big values of k, it should also be able to reduce the effect of the noise.

3.2 Used libraries

For k-NN classification we use *scikit-learn*[2] library. For localisation using the HMM, *hmmlearn*[3] library is used. For data manipulation also *NumPy*[4] and *pandas*[5] are being used. Most of the data visualisations are made using *Matplotlib*[6] and *ggplot2*[7] libraries. All the mentioned libraries except *ggplot2* are written in Python, *ggplot2* is written in R.

3.3 Data pre-processing

There is one structural change for the dataset that needs to be done before the training of HMM and k-NN can start. As described in Section 2, the true room labels are provided by

[2] http://scikit-learn.org
[3] http://hmmlearn.readthedocs.io
[4] http://numpy.org
[5] https://pandas.pydata.org
[6] https://matplotlib.org
[7] http://ggplot2.org

the start and end timestamps but in order to simplify the training and validation process, a script is written to generate a separate room label for every RSSI data sample. This means that the length of RSSI data and the true labels (i.e. correct rooms) become equal.

3.3.1 Data pre-processing for the k-NN

As described in Subsection 1.4, k-NN works by finding the distance between the current test instance and training instances. However, as the RSSI values are noisy, without data pre-processing k-NN often assigns the wrong room label to the data point. To reduce this problem, in addition to experimenting with different values of k, we will not be using the raw RSSI values for training and testing the k-NN. Instead, we will be using the average RSSI value over the last n timestamps for every access point. The parameter n will be named as avg later in this thesis. In addition, different values for this parameter will be tested. As mentioned in Section 2, there are also missing RSSI values when the received signal is lower than -110 dBm. These empty values are replaced with -110 dBm. RSSI value distributions with two different values of n are visualised in Figure 3.

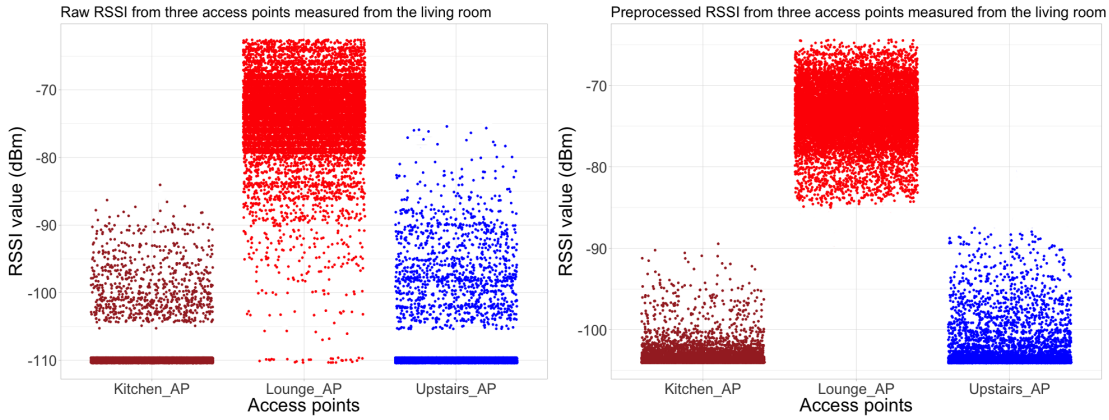

Figure 3. Illustration of RSSI values in the living room, measured by three access points where each dot represents the value of one measurement. On both images, the empty values are already replaced with -110 dBm. On the left image, unprocessed (except the replacement of empty values) RSSI values are shown ($avg = 1$). On the right image, every value represents the average over the last 50 timestamps ($avg = 50$). For visualisation purposes on both images the points have been randomly jittered along the x-axis.

3.3.2 Data pre-processing for the HMM

For the HMM no explicit data pre-processing needs to be done as the missing RSSI values are treated as separate discrete values. A better understanding of how we extracted and found the discrete values from the SPHERE dataset for HMM is covered in Subsection 3.4.1.

3.4 Localisation using HMM

3.4.1 Finding the parameters

Training the HMM means to compute the three required parameters (A, B, π) described in Section 1.3. There exist different ways to solve this problem. One possibility is to use the Baum-Welch algorithm, which is an unsupervised machine learning algorithm [RJ86, Blu04]. But as we have a relatively large labelled training corpus available, we will use a supervised approach, to calculate the parameters based on the seen training data.

Computing the transition matrix As described in Subsection 2.3, we have a total of 8 possible states (bath, bed2, living, kitchen, stairs, toilet, bed1, hall), each representing one room from the testing house. Therefore, our transition matrix will have a dimension of 8×8. We will use two different approaches to calculate the transition matrix.

The first approach uses the labelled data to calculate the probability a_{ij} of going from room i (represented by state s_i) to room j (represented by state s_j). This is done by counting all the instances where room i is followed by room j and then dividing this count by all the samples that had room i associated with them. Mathematically it can be represented as follows:

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i) = \frac{count(i, j)}{count(i)}$$

where $count(i, j)$ represents the number of instances where room i at time $t-1$ (denoted as $q_{t-1} = s_i$) is followed by room j at time t (denoted as $q_t = s_j$) and $count(i)$ represents the total number of samples being in room i. The full transition matrix is calculated by doing the described approach for every room.

The other approach is not to use the actual data but instead use just the room positioning information in the testing house. The probability of going from one room to another depends on whether the rooms are actually connected. The transition probability between two rooms that are not connected in real life, should definitely be 0. For example, as seen from the SPHERE house plan in Appendix II, transition probability between the kitchen and the toilet should be 0. The probability of staying in the same room is quite hard to

estimate. For example in the pre-defined script the transitions are done on average after every 1.6 minutes. As a tradeoff between this and the frequency of the room transitions made by an average person in real life (unknown), we decided to use 5 minutes as the transitioning interval. With this assumption and the fact that data gets recorded at 20Hz, the probability of staying in the same room is set to a constant value of 0.9999. The probability of transitioning to another room is found by dividing the remaining probability $(1 - 0.9999)$ with the number of surrounding rooms that are connected with the current room. As it turns out in Section 4 the transitioning probability has a big impact on the performance of the model. Therefore, when the system is taken into use in real households, it would make sense to calculate the transitioning probability based on how active the specific person is. For less active occupants a smaller transitioning probability between rooms should probably be used.

In the rest of the thesis we call the transition matrix found by this approach the hardcoded transition matrix and the matrix found by the previous approach the non-hardcoded transition matrix.

The first approach is expected to yield better results on the same dataset but at the same time it increases the possibility of overfitting to this particular pre-defined script. The second approach is more likely to perform similarly in different scenarios.

Computing the emission matrix As described in Subsection 1.3, the observation matrix provides information about the probability of observation k inspected from state i. The problem, however, is that we do not have a fixed set of different discrete observations. Instead, we are dealing with continuous sensor values where one observation consists of three different continuous values. There are different solutions for this problem.

The approach used in this thesis converts the continuous values into discrete values. The first step is to convert the values coming from one specific sensor, into discrete ones. The second step is to treat every possible combination of the three discrete values as one distinct observation.

Theoretically, the sensor values can already be viewed as discrete, because as described in Subsection 2.2, one sensor value is represented with an integer that is no smaller than -110 dBm and by the definition of RSSI, no larger than 0 dBm which results in 110 different discrete values for each sensor. However, that means, that by combining the values, there are $110 \times 110 \times 110 \approx 1.3$ million different observations. The probability that the exact same observation is seen both in training and testing data is small. Therefore, when converting one specific sensor value to a discrete one, it is reasonable to split the range of values into smaller sections and treat each section as one discrete value. The optimal number of sections the values should be divided into will be covered later in the thesis and will be called $split\_count$. When splitting the values into sections, we take into account the fact that most RSSI values fall in the range of -110

dBm to -60 dBm. When $split\_count = 2$ we use the center value of this range (in this case -85 dBm) as the splitting point. When $split\_count = 3$ we use -110 dBm and -60 dBm as the splitting points. So we have separated the values smaller than -110 (in this case the empty values), values that fall between -110 dBm and -60 dBm and the values bigger than -60 dBm. For $split_count$s bigger than 3 we additionally divide the range between -110 dBm and -60 dBm into $split\_count - 2$ sections with equal size.

Based on the observed RSSI values from the training data, a total of 24 different probability distributions will be calculated (for every three sensors for every 8 rooms (i.e. states)). The probability distributions for bedroom 2 are illustrated in Figure 4. To compute the probability for any triple of sensor values in any room, the probabilities from corresponding probability distributions are multiplied. For example, the probability of RSSI triple (Kitchen_AP=empty; Lounge_AP=-82; Upstairs_AP=-89) in bedroom 2 is $0.99 \times 0.26 \times 0.54 \approx 0.139$ where 0.99, 0.26 and 0.54 are the corresponding probabilities of the sections that these specific sensor values fall into. By doing this we are treating the values from three sensors as independent, even though in reality they are not. This assumption, however, is necessary for the HMM and in practice yields good results.

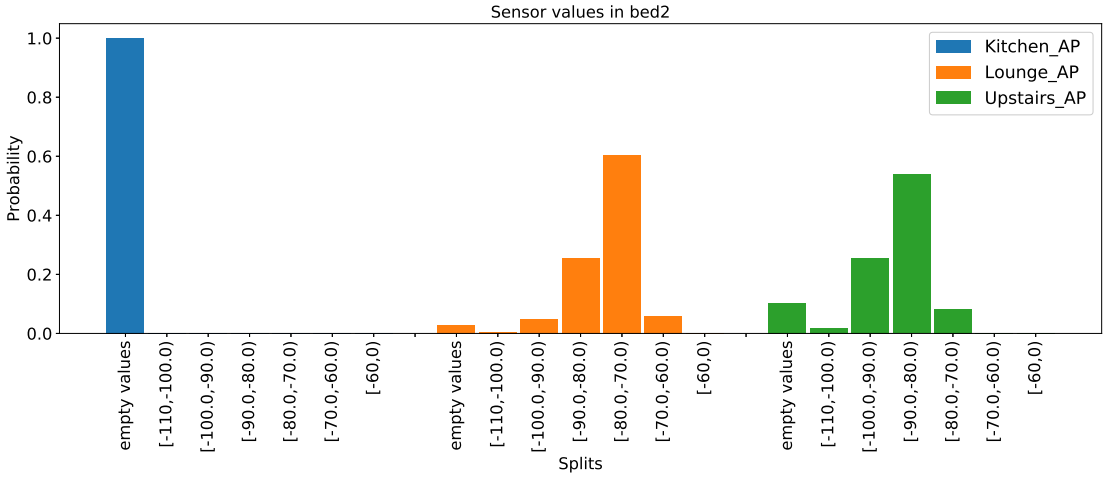

Figure 4. Probability distributions for three sensors in bedroom 2, where each sensor's values are divided into 7 sections ($split\_count = 7$).

Computing the starting probability matrix The calculation to produce the starting probability matrix is probably the most intuitive. The probability of state i being the first state could be calculated by dividing the number of data points having state i by the total number of data points.

To achieve the best results on this particular dataset, it would also seem reasonable to calculate the starting probability matrix based only on the first room of every recording. However, this would result in heavy overfitting as in reality the starting room of the recording can potentially be any room in the house.

In order to prevent overfitting, we decided to assume that the starting probability is equal for every state. This should increase the performance of the model on unseen data.

3.4.2 Predicting the hidden states

Viterbi algorithm is used to predict the hidden state sequence in HMM. The main idea behind the Viterbi algorithm is explained in Subsection 1.3.1. We use the help of *hmmlearn* library to make predictions with the Viterbi algorithm.

The method `hmm.MultinomialHMM()` is used to initialize the HMM. Three required parameter matrices (transition matrix, emission matrix, starting probability matrix) must be provided to the method. For the calculation of these parameters multiple scripts are written that implement the logic described in the previous Subsection 3.4.1. The link to the source code for this and all the other implementations performed in this thesis can be found from Appendix I.

The method `hmm.MultinomialHMM.decode()` is used to get the predicted state sequence based on the observations taken as an argument.

3.5 Localisation using k-NN

The training and testing of k-NN is performed using the *scikit-learn* library. The model is trained using method `neighbors.NearestNeighbors().fit()`. The authors of *scikit-learn* have chosen $k = 5$ as a default value of k. Different values of k will still be tested in this thesis to determine the best value for the used dataset.

3.6 Evaluation of the methods

In this subsection, we will first analyse the difficulties in estimating the performance of our chosen methods. This is followed by a more detailed description of the selected evaluation technique.

3.6.1 Choosing the evaluation method

Due to the nature of the SPHERE dataset and the choice of methods, there are multiple aspects that need to be taken into account when choosing the approach to evaluate the performance and find the best parameters for the models.

As described in Section 2, all the usable data we have consists of five separate recordings of the same pre-defined script. That means that in each recording the order in which the participants visited the rooms is the same. However, our goal is to train models, whose performance is also good at predicting a person's location when a pre-defined script is not followed. Therefore, the available data need to be used as effectively as possible to train a model that is not overfitted to one particular pre-defined script.

The second aspect that needs to be taken into account, is that the SPHERE dataset consists of time-series data. This means that choosing the correct evaluation technique is extremely important as wrong evaluation methods can lead to highly inaccurate or over-optimistic results. The problem with time-series data is that data points in the same recording that are close to each other in terms of time can have very similar characteristics. In case of our dataset, the RSSI values are recorded at 20Hz or after every 0.05 seconds. As 0.05 seconds is a relatively small period, two data points within this timeframe can have very similar RSSI values. This means that the partitioning strategy for the training and testing subsets should not be random, otherwise it is highly likely that a data point with almost the exact same values exists in both the training and testing set. This would result in overly optimistic model accuracy.

The third aspect is that as one of the models used is the HMM, the data used for training and testing must definitely be ordered due to the prerequisites of the HMM. This means that it is not reasonable to shuffle the data samples before training and testing the model, as this would break the sequential nature of the time series.

Finally, we need to consider the actual amount of training data available when the SPHERE project is put into use in hundreds of homes. As the setup process needs to be fast, we assume in this thesis that we do not have more than one recording at once for training.

When taking into account the previously described aspects, a modified version of nested cross-validation is chosen to be implemented in this thesis. A detailed description of this method is described in the next Subsection 3.6.2.

3.6.2 Modified nested cross-validation

In standard cross-validation (CV), the training data are divided into subsets. Then one subset is used for validation and others for training. This process is repeated multiple times until each subset is used at lest once for validation. This enables to test the model on different combinations of training and validation data and therefore reduces the risk of overfitting. The performance of the model is estimated by taking the average accuracy over all the testing iterations. Based on the average accuracy the model parameters can also be tuned.

A usual approach is to finally test the model with tuned parameters on independent

unseen data, that were not included in the cross-validation process.

In our case, it would be ideal if we could perform the cross-validation on the recordings with the pre-defined script and then finally evaluate the model on data that was recorded without the pre-defined script. This would perfectly mimic the real-life scenario where the setup process is done by following some instructions but the model should generalise well on unscripted data. Unfortunately, no data without the pre-defined script is currently available. To cope with this problem, we use a variation of nested cross-validation.

With nested cross-validation first an outer cross-validation is performed to split the data into training and testing folds, and then an inner cross-validation is performed on the training data to determine the best parameters for the model. The best model found with the inner cross-validation is finally evaluated on the testing set that was extracted during the outer cross-validation. It is important to understand, that as the final test set changes with every iteration of outer cross-validation, the result of nested cross-validation is not just one model, but the number of models is determined by the number of iterations in the outer cross-validation.

In our case (illustrated in Figure 5), we perform the outer cross-validation on five recordings. First, we select one recording as a testing set and on the other four we perform the inner cross-validation. In each iteration of inner cross-validation, we select one recording for training and other three for validation. Usually, it would be the other way around - most of the data would be used for training and a smaller subset for validation. However, as described earlier we do not want to use more than one recording for training at once. For validation, we use a custom approach to get an as realistic estimation of the model performance as possible. Instead of testing the model on the whole validation data at once, we use a 10-minute testing window for one evaluation. After every evaluation, the window is moved forward in time by 1 minute and the testing is repeated. Furthermore, for each window of data, the model is also tested with a reversed sequence of the same data. The model performance in one iteration of inner cross-validation is estimated by taking the average accuracy over all the windows. The same process is repeated for every iteration in inner cross-validation. Based on the performance of inner cross-validation the best model parameters are found. The goal for these 10-minute windows and reversing the sequences is to minimise the effect of the pre-defined script. This process is especially important for the HMM, as the sequence and context of data have a big impact on this model. For the k-NN reversing the testing data does not affect the results at all, but to be consistent we perform the exact same evaluation technique for both models.

Finally, the model with best parameters found during the inner cross-validation is evaluated on the recording extracted for testing during the outer cross-validation. The final evaluation is done similarly as the validation for one iteration of inner cross-

validation by splitting the extracted test recording into 10-minute windows. But instead of selecting all the recordings used in the inner cross-validation for training, we train the model four times, each time by selecting a different recording for training. The trained models are evaluated on every 10-minute window of test data. The final accuracy of the model is calculated by taking the average accuracy over all the training sets and test windows.

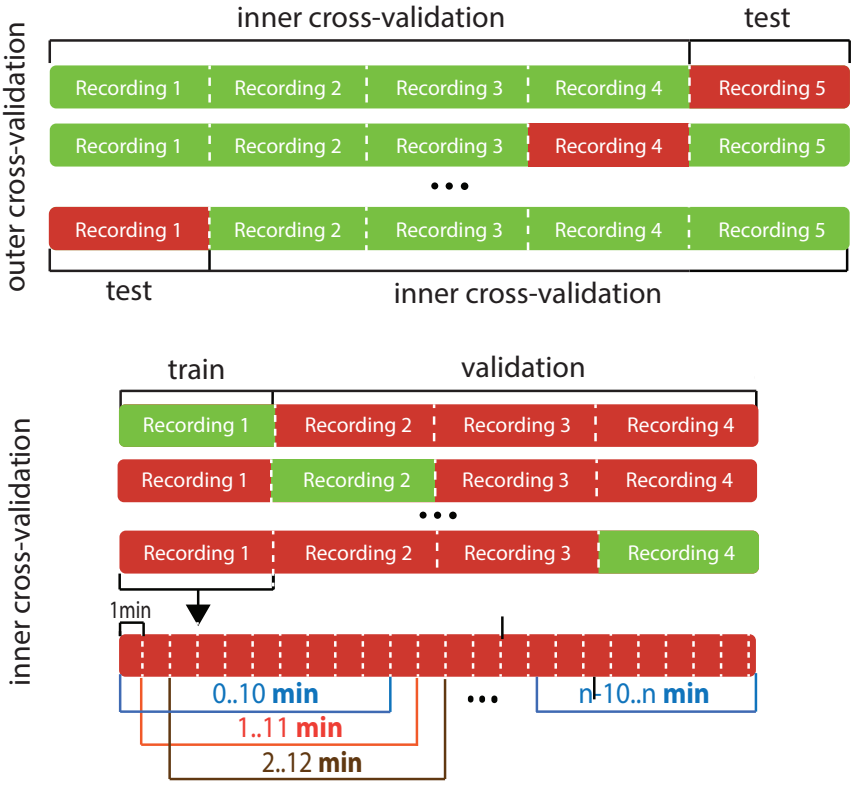

Figure 5. Visualisation of the nested cross-validation used in this thesis. In the outer cross-validation illustration (top) the red colour denotes the extracted test recording and the green colour shows the recordings used for the inner cross-validation. In the inner cross-validation visualisation (bottom) the red colour denotes the recordings used for validation and the green colour shows the recording used for training.

As we have a total of 5 recordings, at the end of the nested cross-validation, we also have a total of 5 models with different parameters and final accuracies.

4 Results and analysis

This chapter shows and analyses separately the results achieved with the k-NN and the HMM. The models are evaluated using the modified nested cross-validation described in Subsection 3.6.2. The chapter ends with a discussion about the limitations and future work.

4.1 Localisation results and analysis with the k-NN

As described previously in Subsection 3.3.1 where we covered the data pre-processing for the k-NN, in addition to using different values of k, we also try to determine the optimal number of timestamps over which the average RSSI values are calculated. For clarity in the following subsections we name this parameter as avg. The correlation between different values of k and the parameter avg will also be analysed.

For parameter k we use values ranging from 1 to 300 with a step of 10 (i.e. $k = 1, 10, 20, 30, ..., 300$) and for avg we try the following values: $1, 5, 10, 25, 50, 75, 100, 125, 150$.

Table 1 represents the results achieved with every iteration of outer cross-validation. For each row in the table, a different recording was selected for the final evaluation of the model which is also indicated by the *Test recording nr* column. *CV recording nrs* indicate the recordings used for inner cross-validation. *Best k* and *Best avg* columns represent the values that yielded the best average accuracy over all the iterations in the corresponding inner cross-validation. The last column shows the accuracy of the model achieved on test data using the *Best k* and *Best avg* values. The final testing is done as described in Subsection 3.6.2.

Table 1. Best parameters and corresponding accuracies for the k-NN in the final testing.

Test recording nr	CV recording nrs	Best k	Best avg	Accuracy on test data
1	2,3,4,5	210	100	0.891
2	1,3,4,5	110	100	0.899
3	1,2,4,5	200	100	0.917
4	1,2,3,5	200	100	0.889
5	1,2,3,4	210	100	0.901
				Avg: 0.899

The testing accuracy as seen in Table 1 was in the worst case 0.889 and in the best case 0.917. The average accuracy over all the models was 0.899. In all cases the best avg

determined during the inner cross-validation was 100 and the best k in most cases was 200 or 210. There was one occasion were the best k was 110. However, when analysing the results from that particular inner cross-validation, the accuracy difference between k being 100 or 200 was minimal.

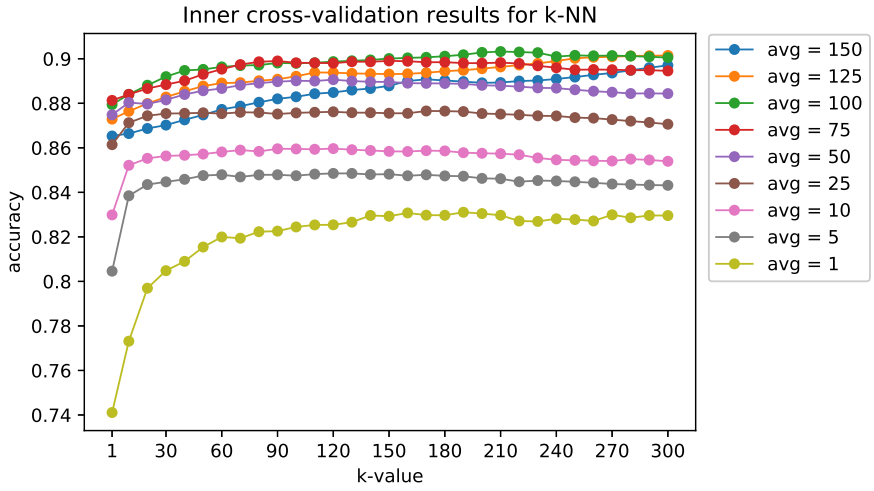

Figure 6. Results representing average accuracies during one of the five inner cross-validations performed with the k-NN model. Note that $avg = 1$ means that no data pre-processing has been done.

Figure 6 illustrates how different combinations of k and avg values affect the average accuracies during one of the inner cross-validations. It is clearly seen that data pre-processing for the k-NN has a big impact. The best average accuracy without the data pre-processing was only about 83% (accuracy achieved with the best k when $avg = 1$). In contrast, after averaging the RSSI data over 100 timestamps (or 5 seconds, in case of data being recorded at 20Hz), we achieved an average accuracy around 90%. The optimal value of k in that particular inner cross-validation was 210. This may seem relatively large but is reasonable when taken into account that the number of data points in one second is relatively high.

The other interesting aspect that should be pointed out, is the correlation between both parameters. It turns out that the value of k has a much bigger impact when there is less or no data pre-processing done. For example when there was no pre-processing done ($avg = 1$), the difference in accuracy among different values of k was approximately in the range of 0.74 to 0.83. In contrast with data pre-processing ($avg = 100$) the accuracy fluctuated approximately only from 0.88 to 0.90. The reason for this is that bigger values of k help to reduce noise, but with data pre-processing by averaging the RSSI values,

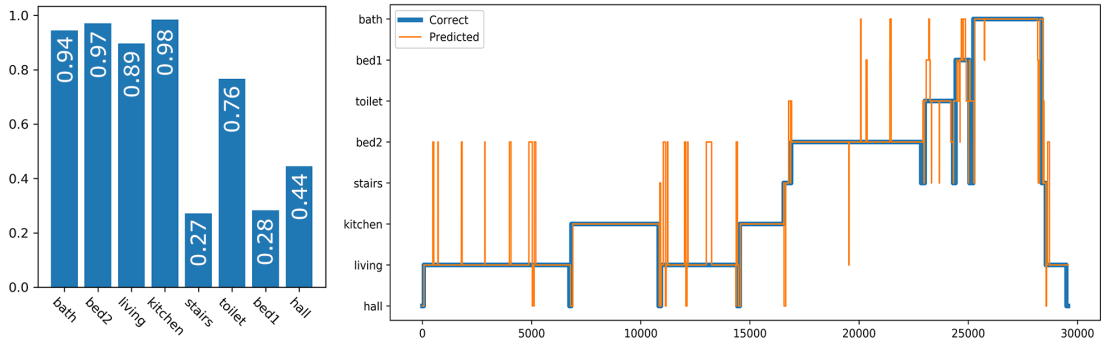

Figure 7. Barchart on the left side visualises k-NN accuracy per room. The thicker blue line on the right image indicates the true path of the person's trajectory. The thinner orange line indicates the predictions.

the noise levels in the RSSI data are already lower. Therefore, the effect of a bigger k is smaller. This also explains the relatively big difference between the best values of k in Table 1.

In Figure 7 the left chart shows the accuracies per room and the right chart visualises the predicted and the real path of a person's movement in the SPHERE house. Despite the relatively good overall accuracy, only half of the rooms had an accuracy over 80% in this example. The reason why the low accuracy in other rooms did not affect the overall results so much, is the relatively small time period the person visits these rooms in the pre-defined script. For clarification, as the exact paths and room-level accuracies varied depending on the chosen training and test data, the visualisations showing the room-level accuracies and the predicted paths (Figures 7, 9 and 10) have been made using the results achieved by the same combination of two full-length recordings for training and testing (recording 1 for training and recording 3 for testing). This makes these visualisations comparable.

4.2 Localisation results and analysis with the HMM

The main tunable parameter for our implementation of the HMM, is the number of sections the sensor values are divided into (described in Subsection 3.4.1). We have named this parameter as $split\_count$. We try the following values for this parameter: $2, 3, 4, ..., 17, 18, 19, 20$. We also analyse how the results with using a hardcoded transition matrix differ from using a non-hardcoded transition matrix that is created by processing the training data. For convenience from now on, the HMM using the hardcoded transition matrix will also be called HMM_{hard} and the HMM using the

non-hardcoded transition matrix $HMM_{non-hard}$. As described previously in Subsection 3.4.1, for the HMM_{hard} we use a constant value of 0.9999 for the probability of staying in the same room.

In addition, we display separately the final evaluation results tested on the reversed test data and non-reversed test data. For example, if the performance on the reversed testing data is similar to the not reversed testing data, it indicates that our model is not overfitted to this particular sequence of observations. Finally, we propose a way to enhance the performance of the HMM model.

4.2.1 Initial results using the HMM

Table 2 represents the HMM results in a similar way as Table 1 for the k-NN did. However, it displays accuracy scores separately for transition matrix being hardcoded and non-hardcoded. Furthermore, for the HMM we analyse the accuracy on reversed and non-reversed testing data separately to have a better understanding of the difference that the hardcoded and non-hardcoded transition matrices make.

Table 2. Best parameters and corresponding accuracies for the HMM in the final testing.

Results using non-hardcoded transition matrix				
Test recording nr	CV recording nrs	Best $split\_count$	Accuracy on reversed test data	Accuracy on non-reversed test data
1	2,3,4,5	10	0.893	0.894
2	1,3,4,5	6	0.914	0.916
3	1,2,4,5	10	0.885	0.885
4	1,2,3,5	6	0.902	0.902
5	1,2,3,4	6	0.910	0.910
			Avg: 0.900	**Avg:** 0.901
Results using hardcoded transition matrix				
Test recording nr	CV recording nrs	Best $split\_count$	Accuracy on reversed test data	Accuracy on non-reversed test data
1	2,3,4,5	7	0.907	0.907
2	1,3,4,5	7	0.928	0.928
3	1,2,4,5	7	0.914	0.914
4	1,2,3,5	6	0.878	0.878
5	1,2,3,4	7	0.914	0.914
			Avg: 0.908	**Avg:** 0.908

The accuracy on the non-reversed data for the $HMM_{non-hard}$ on average was 0.901 and for the HMM_{hard} the average accuracy was 0.908. So the HMM with hardcoded

transition matrix performed slightly better. It would be logical that the HMM with transition matrix calculated using the training data, should have been performed better, but we believe that the main reason is the relatively higher probability to stay in the same room we set when creating the transition probability matrix for the HMM_{hard}. This reduces the number of false transitions between the rooms and therefore increases the accuracy.

The order of the test data did not affect the results at all for the HMM_{hard} as expected but had a small impact on results achieved with the $HMM_{non-hard}$. Even though the difference is minor and in this case and smaller from what we expected, it still indicates that the performance of the HMM with non-hardcoded transition matrix is a little bit dependent on the order of the observation sequence and therefore tends to overfit slightly to the sequence of rooms observed in the training data. We believe that the difference was so small only because the pre-defined script had transitions between rooms always in both ways and therefore reversing the data had not a big impact. The effect of reversing the testing data for both HMM_{hard} and $HMM_{non-hard}$ is also illustrated in Figure 8.

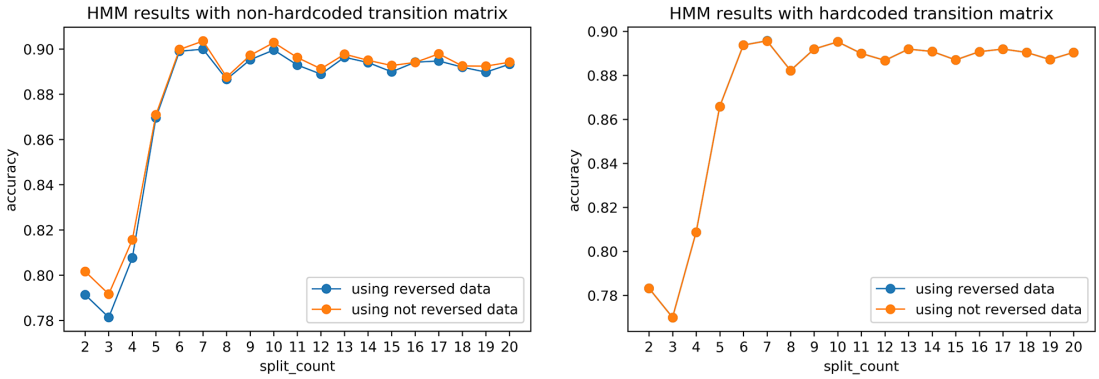

Figure 8. Results from the average accuracies from one complete inner cross-validation performed with the HMM model. On the left image the HMM was created using a non-hardcoded transition matrix and on the right image a hardcoded transition matrix was used. Note that on the right image the results overlap.

The optimal $split\_count$ for the HMM varied but in most cases the best $split_count$s were 6, 7 and 10. A noticeable decrease in performance in all cases was found when using values 5 or smaller for the $split\_count$ parameter.

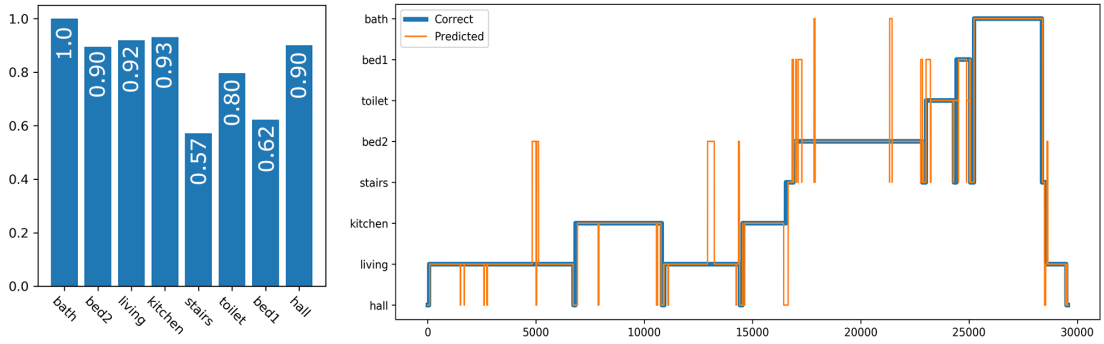

Figure 9. Barchart on the left side visualises HMM accuracy per room. The thicker blue line on the right image indicates the true path of the person's trajectory. The thinner orange line indicates the predictions.

Figure 9 shows the same information for HMM_{hard} as Figure 7 did for the k-NN. Even though the overall average accuracy for the HMM was not bigger by a huge margin, when compared to the k-NN results, there is a relatively big jump in the prediction correctness for the rooms that previously had low accuracy. For example stairs, toilet, bed1 and hall all had noticeably lower accuracies with the k-NN.

4.2.2 Our proposed changes to the HMM and the results

As seen from the right image in Figure 9, there are still plenty of cases where the predicted path does not align with the actual path. There can be numerous reasons for this but we guess that the main reason is that the RSSI values are very sensitive to the exact orientation and activity of the person wearing the sensor as different obstacles around the house can shield the signal. So for example when the person holds their hand with a sensor in a different position relative to their body in the testing data, then for a short period of time the RSSI values can be completely different from what was seen during the training process and therefore be more similar to the RSSI value patterns previously observed from a different room.

One of the solutions we discovered, is to manually modify the transition matrix. It turns out that when we decrease the probability of transitioning from one room to another by many orders of magnitude, the number of false transitions reduces significantly. The reason for this is that it makes the Viterbi algorithm switch to a different room only when the observations have indicated to be in a new room long enough to compensate the increased cost of switching the room.

The following visualisations and results are all achieved with the HMM using the hardcoded transition matrix by dividing the transitioning probability in the initial ver-

sion of HMM_{hard} by a factor of 10^{omega}. For example if the initial probability to stay in the same room is 0.9999 and the probability to transition to another room is 0.0001, then for $omega = 1$ the new probabilities will be 0.99999 and 0.00001 correspondingly. In addition to the $split_count$s tested earlier, we try the following omegas: $2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$.

Table 3 shows the best parameters and final testing accuracies for each of the 5 models. The average accuracy over all the final models was 0.954. The best omegas determined in the inner cross-validations were 70 and 80. The best $split\_count$ was in most cases 10 or 11 but in one case also 6.

Table 3. Best parameters and corresponding accuracies for the HMM with the modified transition matrix in the final testing.

Test recording nr	CV recording nrs	Best $split\_count$	Best omega	Accuracy on test data
1	2,3,4,5	10	70	0.953
2	1,3,4,5	11	70	0.968
3	1,2,4,5	10	70	0.953
4	1,2,3,5	11	80	0.932
5	1,2,3,4	6	80	0.966
				Avg: 0.954

From Figure 10 it is clearly seen that the accuracies per room have increased noticeably. In addition, the predicted path almost exactly matches the real path of the person's movements in the SPHERE house. Six out of eight rooms had an accuracy over 0.97. The only rooms with smaller accuracies were the stairs and the toilet. Both of these rooms were visited very briefly during the pre-defined script.

4.3 Discussion of the results

The average accuracy of the initial version of the HMM was about 90% with the non-hardcoded transition matrix and about 91% with the hardcoded transition matrix. As the average accuracy for the k-NN was also around 90%, there is not a big difference in performance between the k-NN and the initial version of the HMM. Although, it should be noted that for the k-NN there had to be done additional data pre-processing but for the HMM it was not necessary. Moreover, when analysing the accuracies for rooms separately HMM performed better in rooms that had been visited for less time during the pre-defined script. When comparing Figure 7 and Figure 9 then HMM seemed to have slightly less frequent false predictions than k-NN but when analysing the predictions one

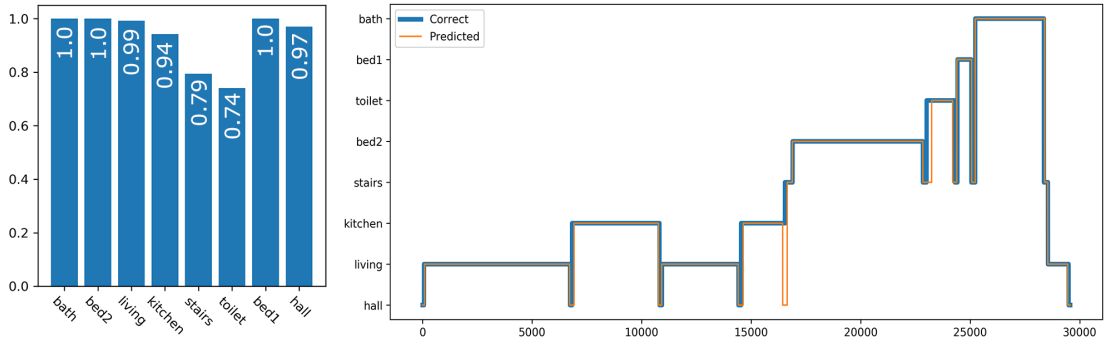

Figure 10. Barchart on the left side visualises the accuracy of the modified HMM per room. The thicker blue line on the right image indicates the true path of the person's trajectory. The thinner orange line indicates the predictions.

by one, it turns out that the HMM spent longer time periods in the wrong room. This is probably due to the specified probabilities of transitions in the HMM, which makes the transition from one room to another less likely. On the other hand, this prevented the HMM from predicting rooms that were not possible for the person to be in when taken into account the room they were in the previous timestamp. With k-NN there were multiple occasions when for example the person is predicted to be upstairs and in the next timepoint straight away downstairs, skipping the stairs, which is logically impossible. This indicates the benefits of context awareness of the HMM. The results also showed that the performance of the hardcoded transition matrix with fixed probabilities was slightly better from the performance of the non-hardcoded transition matrix and was not affected at all by the order of the observation data. Therefore the hardcoded transition matrix should probably be used in real-life scenarios as it tends to be less overfitted to the specific order of data.

A manually modified HMM described in Subsection 4.2.2 was also tested and the performance of this model increased by a big margin. On average the accuracy was a little bit over 95% which is very good when taken into account the noisy nature of the RSSI and the fact that one out of four access points was not working at all. The benefits of this approach should definitely be tested on new data to validate the achieved performance gain. There is a possibility that the model using this approach is relatively more overfitted to this particular data than the initial model.

In the previous research by Lee and Lampe [LL11] that was covered in Subsection 1.5, the accuracy of the k-NN in their case was about 60% compared to the 90% we achieved using this model. However, these results cannot be directly compared because other than the relatively similar environment (9 rooms and 8 APs) there was no data

pre-processing done and in addition they did not specify the value of k used for training the k-NN. The other works conducted on the similar topics were found to be even more different from ours and therefore no reasonable comparison of the results cannot be made.

4.4 Limitations and future work

One of the biggest limitations of performing localisation on the SPHERE dataset was the fact that only three out of four sensors were working at the time of the recordings. Therefore the access points are unevenly distributed with two access points on the ground floor and only one upstairs. This definitely lowered the potential maximum accuracy that was possible to achieve. Not having the possibility to test the trained models on un-scripted data was also one of the biggest difficulties to overcome and therefore the actual real-life performance of the trained models cannot be estimated with full certainty as the evaluation technique (modified nested cross-validation) does not fully eliminate the scripted nature of the data.

This thesis focused on using two main methods to predict the room-level location. Further research with different models and approaches should be conducted to determine the best method for the localisation problem in the SPHERE house. In addition, the models should be tested on un-scripted data to have a more realistic estimation of the real performance of the HMM and the k-NN in a SPHERE-like environment.

5 Conclusion

The main goal for this thesis was to predict a person's room-level location in the SPHERE house using the RSSI data. For this purpose a theoretical background of the RSSI localisation and relevant machine learning methods was first given. Two machine learning approaches were implemented and analysed: the HMM and the k-NN. The HMM was used with the Viterbi algorithm.

We gave a detailed description of the implementation process of both models on the SPHERE dataset. In addition, a comparison and discussion about the two different approaches (hardcoded and non-hardcoded transition matrix) of the parameter calculation for the HMM was given and tested.

Viterbi algorithm with the initial version of the HMM was found to achieve only slightly better results than the k-NN. The average accuracy around 91% for the HMM with the hardcoded transition matrix was only about 1% higher than the accuracy for the k-NN. There was no increase in accuracy almost at all when using the HMM with the non-hardcoded transition matrix. However, a more noticeable difference was found when observing the accuracies for the rooms separately. The HMM had a better performance for rooms that had relatively low accuracies with the k-NN. We also found that reversing the order of sequence of the testing data did not affect the results as much as expected.

We suggested a way to enhance the accuracy of the HMM by modifying its transition probability matrix and decreasing the probability of transitioning to another room by many orders of magnitude. With this modification the HMM achieved accuracies around 95%.

The results of this thesis can be used to help determine the best way for solving the localisation problem in the SPHERE house and in 100 other homes the SPHERE system is currently being deployed to. The findings provide useful information about the potential performance of methods used in this thesis.

References

[Bar05] Paul Barrett. Euclidean Distance - raw, normalized, and double-scaled coefficients. Technical report, Advanced Projects R&D, September 2005.

[Blu04] Phil Blunsom. Hidden Markov Models. *Lecture notes, August 15*, pages 18–19, 2004.

[HC11] Chih-Ning Huang and Chia-Tai Chan. ZigBee-based indoor location system by k-nearest neighbor algorithm with weighted RSSI. *Procedia Computer Science*, 5:58–65, 2011.

[JHB+17] Furqan Jameel, M Asif Ali Haider, Amir Aziz Butt, et al. A technical review of simultaneous wireless information and power transfer (SWIPT). In *Recent Advances in Electrical Engineering (RAEE), 2017 International Symposium on 2017 Oct 24*, pages 1–6. IEEE, 2017.

[KNO+01] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science and Environmental Epidemiology*, 11(3):231–252, 2001.

[KZP07] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised Machine Learning: A Review of Classification Techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[LL11] Kung-Chung Lee and Lutz Lampe. Indoor cell-level localization based on RSSI classification. In *2011 24th Canadian Conference on Electrical and Computer Engineering(CCECE)*, pages 21–26, May 2011.

[LYABW16] Li Li, Wang Yang, Md Zakirul Alam Bhuiyan, and Guojun Wang. Unsupervised learning of indoor localization based on received signal strength. *Wireless Communications and Mobile Computing*, 16(15):2225–2237, 2016.

[MPP09] Antonio Mucherino, Petraq Papajorgji, and Panos M Pardalos. *Data Mining in Agriculture*, volume 34. Springer Science & Business Media, 2009.

[NLLB16] Yepeng Ni, Jianbo Liu, Shan Liu, and Yaxin Bai. An Indoor Pedestrian Positioning Method Using HMM with a Fuzzy Pattern Recognition Algorithm in a WLAN Fingerprint System. *Sensors*, 16(9):1447, 2016.

[RJ86] Lawrence R Rabiner and Biing Hwang Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[Sau17] Martin Sauter. *From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband*. John Wiley & Sons, 3 edition, August 2017.

[STSK16] Ahmed H Salamah, Mohamed Tamazin, Maha A Sharkas, and Mohamed Khedr. An Enhanced WiFi Indoor Localization System Based on Machine Learning. In *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on 2016 Oct 4*, pages 1–8. IEEE, 2016.

[TDK+16a] Niall Twomey, Tom Diethe, Meelis Kull, Hao Song, Massimo Camplani, Sion Hannuna, Xenofon Fafoutis, Ni Zhu, Pete Woznowski, Peter A. Flach, et al. The SPHERE challenge: Activity Recognition with Multimodal Sensor Data. *CoRR*, abs/1603.00797, 2016.

[TDK+16b] Niall Twomey, Tom Diethe, Meelis Kull, Hao Song, Massimo Camplani, Sion Hannuna, Xenofon Fafoutis, Ni Zhu, Pete Woznowski, Peter A. Flach, et al. The SPHERE Challenge: Activity Recognition with Multimodal Sensor Data. *arXiv preprint arXiv:1603.00797*, 2016.

[Xia11] Qingjun Xiao. *Range-free and range-based localization of wireless sensor networks*. Hong Kong Polytechnic University (People's Republic of China), 2011.

[ZF13] Xiuyan Zhu and Yuan Feng. RSSI-based Algorithm for Indoor Localization. *Communications and Network*, 5:37–42, 2013.

[ZM09] Walter Zucchini and Iain L. MacDonald. *Hidden Markov Models for Time Series: An Introduction Using R*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Chapman and Hall/CRC, 1st edition, 2009.

Appendix

I. Source Code

The source code for data pre-processing, implementation of methods and visualisations can be found from the Bitbucket repository:

- https://bitbucket.org/henryteigar/indoor-localisation/src/master/.

II. Floorplan of the SPHERE house

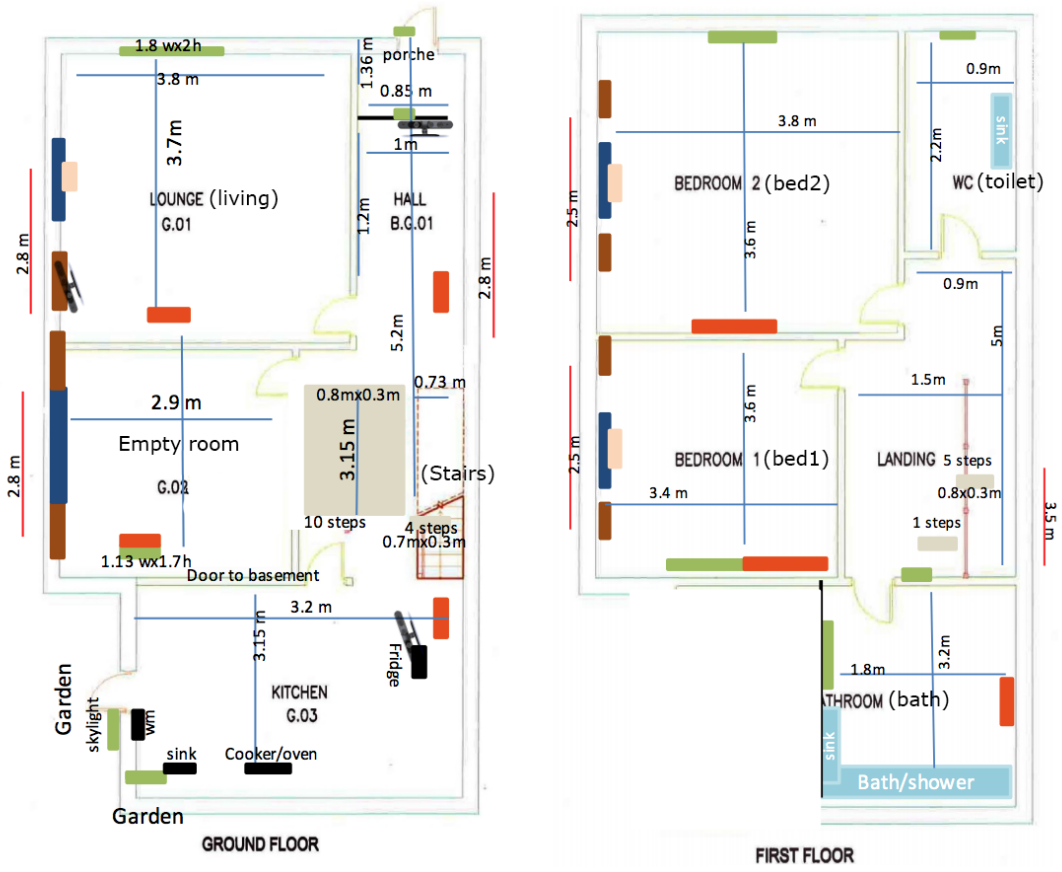

Figure 11. The floorplan of the SPHERE house. Correct room labels have been added to the parentheses for rooms that had a different name from what was used in the SPHERE dataset (bath, bed2, living, kitchen, stairs, toilet, bed1, hall).

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Henry Teigar (date of birth: 19th of February 1995)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Indoor Localisation Using Received Signal Strength

   supervised by Meelis Kull

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2018