Master's Thesis
Evaluation of Machine Learning techniques for Passive Presence Detection

Andreas Sofos
443760

August 30, 2017

Academic Supervisor:
Dr. Michel van de Velden

Company Supervisor:
Anne C. van Rossum

MSc Econometrics and Management Science: Operations Research and Quantitative Logistics
Erasmus School of Economics, Erasmus University Rotterdam

ERASMUS UNIVERSITEIT ROTTERDAM
ERASMUS SCHOOL OF ECONOMICS

CROWNSTONE

The work in this report was performed at Crownstone. Their cooperation is hereby gratefully acknowledged.

Abstract

Passive Presence Detection is the task of detecting when a person is present or not, without the need for them to carry any wireless device. For that, a wireless sensor network consisting of Bluetooth Low Energy (BLE) devices, often called nodes or beacons must be employed in an area of interest in which their positions are not considered known. Beacons transmit and receive BLE signals and they report the Received Signal Strength (RSS) of them, counted in decibels. The RSS data can be explored in search of patterns that can classify presence versus non-presence of people near a wireless sensor network. Machine Learning and Signal Processing techniques are used to tackle the classification problem of Passive Presence Detection. Specifically, Random Forests and two versions of a Sparse Representation-based classification model are employed. Random Forests create an ensemble of classification trees, based on random features. Each of the trees votes for a class. The Sparse representation-based classifier exploits the theory of compressed sensing to find sparse representations of the RSS data, i.e new data matrices which contain only a few non-zero elements. Compressed sensing states that if a data set has a sparse representation in some overcomplete basis, then a high quality reconstruction is possible with much fewer measurements than we would normally need. This method finds an overcomplete basis for each class and classifies the input data based on the most accurate reconstruction. Two data sets were used under various experimental settings to verify the efficacy of the proposed methods. Both Random Forests and the Sparse Representation-based classifier report 100% classification accuracy.

Contents

1 Introduction

Wireless sensor networks have become popular in academic research as well as in practice as they can be easily applied to a wide range of tasks in our lives. They consist of distributed sensors that can monitor the state of an area of interest and act in a cooperative fashion in order to collect and process data, thus creating a smart environment.

Normally, for object detection applications, a person needs to carry a smartphone or smartphone peripheral (a smart watch, bracelet, etc.). However, a Bluetooth radio can also be used in a passive setup. By continuously monitoring the signal drops from other sensors it is possible to infer if there are objects that interfere with the Bluetooth signals.

This study examines the task of Device-free Passive (DfP) Presence Detection, that is the detection of a person without the need for any physical devices i.e. tags or sensors. It is a subject that has become an emerging research field. The requirement of a person holding a device which can assist the localization process can hardly be met in many applications. For example, in an intrusion detection system, one cannot expect a thief to carry any device. Also, in applications which aim to use technology to improve a person's daily living, the detection process cannot rely on a person's willingness or ability to keep any type of hardware close to them all the time.

The variations of the Received Signal Strength (RSS) data, gathered from a DfP detection indoor environment are examined as part of this study. Machine learning methods are exploited with the purpose of detecting human presence based on different data patterns.

2 Problem Description

One goal of home and office automation is that devices respond to someone's presence. For instance, lights can be turned on when a person enters a room. To perform such functions, it is necessary to know when people are inside the building.

In DfP detection and localization systems, Bluetooth Low Energy devices can be employed. These devices, often called radio frequency sensors, nodes or beacons, use wireless personal area network technology for transmitting data over short distances. By capturing the changes of the RSS indicators reported by these devices, we try to derive the presence of a subject based upon these changes. RSS is a measure of quality of a received signal to a Bluetooth Low Energy device, such as a Crownstone chip, counted in decibels. The fundamental idea, is that objects cause disturbances in the radio environment which can be exploited to detect their presence. The human body, for instance, changes the pattern of radio frequency signals when present. Due to the fact that the human body contains about 70% water, it acts as an absorber by attenuating the signal. Signal strength is also influenced by obstacles such as walls and doors, which may provide additional information.

This research aims to use machine learning techniques to detect presence in an environment where Crownstone chips are installed. Detection refers to changes in an area of interest, so the purpose of this study is to analyze the spectrum and see how changes in RSS can be robustly matched with the presence of people.

This thesis internship is an initiative of Crownstone (2016), a startup in Rotterdam that brings technology to detect people and devices into every building. The Crownstones up to now can estimate a person's position only through their smartphone. The Crownstones (Figures 1-2) measure current and voltage to detect devices and examine their usage. (`http://crownstone.rocks`)

Figure 1: Crownstone Plug

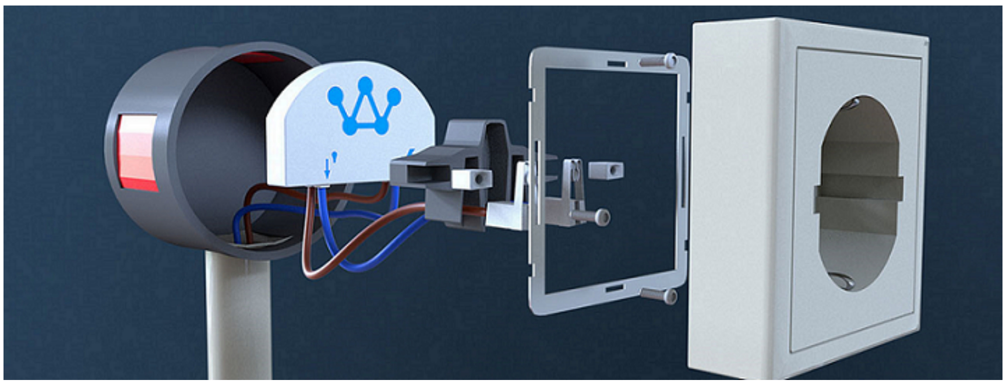


Figure 2: Built-in Crownstone

The problem as described above leads to the following research question:

Which Machine Learning and Signal Processing methods perform best under various experimental settings for the task of device-free presence detection?

In particular, this research examines how methods are affected by the number of sensors in a wireless sensor network and how quickly algorithms accurately predict the state of whether there is presence of a human or not in an area of interest.

3 Literature Review

Presence detection and localization techniques based on radio signals have attracted the attention of researchers for years and many methods have been proposed. However, most of them require the target to carry a tracking device so that it actively participates in the localization process. On the other hand, device free localization is an emerging research field which was first introduced by Youssef, Mah, & Agrawala (2007). For the purpose of this research, I cannot rely either on any active participation of people in a training phase, such as radio map construction or in predetermined node locations. This research aims to exploit the benefits of modern tools of Machine Learning and Signal Processing theory, working on RSS data sets. At least to our knowledge, this approach is quite novel in the field of Device-free Presence Detection. Literature on Image Classification is more related to the methods used in this study.

Youssef et al. (2007) use statistical methods, such as moving average and moving variance in order to detect changes in the RSS by comparing the static state of the environment, i.e the long term signal behavior to the current state, based on threshold values. Their approach involves a training process that consists of a construction of a passive radio map. It requires a training procedure in which a person, without the assistance of any device, remains in the area of interest and the signal strength measurements are recorded.

Another popular application in the field of Device-free Presence Detection is called Radio Tomographic Imaging (Wilson & Patwari, 2008). It is based on the fact that as an object is moving within a wireless network, the attenuation it causes can be imaged as a function of space for detection purposes. This is possible by measuring the RSS on many different paths in the network. The linear model that is suggested by Wilson and Patwari (2008) is based on the known locations of the nodes in a wireless network. In the context of this research, information about the locations of the nodes is not provided.

A Robust Face Recognition method, based on sparse representations of images is presented by Wright, Yang, Ganesh, Sastry, & Ma (2009). Compressed sensing theory (Donoho, 2006) is applied for computing sparse representations of data based on an overcomplete dictionary of base vectors. This dictionary is comprised of training samples from all classes. The intuition behind this is that finding the sparsest representation of an image for a given dictionary can reveal the class the image comes from. The test data can be expressed as a linear combination of the training samples from the same class. One can classify the input test data by calculating these linear approximations, each time using only training samples from a single class. The best approximation in terms of the minimum residual between the initial and the reconstructed input samples reveals the right class. The algorithm correctly identified the subjects even when the images were corrupted (beyond 50%), a case in which the human eye can barely recognize images as face images.

Wang, Guo and Zou (2016) attempted to use this methodology for face recognition to formulate the Device-free Localization Problem as a Sparse Representation-based Classification problem. Again, there are important differences compared to this study as mentioned before, i.e the position of the nodes are known and the training sets are constructed by a fingerprinting method of collecting RSS values by moving a target in every possible position. However, by reviewing the RSS matrix as an image matrix they managed to turn the DfP concept into an image classification problem and solve it even when the image is influenced by noise and outliers (as in corruption and disguise in image classification).

In this research, I utilize the theory of sparse representations by experimenting on different dictionaries and applying sparse representation based classification methods to two datasets. At the same time, regular machine learning classification techniques are tested and their performance is evaluated.

4 Data

This study uses two different data sets, an online data set called Radio Tomographic Imaging (RTI) Data Set and a data set (Crownstone Data Set) collected for the purpose of this research at Crownstone offices. The RTI data set contains RSS values collected by 28 nodes. This gives us the opportunity to obtain data sets of different dimensions, by keeping RSS values from many different subsets of nodes and perform a series of experiments that can give information on how many nodes I should employ in the context of creating our own data set. The Crownstone data set contains RSS values collected by five nodes, which is a more realistic number of nodes for a smart home environment. A lower dimension data set is produced from the Crownstone data set, for the purpose of this study.

In the following subsections I present the set ups of the data collection processes, the format of the data and a description of the files of the data sets.

4.1 Experiment Description

4.1.1 RTI Data Set

The RTI Data Set (Wilson & Patwari, 2008) is used for the purpose of this study. This data is collected through 28 Crossbow TelosB nodes, placed in an outdoor environment, in a square perimeter of 21×21 feet. Each node is placed at three feet distance from its neighboring node. Inside this area, there are two trees. Figure 3 shows the map and a photo of this area. The transmission protocol is the IEEE 802.15.4, in the 2.4GHz frequency band. Only one node transmits at a time, and each one of them has a unique ID attached, from 0 to 27.

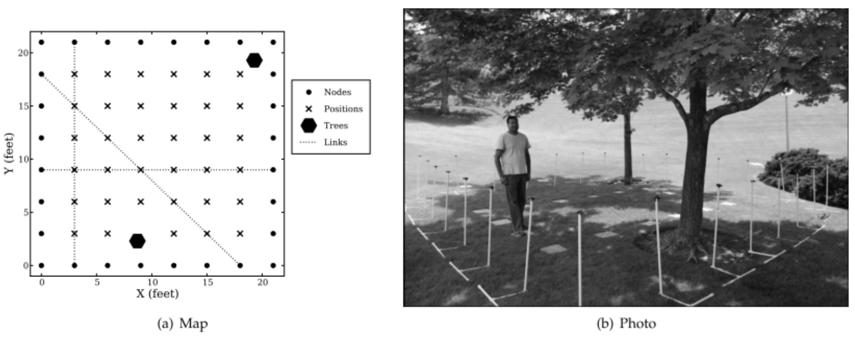


Figure 3: RTI map and photo (Wilson & Patwari, 2008)

4.1.2 Crownstone Data Set

This data set is collected at Crownstone offices, in an area which consists of two rooms forming an L shape and separated by a bookcase in between. Only two tables were inside this area, as depicted by the rectangle shapes. The floorplan is shown in Figure 4. In the whole area, five crownstones were deployed at random distances between them, determined by an estimation of the possible distances of the sockets in a house, in order to realistically simulate a real smart home environment.

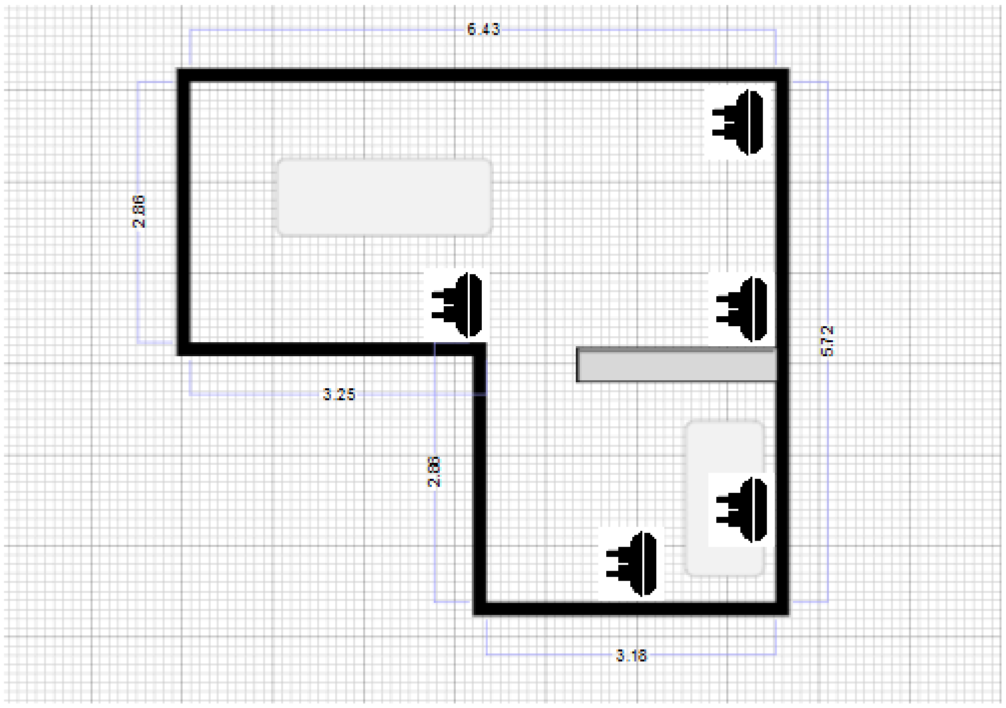


Figure 4: Crownstone floorplan

4.2 Data Format

4.2.1 RTI Data Set

Each line in a data file represents the RSS values from each of the nodes to the reporting one, the ID of which is indicated in the first column of every line. The last three values are the time the measurement was received, in the format of HR MIN SEC. In total each file contains 32 columns. Since a node cannot transmit to itself, this cell is filled with the value -45, as this column is meaningless and is given a value close to the mean (RSS values vary approximately between 0 and 100). An example line looks like this:

3, -44, -52, -55, -45, -76, -93 ... 10 2 46

The data collection rate varies between 70-80 measurements per second.

4.2.2 Crownstone Data Set

Similarly to the format of the Radio Tomographic Imaging Data Set, each line represents the RSS values from each of the nodes to the reporting one, the ID of which is indicated in the first column of every line. Again, the value of -45 was given to the cells that represent a pair of the same node ID.

4.3 File Descriptions

4.3.1 RTI Data Set

empty.csv: The area is completely vacant. Each node has reported around 137 RSS vectors and in total the file contains 3785 measurements spanned over a 53 seconds period.

m-1.csv: One person walks at constant velocity in two loops around a square loop path. Each node has reported around 70 RSS vectors and in total the file contains 1982 measurements spanned over a 28 seconds period.

4.3.2 Crownstone Data Set

baseline.csv: The area is completely vacant. Each node has reported around 1000 RSS vectors and the file in total contains 5759 measurements spanned over a period of approximately 20 minutes.

presence.csv: One person is walking around the area, spending half of the time of this experiment in each of the two rooms. Each node has reported around 500 RSS vectors and the file in total contains 2729 measurements spanned over a period of approximately 10 minutes.

5 Methodology

The task of Passive Presence Detection is a classification problem. A method should be able to process input data, which in this case is RSS data, and deliver a prediction on whether the data reveal human presence or not. For this reason, two main classification methods are employed in this research. The first one is a machine learning method, called Random Forests that works on the raw data sets, as they are described in Section 4. The second method, called Sparse Representation-based classification, process the data in a way that it produces higher-level representations of it and then performs the classification task. This method is based on Signal Processing theory which is discussed in the current section. Two versions of this method are presented, as well as algorithms that are involved. The Sparse Representation-based classification is compared with the k-Nearest Neighbor algorithm, in terms of finding similarities and differences that can explain the choice of the second method for this study.

As a general data preprocessing step, the first column containing the ID of the reporting node as well as the last three columns reporting the time were removed.

A Matlab script was created that reduces the size of the data set by keeping information only from a subset of nodes. In more details, the script keeps only the rows and columns of the file that correspond to the reporting and transmitting nodes of a by the user chosen subset respectively. This gives us the opportunity to experiment with different number of deployed nodes, and to compare the accuracy of the results. The fewer number of nodes used, the more realistic and interesting a presence detection application becomes.

5.1 Classification on the Raw Data Sets

5.1.1 Data Processing

The 'empty.csv' and 'm-1.csv' data sets include samples from the 'non-presence' class and the 'presence' class respectively.
After having applied the preprocessing step to both files, thus removing the first column including the number of the reporting node and the last three columns including the time (as described in subsection 4.2), the files were merged and every single line measurement was given a label, either '0', corresponding to 'non-presence', if the line comes from the 'empty.csv' file or '1', corresponding to 'presence', if the line comes from the 'm-1.csv' file. The exact similar data processing procedure was performed to the second data set.

Each instance of the data has as many attributes as the number of nodes used in a specific experiment.

5.1.2 Random Forests Classification

For the classification task, the mechanism of Random Forests was used, that is "a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest" (Breiman, 2001, p. 1). In the following description of this method, I first introduce the decision tree classification method (Tan, Steinbach, & Kumar, 2014).

A decision tree consists of nodes and leaves. The first node is called root and the final nodes are called leaves. Each node is split into either two partitions (binary split) or more than two non-overlapping partitions (multi-way split). In order to explain how a split is determined and which attribute should be used for the split, an impurity measure term is introduced. A tree should go from high impurity to low impurity. Minimum impurity means that each leaf corresponds to exactly one class. As we move from the root to the leaves impurity levels should decrease. Minimum impurity means that each leaf corresponds to exactly one class. For continuous data, as in the case of the current research, the numbers must be discretized in some form, usually by finding a threshold which will determine the split. Different measures of impurity can be employed in order to determine the best split, such as the Gini Index, the Classification Error and the Entropy. If I let $p(i|t)$ denote the proportion of cases belonging to class i at tree node t and let c denote the number of classes, then:

$$Gini(t) = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{1}$$

$$ClassificationError(t) = 1 - max_i p(i|t) \tag{2}$$

$$Entropy(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t) \tag{3}$$

Finally, a stopping rule must be set regarding the growth of a tree. A node cannot be further expanded if all cases belong to the same class or have similar attribute values or the number of cases is below some threshold value.

After the construction of the tree classifier its performance must be evaluated. The training data is a part of the data set used to train the classifier. After this process, another part of the data set, called test set which is never seen before by the classifier, is used to assess the model's predictive performance. The error on training data and the generalization error (on test data) could be considered in this case. If we keep adding nodes to a tree until all cases belong to the same class, the training error will be minimized, however we come across the problem of data overfitting. This means that the classifier is not optimal for the test set. On the other hand, a very simple classification tree might lead to misclassification of many instances. Consequently, there is a trade off between size and stability. In order to deal with this issue, a procedure called cost-complexity pruning is utilized. The tree is grown to its full extend, which means that splitting no leaf node further will improve the accuracy of the training data. As a second step, subtrees that are not contributing significantly towards classification accuracy are removed. The term cost-complexity is defined as the error rate divided by the number of nodes in the tree and one should decide at which value of this term to stop. If the error rate is denoted by ER and NoN is the number of nodes in a tree then the cost-complexity measure, denoted by CC, is simply given by:

$$CC = \frac{ER}{NoN} \tag{4}$$

For that, the out-of-sample predictions of pruned trees need to be determined. The accuracy measure (5) is used in this research to evaluate the performance of a classification tree, as well as that of a Random Forest. It is defined as the fraction of the number of correct predictions (true positive (TP) and true negative (TN)) over the total number of predictions (NoP).

$$Accuracy = \frac{TP + TN}{NoP} \tag{5}$$

Following this short introduction on classification trees, I proceed by with the Random Forests algorithm. Based on a training set T, new bootstrap training sets T_k are created by sampling with replacement. The algorithm creates as many bootstrap training sets as the number of trees we want to grow to form a Random Forest. Each training set is then used to grow a tree-classifier. If the data consists of N input variables, then a subset M<N of these input variables is randomly selected at each node to determine the best split. The parameter M is held constant during the whole forest growing procedure.

As far as the accuracy of Random Forests is concerned, Breiman (2001) introduced the definition of the generalization error of a forest in terms of the margin function

$$mg(X,Y) = av_k I(h_k(X) = Y) - max_{j \neq Y} av_k I(h_k(X) = j) \tag{6}$$

where the training set is randomly created from the distribution of the random vector Y,X and $h_1(x), h_2(x), ..., h_k(x)$ form the ensemble of classifiers in the forest. The margin function measures the difference between the average number of votes for the right class and the average vote for any other class. The generalization error

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \tag{7}$$

is then defined as the probability that the margin function is negative. From the Strong Law of Large Numbers, as the number of trees increases, the generalization error converges, so Random Forests do not overfit.

The generalization error depends on two factors. The strength of the individual classifiers and the correlation between any two classifiers in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the trees decreases the error rate of the forest. On the other hand, increasing the correlation between two classifiers increases the forest's error rate.

The Random Forest mechanism can internally calculate an unbiased estimation of the generalization error. For each bootstrap training sample that is drawn from the initial training set with replacement, 30% of the observations are left out of the sample and are not used in the construction of this specific classifier. These are used as test sets and are fed to the classifiers that were constructed from a training set that does not contain these observations. This is called out-of-bag classifier and the error rate of this classifier is the out-of-bag estimation of the generalization error. It should be noted that the out-of-bag error is a statistical measure rather than an actual classifier.

The data set is split into a training set containing 70% of the observations and a test set of the remaining 30%, randomly chosen without replacement that is used to assess the performance of the models. In another experiment, only a few random measurements, namely 210 samples are used as a test set and the rest form the training set. A Random Forest is trained, each time using a different value of the M parameter, among all possible ones. The number of trees are fixed at 500 in all experiments. Finally, the test set is used for predicting the right class and the accuracy of this process, which is defined as the percentage of the successful predictions over the number of samples of the test set, is calculated.

The code implementing this method is written in R programming language for statistical computing with the use of the 'randomForest' package (Liaw & Wiener, 2002).

5.2 Sparse Representation-based Classification

The main idea that led us to explore the methodology of Sparse Representation-based Classification, a popular technique in the field of Image Processing, is that a matrix with RSS values can be seen as an image in which each single RSS value corresponds to a pixel value. For example, a state in which no people are present in an area of interest and a state in which one or more people are present can be considered as different images, in terms of the RSS values. In the next section, I aim to provide an insight on why this method may be useful for this research by presenting the basic theory of compressed sensing.

5.2.1 Data Processing

All data matrices are rotated, as this was an input data requirement of the main algorithms employed in this section, so from now on I refer to a data set with n1 measurements and n2 attributes (number of nodes) as $n2 \times n1$.

5.2.2 Compressed sensing

In signal processing theory, sampling, i.e the transformation of a continuous time signal to a discrete time signal is of vital importance in many applications. The Nyquist-Shannon sampling theorem (Shannon, 1949) states that a band-limited function, i.e a function whose Fourier transform is zero above a certain finite frequency, can be exactly restored from equally spaced samples if the sampling frequency is at least twice the highest frequency contained in the signal. The compressed sensing theory (Donoho, 2006) which has gained significant recognition in the last years, uses sparse representations (Elad, 2010) to sample a signal at a much lower than the Nyquist rate. The term sparse implies that most of the elements of the data are zero and only a small number of elements have a non-zero value. The key idea of compressed sensing is that if a signal has a sparse representation in some overcomplete basis, then a high quality reconstruction is possible with much fewer measurements than we would need according to Nyquist. To explain overcompleteness, let $V = \{v_i\}_{i \in J}$ be a set of vectors on a Banach space B. A subset of vectors in V is complete if every element in B can be approximated in norm by finite linear combinations of elements in V. If we remove a v_j from the subset V and the remaining vectors still form a complete set, then V is called overcomplete. A Banach space is a vector space in which a norm is defined.

For the purpose of this research, a dataset can be considered as a signal so both terms are equivalent. Assume that a signal with n elements that contains only m non-zero components, where m is much smaller than n, will be called m-sparse. A straightforward approach to find these m elements is to iterate over all n elements of the signal. The objective of compressed sensing is to capture this information in the m-sparse signal of length n with many fewer than n measurements. Sparsity is the key in the success of compressed sensing, otherwise the regular sampling theorem must be applied. If the signal is assumed to be m-sparse, then on average only O(mlog(n)) measurements are required to reconstruct it.

In order to proceed in more depth in the theory of compressed sensing the following notations are used. Consider a matrix $Y \in \mathbb{R}^{n\times N}$, a matrix $D \in \mathbb{R}^{n\times K}$ and a T_0-sparse matrix $X \in \mathbb{R}^{K\times N}$. Y represents an RSS matrix in which n is the number of nodes used in the experiment to collect the RSS values and N is the number of observations. X is the sparse representation of Y with respect to an overcomplete basis D, usually called dictionary in this field. The problem is to find X when Y and D are given and there is an underdetermined system of equations, that is when n<K. Based on compressed sensing theory, this is possible under certain conditions of the matrix D and the sparsity level of the data. The higher the sparsity level the higher the probability of a successful recovery of X.

If we know beforehand that X is sparse we can reconstruct it by solving the following optimization problem (Kanso, 2009, p. 17):

$$\begin{gathered}\hat{X} = argmin_X \|X\|_0 \\ subject\, to\, Y = DX\end{gathered} \tag{8}$$

where $\|X\|_0$ is the l^0-norm of X, defined as the number of non-zero elements in X. This problem is however a non-convex NP-hard problem. In order for this problem to become computationally tractable the l^0-norm of X is replaced by the l^1-norm of x. The l^1-norm is defined as the summation of the absolute values of all components of a vector or a matrix. In mathematical notation $\|p\|_1 = \sum_{j=1}^{n} |p_j|$ for a vector p $\in \mathbb{R}^n$. The intuition behind the hypothesis that the l^1-minimization problem induces sparsity is explained in the following example.

Suppose we are looking for a line that matches two points in a two-dimensional space but only one point is known. Then there will be an infinite number of solutions. If for instance, the point is [10,5] and a line is defined as y=ax+b, then we want to find a solution to the equation:

$$(10\ 1) \times \begin{pmatrix} a \\ b \end{pmatrix} = 5$$

Obviously, all points belonging to the line b=5-10a are solutions.

The points for which the l^1-norm equals a constant c look like this rotated red square.

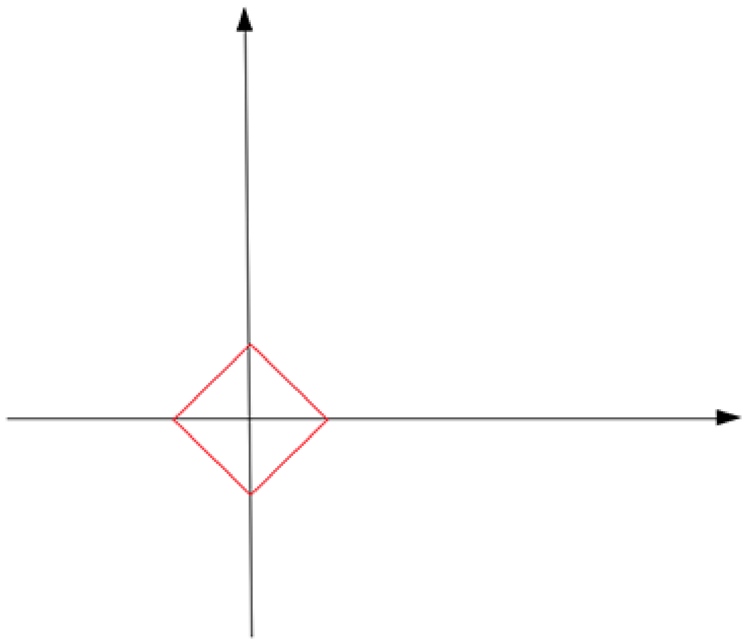


Figure 5: l^1-norm shape

The only sparse points on this shape are the ones on the tips, where either the x or the y coordinate is zero. The process of finding a sparse solution could geometrically be seen as making the c constant large enough so that the red shape touches the blue line (b=5-10a), as shown below.

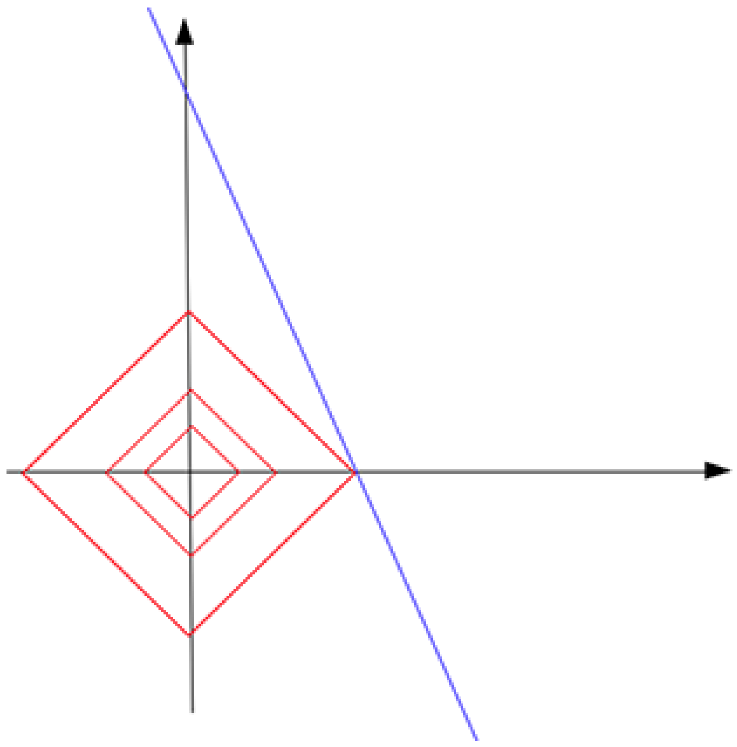


Figure 6: Sparse Solution

Because of the nature of the l^1 shape, there is a very high probability that the touch point will be at a tip of the shape and since all tips are sparse points, the solution will also be sparse. In this example, the touch point is the sparse vector (0.5 0), which is the solution with the smallest l^1-norm out of all the possible solutions in the blue line.

In high dimensional space the probability of touching a tip could be even higher, since the l^1 shape will have more tip points and this explains the reason why the l^1-minimization problem might be capable of inducing sparsity.

As a result, the initial l^0-minimization problem is replaced by

$$\begin{aligned} \hat{X} &= argmin_X \|X\|_1 \\ subject\, to\, Y &= DX \end{aligned} \tag{9}$$

which is a convex optimization problem. In case the data contain inaccurate measurements or some form of noise in general, as it usually happens in real cases then the optimization problem becomes

$$\hat{X} = argmin_X \|X\|_1 \quad subject\,to\,\|Y - DX\|_2 \leq e \tag{10}$$

where e depends on the amount of noise. Eldar and Kutyniok (2012, p. 30) present some common noise models under which sparse signal recovery can be stably performed. The l^1-minimization problem can be solved by many efficient algorithms, such as the so called pursuit algorithms, one of which will be discussed in the following subsection.

As it may be obvious already, one factor that affects the success of finding a sparse representation of a signal is the choice of the dictionary D. It is the nature of this basis that makes X a sparse matrix and so finding the sparsifying domain is crucial in compressed sensing applications. For example, a sinusoidal wave is not sparse in the time domain, but in the frequency domain it is just an impulse, as shown in Figure 7.

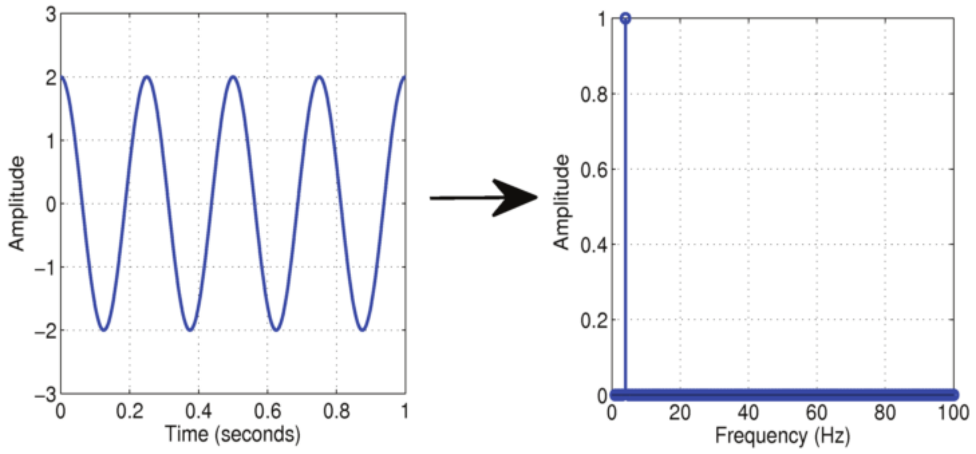


Figure 7: Sparsifying domain of sinusoidal wave (Kanso, 2009)

The right graph of Figure 7 is the result of Fourier Transform of the sinusoidal wave, shown in the left graph. The formula of Fourier Transform is

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{11}$$

which transforms a signal from time to frequency domain, where ω is the angular frequency. For the sinusoidal wave the result of this transformation is shown below:

$$x(t) = sin(2\pi st) -> X(\omega) = \frac{1}{2}[\delta(\omega + s) + \delta(\omega - s)] \tag{12}$$

where $\delta(x)$ is the Dirac delta function defined by the properties:

$$\delta(x) = \begin{cases} 0 & x \neq 0 \\ \infty & x = 0 \end{cases} \tag{13}$$

and

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{14}$$

Figure 7 visualizes the result of the above analysis for a certain time and frequency window. Since the two representations of the sinusoidal wave are equivalent, one can say that when the signal is represented in the frequency domain, which can be considered as the dictionary, then the signal clearly becomes sparse.

Similarly, in the current research, a dictionary must be found with respect to which an RSS matrix can provide a sparse representation. If there is no prior knowledge of this dictionary, one has to find the optimal dictionary, in terms of the sparsest representation that this dictionary can yield. One such dictionary learning algorithm is the K-SVD algorithm, presented in Section 5.2.5.

5.2.3 The Orthogonal Matching Pursuit (OMP) algorithm

The Orthogonal Matching Pursuit algorithm deals with underdetermined equations in a greedy way in order to recover sparse representations of input data. Elad (2010, p. 37) gives a detailed description of the steps of the algorithm which is presented below.

In order to explain this greedy approach I assume that (1) has an optimal solution value equal to 1, meaning that X has only one non-zero element in the optimal solution. Thus, Y is a multiple of some column in D. This column can be found by applying K tests, one per column of D. The notation $\|A\|_F$ is the Frobenius norm defined as $\|A\|_F=\sqrt{\sum_{ij} {A_{ij}}^2}$. The j-th test can be performed by minimizing $e(j) = \|d_j z_j - Y\|_F$, leading to $z_j = \frac{{d_j}^T Y}{\|d_j\|_F^2}$. The Sweep step of the algorithm calculates the error values in a similar fashion. If I plug z_j into the error expression I have:

$$e(j) = min_{z_j} \|d_j z_j - Y\|_F^2 = \| \frac{d_j^T Y}{\|d_j\|_F^2} d_j - Y\|_2^2 = \|Y\|_F^2 - 2\frac{(d_j^T Y)^2}{\|d_j\|_F^2} + \frac{(d_j^T Y)^2}{\|d_j\|_F^2} = \|Y\|_F^2 - \frac{(d_j^T Y)^2}{\|d_j\|_F^2} \tag{15}$$

In case the error is zero then a solution has been found. The above test is equivalent to $\|d_j\|_F^2 \|Y\|_F^2 = (d_j^T Y)^2$. However, if (1) is known to have an optimal solution value of T_0, then there are K^{T_0} subsets of T_0 columns of D, and it might be computationally expensive to enumerate all these subsets (O(K^{T_0})). As a result, the greedy approach of OMP searches for local optima in order to approximate Y from a set of columns of D. If the error is below a predefined threshold, the algorithm stops.

The so-called Update Provisional Solution step minimizes the term $\|DX - Y\|_F^2$ with respect to X, such that its support is S^k. The algorithm forces the columns in D that are part of the support S^k to be orthogonal to the residual r^k and this is why the algorithm is called Orthogonal Matching Pursuit.

Algorithm 1 Orthogonal Matching Pursuit Algorithm

```
1: Initialization: Set k=0, initial solution $X^0 = 0$, initial residual $r^0 = Y - DX^0 = Y$, initial solution support $S^0 = Support\{X^0\} = \emptyset$
2: while $\|r^k\|_F \geq e_0$ do
3:     k=k+1.
4:     Sweep: Compute the errors $e(j) = min_{zj}\|d_j z_j - r^{k-1}\|_F^2$ for all j using the optimal choice $z_j = \frac{{d_j}^T r^{k-1}}{\|d_j\|_F^2}$
5:     Update Support: Find a minimizer, $j_0$ of e(j): $\forall j \notin S^{k-1}, e(j_0) \leq e(j)$, and update $S^k = S^k \cup \{j_0\}$.
6:     Update Provisional Solution: Compute $X^k$, the minimizer of $\|DX - Y\|_F^2$ s.t. $Support\{X\} = S^k$.
7:     Update Residual: Compute $r^k = Y - DX^k$.
```

5.2.4 The K-means algorithm

A short description of the K-means algorithm will be given in this section. It is important to note that K-means is not used in the current research, however it is mentioned in order to help the reader better understand the functionality of the K-SVD algorithm (5.2.5), which is a generalization of K-means and is used in this paper.

The procedure of K-means assigns a set of data points to K clusters, where the K parameter is specified by the user. Each cluster is represented by a group average (centroid). Let Y be the $N \times n$ observation matrix, D the $n \times K$ centroid matrix and X a $N \times K$ indicator matrix assigning observations to the K clusters. As an initialization step, K initial centroids should be specified to form $D^{(0)}$. Then, the algorithm associates each point of Y to the closest centroid by solving the following minimization problem, where D^T and X^T are the transpose of D and X respectively.

$$\begin{gathered} min_X \|Y - XD^T\|_F^2 \\ subject\ to\ D = (X^T X)^{-1} X^T Y \end{gathered} \tag{16}$$

As soon as all observations have been assigned to a centroid, the first step of K-means has been completed for a certain iteration. The second step involves the update of the centroid matrix D, in which the new cluster centers are calculated based on the points that already belong to each cluster. The algorithm iterates between these two steps until the centroid matrix D no longer changes.

Algorithm 2 K-means algorithm

```
1: Initialization: Set the initial centroid matrix $D^{(0)} \in \mathbb{R}^{n \times K}$. Set J=1.
2: while Centroid matrix changes do
3:     Assign each object to the group that has the closest centroid.
4:     When all objects have been assigned, recalculate the centroids.
5:     J=J+1.
```

5.2.5 The K-SVD algorithm (with OMP)

The most popular and widely used algorithm for dictionary learning is the K-SVD algorithm (Aharon, Elad, & Bruckstein, 2006), a generalization of the K-means clustering process. Instead of using a predetermined dictionary, this unsupervised method tries to fit a dictionary to the data, so that it best represents each member of the training signals set. At each iteration it alters between sparse coding of the data and an update of the vectors or, as they are usually called in this setting, atoms, of this dictionary as a learning process to better fit the data. The goal is to find a dictionary that yields sparse representations for a training signal. This process has been proved to create dictionaries that outperform predetermined dictionaries (Aharon et al., 2006).

The K-means clustering method, as seen in the previous section, applies two basic steps in each iteration of the algorithm. First, given several vectors $\{d_k\}$, k=1,...,N, it clusters the training instances to the nearest neighbor and after that it updates these vectors to better fit future instances. K-SVD follows a similar two-step procedure by finding the coefficient matrix X given a dictionary D and then updating the dictionary in a second stage. While in K-means each sample is represented by only one of those vectors, based on their Euclidean distance, in K-SVD each example can be expressed as a linear combination of several vectors of the dictionary. This explains why it is considered a generalization of K-means. In the extreme case that the algorithm is forced to use one dictionary column per sample and a unit coefficient for this column, it exactly behaves like the K-means algorithm. However, because of the fact that not only one column of the dictionary can represent a sample, it should be noted that the algorithm does not actually perform clustering.

The K-SVD algorithm is flexible and can operate together with a pursuit algorithm, which in our case is the Orthogonal Matching Pursuit. It aims to find the best possible dictionary by solving the following minimization problem:

$$\begin{gathered} min_{D,X} \|Y - DX\|_F^2 \\ subject\,to\ \|x_i\|_0 \leq T_0,\ \forall i \end{gathered} \tag{17}$$

As defined in section 5.2.2, $Y \in \mathbb{R}^{n \times N}$ is a matrix of N observations with n attributes each, $D \in \mathbb{R}^{n \times K}$ is the dictionary and $X \in \mathbb{R}^{K \times N}$ is a T_0-sparse vector which contains the representation coefficients of Y.

K-SVD restricts the value of K to be less than the number of measurements of Y, so it must hold that $K < N$, otherwise the solution is considered trivial. The algorithm applies the two steps described above. It employs a pursuit algorithm to tackle

$$\begin{gathered} min_{x_i} \|y_i - Dx_i\|_F^2 \\ subject\ to\ \|x_i\|_0 \leq T_0 \end{gathered} \tag{18}$$

and then updates one column of D, both at each iteration. The updating process involves fixing all columns apart from one, d_k and finding a new column to replace d_k as well as new values for its coefficients that best reduce the overall mean squared error $E = \|Y - DX\|_F^2$. In order to describe this in more details, the objective function can be rewritten as:

$$\|Y - DX\|_F^2 = \|Y - \sum_{j=1}^{K} d_j {x_T}^j\|_F^2 = \|(Y - \sum_{j \neq k} d_j {x_T}^j) - d_k {x_T}^k\|_F^2 = \|E_k - d_k {x_T}^k\|_F^2 \tag{19}$$

where ${x_T}^k$ is the kth row in X, or else the coefficients corresponding to the d_k column of the dictionary D. Consequently, the matrix E_k expresses the total error when the kth vector is removed. If I restrict this matrix to represent only data points that actually use d_k, defined as $\omega_k = \{i | 1 \leq i \leq N,\ x_T^k(i) \neq 0\}$ and name this new matrix E_k^R, I receive the equivalent objective function:

$$\|E_k^R - d_k {x_R}^k\|_F^2 \tag{20}$$

As a final step of the algorithm, the Singular Value Decomposition (SVD) can now be applied to E_k^R, decomposing it to E_k^R=UΔV^T. The new column that updates the dictionary, denoted by d'_k will be the first column of U and the new coefficient vector x_R^k will be the the first column of V multiplied by Δ(1,1).

The algorithm repeats the whole process, as previously described, until a stopping rule is met. In this paper, K-SVD is restricted to perform a certain number of iterations which is given as a parameter to the algorithm. I present an overview of the algorithm from (Aharon et al., 2006, p. 4317) below.

Algorithm 3 K-SVD Algorithm

```
1: Initialization: Set the dictionary matrix $D^{(0)} \in \mathbb{R}^{n \times K}$ with Frobenius normalized columns. Set J=1.
2: while Stopping rule not met do
3:     Sparse Coding Stage: Use OMP to compute the representation vectors $x_i$ for each example $y_i$, by
       approximating the solution of $min_{x_i} \|y_i - Dx_i\|_F^2$ subject to $\|x_i\|_0 \leq T_0$.
4:     Dictionary Update Stage: For each column k=1,2,...,K update it by:
       -Define the group of examples that use this atom, $\omega_k = \{i | 1 \leq i \leq N, {x^k}_T(i) \neq 0\}$.
       -Compute the overall representation error matrix $E_k$, where $E_k = Y - \sum_{j \neq k} d_j {x_T}^j$.
       -Restrict $E_k$ by choosing only the columns corresponding to $\omega_k$ and obtain ${E_k}^R$.
       -Apply SVD decomposition ${E_k}^R$=U$\Delta V^T$. Choose the updated dictionary column $d'_k$ to be the first
       column of U. Update the coefficient vector ${x_R}^k$ to be the first column of V multiplied by $\Delta$(1,1).
5:     Set J=J+1.
```

5.2.6 K-SVD vs PCA

Principal Components Analysis (PCA) is a widely used unsupervised method of finding low-dimensional representations of data, a process that is called dimensionality reduction. Suppose X is the original $n \times N$ data set, where n is the number of dimensions and N is the number of observations. The objective is to find another $n \times N$ matrix Y, linearly related to X with respect to a basis P, in order to represent the data set. The rows of P, $\{p_1, p_2, ..., p_n\}$ are called principal components and PCA assumes that directions with the largest variance in the data are the most important in order to express the information that is hidden in the data. Thus, PCA searches the directions along which variance is maximized and saves these ordered vectors in P. The process is repeated until n vectors are found. Principal components are restricted to be orthogonal. In mathematical notation the problem translates to:

$$\begin{aligned} & min_{Y,P} \|Y - PX\|_F^2 \\ & subject\ to\ P^T P = I \end{aligned} \tag{21}$$

A main reason why PCA is extensively applicable is that the orthogonality constraint allows linear algebraic decompositions to easily solve it. However, there are two important limitations associated with PCA, regarding its use in the context of this research. Firstly, principal components are restricted to be a linear function of the input data. Secondly, the number of principal components cannot be greater than the dimensionality of the data.

These limitations do not apply in K-SVD. Since l^1-minimization is involved, the sparse representations are certainly a non-linear function of the input data. In addition, the dictionary consists of more column vectors than the dimensionality of the data. This is a crucial assumption in order for sparsity to be achieved, along with high accuracy in reconstruction. Since only a small number of the dictionary's columns are actually used, representations become of higher level, in terms of the many possible different patterns that might be hidden in the data. The more columns a dictionary has, the more choices OMP has to pick the best ones, in order to create a sparse representation with the minimum possible reconstruction error.

In the following sections, two similar classifiers will be presented, which only differ in the nature of the dictionary they use. Figure 8 shows a block diagram of the Sparse Representation-based classifier, whose two versions are presented in the following subsections.

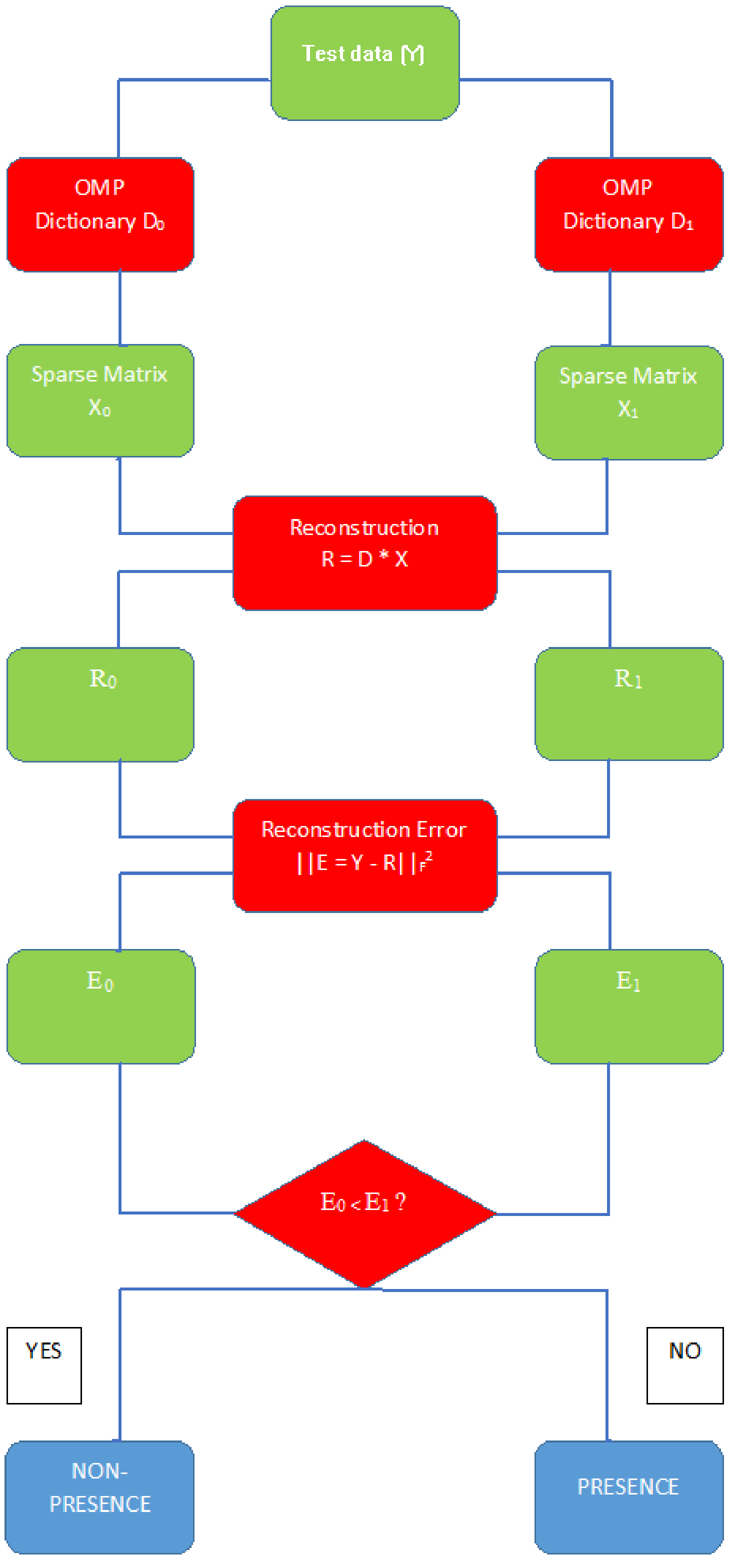


Figure 8: Sparse Representation-based Classification Algorithm

5.2.7 Sparse Representation-based Classification Algorithm using K-SVD

The first step of this method is to train a dictionary for each class ('non-presence', 'presence') using the K-SVD algorithm. The two data sets representing the two classes (empty.csv, m-1.csv) are used separately in this method. They are split into a 70% training set and a 30% test set. A dictionary is trained on each training set by the K-SVD algorithm. Then, for every input test data set the Orthogonal Matching Pursuit algorithm is employed to solve the l^1-minimization problem (1), each time using a different dictionary, obtained by K-SVD. Each one of the two resulting sparse representations are used to reconstruct the initial data set. The reconstruction error $\|Y - DX\|_F^2$ is calculated for each of the two cases and the test data set is classified based on the minimum of the two reconstruction errors.

Algorithm 4 below summarizes the complete procedure.

Algorithm 4 Sparse Representation-based Classification with K-SVD

```
1: train **Dictionary0** using K-SVD on 'non-presence' training data set
2: train **Dictionary1** using K-SVD on 'presence' training data set
3: **SparseMatrix0**=OMP (Dictionary0, testdata)          ▷ $l^1$-minimization problem
4: **SparseMatrix1**=OMP (Dictionary1, testdata)          ▷ $l^1$-minimization problem
5: **if** $\|testdata - Dictionary0 * SparseMatrix0\|_F^2 < \|testdata - Dictionary1 * SparseMatrix1\|_F^2$ **then**
   **return** 0                                           ▷ 'non-presence'
6: **else**
7: **return** 1                                           ▷ 'presence'
```

5.2.8 Sparse Representation-based Classification Algorithm using the Training Sets as Dictionaries

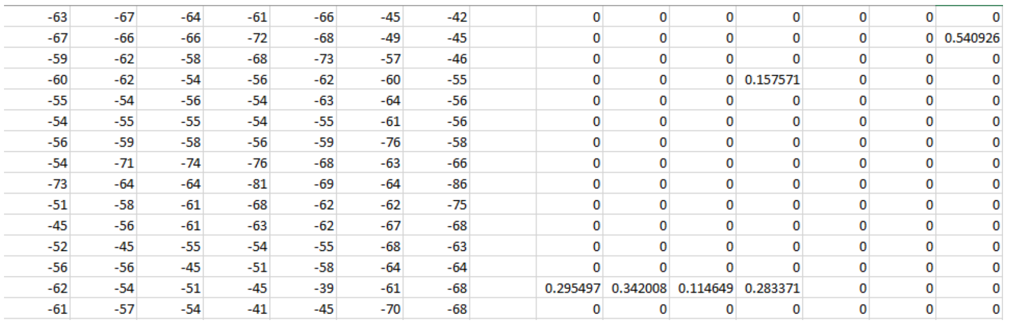
The second algorithm proposed in this section differs from Algorithm 1 only in the nature of the dictionaries. This time I do not apply K-SVD to train dictionaries and the training sets from the two classes are directly fed into the Orthogonal Matching Pursuit algorithm, playing the role of dictionaries. Since any input data set will share similar features with the training set-dictionary coming from the same class, I expect that this method will manage to provide a sufficient discrimination criterion. The sparse matrix generated by the dictionary from the same class is expected to have less and smaller non-zero coefficients, thus leading to a successful classification method. An overview of Algorithm 5 is presented below.

Algorithm 5 Sparse Representation-based Classification with Training Sets as Dictionaries

```
1: Consider the 'non-presence' training data set as **Dictionary0**
2: Consider the 'presence' training data set as **Dictionary1**
3: **SparseMatrix0**=OMP (Dictionary0, testdata)                ▷ $l^1$-minimization problem
4: **SparseMatrix1**=OMP (Dictionary1, testdata)                ▷ $l^1$-minimization problem
5: **if** $\|testdata - Dictionary0 * SparseMatrix0\|_F^2 < \|testdata - Dictionary1 * SparseMatrix1\|_F^2$ **then**
   **return** 0                                                 ▷ 'non-presence'
6: **else**
7: **return** 1                                                 ▷ 'presence'
```

The algorithms in this section were developed in Matlab with the use of the KSVD_Matlab_Toolbox from Aharon et al. (2006).

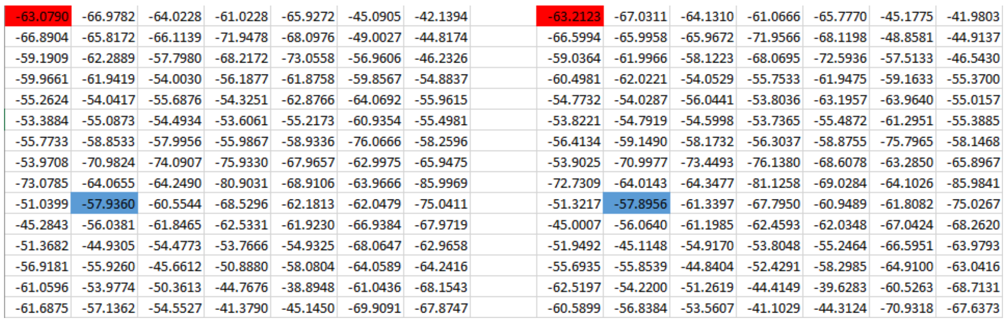
A visualization of the intuition behind the Sparse Representation-based classifier is provided in the following two figures.



-63	-67	-64	-61	-66	-45	-42		0	0	0	0	0	0	0
-67	-66	-66	-72	-68	-49	-45		0	0	0	0	0	0	0.540926
-59	-62	-58	-68	-73	-57	-46		0	0	0	0	0	0	0
-60	-62	-54	-56	-62	-60	-55		0	0	0	0.157571	0	0	0
-55	-54	-56	-54	-63	-64	-56		0	0	0	0	0	0	0
-54	-55	-55	-54	-55	-61	-56		0	0	0	0	0	0	0
-56	-59	-58	-56	-59	-76	-58		0	0	0	0	0	0	0
-54	-71	-74	-76	-68	-63	-66		0	0	0	0	0	0	0
-73	-64	-64	-81	-69	-64	-86		0	0	0	0	0	0	0
-51	-58	-61	-68	-62	-62	-75		0	0	0	0	0	0	0
-45	-56	-61	-63	-62	-67	-68		0	0	0	0	0	0	0
-52	-45	-55	-54	-55	-68	-63		0	0	0	0	0	0	0
-56	-56	-45	-51	-58	-64	-64		0	0	0	0	0	0	0
-62	-54	-51	-45	-39	-61	-68		0.295497	0.342008	0.114649	0.283371	0	0	0
-61	-57	-54	-41	-45	-70	-68		0	0	0	0	0	0	0

Figure 9: Original RSS data and a sparse representation of it

Figure 9 depicts a randomly chosen subset of an original RSS data set (Y) and a sparse representation of it (X) with respect to a dictionary (D). Without any significant loss, the important information that is hidden in Y is gathered in the six non-zero elements of X, together with their position in the matrix. This statement can be justified by Figure 10 that shows the result of the Sparse Representation-based Classification Algorithm (Algorithm 5) which was applied to the original data set of Figure 9 with respect to two different dictionaries (training sets).



-63.0790	-66.9782	-64.0228	-61.0228	-65.9272	-45.0905	-42.1394
-66.8904	-65.8172	-66.1139	-71.9478	-68.0976	-49.0027	-44.8174
-59.1909	-62.2889	-57.7980	-68.2172	-73.0558	-56.9606	-46.2326
-59.9661	-61.9419	-54.0030	-56.1877	-61.8758	-59.8567	-54.8837
-55.2624	-54.0417	-55.6876	-54.3251	-62.8766	-64.0692	-55.9615
-53.3884	-55.0873	-54.4934	-53.6061	-55.2173	-60.9354	-55.4981
-55.7733	-58.8533	-57.9956	-55.9867	-58.9336	-76.0666	-58.2596
-53.9708	-70.9824	-74.0907	-75.9330	-67.9657	-62.9975	-65.9475
-73.0785	-64.0655	-64.2490	-80.9031	-68.9106	-63.9666	-85.9969
-51.0399	-57.9360	-60.5544	-68.5296	-62.1813	-62.0479	-75.0411
-45.2843	-56.0381	-61.8465	-62.5331	-61.9230	-66.9384	-67.9719
-51.3682	-44.9305	-54.4773	-53.7666	-54.9325	-68.0647	-62.9658
-56.9181	-55.9260	-45.6612	-50.8880	-58.0804	-64.0589	-64.2416
-61.0596	-53.9774	-50.3613	-44.7676	-38.8948	-61.0436	-68.1543
-61.6875	-57.1362	-54.5527	-41.3790	-45.1450	-69.9091	-67.8747

-63.2123	-67.0311	-64.1310	-61.0666	-65.7770	-45.1775	-41.9803
-66.5994	-65.9958	-65.9672	-71.9566	-68.1198	-48.8581	-44.9137
-59.0364	-61.9966	-58.1223	-68.0695	-72.5936	-57.5133	-46.5430
-60.4981	-62.0221	-54.0529	-55.7533	-61.9475	-59.1633	-55.3700
-54.7732	-54.0287	-56.0441	-53.8036	-63.1957	-63.9640	-55.0157
-53.8221	-54.7919	-54.5998	-53.7365	-55.4872	-61.2951	-55.3885
-56.4134	-59.1490	-58.1732	-56.3037	-58.8755	-75.7965	-58.1468
-53.9025	-70.9977	-73.4493	-76.1380	-68.6078	-63.2850	-65.8967
-72.7309	-64.0143	-64.3477	-81.1258	-69.0284	-64.1026	-85.9841
-51.3217	-57.8956	-61.3397	-67.7950	-60.9489	-61.8082	-75.0267
-45.0007	-56.0640	-61.1985	-62.4593	-62.0348	-67.0424	-68.2620
-51.9492	-45.1148	-54.9170	-53.8048	-55.2464	-66.5951	-63.9793
-55.6935	-55.8539	-44.8404	-52.4291	-58.2985	-64.9100	-63.0416
-62.5197	-54.2200	-51.2619	-44.4149	-39.6283	-60.5263	-68.7131
-60.5899	-56.8384	-53.5607	-41.1029	-44.3124	-70.9318	-67.6373

Figure 10: Reconstructed data sets

The data set on the left of Figure 10 is the reconstructed data set, based on the training set from its own class, while the one on the right is the reconstructed data set, based on the training set from the other class. It can be observed that the values of both data sets are close to the original ones. However, if someone compares each cell separately, they will realize that the values in the left data set are usually more close to the values of the original one. Two examples are shown in the red and blue cells. As a result, the representation error that is calculated by the algorithms can reveal the class a data set belongs to.

5.2.9 Sparse Representation-based Classification vs k-Nearest Neighbor

Both Sparse Representation-based Classification algorithms can be considered as a generalization of the Nearest Neighbor (NN) classification method. NN is the simplest version of k-NN, a non-parametric algorithm that does not require any learning process as part of the training phase. It uses the entire training set to make its decisions, which consists of a set of vectors and a class label corresponding to each one of these vectors. k-NN can tackle both binary and multi-class classification tasks. The choice of k corresponds to the number of the closest neighbors, based on some distance metric that will affect the decision of the classifier. For binary classification, k usually is an odd number. The algorithm finds the k closest to a test point x training points and performs a majority voting in order to assign the most popular class label to x. Consequently, each training point votes for their class and the class with the most votes is assigned to x. The case where k=1 is referred to as nearest neighbor classification.

NN decides on the class label of a test point based on a single training point. The Sparse Representation-based Classification algorithm takes into account all possible dictionary columns and tries to find the minimum number of training samples that can represent a test sample so that this representation can be considered sparse. Similarly to k-NN, the choice of the feature space does not have any effect on the classifier. The two critical factors are the dimension of the feature space, which has to be significantly large and the proper calculation of the sparse representation. A serious disadvantage of k-NN is that it suffers from the curse of dimensionality. The volume of the input space increases exponentially as the number of dimensions increases and k-NN would struggle in that case.

6 Results

The methods presented in Section 5 are applied to the data sets (Section 4) under various experimental settings and the results are reported for each one of the methods in the following subsections respectively. The most valuable conclusions are discussed with respect to the research objective. Thus, I evaluate the predictive power of the algorithms and how it is affected by the different number of nodes that are employed in an area of interest. Secondly, an estimation is made regarding how quickly each algorithm accurately predicts the outcome of the classification task. This estimation relies on three different factors which are the number of measurements that are sufficient for a correct prediction, the computational time of the training phase and the computational time the algorithm takes to predict the class (after the completion of the training phase). These factors are discussed in each of the following subsections.

6.1 Classification on the Raw Data Sets

The methodology is applied to the initial RTI data set, which contains the RSS vectors for all 28 nodes, as well as three smaller RTI data sets which represent the cases where 10, 5 and 2 nodes, randomly chosen, are employed respectively and also to the Crownstone data set. The results of each implementation are shown and discussed below. It should be mentioned that only the values of M for which the highest accuracy was achieved are reported.

Number of Nodes	Accuracy (30% test set)	M (30% test set)	% Accuracy (210 obs. test set)	M (210 obs. test set)	Training Time (sec)	Number of Observations	Data Set
28	100	1-6, 8-12	1.0000	all	13.4289	5766	RTI
10	96.49	3, 4	0.9333	1, 2	1.2959	2088	RTI
5	91.26	3	0.9095	1-4	0.2903	1028	RTI
2	80.95	2	0.8285	2	0.0343	417	RTI
5	86.10	1	0.8571	2, 4, 5	0.2992	8487	Crownstone

The slight variation in accuracy is due to the change of the percentages in which the data set is split into training and test set. The results show that even when only 210 observations used as a test set, the classifier proved to be good enough in prediction. This small number of samples is specifically chosen, as it only takes approximately three seconds for them to be collected. Consequently, it makes the application realistic because it can identify the change of the state from ’non-presence’ to ’presence’ and vice versa. It should be noted that since each single RSS vector has its own label and 70-80 such vectors are collected per second, accuracy levels of 80% are sufficient to correctly predict whether someone is present or not. The training phase in each of the experiments does not take more than 13.4289 seconds (28 nodes case). The algorithm predicts the outcome in less than a second.

One might observe that, in comparison with the results of the RTI data set when the same number of nodes (five) is used, accuracy in the Crownstone Data Set is around 5% lower. A possible explanation lies to the fact that the Crownstone data contain many RSS vectors whose values are reported within a range of some seconds and not in the exact same second. In other words, I could consider this data set as being more noisy compared to the first one. Finally, it can be concluded that the classification task of presence detection is successful even in the case when only two nodes are deployed in an area of interest.

6.2 Sparse Representation-based Classification Algorithm using K-SVD

In this application, data sets of different dimensions are used as input. In more details, from the initial data set, new ones are created by only keeping RSS values from randomly selected subsets of nodes, as discussed in the beginning of Section 5. In addition, different values for the K parameter are tested, in order to obtain an estimate of the size of a strong dictionary. The term strong implies that the algorithm can provide correct predictions in high accuracy, with respect to the dictionary. Also, the fewer measurements the algorithm needs to correctly predict the class, the stronger the dictionary. Thus, an evaluation data set is used and every time the algorithm predicts the right class, the size of the evaluation set is reduced and the same action is repeated. This procedure shows the minimum size of a data set for which the algorithm can correctly predict the class it comes from. The following table summarizes the important parameters and the results of the experiments. K_0 and K_1 represent the K parameters for dictionaries D_0 and D_1 respectively, D_0 time and D_1 time correspond to the time the algorithm takes to train the dictionaries, expressed in minutes and #Iter is the number of iterations of K-SVD at each training procedure.

Number of Nodes	K_0	D_0 time	K_1	D_1 time	#Iter	Classification Outcome	Data Set
28	300	5	300	5	80	FAILURE	RTI
28	1000	10	1000	10	40	SUCCESS	RTI
15	700	5	700	5	40	SUCCESS	RTI
10	400	1	400	1	40	FAILURE	RTI
10	900	6	500	1	40	FAILURE	RTI
10	900	15	500	4	80	FAILURE	RTI
5	550	3	250	1	40	FAILURE	RTI
5	1000	10	1000	10	40	FAILURE	Crownstone
5	2000	65	2000	67	40	FAILURE	Crownstone
5	3500	420	2000	67	40	FAILURE	Crownstone

The classification outcome is considered successful when test sets from both classes can be assigned the true label. In any other case, an experiment fails. As it can be seen in the table, the algorithm proves to be an accurate classifier when the RTI data set includes information gathered by at least 15 nodes. In these cases, the right class is predicted for both test sets and for every possible size of them, even when only one RSS vector is used as input. The experiments on the Crownstone data are unsuccessful. Although the size of the dictionaries is close to their upper bound (based on K-SVD's constraint for which it holds that $K < N$), classification cannot be performed correctly, most likely because a data set with only five attributes requires larger dictionaries to be trained with respect to the training sets. However, this process is quite expensive in terms of computational time. It should be noted that, once the dictionaries are trained, the algorithm needs less than a second to deliver its output. What appears to be an important conclusion is that the classification process seems to be highly dependent on the size of the dictionaries. The truth of this statement is explored further in the next Sparse Representation-based Classification application, in which there are not any constraints on how large a dictionary can be, as in K-SVD.

6.3 Sparse Representation-based Classification Algorithm using the Training Sets as Dictionaries

The sequence of the experiments in this section and the choice of the parameters, as shown in the table below, follow the same intuition as in the K-SVD-based Sparse Representation classifier. The reader should recall that the difference of this algorithm from the K-SVD-based one is that there is no dictionary training phase. The training sets represent the dictionaries and therefore they are directly fed into the Orthogonal Matching Pursuit Algorithm.

Number of Nodes	#Samples of Dictionary0	#Samples of Dictionary1	Classification Outcome	Data Set
28	2650	1387	SUCCESS	RTI
28	1000	1000	SUCCESS	RTI
20	1880	1000	SUCCESS	RTI
20	1000	1000	SUCCESS	RTI
15	1500	800	SUCCESS	RTI
10	961	572	FAILURE	RTI
10	2000	1000	SUCCESS	RTI
5	5659	2629	SUCCESS	Crownstone
5	4000	1900	FAILURE	Crownstone
2	1607	700	FAILURE	Crownstone

As far as the RTI data set is concerned, the classification task of presence detection is successful with as few as 10 nodes. About the Crownstone data set, which includes a lot more samples than the RTI data set of the same dimensions (number of nodes), I manage to obtain strong dictionaries and this leads to accurate classification performance.

In some of the successful experiments, the algorithm sometimes misclassifies test sets that contain a very small number of RSS vectors (17 or less). However, because of the high data collection rate, this can be considered as a slight malfunction of the algorithm that cannot appear in practice. In only one second, a lot more samples can be collected, so it is an extreme case that should be tested in the context of the current research, but it certainly cannot affect the algorithm's overall successful performance. The algorithm delivers its output in less than a second and the performance of the classifier is highly dependent on the size of the training sets.

6.4 Results Comparison

Two main methods are compared in this research, which are the Random Forests and the Sparse Representation-based classifier. In the second method, two algorithms are developed, which only differ in the source the dictionaries come from. Algorithm 4 trains the dictionaries using K-SVD and Algorithm 5 uses the training sets as dictionaries.

Due to the different nature of the two main methods, there is not a straightforward way to compare their performance in classification. Random Forests work on data sets in which each RSS vector represents a different sample. The input of the Sparse Representation-based classifier is a data set for which a prediction must be made. Thus, in this case a data set can be considered as one sample. If I consider the accuracy of Random Forests as a majority vote for a class label, then there is a common performance measure. In that case, Random Forests always predict the right class label, while the Sparse Representation-based classifier does not. However, once strong dictionaries are found, the Sparse Representation-based classifier can be directly compared with Random Forests, as both give 100% correct predictions. In the context of the current research, it is crucial that a classification algorithm can perform exceptionally in the classification task. In smart home applications, where devices should respond to our presence, there is no room for a false prediction. If for example, crownstones are responsible for cutting off the power usage so that kitchen appliances do not work when someone is not there, then it would be dangerous if the system fails to achieve it.

Regarding the minimum number of nodes that need to be deployed in an area of interest in order for an algorithm to correctly predict the state of 'non-presence' or 'presence', Random Forests outperform the two Sparse Representation-based algorithms. Two nodes are sufficient for the Random Forest mechanism to detect presence in an area of interest. The Sparse Representation-based classifier is successful with five or more nodes, in the case where the training sets are used as dictionaries. The K-SVD-based classifier does not succeed with fewer than 15 nodes. The fewer number of nodes installed the more realistic and affordable a smart home concept would become.

Regarding how quickly a method predicts the classification outcome, all methods can deliver the predicted class label in less than a second. They correctly predict the label of a data set of any possible size. It should be noted that it takes only milliseconds for a data set with a few RSS vectors to be collected. An important difference is in the training phase. The Sparse Representation-based classifier that uses K-SVD shows by far the worst performance. In our experiments, the training of a dictionary varies between one minute and seven hours. Random Forests need no more than 13.4289 seconds for the training phase at the worst case scenario. Finally, when the training sets are used as dictionaries in the Sparse Representation-based classifier, no training phase is associated with this mechanism. Negligible computational time is necessary in real applications when someone enters or exits an area where a wireless sensor network is deployed, since the system can respond to this change immediately and subsequently perform any kind of actions.

7 Conclusion

This paper uses machine learning and signal processing methods to achieve passive presence detection. This is to detect whether someone is present or not in an area of interest. The methods process RSS data gathered by Bluetooth Low Energy nodes placed inside the area. The Random Forest ensemble classifier is used as the first method, which adopts random features to grow a forest of classification trees in order to predict the class label. The second method is a custom classification algorithm which exploits the theory of compressed sensing by generating sparse representations of the input data. Experimental results show that the task of passive presence detection is feasible, as the change of the state from presence to non-presence and vice versa can be detected in milliseconds even when only two Bluetooth Low Energy nodes are used and no knowledge about the position of the sensors exists. The outcome of this research allows for applications in a smart home environment. Indicatively, it offers advanced safety when someone forgets to switch off a kitchen appliance before they leave or it can act as a low-cost intrusion detection system.

Future research could possibly involve not only detecting presence but also finding approximations of the number of people that are present in an area of interest. For instance, in a hospital certain rooms are meant for physicians to do administrative work while others are meant for physicians to treat their clients. Physicians might use those latter rooms to do administrative work for their own convenience. Knowledge about the presence of a single physician versus the presence of multiple people would give a hospital the ability to optimally manage the availability of the rooms by gently advising the physician to move to the appropriate room and thus increasing the rate of patients who receive treatment per hour.

Further investigation for the applicability of a passive presence detection system is also required in cases where there are limitations regarding placing of the infrastructure. In a formal care housing it is important to know where people with dementia for example, are in a building. In such cases, vertical localization is necessary. If a person is on a different floor than the one they should be, then it takes more time for a caretaker to find them. The challenge in this case is for the system to obtain the required vertical resolution.

8 References

Youssef, M., Mah, M., & Agrawala, A. (2007). Challenges: Device-free Passive Localization for Wireless Environments. *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking - MobiCom '07*, 222-229, 2007.

Aharon, M., Elad, M., & Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11), 4311–4322.

D.L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

Eldar, Y. C., & Kutyniok, G. (2012). *Compressed Sensing: Theory and Applications.*

Wilson, J., & Patwari, N. (2008). Radio Tomographic Imaging with Wireless Networks, 9(5), 621–632.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry & Y. Ma, "Robust Face Recognition via Sparse Representation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210-227, Feb. 2009.

D. S. Wang, X. S. Guo & Y. X. Zou, "Accurate and robust device-free localization approach via sparse representation in presence of noise and outliers," *2016 IEEE International Conference on Digital Signal Processing (DSP)*, Beijing, 2016, pp. 199-203.

Kanso, M. A. (2009). Compressed RF Tomography : Centralized and Decentralized Approaches, (October).

Elad, M. (2010). *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing (1st ed.).* Springer Publishing Company, Incorporated.

Tan, P., Steinbach, M., & Kumar, V. (2014). *Introduction to data mining* (Pearson New International edition. First edition. ed., Pearson custom library). Essex: Pearson.

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

C. E. Shannon, ”Communication in the Presence of Noise”, Proc. Institute of Radio Engineers, vol. 37, no. 1, 1949, pp. 10-21.