Relative Estimation of Internet of Things Sensor Nodes Based on Sensor Readings

Author:
Hans Hellzen Melby

Supervisors:
Pierluigi Salvo Rossi, IET, NTNU
Sigve Tjora, Disruptive Technologies Research and Development

TTT4510 - Signal Processing, Specialization Project

Faculty of Information Technology, Mathematics, and Electrical Engineering

Trondheim
19. December 2016

Abstract

The Internet of Things has been referred to as the next Industrial Revolution that will change how businesses, consumers, and governments work [6]. In order to help meet this future demand, Disruptive Technologies Research and Development is in the process of developing cheap and easily configurable IoT-sensors that customers can buy in bulk an place around their homes or other locations without any need for technicians to help in the installation and deployment process. However, since the customers themselves place the sensors in a location of their choosing, and since the sensors only transmit raw data without information related to their respective location, no specific localization information will be recorded. The aim of this project is to explore the possibility of inferring relative localization information about the sensors from looking only at the sensor data measurements themselves.

Contents

List of Figures

1 Introduction

1.1 Definitions and current scope

Oxford Dictionary defines the Internet of Things (IoT) as "The interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data" [2]. It is estimated that 50 billion devices will be connected to the Internet by the year 2020 [5]. Therefore, Internet of Things are one of the hottest buzzwords of the current Information and Communication Technology (ICT) scope. According to a Cisco survey from 2011, the IoT concept was "born" sometime between 2008 and 2009 [5]. As both homes and businesses are embracing the idea of monitoring and controlling various devices over the Internet, the demand for cheap and easily configurable IoT devices will increase exponentially in the foreseeable future. However, many of today's existing solutions face numerous challenges, such as being overly complex, unreliable, and having too short of a lifespan.

1.2 Motivation

Disruptive Technologies Research and Development is a company currently focusing on solving many of the mentioned problems by providing cheap, easily configurable sensors that users can buy in bulk and place in a location of their choosing with minimal effort, i.e. without any need for technicians or installation professionals to assist in the process. However, since no third party is involved in the installation and configuration process, no information will be recorded in regards to the location of the placement of these sensors. It is therefore of interest to explore the possibility of inferring the relative locations of these sensors based only on the sensor data. A typical scenario might be that data from a certain set of sensors over a certain amount of time is stored on a cloud database, and one wants to use this data to estimate the distance between these sensors. If such a method can be proven to be reliable, one might be able to use this information to make conclusions about where in a certain location (for instance, a home) each of these sensors are located, and thus provide customers with for instance a web interface where they can view sensor information from the different rooms in their house without ever needing to manually specify where the sensors are located.

1.3 Related work

There exists vast amounts of research papers and literature describing methods for inferring the locations of sensors. However, nearly all of the existing material uses localization-specific, such as received signal strength (RSS), time-of-flight (TOF), and Global Positioning System (GPS) information (ex: [7], [8]), whereas the problem at hand is limited to looking at stored sensor data only. Therefore, a new procedure based on existing theory in signal processing and machine learning had to be developed.

1.4 Outline

Section two describes the theory related to the methods used in the implementation. The third section describes in more detail the main problem at hand in this project thesis and a detailed, stepwise description of the methods used to model, simulate, and obtain results. The fourth section presents the result obtained from the method implemented. The fifth section discusses the results obtained and presents possible improvements to the current method. The sixth, and final section provides a conclusion and describes possible future work to build on the current solution.

2 Theory

This section describes the theory used for the implementation of the system. Most of it is from fields such as signal processing, statistics, machine learning, and linear algebra.

2.1 Autoregressive model

An autoregressive (AR) model can be described as a stochastic process where future values are estimated based on a weighted sum of past values. The order of the model denotes how many previous values are used to estimate the next value. If the order of the model is denoted as p, and X_i is the result of an autoregressive model, then an AR[p] model of X_i can be written as:

$$X_i = \sum_{k=1}^{p} a_k X_{i-k} + \epsilon_i \tag{1}$$

Where a_k are the weighing coefficients, X_{i-k} represents the k'th past value of the recorded sequence X, and ϵ_i is an error term following some statistical distribution

A variation of this will be used to generate realistic, synthetic time-series sensor data, where the weighing coefficients are used to represent both the correlation and the distance between the different sensors.

2.2 Gaussian distribution

The Gaussian distribution is a continuous probability distribution that is often used to represent real-valued random variables whose distribution is not exactly known. It contains two parameters: σ^2, which represents the variance, and μ, which represents the mean. The distribution is modeled through the following function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

If a random variable X follows a Gaussian distribution, the following notation is commonly used:

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{3}$$

Below is a graph with several Gaussian distributions of different means and variances.

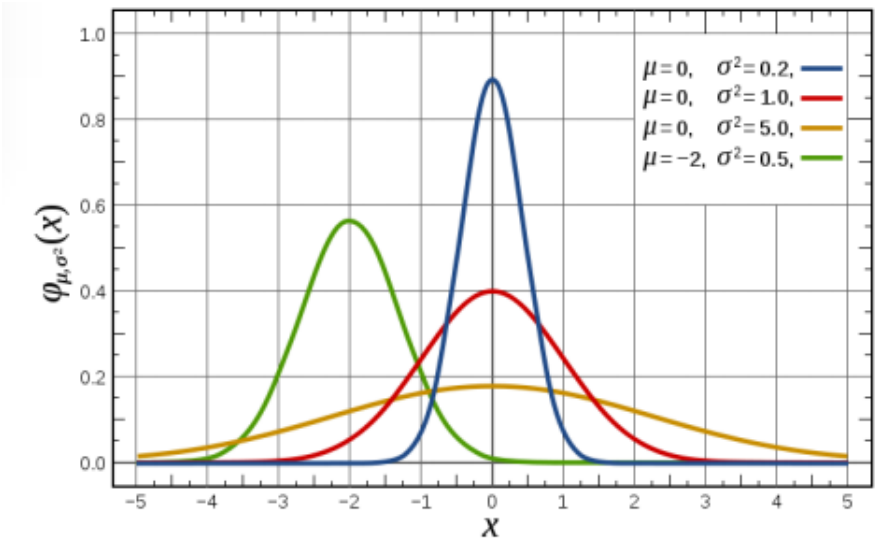

Figure 1: Various Gaussian distributions [9]

The Gaussian distribution is often used to represent noise in approximation models.

2.3 Pearson product-moment correlation coefficient

The Pearson product-moment correlation coefficient, or PPMCC, is the measure of the linear dependence between two variables. The PPMCC gives a value between -1 and +1, where -1 is negative correlation, +1 is positive correlation, and 0 is no correlation. The PPMCC $\rho_{X,Y}$ for the two time

series realizations X and Y can be written as:

$$\begin{aligned}\rho_{X,Y} &= \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \\ &= \frac{cov(X,Y)}{\sigma_X \sigma_Y}\end{aligned} \tag{4}$$

Where cov(X, Y) is the covariance between X and Y, and σ denotes the standard deviation of the respective time series realizations.

This coefficient will be used to calculate the correlation between the sensor data realizations.

2.4 Mean squared error

The mean squared error (MSE) is a measure of the average of the squares of the deviations between an estimate and its true value. This deviation can occur due to randomness or because the estimator does not account for information that can provide a better estimate. The MSE is defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{5}$$

Where $Y_1, ...Y_n$ is the observed values and $\hat{Y}_1, ..., \hat{Y}_n$ are the corresponding estimates. The closer an MSE is to zero, the better the estimator.

2.5 Regression analysis

Regression analysis is a statistical method for estimating the relationship between certain variables. More specifically, it aims at determining the relationship between some dependent variable and one or more independent variables, where the independent variables are used as prediction terms. The function that estimates said relationship is called a regression function.

2.5.1 Linear regression

In the case of simple linear regression, the relationship can be modeled as

$$y = a_0 + a_1 x + \epsilon \tag{6}$$

Where a_0 and a_1 are weighing terms determined from some regression analysis model, and ϵ is a random, unobserved error term with zero mean.

2.5.2 Polynomial regression

In many cases, the relationship between some dependent variable and the set of independent variables cannot be modeled as a linear relationship. In such cases, polynomial regression becomes useful. Polynomial regression is a form of multiple linear regression where the relationship between two variables is modeled as an nth degree polynomial. If the independent values $\mathbf{x}$=x_1, x_2,..., x_N are considered to be the "features" of the model, and $\mathbf{y}$=y_1, y_2,..., y_N are the dependent output values, then one can model the relationship as

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + ... + a_n x_i^n \tag{7}$$

Where

$$\mathbf{x_i} = [1 \; x_i \; x_i^2 \; ... \; x_i^n] \tag{8}$$

Is the feature vector for each independent variable x_i. The feature vectors can then be stacked on top of each other and multiplied by the alpha coefficient column vector which then equals the dependent output values. The equation thus becomes

$$\begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^k \\ 1 & x_2 & x_2^2 & ... & x_2^k \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ 1 & x_n & x_n^2 & ... & x_n^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} \tag{9}$$

Or written in simpler terms:

$$\begin{bmatrix} \mathbf{x_1}^\top \\ \mathbf{x_2}^\top \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x_n}^\top \end{bmatrix} \mathbf{a} = \mathbf{y} \tag{10}$$

Since the feature vectors $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$, and the output vector $\mathbf{y}$ is known, the next step becomes solving for the alpha weighting coefficients.

2.5.3 Moore-Penrose Pseudoinverse

The problem with equation 9 in the case of $n > k$, which often is the case in regression problems, is that the system becomes overdetermined, meaning there are more equations than unknowns. One way to solve this is to compute a best-fit solution using the Moore-Penrose Pseudoinverse (MPI), which is a generalization of the inverse of a matrix. The MPI of a matrix A is denoted as A^+, and is computed as:

$$A^+ = (A^* A)^{-1} A^* \tag{11}$$

With the important property that

$$A^+ A = I \tag{12}$$

Where I denotes the identity matrix. In our case, if X denotes the feature matrix, equation 10 becomes:

$$X\mathbf{a} = \mathbf{y} \tag{13}$$

Multiplying by the MPI of X on both sides gives

$$\begin{aligned} \underbrace{X^+ X}_{=I}\mathbf{a} &\approx X^+ \mathbf{y} \\ \mathbf{a} &\approx X^+ \mathbf{y} \end{aligned} \tag{14}$$

2.6 Mapping coordinates in Euclidean space

The following method is based on the article "Rapid Calculation of Coordinates from Distance Matrices" by G. M. Crippen from the University of California [4].

Given a distance matrix $\mathbf{D}$ of distances d_{ij} where i denotes the row number and j the column number. Assume the following constraints:

$$\begin{aligned} d_{ij} &= d_{ji} \\ d_{ij} &= 0 \qquad i = j \\ d_{ij} &> 0 \qquad i \neq j \end{aligned} \tag{15}$$

Define a matrix $\mathbf{M}$ of same size as $\mathbf{D}$ with elements m_{ij} where

$$m_{ij} = \frac{d_{1j}^2 + d_{i1}^2 - d_{ij}^2}{2} \tag{16}$$

Decompose the matrix $\mathbf{M}$ by eigenvalue decomposition such that

$$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{U}^\top \tag{17}$$

The number of dimensions needed for exact representation of the points is equal to the number of nonzero singular values of the matrix $\mathbf{S}$. The coordinate matrix $\mathbf{X}$ can now be found through

$$\mathbf{X} = \mathbf{U}\sqrt{\mathbf{S}} \tag{18}$$

With $\sqrt{\mathbf{S}}$ denoting taking the square root of each element in $\mathbf{S}$. The coordinates for each point is now represented on each row of $\mathbf{X}$. If one wants to represent the coordinates in fewer dimensions (at a cost in accuracy), one can choose to only keep a certain amount of the singular values of $\mathbf{S}$, and set the rest to zero. For instance, if rank($\mathbf{S}$)=n, and one wishes to map the coordinates in k dimensions where $k < n$, then one can create a new matrix $\mathbf{S}$', which is of equal size as $\mathbf{S}$, but where only the k'th first singular values are kept, and the rest are set to zero.

3 Description and implementation

3.1 Defining the task at hand

Other than the project title, very little information was given about the exact project scope and preferable methods to use. This meant that the project was quite open to interpretation and choice in methods, but it also meant one had to appropriately make limitations and assumptions about which scenarios to explore.

The scenario eventually chosen to further explore was looking at the location of a certain set of sensor *in relation to each other*. Thus, one does not make any conclusions about *where* in a location (such as an office, home, etc...) a certain sensor is located, but rather the *approximate distance* from another sensor (or sensors) based on the sensor reading.

3.1.1 Assumptions and simplifications

As mentioned in the introduction, the sensors are still in the development phase, and thus no sensors were available to provide measurement data. Therefore, this had to be generated synthetically. Before this was done, the following assumptions were made:

- A certain set of sensors are known to be located in the same location
- Timestamps are available for each reported sensor value
- The sensors report their values approximately 100 times per day (Disruptive Technologies have informed that the sensors will transmit around every 15 minutes, which approximately corresponds to 100 measurements per 24 hours)
- The distances between sensors are chosen such that a distance of 20 meters between the sensors are assumed to have very little correlation, and as the distance decreases, the correlation increases (as a nonlinear relationship). While perhaps unrealistic, this is done to have a comparison basis.

For simplicity, the scenario has been limited to only generating the same type of sensor data (such as temperature) for each sensor and try to infer relative distance information by exploring sensor data relationships. In

other words, the methods used are all based on the sensors producing the same type of data.

3.2 Implementation overview

The project implementation is divided into the following parts

1. **Data generation**: Since the sensors are still in development phase, no sensors were available to provide data recordings. Therefore, a method had to be developed to generate synthetic, realistic data from a set of sensors that somehow models the relative distance between them.
2. **Sensor relationship correlation**: The relationship between the sensors over a fixed amount of time is computed and stored for further use.
3. **Distance computation**: A function for relating the sensor correlation to their corresponding distance is trained using statistics and machine learning techniques.
4. **Topology relationship mapping** The distance between the sensors is mapped in two-dimensional Euclidean space.

All scripts were written in Matlab. The relevant Matlab files and data sets can be found in the attached compressed folder. The file script_explanations.txt contains a brief explanation of the purpose of each file.

3.3 Data generation

Generating realistic, synthetic sensor data can be a tricky task, as it is necessary to somehow model the data samples with respect to both its previous values and the values of the sensors in close proximity. The method chosen was to generate the data for each sensor as a variant of an autoregressive (AR) process, modified to include not only past values of of its own data vector, but the data vector of the other sensors in close proximity as well. If $t_i(n)$ denotes the n'th sample of sensor i in a topology with a total of M

sensors, then

$$t_i(n) = \sum_{j=1}^{M} \alpha_{ij} t_j(n-1) + \epsilon_i(n) \tag{19}$$

Where α_{ij} are the weighting coefficients between sensor i and j, $t_j(n-1)$ is the previous value of sensor j, and $\epsilon_i(n) \sim \mathcal{N}(0,\ 0.2)$ and is included as a noise term. This formula is thus applied recursively for each sensor data sample until the desired number of samples is reached.

The alpha values α_{ij}, where $i \neq j$, will henceforth be referred to as the cross-alpha coefficients, and for the case of i = j as self-alpha coefficients.

3.3.1 Alpha weighting coefficients

The α_{ij} weighting coefficients from equation 19 are meant to model how much of an "environment" two sensors share. If an α_{ij} coefficient is high, the two sensors i and j are presumed to have much of the same environment and thus be highly correlated and close in distance. For low α_{ij} coefficients, vice versa. Let $\mathbf{A}_M$ denote a symmetric MxM matrix containing the alpha coefficients. $\mathbf{A}_M$ can then be written as

$$\mathbf{A}_M = \begin{bmatrix} \alpha_{11} & \alpha_{12} & ... & \alpha_{1M} \\ \alpha_{21} & \alpha_{22} & ... & ... \\ ... & ... & ... & ... \\ \alpha_{M1} & ... & ... & \alpha_{MM} \end{bmatrix} \tag{20}$$

For the data to be considered realistic over a 100-sample realization, the following constraints are put on the alpha values:

$$\begin{aligned} \alpha_{ij} &= \alpha_{ji} \\ 0 \leq \alpha_{ij} &\leq 0.5 \qquad i \neq j \\ \sum_{j=1}^{M} \alpha_{ij} &= 1 \end{aligned} \tag{21}$$

The first constraint simply states that sensor i's spatial relationship to sensor j must be equal to sensor j's spatial relationship to sensor i. The

second constraint is for practical issues in relating the cross-alpha values to the sensor data correlation. From testing, it could be observed that when the cross-alphas were set to values greater than 0.5, the correlation between the sensors started to decrease. For instance, in the case of M=2 sensors, if one initially sets the the two cross-alpha coefficients α_{12} and α_{21} to 0, and the self-alpha coefficients α_{11} and α_{22} to 1, and step-wise increments the cross-alpha coefficients by 0.05 (and thus decreases the self-alpha values by the same amount to satisfy the third constraint in equation 21), generates a data realization of 100 samples for each sensor with equation 19, and computes the correlation, one gets the behavior shown in figure 2:

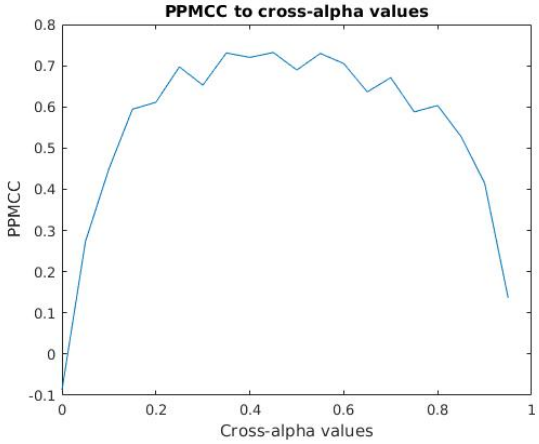

Figure 2: Illustration of correlation behaviour with different cross-alpha values

One can see that from the graph that the correlation between the sensors starts to decrease again after the cross-alpha coefficient reach values of 0.5 and upwards.

The third constraint of equation 21 is to ensure stability in the sensor data generation. If the alpha coefficient sum is less than one, the data will converge towards zero, whereas if greater than one, the data will go towards infinity. This is shown in figure 3, this time for the case of $M = 3$ sensors, each with an initial value of 20, where the sum of alphas is equal to 1.00, 0.95, and 1.05, respectively. As can be seen, if the sum of alphas is equal to one, the data will have realistic variations around its initial value, whereas in the other cases the data will either converge to zero or diverge towards infinity fairly quickly.

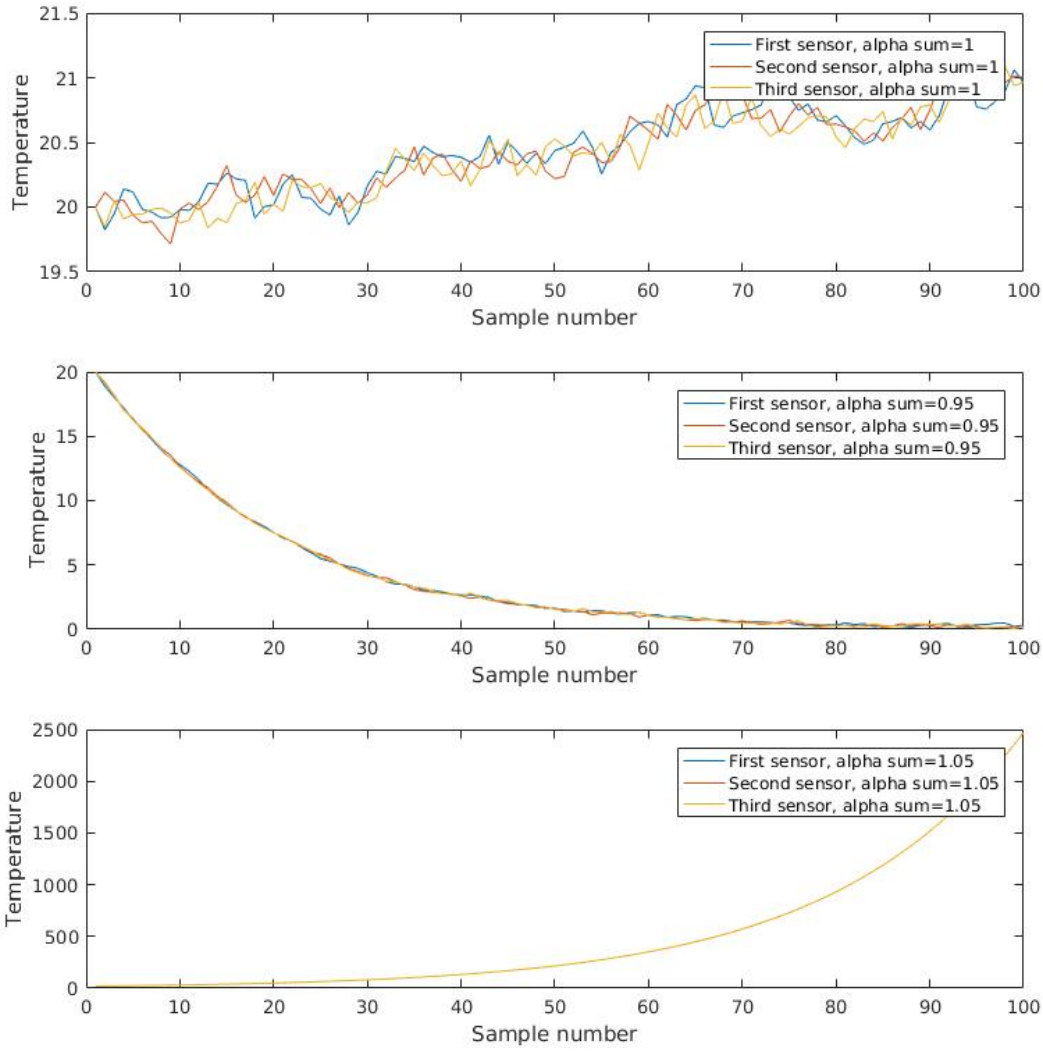

Figure 3: Illustration of data generation behaviour for different alpha value sums

3.3.2 Alpha-distance relationship

In a scenario where one has sensors available to do measurements, exploring the relationship between the distance between sensors and the measurement correlation is a simple task. However, when generating synthetic data, one can obviously not measure the distance between the sensors, it has to be modeled in some way. In the case of this project, this was done through assuming a distance from a combination of the cross-alpha values. Firstly, an alpha-distance function had to be assumed. An assumption was made that the relationship between the correlation and the distance would be a type of non-linear decreasing function, since no correlation would mean the sensors are not in each others vicinity and thus have a high distance between each other, whereas if some positive correlation is measured, the distance decreases rapidly with the distance converging to zero as the PPMCC approaches one. The function chosen to represent this relationship was an exponentially decaying function of the form

$$f(x) = \beta e^{-\phi(x-x_0)} + \gamma \tag{22}$$

with β, ϕ, x_0 and γ being constants (whose values will be discussed on the next page). The initial plan was to simply insert each individual cross-alpha value into the function above and obtain a reference distance between each sensor. However, this turned out not to be a viable method, the reason being that the cross-alpha values will be different for the same type of distance modeling for different numbers of sensors due to the second constraint of equation 21. For instance, consider the two cases of M=3 and M=4 sensors, where one in both cases wants to model the sensors as being as close to each other as possible. For M=3 sensors the alpha matrix would be as follows:

$$\mathbf{A}_3 = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix} \tag{23}$$

Whereas for M=4, it would be:

$$\mathbf{A}_4 = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \tag{24}$$

Both matrices model the same scenario of all the sensors being very close to each other, but they contain different alpha values. A way of normalizing the alpha values to be independent of the number of sensors was therefore necessary. The following method was devised: consider the alpha value α_{ij}, where $i \neq j$, and let α_{ij}' be the normalized value of α_{ij}. Let $\alpha_{sum,ij}$ be the sum of the other cross-alpha values for sensor i excluding α_{ij}. α'_{ij} is thus:

$$\alpha'_{ij} = \frac{\alpha_{ij}}{1 - \alpha_{sum,ij}} \tag{25}$$

One can think of the normalized cross-alpha values as how much that value contributes to the "shared" environment for a particular sensor i. A value close to one will mean the two sensors are very close in distance, and for values close to zero, vice versa. The normalized alpha matrices $\mathbf{A_3}'$ and $\mathbf{A_4}'$ thus becomes:

$$\mathbf{A_3}' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{26}$$

and

$$\mathbf{A_4}' = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \tag{27}$$

With some testing of varying the parameters from equation 22 and plotting the results, the parameters were eventually chosen as $\beta = 3$, $\phi = 2$,

$x_0 = 1$, and $\gamma = -2.5$. Writing equation 22 as a function of the normalized cross-alpha values, the reference distance formula becomes:

$$d_{ij,ref} = f(\alpha'_{ij}) = 3e^{-2(\alpha'_{ij}-1)} - 2.5 \tag{28}$$

Which, when plotted, can be seen in figure 4.

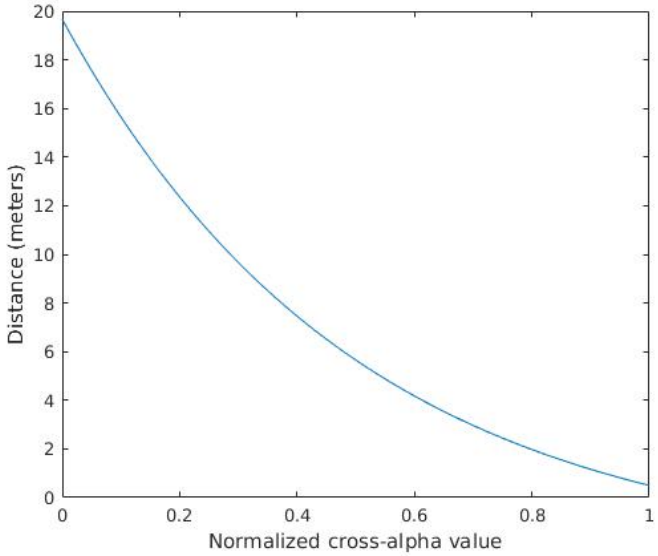

Figure 4: Normalized cross-alpha values to reference distance

The reference distances between a certain set of sensors can be written in matrix form as follows:

$$\mathbf{D}_{ref} = \begin{bmatrix} 0 & d_{12,ref} & ... & d_{1M,ref} \\ d_{21,ref} & 0 & ... & d_{2M,ref} \\ ... & ... & ... & ... \\ d_{M1,ref} & ... & ... & 0 \end{bmatrix} \tag{29}$$

Where $d_{ij,ref} = d_{ji,ref}$.

3.3.3 Data generation procedure

With a formula for generating data samples, and a way of choosing the AR coefficients to model correlation between sensors, a method for efficiently generating a certain amount of data samples for a set of sensors will now be discussed. Let N be the number of data samples to be generated, and again, let M be the number of sensors. Let $\mathbf{T}$ be an MxN matrix containing data with each row representing a sensor and each column representing data samples at a particular point in time. Procedure 1 creates the full data matrix $\mathbf{T}$.

Algorithm 1 Data generation (NOTE: 1-indexing is assumed)

```
1: procedure GENERATEDATAMATRIX
2:     T ← Matrix of M rows, N columns of zeros
3:     A ← Alpha coefficients matrix A of M rows, M columns
4:     B ← Mx1 column vector of initial sensor values
5:     Column 1 of T ← B
6:     for i=2 to N do
7:         E ← Mx1 vector of random Gaussian noise values
8:         Column i of T ← A*[Column i-1 of T] + E
9:     end for
10: end procedure
```

3.4 Sensor relationship correlation

After generating data realizations for each sensor using procedure 1, computing the correlation between the different sensor realizations is a simple task. Simply apply equation 4 between each set of sensor data vectors. Let $\mathbf{T}$ still represent the matrix containing the data realizations for each sensor, which can be written as:

$$\mathbf{T} = \begin{bmatrix} t_{1,1} & t_{1,2} & ... & t_{1,N} \\ t_{2,1} & t_{2,2} & ... & t_{2,N} \\ ... & ... & ... & ... \\ t_{M,1} & t_{M,2} & ... & t_{M,N} \end{bmatrix} = \begin{bmatrix} \mathbf{t_1}^\top \\ \mathbf{t_2}^\top \\ . \\ . \\ \mathbf{t_M}^\top \end{bmatrix} \tag{30}$$

The correlation between two sensor data realizations $\mathbf{t_i}^\top$ and $\mathbf{t_j}^\top$, can

be written as a scalar value c_{ij}, calculated with equation 4 as:

$$c_{ij} = \frac{cov(\mathbf{t_i}^\top, \mathbf{t_j}^\top)}{\sigma_{\mathbf{t_i}^\top} \sigma_{\mathbf{t_i}^\top}} \tag{31}$$

The correlation between all the sensor data realizations can be written as a symmetric MxM matrix with diagonal elements equal to one, represented as follows:

$$\mathbf{C} = \begin{bmatrix} 1 & c_{1,2} & ... & c_{1,M} \\ c_{2,1} & 1 & ... & c_{2,M} \\ ... & ... & ... & ... \\ c_{M,1} & c_{M,2} & ... & 1 \end{bmatrix} \tag{32}$$

3.5 Distance computation

Now that a reference distance and the correlation has been calculated for each set of sensors, the next step is finding a function to relate the two. In cases of relating the correlation between data sets to some output value, regression analysis is often a good choice [1]. Since the relationship is assumed to be non-linear, a polynomial regression approach is used.

Given a training set with a vector of correlation values $\mathbf{c} = [c_1, c_2, ..., c_n]$, and let $\mathbf{d}_{ref} = [d_{1,ref}, d_{2,ref}, ..., d_{n,ref}]$ be the corresponding vector of reference distances assumed to be in meters. Using the method described in 2.5.2, let $\mathbf{c}$ be the independent input values and $\mathbf{d}_{ref}$ be the dependent output values. The k'th order polynomial regression function thus becomes:

$$\begin{bmatrix} 1 & c_1 & c_1^2 & ... & c_1^k \\ 1 & c_2 & c_2^2 & ... & c_2^k \\ & \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & \\ 1 & c_n & c_n^2 & ... & c_n^k \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \cdot \\ \cdot \\ \cdot \\ w_k \end{bmatrix} = \begin{bmatrix} d_{1,ref} \\ d_{2,ref} \\ \cdot \\ \cdot \\ \cdot \\ d_{n,ref} \end{bmatrix} \tag{33}$$

With $w_0, w_1, ..., w_k$ being the function weights. Using the MPI from

2.5.3, the function weights become:

$$\begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_k \end{bmatrix} \approx \begin{bmatrix} 1 & c_1 & c_1^2 & ... & c_1^k \\ 1 & c_2 & c_2^2 & ... & c_2^k \\ & & . . . . . & & \\ & & . . . . . & & \\ & & . . . . . & & \\ 1 & c_n & c_n^2 & ... & c_n^k \end{bmatrix}^+ \begin{bmatrix} d_{1,ref} \\ d_{2,ref} \\ . \\ . \\ . \\ d_{n,ref} \end{bmatrix} \tag{34}$$

Giving the polynomial regression function relating the distance to the correlation between two sensors i and j as:

$$d_{ij}(c_{ij}) = w_0 + w_1 c_{ij} + w_2 c_{ij}^2 + ... + w_k c_{ij}^k \qquad c_{ij} \in [-1, 1] \tag{35}$$

3.6 Distance topology mapping

Using equation 35, one can now obtain a distance estimate matrix for an arbitrary set of sensors. These can then be mapped in two-dimensional Euclidean space using the method described in 2.6. Unfortunately, like described in said section, in many cases an exact representation is not possible in two dimensions due to the rank of the singular matrix often being greater than two. However, it is usually the case that one can eliminate (set to zero) the singular values λ_i where $i > 2$ and still get an estimate in two-dimensional space, often well within a reasonable margin of error. Take the following example with the distance matrix $\mathbf{D}$ given as:

$$\mathbf{D} = \begin{bmatrix} 0 & 2.6938 & 2.7522 & 2.7391 & 19.2914 \\ 2.6938 & 0 & 2.7659 & 2.6972 & 19.7667 \\ 2.7522 & 2.7659 & 0 & 2.4647 & 19.4754 \\ 2.7391 & 2.6972 & 2.4647 & 0 & 20.0920 \\ 19.2914 & 19.7667 & 19.4754 & 20.0920 & 0 \end{bmatrix} \tag{36}$$

The rank of the singular value matrix obtained from doing the Eigenvalue decomposition and described in 2.6 turns out to have a rank of four, meaning four dimensions are need for exact representation of the coordinates. Following is an analysis of how much information is lost when reducing the number of dimensions to two. The coordinate matrix $\mathbf{X}$ in two

dimensional space obtained from equation 18 by only using the first two singular values becomes:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 \\ -0.2158 & -2.9149 \\ -0.0500 & -0.4748 \\ -0.1173 & -1.2714 \\ -19.6611 & 0.0408 \end{bmatrix} \tag{37}$$

Recalculating the distances from $\mathbf{X}$ using Euclidean distance, one gets the following distance matrix estimate $\hat{\mathbf{D}}$:

$$\hat{\mathbf{D}} = \begin{bmatrix} 0 & 2.9228 & 0.4774 & 1.2767 & 19.6611 \\ 2.9228 & 0 & 2.4457 & 1.6464 & 19.6566 \\ 0.4774 & 2.4457 & 0 & 0.7994 & 19.6159 \\ 1.2767 & 1.6464 & 0.7994 & 0 & 19.5878 \\ 19.6611 & 19.6566 & 19.6159 & 19.5878 & 0 \end{bmatrix} \tag{38}$$

Calculating the MSE for between the original distance matrix $\mathbf{D}$ and its estimate $\hat{\mathbf{D}}$:

$$\begin{aligned} MSE(\mathbf{D}, \hat{\mathbf{D}}) &= \frac{1}{n} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} (\hat{d}_{ij} - d_{ij})^2 \\ &= \frac{1}{10} \sum_{i=1}^{4} \sum_{j=i+1}^{5} (\hat{d}_{ij} - d_{ij})^2 \\ &= \frac{1}{10}(0.052 + 5.175 + 2.5998 + 0.1367 + 0.1025 + 1.1041 + 0.0121 + 2.7732 \\ &\quad + 0.0197 + 0.2542) \\ &= 1.2229 \end{aligned} \tag{39}$$

Figure 5 shows the above coordinates for the sensors plotted in two-dimensional euclidean space with the circles representing the square root of the above calculated MSE, which can be viewed as the area of uncertainty.

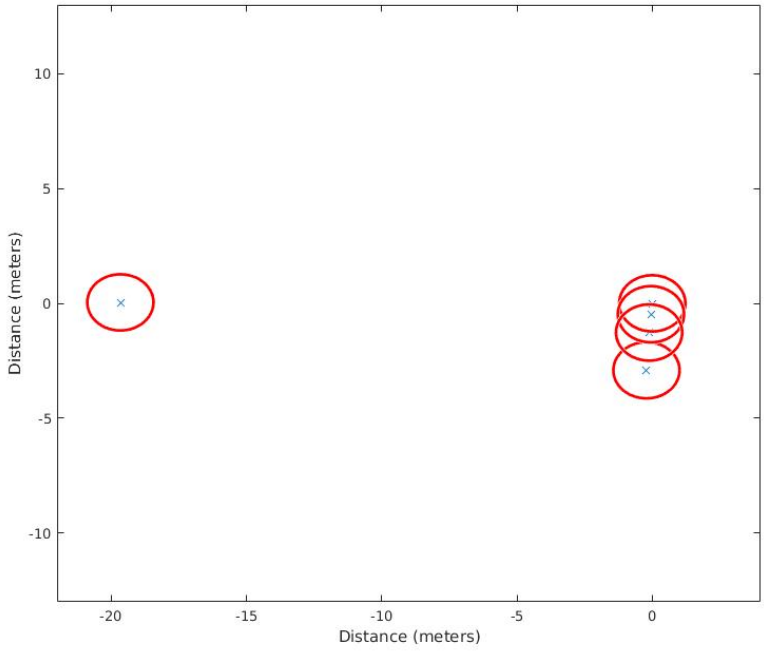

Figure 5: Relative location of sensors plotted in Euclidean space

4 Results

4.1 Training the polynomial regression function

In order to train a polynomial regression function, one needs to have training data available. This was done by creating lots of different alpha values for different scenarios, generating data realizations, calculating reference distances, and using this to train an n'th order polynomial function. The procedure can be seen in figure 6.

The scenarios for data generation were as follows:

- Three and two sensors close to each other
- Five sensors close to each other
- Five sensor far away from each other
- Four sensors close, one far away
- Three sensors close to each other, two far away
- Five sensors medium distance from each other
- Four sensors close to each other
- Four sensor medium distance from each other
- Three sensors close to each other, one far away
- Three sensors close to each other
- Two sensors close, one far away
- Three sensors medium distance from each other

The exact choice for alpha values can be seen in the Matlab files manual_alpha_generation_five_sensors.m, manual_alpha_generation_four_sensors.m., and manual_alpha_generation_three_sensors.m.

Figure 7 shows a plot with the average correlation plotted against its corresponding reference distance. Each 'x' represents the relationship between two sensors, with the different colors representing the different scenarios described above.

Figure 8 shows plots of polynomial functions of various orders obtained from regression analysis on the training data.

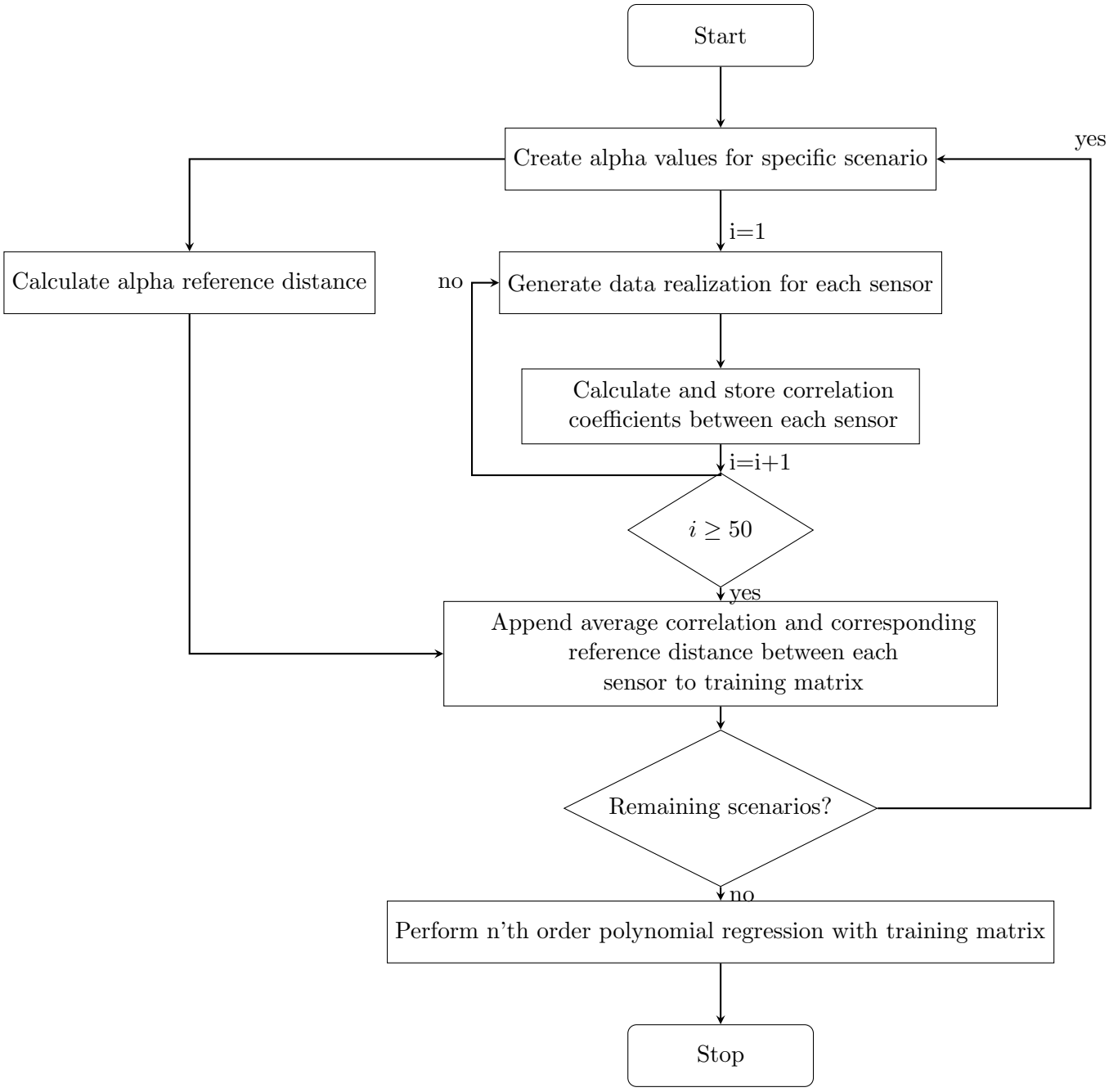

Figure 6: Flowchart for polynomial regression training procedure

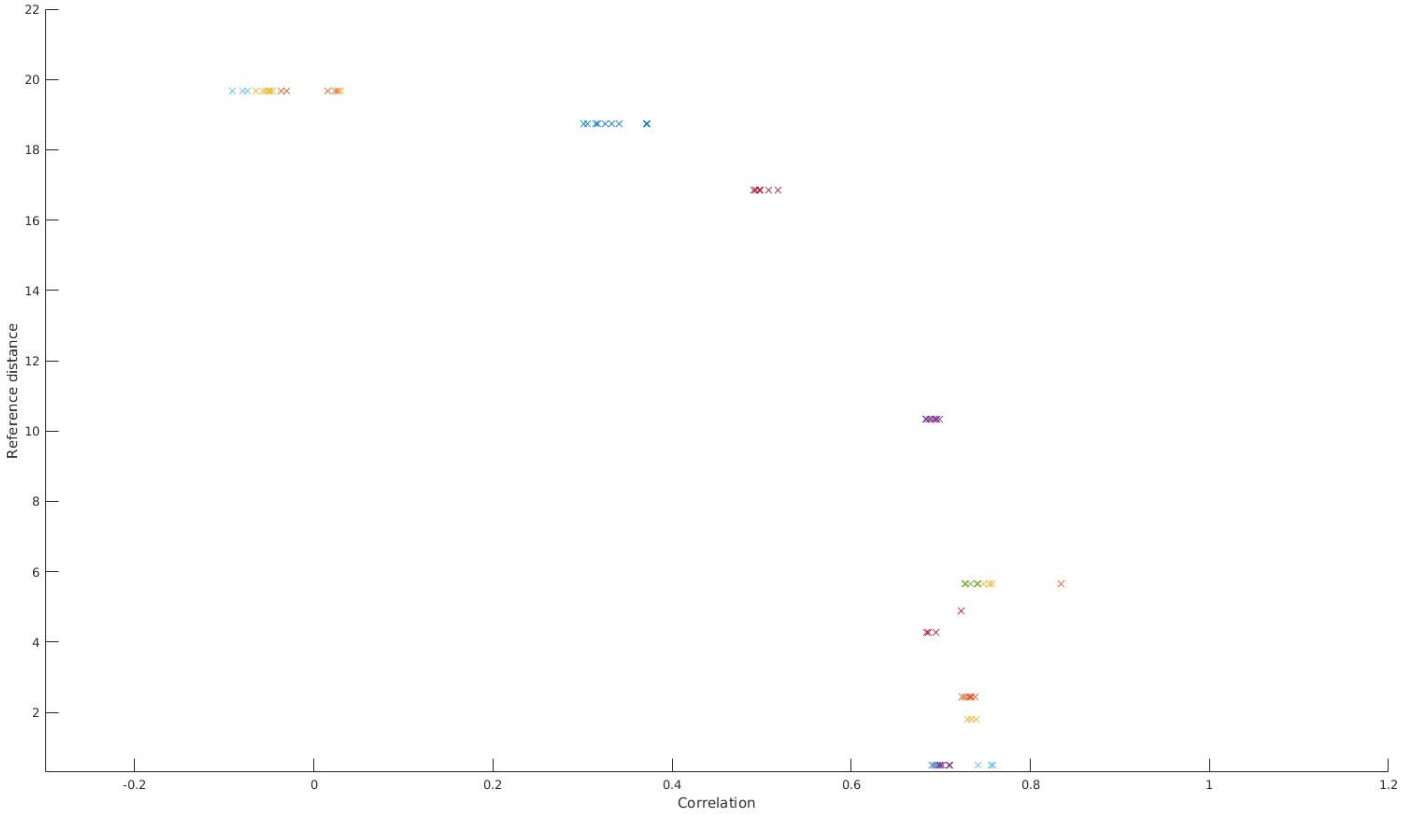

Figure 7: Training data

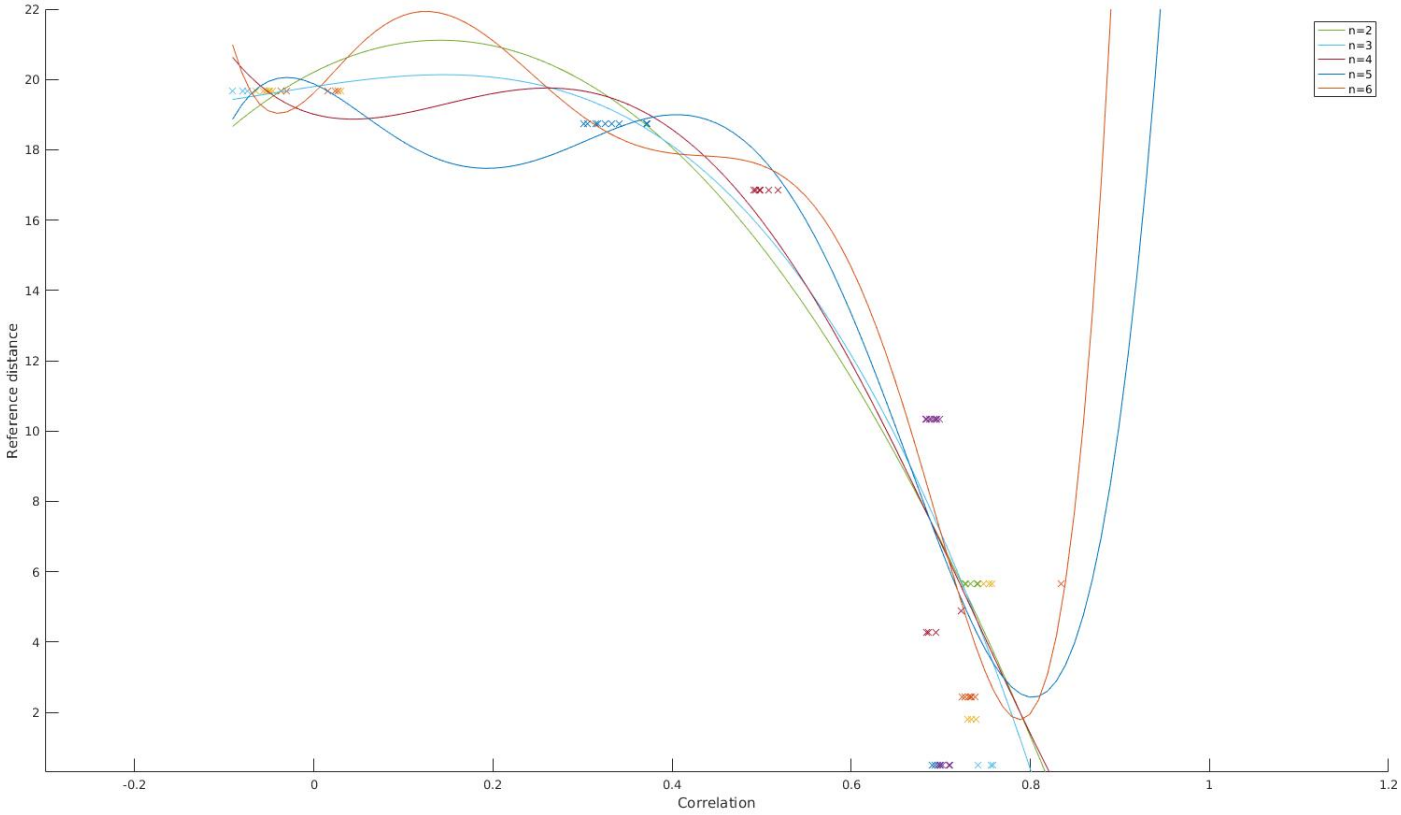

Figure 8: Training data plotted against regression functions of various orders

As can be seen, polynomial functions of orders five and six are clearly not good fits as they start drastically increasing towards the higher end of the correlation scale. Order four also shows undesirable behaviour as it quite significantly increases in the correlation range of 0.2 to 0.4. Out of the remaining functions of order two and three, order three seems to be the best fit as it stays relatively flat before it starts decreasing from a correlation value of approximately 0.4. Therefore, order three seems to be the best fit.

After running a few more training iterations and analyzing the obtained functions, the third order function in figure 9 seemed to have the desirable properties.

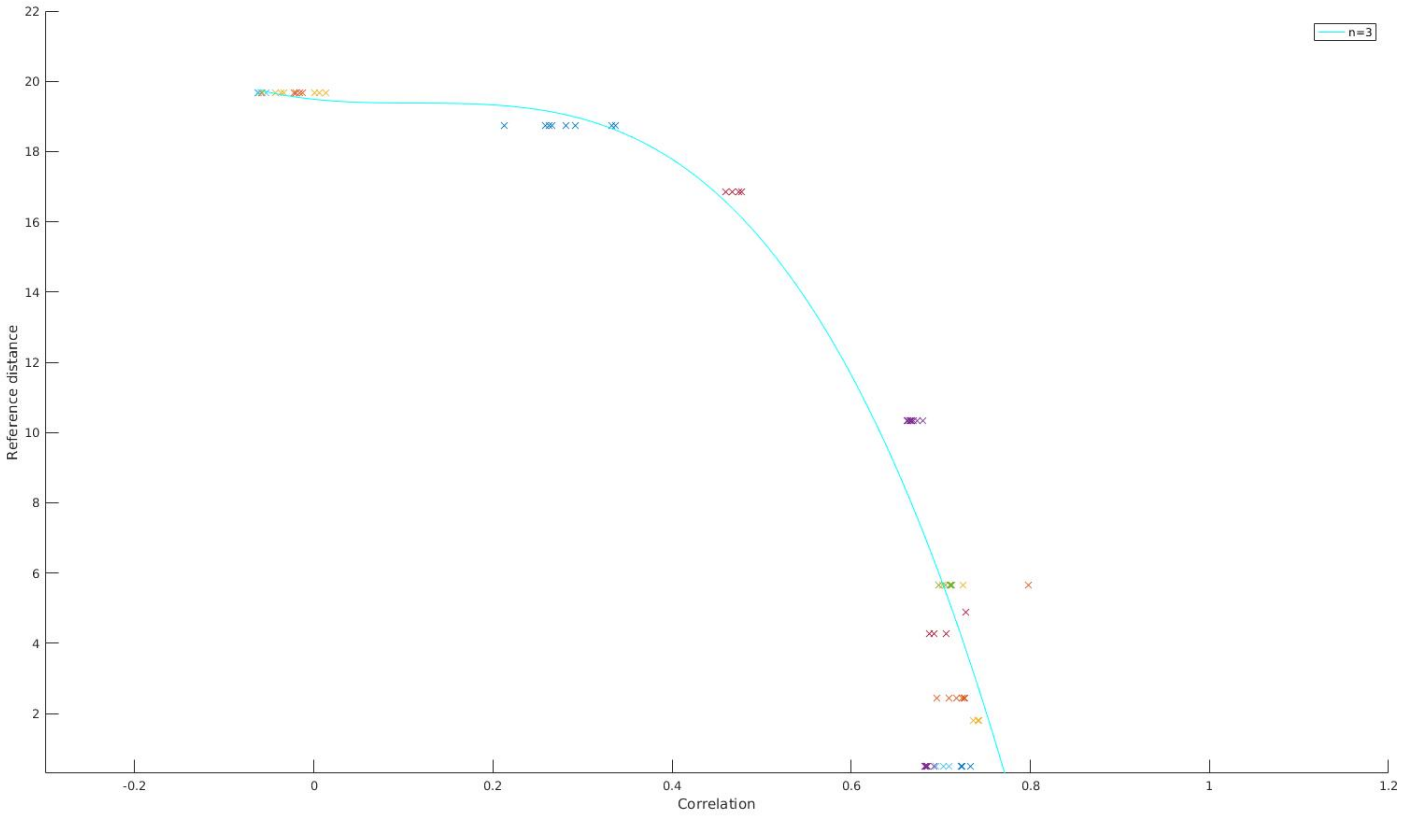

Figure 9: Training data plotted against third order regression function

With function weights $w_0 = 19.5695, w_1 = -1.5639, w_2 = 18.3944$, and $w_3 = -62.7428$, the function becomes:

$$d_{ij}(c_{ij}) = 19.5695 - 1.5639c_{ij} + 18.3944c_{ij}^2 - 62.7428c_{ij}^3 \qquad c_{ij} \in [-1, 1] \quad (40)$$

4.2 Testing and evaluating the polynomial regression function

In order to test the performance of the trained regression function in equation 40, a separate test set with other scenarios had to be made. These can be seen in the Matlab files manual_alpha_gen_three_sensors_testing.m, manual_alpha_gen_four_sensors_testing.m, and manual_alpha_gen_five_sensors_testing.m. Since having a negative distance between sensors does not make sense, which is the case with the trained function obtained in 4.1 (which is an obvious area of improvement which will be a topic in the discussion), equation 40 was slightly modified to take the maximum value of equation 40 and zero. The modified function thus becomes.

$$d_{ij}\prime(c_{ij}) = \max(d_{ij}(c_{ij}), 0) \qquad c_{ij} \in [-1, 1] \quad (41)$$

The results, again with averaging over 50 realizations and using the average correlation as input to the modified distance function, can be seen in figure 10.

The vertical lines represent the difference between the computed reference distances from the the alpha values and the distances calculated from equation 41. The exact values can be seen in the attached file function_eval_data.csv. The deviation between the two, if modeled as the mean squared error with d_{reg} being the distance obtained from the regression function, becomes:

$$\begin{aligned} MSE(d_{ref}, d_{reg}) &= \frac{1}{N}\sum_{n=1}^{N}(d_{n,ref} - d_{n,reg})^2 \\ &= \frac{1}{38}\sum_{n=1}^{38}(d_{n,ref} - d_{n,reg})^2 \\ &= 14.4126 \end{aligned} \quad (42)$$

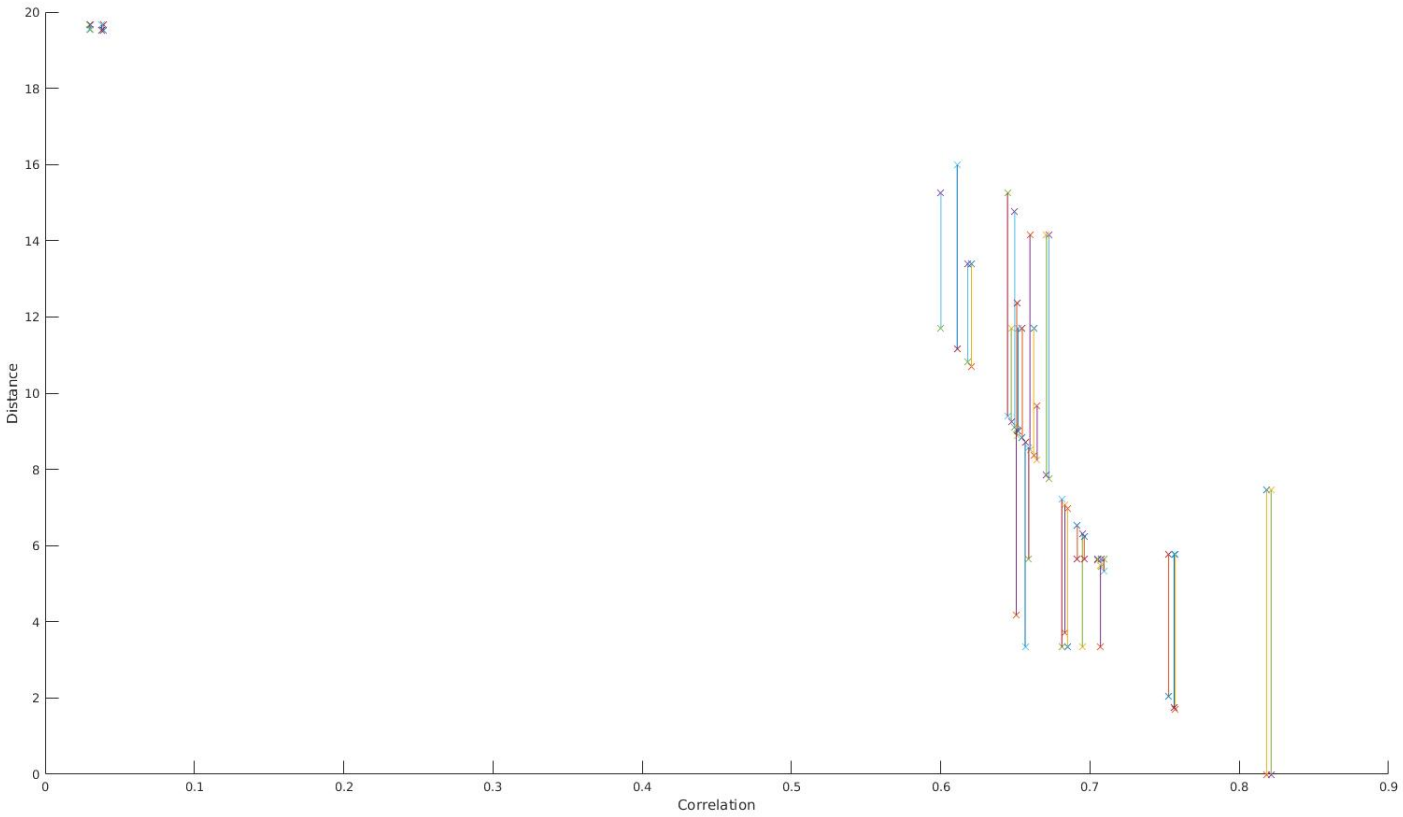

Figure 10: Correlation plotted against reference distance and distance from regression function

With the root mean square error (RMSE) being equal to $\sqrt{MSE} = \sqrt{14.4126} = 3.7964$.

4.3 Plotting in 2-dimensional Euclidean space

Following the procedure described in 3.6, it is quite straightforward convert a distance matrix to a 2-dimensional of the sensors. Following is an example from the scenarios described in the file manual_alpha_gen_five_sensors_testing.m (the code for producing the plots can be seen in generate_2d_plot.m).

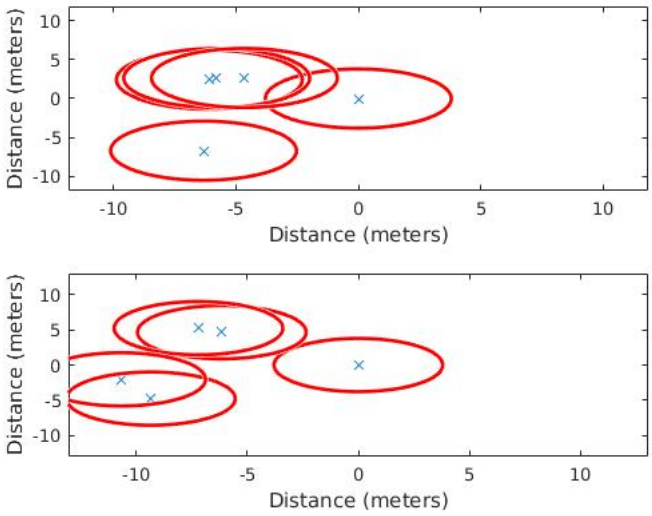

Figure 11: Sensor distance topology for five sensors randomly spread out

With the radius of the red circles being the RMSE calculated from the MSE in equation 42. Another example is shown below, this time with four sensors, with the scenarios described in the file manual_alpha_gen_four_sensors_testing.m.

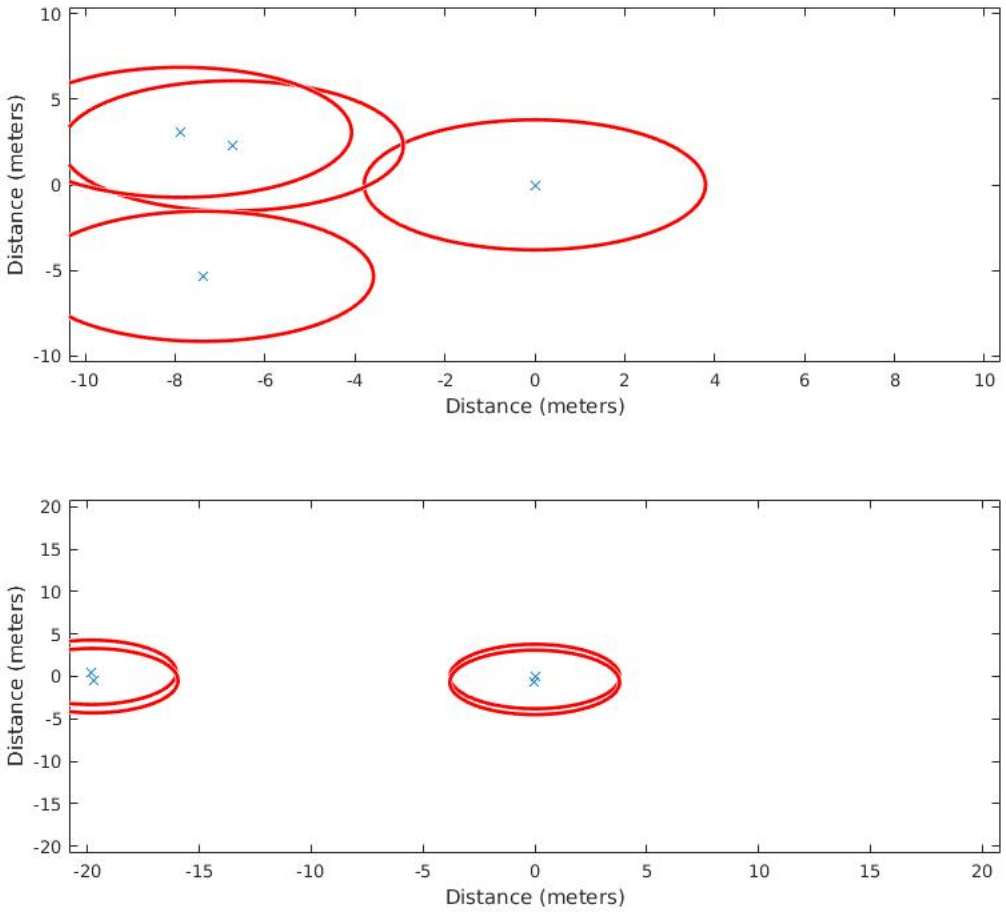

Figure 12: Sensor distance topology for four sensors, top graph illustrating four sensors spread out, bottom graph illustrating two and two sensors grouped together

5 Discussion

The steps can be summarized into the following four parts:

1. Generate synthetic data
2. Compute correlation between synthetic data
3. Perform regression analysis to obtain a function to relate the correlation to the distance between sensors
4. Map the sensors as a distance topology in 2-dimensional Euclidean space

Furthermore, the obtained regression function was tested with a small, separate data set.

Generating synthetic data by using the method described in 3.3 seemed to work fairly well, given that one chooses appropriate alpha values for the correct scenarios. One of the bigger challenges was finding a proper function to relate the alpha values to a reference distance. The method for this chosen in combining equation 28 and equation 25 was a result of making an educated guess on the relationship between the correlation and the alpha values. Given more time, the model for this relationship is an obvious area of improvement, as it directly affects the rest of the procedure. The regression function shown in figure 9 does generalize the data to a certain extent, but also has obvious weaknesses. The correlation values are defined in the interval of $c_{ij} \in [-1 \quad 1]$ (although values significantly lower than zero are highly unlikely), and the distances between sensors only make sense for non-negative values. Yet, for correlation values higher than approximately 0.78, the distances, according to the function, will be negative. Therefore, the adjusted function in equation 41 had to be be used. This was done instead of trying to improve other parts in the process as the latter would have been too time consuming in the later stages of the project.

The results from the test set shown in figure 10 leaves a lot to be desired. An RMSE value of 3.7964 is a bit misleading since the MSE distance components in the lower end of the correlation scale are very small due to the function being essentially flat, whereas the MSE distance components in the higher end of the correlation scale are several magnitudes bigger due to the steepness of the curve where a slight correlation change will cause a large shift in distance. There are, however, certain scenarios that gets modeled quite well. Consider, for instance, the bottom scenario in figure 12,

where the data is initially created to model four sensors with two and two being close to each other. The results produced by both equation 41 and by the distance topology mapping for this case seems to be quite accurate. Thus, for certain scenarios, the model does quite well. Another point to be made regarding the regression function obtained in equation 40, as any data scientist will tell you, is that the size of both the training set and test set is too low. Generating appropriate data to cover many scenarios is, however, a time-consuming process. Given more time, this would have been another area to focus on.

6 Conclusion and future work

The aim of the project was to create a simple representation of how features between raw sensor data can be used to find relative distance between sensors. Due to the work being purely theoretical due to lack of actual sensor data, every step of the process had to be modeled by using methods from fields such as signal processing, statistics, machine learning, and linear algebra. As this was quite a constrained project both in terms of time and credits, the focus was made on creating an initially basic model, with potential of expanding the model later. Such simplifications included: assuming all sensors produce the same type of data, only generating 100 samples at a time for each sensor, only extracting one type of comparison between sensors (correlation), and not taking into account possible obstacles (such as walls) between sensors.

There's lots of future work to be done in order to improve the model. The most obvious step is to use real, instead of synthetic, sensor data. Hopefully real data will have some resemblance to the synthetic data such that the model only needs to be altered and not thrown out completely. Secondly, more than one type of sensor data can be incorporated to try and make more accurate predictions. Thirdly, while a polynomial regression approach seemed to be the most logical in this scenario, it is definitely not certain that it is the method that will provide the best results. Other types of regression models, such as ridge regression, may be worth trying. Also, remodeling the input-output relationship as a classification problem solved by training a neural network where each output neuron represents a distance range can also be an interesting approach.

As closing words to this project thesis: Despite the weaknesses described, hopefully this project can serve as a basis to build on in terms of further modeling the relationship between raw sensor data and relative distances in groups of sensors.

Bibliography

[1] Introduction to correlation and regression analysis. University lecture, 2013. URL http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Multivariable/BS704_Multivariable5.html.

[2] Internet of things, 2016. URL https://en.oxforddictionaries.com/definition/Internet_of_things.

[3] Polynomial regression, 2016. URL http://docs.rapidminer.com/studio/operators/modeling/predictive/functions/polynomial_regression.html.

[4] G. M. Crippen. Rapid calculation of coordinates from distance matrices. *Journal Of Computational Physics*, 26:449–452, 1978. doi: http://www.sciencedirect.com/science/article/pii/0021999178900815.

[5] Dave Evans. The internet of things: How the next evolution of the internet is changing everything, 2011. URL https://en.oxforddictionaries.com/definition/Internet_of_things.

[6] John Greenough. How the 'internet of things' will impact consumers, 2016. URL http://nordic.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10?r=US&IR=T.

[7] Khanh Ngo Tan Vu Ha Nguyen Thi. A positioning algorithm in the internet of things, 2016.

[8] Karin Avnit Francois Septier Laurent Clevier Ido Nevat, Gareth W. Peters. Location of things: Geospatial tagging for iot using time-of-arrival, 2016.

[9] Sun Yuhang. Spectrum sensing in cognitive radio systems using energy

detection. Bachelor thesis, 2011. URL http://www.diva-portal.org/smash/get/diva2:451175/fulltext02.