Author
Arpan Govindraju (a.govindaraju@student.tudelft.nl)

Title
TODO TITLE

MSc presentation
17th July 2016

Graduation Committee

TODO GRADUATION COMMITTEE	Delft University of Technology
TODO GRADUATION COMMITTEE	Delft University of Technology

Abstract

TODO ABSTRACT

Preface

Sesnor localization is one of the

I would like to thank my supervisor Ashish Pandharipande(Philips Research), David(Philips Research) , Venkatesh Prasad(TU Delft) for all the support and guidance provided during the course of this 9 months. I would also like to thank my parents for all the support and love they have provided me all these years. A special thanks to all the people of FO 52 past and the present who made the stay in Eindhoven so much more pleasant.

Arpan Goindaraju

Delft, The Netherlands
17th July 2016

Contents

Chapter 1

Introduction

1.1 Motivation

The European Union (EU) has pledged to cut the consumption of primary energy by 20%, by the year 2020. It is estimated that buildings consume 40% of the energy produced[1]. This has resulted in an increase in the demand to reduce the energy consumption of buildings. To reduce the consumption of energy, building automation systems (BAS) are being widely employed. BAS are computer-based systems that help to manage, control and monitor building technical services (HVAC, lighting etc.) and the energy consumption of devices used by the building. BAS deploy huge amount of sensors, which provide inputs to perform efficient control of various services like HVAC and lighting. BAS brings with it various benefits, at the same time, offers numerous challenges too. For BAS sensor measurements alone are not sufficient to understand the condition of the facilities, unless combined with additional contextual information. One of the important meta-data required is that of the physical location of the sensor as discussed in [1]. As the size and distribution of the deployed sensors are high, it is highly cumbersome and error prone to manually maintain the meta-data about the sensor placement. Apart from being error prone, the manual configuration has to be repeated every time there is a change in the meta-data. Change in meta-data can be due to various reasons, such as change in the office setup,replacement and/or relocation of sensors. All these factors result in inaccurate information about the location of the sensors. Without the accurate information of the location of the sensors, interpreting the data collected from the sensor is difficult and also can be misleading. This could result in the decrease in the effectiveness of the deployed BAS systems. Hence there is a need to develop methods to accurately determine the location of the sensors within the building.

[1]according to value published at https://ec.europa.eu/energy/en/topics/energy-efficiency/buildings

1.2 Problem Statement

Several studies have been carried out to infer the sensor location from sensor data. Most of the methods developed so far have identified ways to cluster the sensors that are located within the same room; however the methods do not identify these rooms within the building [2]. Therefore the research question that is being tackled in this work is:

How to Automatically determine the sensor location utilizing the data from the sensors and the information about the grid (coordinates of the vertices constituting the grid)

1.3 Contribution of the thesis

This work makes the following contribution:

- Using energy of the sensor data stream as a feature to distinguish neighboring sensors from non neighboring sensors for binary occupancy sensor data.
- Reducing the problem of determination of sensor locations on the grid to a problem of graph matching.

1.4 Outline of thesis

Rest of the thesis is organized as below in Chapter 2 we give a brief overview of related work. In chapter 3 we present the method that has been developed. In the chapter 4 we describe our testbed setup. Next in chapter 5 we present the results obtained by applying the method that we have developed on actual sensor data obtained from 2 different testbeds. In the end in chapter 6 we conclude the work done and discuss the future work.

Chapter 2

Literature Review

There has been an extensive body of research that aims at obtaining the location information of the sensors in WSN's. All the approach that has been taken till now has been categorized by the Wang et al. in [3] to fall into one of the following categories:

- Proximity based localization
- Range based localization
- Angle based localization

In proximity based localization, WSN is represented as a graph G(V,E). Location of the subset of the nodes, $H \subset G$ is known. The goal is to estimate the location of the remaining V-H nodes relative to the position of the H nodes. Range based localization make use of the ranging techniques using RSSI , time of arrival and time difference of arrival of the signals. The distance is computed using the ranging technique but knowing the distance between the nodes computing the location of the nodes is non trivial. Angle based localization adds the information of angle between the two sensor nodes. The information of distance and angle is used to determine the location of the sensors.
Apart from the techniques mentioned above, recently there have been few works which look at the data measured by the sensors to obtain the information about the location of the sensor. As our approach towards determining the sensor location uses sensor data as input, in this chapter we provide a brief overview about the works which take the same approach.
Various approaches have been taken to automate the process of determining the sensors location using the data from the sensors in buildings using different data analytics and signal processing tools. Hong et al.[4] apply empirical mode decomposition to 15 sensors in 5 rooms to cluster sensors which belong to the same room by analyzing the correlation coefficients of the intrinsic mode functions. They characterize the correlation coefficient

distribution of sensors that are located in the same room and different rooms. They were able to show that there exists a correlation boundary analogous to the physical boundary which can be discovered empirically. In [2] Akinci et al. propose a feature: energy content in HVAC delivered air, which can be derived from HVAC system sensors which could lead to identification of the space in which the sensors are located . They combine sensor measurements and building characteristics(floor area) for the identification of the space in which the sensors are present. In [5] Koc et al. propose a method to automatically identify the zone temperature and discharge temperature sensors that are closest to each other by using statistical method on the collected raw data. They explore whether linear correlation or a statistical dependency measure(distance correlation) are better suited to infer spatial relationship between the sensors. They carry out there analysis on three different testbeds. They also investigate the effects of distance between sensors and measurement periods on the matching results. In the end the authors conclude that linear correlation coefficient provides better matching results compared to distance correlation. The authors also conclude that as the distance between the sensors increase, the data size needed to infer spatial relationships also increase.
Lu et al.[6] describe a method to generate representative floor plans for a house. Their method clusters sensors to room and assigns connectivity based on simultaneous firing of the sensors placed on the door and window jambs. The algorithm gives a small set of possible maps from which the user has to choose the right map. Method requires special placement of the sensors . The authors were able to calculate the floor map of 3 out of the 4 houses they evaluated. Ellis et al.[7] proposed an algorithm to compute the room connectivity using PIR and light sensor data. They compute room connectivity based on the artificial light spill over between rooms; occupancy detection due to movements between rooms; and fusion of the two. They calculate the transition matrix for light sensor and occupancy sensor. Fuse both the data together to compute the connectivity graph. Here the authors have considered a situation where there is only one PIR and light sensor per room. Müller et al. define sensor topology as a graph with directed and weighted edges. All pairs of consecutive sensors triggers are are interpreted as as a user walking from the former to the latter sensing region indicated in the sensor graph by a edge from the former to the latter. Every time a consecutive edge triggers are observed the weight of the edge between the sensors are incremented. They define a method to filter out erroneous edges. In [9] Marinakis et al. obtain the sensor network topology using Monte Carlo Expectation Maximization. They assign activity to people present in the space to obtain the graph topology. The algorithm requires number of people present in the space as the input. They demonstrate the effectiveness of the algorithm using various simulated data.

Chapter 3

Methodology

This chapter explains the method developed to locate the sensors in a sensor grid, using the information about the locations of the vertices of the grid and data obtained from the sensors. Before we explain the method we introduce the definitions of the terms that we use in our thesis.

Definition 3.0.1. Neighboring sensors: Two sensors are said to be neighbors if they have overlapping field of view.

Definition 3.0.2. Grid Adjacency Matrix: We represent the sensor grid in the form of an adjacency matrix. Two vertices of the grid i and j are adjacent if the distance between them ($d_{i,j}$) is less than twice the radius(r) of the sensors field of view.

$$GAM_{i,j} = \begin{cases} 1, & \text{if } d_{i,j} < \text{ 2r } \forall\ i \neq j \\ 0, & \text{otherwise} \end{cases}$$

$d_{i,j}$ is the euclidean distance between the vertex i(x_i,y_i) and j(y_j,y_j).

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (x_j - y_j)^2}$$

From the definitions 3.0.2 and 3.0.1 we can say that neighboring sensors will always be placed on neighboring vertices.

3.1 Feature

Consider a m $\times$ n grid as shown in the figure 3.1. There are $N = (m \times n)!$ ways in which the sensors can be uniquely placed on the grid. Out of these N ways we have to identify the actual arrangement of the sensors in the grid.

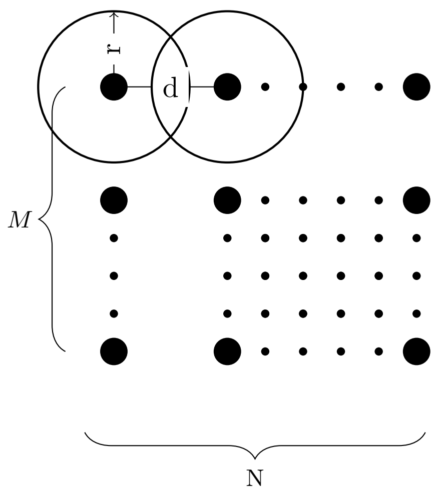


Figure 3.1: A M × N grid having a sensor with field of view r and distance between the nodes being d.

Along with the information of the vertex locations we also have the data from the sensors available. Using the sensor data along with information available about the grid we develop a method to identify the sensor lcoation of the grid
As neighboring sensors have an overlapping field of view, they observe the same events and thus are highly correlated. On the raw signals of the PIR data stream we use a sliding window with 50% overlap between the consecutive windows and compute the energy feature using the equation 3.1

$$E_s = \sum_{n=0}^{k} |x(n)|^2 \tag{3.1}$$

For a PIR binary data computing energy reduces to counting the number of triggers observed within the window.

With the newly computed energy data stream, we compute the cross correlation between all the sensors using equation 3.2

$$r(x,y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{3.2}$$

X and Y are data streams.
$\overline{X}$ and $\overline{Y}$ is the mean value of X and Y respectively.
n is the number of samples.

3.2 Grid Correlation Sum

Using the cross correlation values between the sensors, we define correlation matrix R between the n sensor nodes as shown in the equation 3.3 .

$$R = \begin{bmatrix} r(1,1) & r(1,2) & \dots & r(1,n) \\ r(2,1) & r(2,2) & \dots & r(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ r(n,1) & r(n,2) & \dots & r(n,n) \end{bmatrix} \tag{3.3}$$

r(α,β) represents correlation value between the sensor α and β.

Using correlation matrix(R) and grid adjacency matrix(GAM) we define a quantity called $GridCorrelationSum(GCS)$ as given in the equation 3.4. GCS represents the sum of correlation value of the sensor pair residing on neighboring vertices of the grid. As the matrices are symmetrical along the diagonals we only consider the elements of the upper triangle of the matrices excluding the diagonal elements.

$$\text{GCS} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{R}(\phi(i), \phi(j)) \times \ \text{GAM(i,j)} \tag{3.4}$$

i, j represent i^{th} and j^{th} vertex on the grid.
ϕ is a mapping functions which gives the sensor present at vertex i.
$R(\alpha, \beta)$: correlation coefficient between the sensor α and β.
GAM: Adjacency matrix of the grid as per definition 3.0.2.

We use GCS to identify the correct arrangement out of all the N possible arrangement. If we compute GCS for all the possible arrangements, the arrangement with the maximum GCS will represent the actual arrangement of the sensors on the grid. If two non neighboring sensors are kept on neighboring verticies or vice versa then the correlation value between those two sensors will be low and thus decreasing the GCS. To illustrate this consider a sensor grid as shown in the figure 3.2. It consists of 3 $\times$ 3 grid. GCS for the the grid is :

GCS_1 = r(1,2)+ r(1,4)+...r(2,3)+...r(5,6)+r(6,9)...r(8,9)

Now consider the arrangement shown in figure 3.3, where the position of the sensors 3 and 6 are interchanged

GCS_2 = r(1,2)+r(1,4)+...+**r(2,6)**+...+**r(3,5)+r(3,9)**....r(8,9)

GCS_1 will be greater than GCS_2 as r(2,3) $>$ r(2,6) ,r(5,6)>r(3,5) and r(6,9) $>$ r(3,9) as sensors 2 and 6 , 3 and 5, 3 and 9 are non neighboring

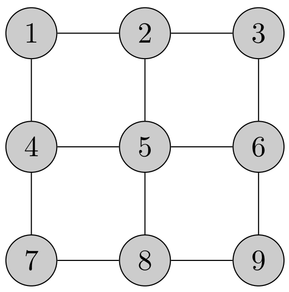


Figure 3.2: Correct arrangement

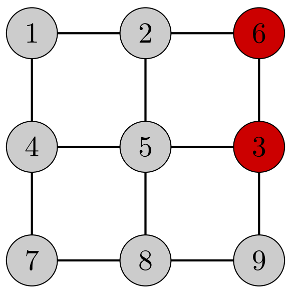


Figure 3.3: Incorrect arrangement

sensor nodes placed on neighboring vertices. Hence we can say that GCS will be maximum for a mapping which maps sensors onto the grid accurately.
When the number of sensors is low we can identify the correct mapping by computing the GCS value for all the possible N mappings. As the number of sensors increase the possible mappings to be checked also increases. Hence we need to find a method to reduce the search space.

3.3 Maximum spanning Tree

Having established a criterion to determine the right arrangement of sensors on the grid from possible N arrangements. The next step is to reduce the search space. For which we compute the maximum spanning tree(MST) for the correlation matrix R. If we consider R as an adjacency matrix for a graph G, then the sensors represents the vertices and the correlation values between the sensors represents the weights of the edges between the corresponding sensors. Now if we compute a maximum spanning tree for the graph, then the tree consists of all the sensor nodes connected to atleast one another node by an edge. The MST chooses an edge for every sensor such that the weight of the edge outgoing from the sensor α to sensor β is the maximum among all the edges starting from sensor α, in graph G. As the weights represent correlation values it implies that the maximum spanning tree connects a node to another with which it has the maximum correlation. As the correlation between the neighboring nodes is maximum we can say that MST connects the neighboring sensors.
To compute the MST we make use of prim's algorithm[10]. As prim's algorithm was originally developed to compute the minimum spanning tree we negate the weights for the correlation matrix and obtain the minimum spanning tree which will represent the maximum spanning tree for the original graph G.
As the sensors are placed on the grid and there exists an edge between the neighboring vertices of the grid. The MST over R represents one of the spanning tree for the grid. As illustrated in the figure 3.4. It is possible that

for a given grid there may be several spanning trees with the same structure. Figure 3.5 shows few of the spanning tree similar to the MST obtained for the correlation matrix. If we compute all the possible spanning trees for the grid which maintains the same structure as that of the maximum spanning tree for the correlation matrix and map each vertex on the MST to the corresponding spanning tree of the grid as shown in the figure 3.6, we obtain several possible mappings of the sensors to grid location.Among the possible mappings we have to choose the correct arrangement. To choose the true arrangement of the sensor on the grid we make use of GCS. The number of arrangements that are obtained from this method is lesser than the N possible arrangements that we had to check previously.
The main reason for decrease in the possible mapping can be explained as follows: When we are computing various mappings between the sensors maximum spanning tree and spanning tree for the grid suppose we assign one of the nodes(S_x) on the spanning tree to one of the nodes on the grid(G_x) then in the next step the node which is connected to S_x can be mapped only to the neighboring nodes of the grid node G_x .

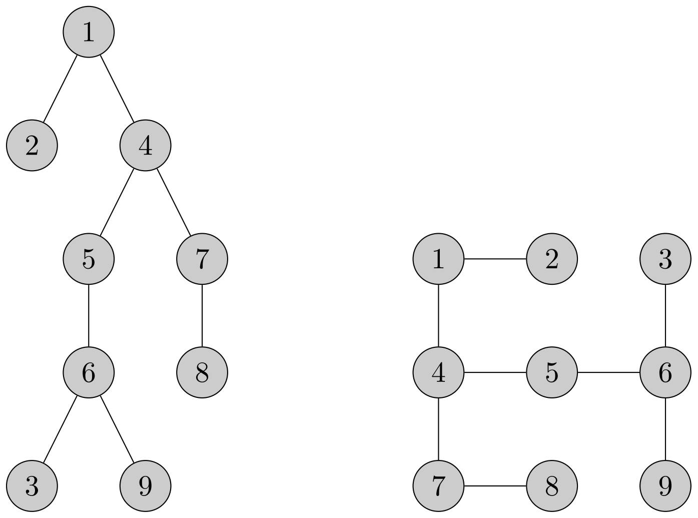


Figure 3.4: A maximum spanning tree incident on the grid

3.4 Graph Matching

In the previous section we obtain a maximum spanning tree from the correlation matrix and saw how this represents a spanning tree for the grid. In this section we describe method to obtain all the possible spanning tree that has the same structure as the MST obtained from the correlation matrix of the sensors. The task of checking if a pattern graph H is contained in a base graph G and obtaining all the mappings from G to H is a standard problem

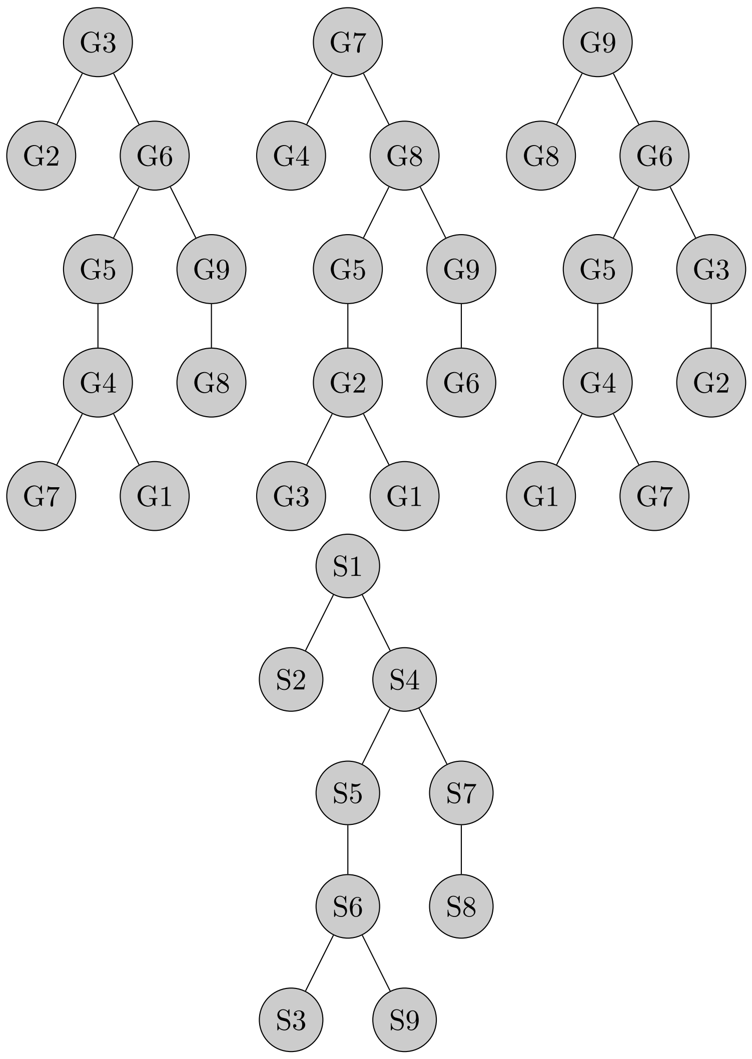


Figure 3.5: Various spanning tree for the grid with similar structure of the MST of the correlation matrix.

of graph matching.

A graph matching process between two graphs $G = (V_G, E_G)$ and H $= (V_H, E_H)$ consists of determining a mapping M which associates nodes of the graph G to H and vice versa. Different constraints can be imposed onto M which results in different mapping types: monomorphism, isomorphism, graph-subgraph isomorphism are the most popular ones. Graph matching is widely used in pattern recognition for comparing the graph representing an object to a model graph, or prototype. Our problem falls under the category of graph monomorphism .

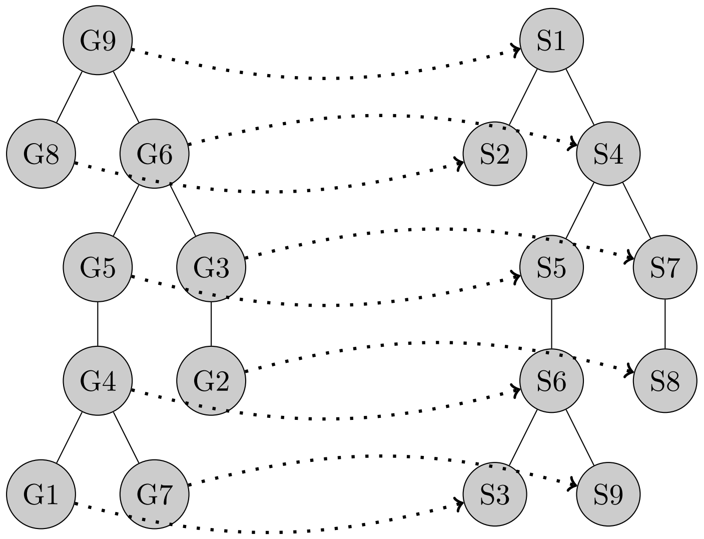


Figure 3.6: Mapping between the grid spanning tree and the MST for the correlation matrix

Two graphs $G = (V_G, E_G)$ and H $=(V_H, E_H)$ are monomorphic if and only if there exists an injective (node) mapping $\phi(V_G) \to V_H$: for which $\forall v, w \in V_G : (v, w) \in E_G \Rightarrow (\phi((v), \phi(w)) \in E_H$.
holds true.

Monomorphism is often confused with subgraph isomorphism. Monomorphism is weaker kind of subgraph Monomorphism.
Two graphs $G = (V_G, E_G)$ and H $=(V_H, E_H)$ are sub graph isomorphic if and only if there exists an injective (node) mapping $\phi(V_G) \to V_H$: for which $\forall v, w \in V_G : (v, w) \in E_G \Leftrightarrow (\phi((v), \phi(w)) \in E_H$.
holds true. The relationship between edges of the graphs are equivalence for subgraph isomorphism, and for Monomorphism relationship is an implication.
A problem of finding all the Monomorphisms of the pattern graph into the base graph is defined as Monomorphism problem. Graph Monomorphism is a NP-Complete problem [11].

3.4.1 Related Work

Graph matching is widely used in pattern recognition and image processing applications. Graph matching also finds its application in the field of biomedical and biological applications. Most of the algorithms for graph matching are based on some form of tree search with backtracking. The basic idea is that the partial mapping (initially empty) is iteratively expanded by adding to it new pairs of matched nodes; the pairs are chosen based on some conditions employed to satisfy the conditions of the matching type.

The key principle is that there will be a partial mapping(initially empty)and is expanded iteratively by adding to it a new pairs of matched node. The matched nodes are chosen based on the certain conditions that depend on the matching type.

The first prominent and one of the most widely used algorithm in the area of graph matching was proposed by Ullmann [12]. Ullmann algorithm addresses the problem of graph isomorphism, sub graph isomorphism , graph monomorphism. Ullmann algorithm is based on depth first search with backtracking. To prune unfruitful matches the author proposes a refinement procedure, which employ conditions based on the knowledge of the adjacent nodes and the degree of the nodes that are being matched. Ghahraman et al. in [13] propose a graph monomorphism algorithm. The authors formulate the graph monomorphism problem as a minimum weight clique problem of weighted nets . The major drawback of the algorithm is that the algorithm uses a $N^2 \times N^2$ matrix to represent netgraph . N representing the number of nodes of the largest graph. As a result of this only small graphs can be dealt with using the algorithm. A more recent algorithm for graph isomorphism, subgraph isomorphism and monomorphism was proposed by Cordella et al. in [14], popular as VF algorithm. The authors define a heuristic that is based on the analysis of sets of nodes adjacent to the ones already considered in the partial mapping. The authors further improved the algorithm in 2001 in their paper [15]. The authors proposed a modification to the algorithm and called it VF2. The authors introduce a method to restore data structure, which enabled them to reduce the memory consumption from $O(N^2)$ to O(N) where N is the number of nodes in the graph, thus enabling the algorithm to work with large graphs. In our work we use the VF2 algorithm to obtain the mappings between the spanning tree and grid graph.

3.4.2 VF2

In this section we give a brief description about VF2 algorithm.

VF2 algorithm is a graph matching algorithm to solve graph isomorphism, monomorphism and subgraph isomorphism problem. VF2 is a depth search first method to iterate through all the nodes and recursive backtracking technique to check for all the possible mappings. A process of matching a base graph G to a pattern graph H consists of determining a mapping M which associate nodes of base graph (G) with nodes of the pattern graph (H) and vice versa, with some constraints. Mapping is expressed as a set of pairs of node (n,m) with n $\in$ G and m $\in$ H. In VF2 algorithm the process of finding the mapping function is described by a State Space Representation (SSR). Each state s of the matching process can be associated to a partial mapping solution M(s), which contains only a subset of M. A transition from current state(s) to the next state(s') represents the addition of a mapping

```
**Procedure** Match(s)
  **INPUT:** an intermediate state s; the initial state $s_0$ has $M(s_0) = \emptyset$
  **OUTPUT:** the mappings between two graphs
  **Match(s)**
  **IF** M(s) covers all the nodes of $G_2$ **THEN**
    **OUTPUT** M(s)
  **ELSE**
    Compute the set P(s) of the pairs of candidates for inclusion in M(s)
    **FOREACH** (n,m) $\in$ P(s)
      **IF** F(s,n,m) **THEN**
        Compute the state s' obtained by adding (n,m) to M(s)
        **CALL** Match(s')
      **ENDIF**
    **ENDFOREACH**
    Restore data structure
  **ENDIF**
```

Figure 3.7: Pseudo code for VF2 algorithm

(n,m) to the state s.

VF2 algorithm introduces a set of rules which helps to prune the number of possible SSR that needs to be checked before obtaining a valid mapping. Figure 3.7 gives a high-lvel description of the VF2 algorithm. There are 3 important functionalities in the algorithm:

- generation of possible mappings(P(s)) .
- checking of the validity of the mapping(F(s,n,m)).
- Restore data structure.

3.4.3 Computation of candidate pair set P(s)

This section explains the method to compute the candidate pair set P(S) for an undirected graph G and H. For every intermediate state s the algorithm computes P(s), a set of possible mapping pairs. For each pair p belonging to P(s) the feasibility of its addition F(s,n,m) is checked : if the check is successful the next state $s' = s \cup p$ and the whole process recursively applies to s'.

To compute P(S) set $T_1(s)$ and $T_2(s)$ are defined for Graph G and H respectively. T_1 and T_2 are the set of nodes in G and H, which are neighbors of the set of nodes included in the partial mapping state M(s) but are not included in M(s). Set P(s) will be made of all the the node pairs(n,m) with n being a node in T_1 with the smallest label and m being a node in T_2 . If T_1 and T_2 sets are empty then the set P(s) be calculated by using the sets of nodes not contained in either G(s), H(s).

3.4.4 Feasibility Rules

Feasibility Rules are used to check the consistency of the partial solution s' obtained by adding nodes n,m and prune the search tree. The functionality of the rules are as explained below.

In vf2 algorithm, those candidates(m,n) are pruned if m is not connected to already matched nodes in G_q (i.e nodes of G_q included in M(s)). Subsequently, the pruning step also removes those node-pairs(m,n) in which n is not connected to the matched nodes in the data Graph G. These pruning rules assume that the query and data graph are connected.

The algorithm also compares the number of neighboring nodes of each n and m that are connected to nodes in M(s) but are not included in M(s). The number of such nodes in the data graph must be greater than or equal to the number of such nodes in the query graph.

Finally the number of neighboring nodes of each of n and m that are not directly connected to nodes in M(s) are compared. The number of such nodes in the data graph must be greater than or equal to the number of such nodes in the query graph.

3.4.5 Restore data structure

In VF2 algorithm all the vectors that are used to store data have the following property: If an element is non-null in a state s, it will remain non-null in all states descending from s. This property, together with the depth-first strategy of the search is, is used to avoid the need to store a different copy of the vectors for each state: when the algorithm backtracks, it restores the previous value of the vectors. Hence the space complexity of VF2 algorithm is of the order O(N). This is illustrated in the figure 3.8.As can be seen at the last state if a variable was created at every state then we will end up with N^2 variables.

3.4.6 Computation Complexity

Computational complexity of the VF2 can be classified as depends on two factors:

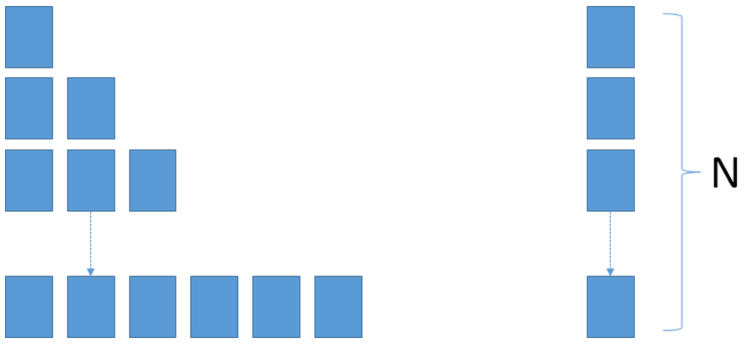


Figure 3.8: Memory cells occupied at each state with and without restore data

- The time required to calculate the feasibility of every SSR's , computation of the node pair mappings.

- Number of SSR visited.

According to the authors the former has a time complexity is $O(N)$. For the latter the authors consider a best case; where only only one of the potential mappings are feasible is $O(N)$ and thus making the total complexity $O(N^2)$. In the worst case, the algorithm has to explore all the states. This can happen when every node is connected to every other node (base graph is a complete graph) and the graph exhibits strong symmetry. The authors show that the complexity in such a condition is $O(N!)$ and thus making the total complexity $O(N!N)$. In our case we show the order of complexity for a 8-neighborhood grid. The worst case for such a grid will arise when at every stage of the search tree the algorithm has to explore every possible branch. Since it's a 8-neighborhood grid all the nodes except the ones at the edges will have 8 neighbors. As one of the neighboring node would be the parent node the node at stage i will have 7nodes to explore and stage N there will be 7^n possibilities to explore . The total number of states to explore will be given by the sum:

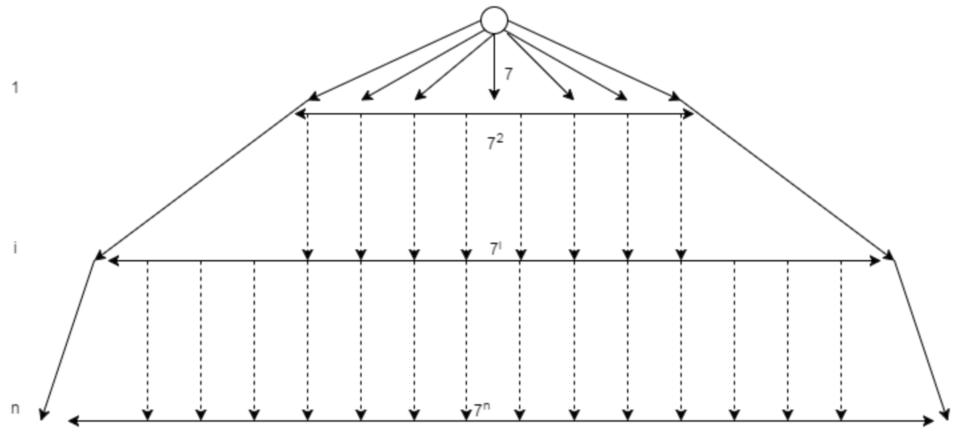


Figure 3.9: All the visited states in case of a 8 neighborhood grid with n nodes.

$$1 + 7 + 7^2 + ... + 7^i + ...7^n$$

This represents a geometric progression with a= 1, r =7.

Sum of a geometric progression is given by:

$$S_n = \frac{a(r^n - 1)}{r - 1}$$

$$S_n = \frac{7^{n+1} - 1}{7 - 1}$$

When n is large

$$S_n \approx 7^N$$

Thus computational complexity is $O(7^N N)$

3.5 Determination of sensor placement

After we obtain the various mappings. We need to decide which out of the various mappings gives the actual placement of the sensors on the grid. To identify the correct mapping we compute GCS as explained in section 3.2. We calculate the grid correlation sum for all the mappings obtained and the mappings which gives the maximum sum will be the actual placement of the sensors on the grid.

3.5.1 Limitations

Our method can identify the location of the sensors upto rotational symmetry. As the GCS remains the same for a sensor arrangement which is rotationally symmetrically to the original sensor arrangements. Consider the two arrangements that are shown in the figure 3.10 and 3.11. Therefore when we pick the mappings which has the highest GCS we get multiple mappings. The multiple mappings correspond to the arrangement which are all rotationally symmetric to each other.

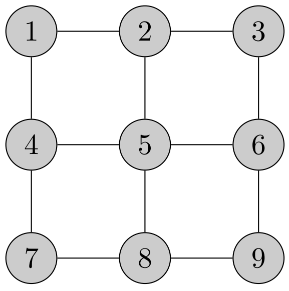


Figure 3.10: Actual arrangement

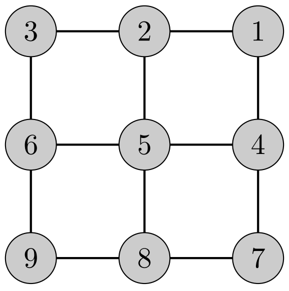


Figure 3.11: Sensors arrangement rotationally symmetric to the actual arrangement

3.5.2 Special case when the grid is $n \times 2$ or $2 \times n$

Consider a sensor $2 \times n$ sensor grid with diagonally opposite nodes connected as shown in the figure 3.12. These grid structure posses a special case of symmetry. As shown in the figure 3.13 the nodes 1 and 2 are interchanged; it is as if the whole structure was rotated with edge (1,2) fixed. The diagonally opposite edges (1,3) in figure 3.12 becomes non diagonal edges in the figure 3.13. To overcome this error we use weighted grid adjacency matrix instead of a binary grid adjacency matrix. We compute $WeightedGridAdjacecnyMatrix(WGMA)$ as follows: First we normalize the distance between all the sensor nodes which has an edge between them in the GAM by dividing all the distances between the sensor nodes by the minimum distance(d_{min}) that exists between the nodes in the entire grid. Then we take the reciprocal of the normalized distance to obtain the WGAM. All the members of the WGAM value lies between the 0 and 1. 1 signifying the 2 sensors are close to each other and 0 signifying that the 2 sensors are far away from each other.

$$WGAM_{i,j} = \begin{cases} \frac{d_{min}}{d_{i,j}}, & \text{if } GAM_{i,j} = 1 \\ 0, & \text{otherwise} \end{cases}$$

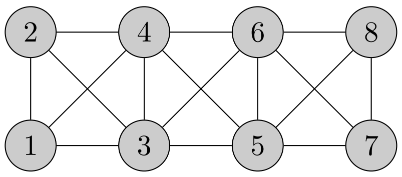


Figure 3.12: Actual arrangement of 2×4 grid.

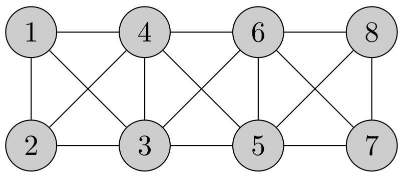


Figure 3.13: Incorrect arrangement of 2×4 grid with the same GCS values.

We use $WGAM$ instead of GAM to compute GCS in equation 3.4. The reasoning behind the use of WGAM matrix instead of the GAM is that the correlation values for the sensor which are closer is higher than the one compared to sensors which are further apart within the sensors overlapping field of view. By assigning weights to the edges in this manner, more importance is given to the edges between the sensors that are closer to the each other. Doing so the sensors which are close by contribute more to the GCS . Now if we consider arrangement as shown in the figure **??** and 3.13 though there GCS remain the same with GAM, GCS won't remain the same when $WGAM$ is used. The contribution of the correlation between the sensors placed on the non diagonal edges are more compared to the contribution made by the diagonal edges. Therefore if we place the non diagonally located sensors on the diagonals of the grid GCS decreases. Thus by using a weighted grid adjacency matrix we are able to obtain the mapping of the sensors onto the grid upto rotational symmetry.

3.6 Summary

In this chapter we describe a method to locate the sensor on the grid making use of sensor data and grid vertices information upto rotational symmetry. In figure 3.14 we summaries the steps involved in the form of a flowchart.

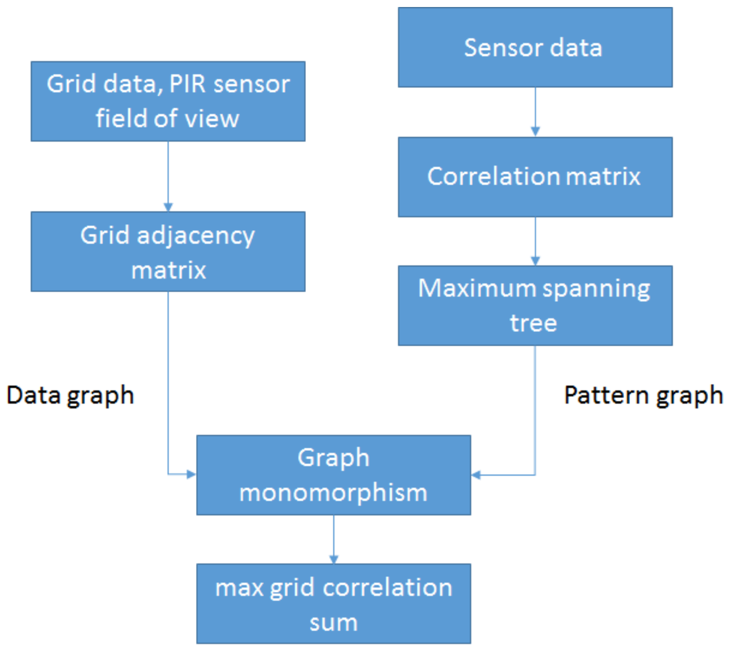


Figure 3.14: Flowchart for the method developed

Chapter 4

Test Bed

We setup our testbed in a 8m X 6m office space. Sensors are mounted on the ceiling which is at a height of 3m. The sensors are placed 2.5m apart in the horizontal direction and 2.2m apart in the vertical direction. We used PIR sensor EKMB1101112 from Panasonic. The output of the sensor is binary. With 1 indicating occupancy and 0 if the region is unoccupied. We use eZ430-RF2500 toolkit which consists of MSP430 micro-controller and CC2500 multi channel RF transceiver which is designed for low power wireless applications. The testbed uses simpliciTI a a TI proprietary low-power RF network protocol. To avoid collision we employ CSMA/CA. To decrease the packet loss we enable acknowledgment. An acknowledgment is sent from the AP if the packets are received. If no acknowledgment is received by a node, the node retransmits its data a maximum of 6 times until an acknowledgment is received from the AP. In the figure 4.2 we can see that the there was a decrease in the packet loss per node. The maximum packet loss before enabling acknowledgment was 23%, after acknowledgment was enabled maximum packet loss for a node was 2%.

We place 8 sensors in the room as shown in the figure 4.1. All the 8 nodes communicate directly to a centrally located access point which is connected to a laptop, which stores the data. We sample the PIR sensor

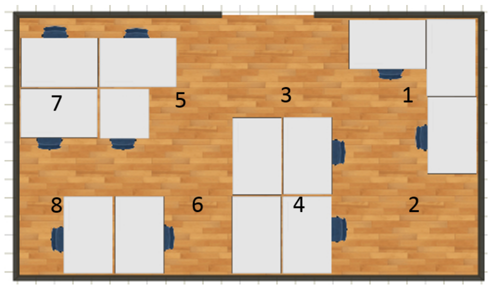
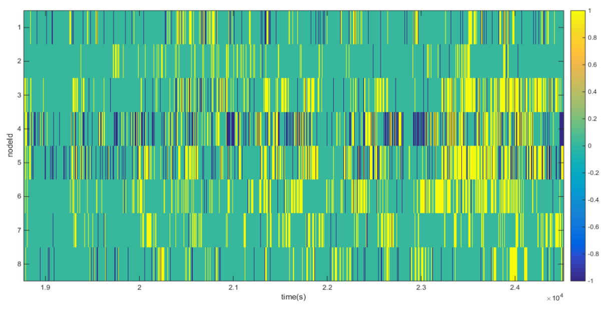
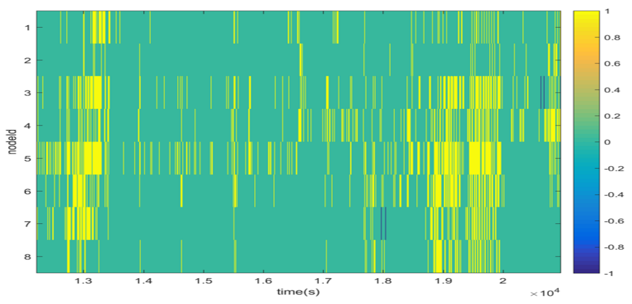


Figure 4.1: Room layout of the testbed.



(a) Data before ack was enabled



(b) Data after ack was enabled

Figure 4.2: Data from the test bed. - 1 indicates the lost data , 1 indicates occupancy, 0 indicates unoccupied space

0	1	2	3	4	5	6	7	8	9	10	11	12	13
Mac id	packet number	start time				end time				data			

Figure 4.3: Packet Structure

every 100ms. All the nodes have a local clock running on them. With 1 clock tick corresponding to 1ms. All the local clocks are synchronized to one global clock.

Since the data is binary we combine 32 samples and transmit the packet every 3.2s. Data packet is an array of 8 bit. The packet structure is as shown in the figure 4.3

The transmitted packet is an array of 1 byte. Consisting of 14 bytes of payload. The payload structure is as shown in the figure 4.3

- Mac Id - unique identifier for each node.
- Packet number- Keeps track of the packets that are being transmitted from a node. It helps to identify the lost packets as well as repeated packets.

- Start time - The time stamp corresponding to the first PIR sample of the PIR sensor in the packet.
- End time - The time stamp corresponding to the last sample of the PIR sensor in the packet.
- Data - 32 samples of the PIR sensor.

Data was collected for a span of 2 months. During the course of the evaluation the occupants were notified of data gathering, but were never instructed to behave in any particular way or was there any constrained applied on the activity or the number of people in the room.

Chapter 5

Results and Discussion

In order to verify the proposed method we test our method with two datasets. First we apply our method to the data collected from the test bed as described in the chapter 4 . Next we make use of the data released by the WSU smart home project[1] [16].

5.1 HTC34 testbed

As described earlier the testbed consists of a 8 sensors arranged in a 4×2 grid. The sensor has a filed of view of radius 2.6m the calculation is explained in appendix **??**. The overlap between the field of view of sensors placed on the grid is as shown in the figure 5.1 With the help of the co-ordinates of the sensors we obtain the adjacency matrix of the grid as shown in the figure **??**.

5.1.1 Determination of Window length

The first step in evaluation of our method is to obtain the energy stream from the raw data. To obtain the energy stream we use a moving window with 50% overlap. We determine this window length empirically. As we already know the neighbors of every sensor We determine the window length empirically. We calculate

5.2 Convergence Time

An important aspect to know in an data driven method is to know how much of data is required in order to get accurate results. For this purpose we

[1]http://ailab.wsu.edu/casas/datasets/

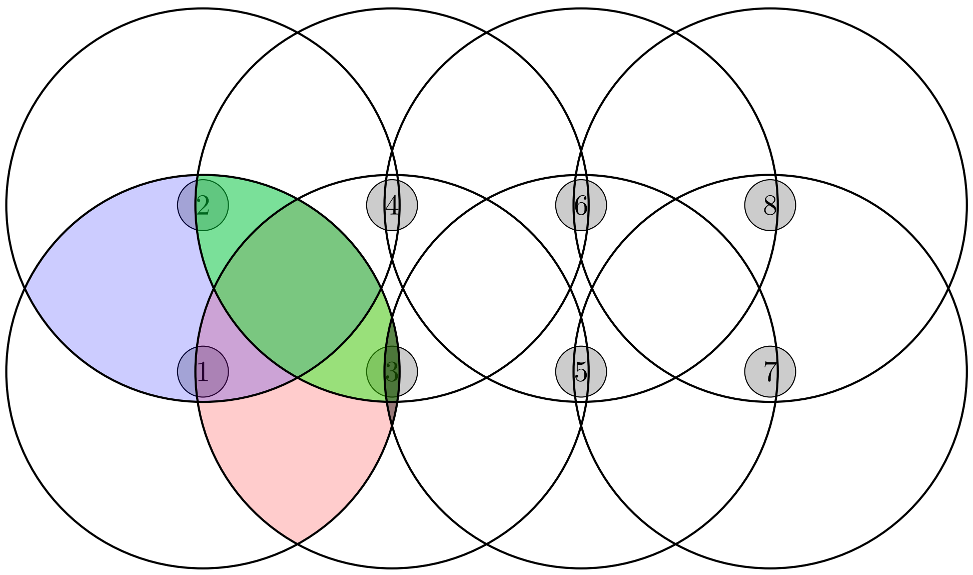


Figure 5.1: Overlapping field of view of the sensors on the vertex with the overlapping field of view for sensor 1 highlighted in color

$$\begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{bmatrix}$$

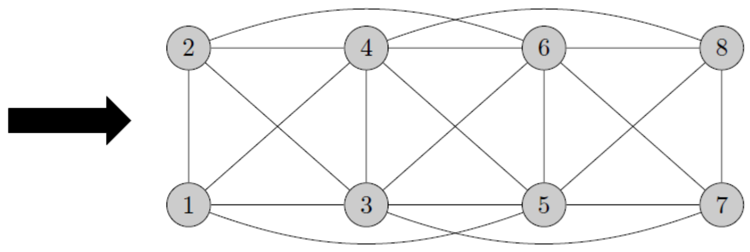


Figure 5.2: Grid adjacency matrix for the grid and its corresponding adjacency graph

dataset	Number of states visited	Mappings	Solution Mapping	Error
1	11888	2744	4	0
2	12360	2744	4	0
3	12144	2664	4	0
4	11984	2752	4	0
5	11736	2592	4	0
6	11888	2744	4	0
7	11776	2736	4	0
8	11368	2664	4	0
9	11752	2656	4	0
10	11368	2664	4	0
11	11368	2664	4	0

Table 5.1: REsults obtained from the HTC34 testbed.

5.3 something

The Grid adjacency matrix for the grid and corresponding grid graph is as shown in figure 5.2. The computed correlation values for one of the dataset of 8hrs length is as shown in the figure 5.3. We evaluate the performance with 10 different datasets of 8hrs each. We were able to map the sensors onto the grid accurately upto rotational symmetry using the weighted grid adjacency matrix and calculating the GCS. The mappings obtained are as shown in the figure **??**. Table 5.1 provides details about the number of states visited, number of mappings between the MST and the grid, and the number of mappings obtained as the solution from the developed method and the percentage error. As can be seen from table we obtain 0% error for all the datasets. The number of states visited(includes all the intermediate states) and the number of mappings is lesser than number of possible arrangements which is $8! = 40320$. Number of mappings is almost 14 times lesser than brute force method.

5.4 WSU Kyoto testbed

We evaluate our method using the WSU smart workplace testbed[17], a publicly available data set. The testbed is located in a 12.2m × 10.9m lab, where students performed their normal work routine. The data was collected over a period of 4 months. The testbed consists of 43 motion sensors, placed as shown in the figure 5.4. The sensors field of view is $1.2m \times 1.2m$. In appendix **??** we provide a detailed explanation about how we obtain the grid adjacency matrix for the testbed is provided. First we choose a 4 times 3 grid which includes sensors 27,22,15,28,23,16,29,24,17,30,25,18. This grid

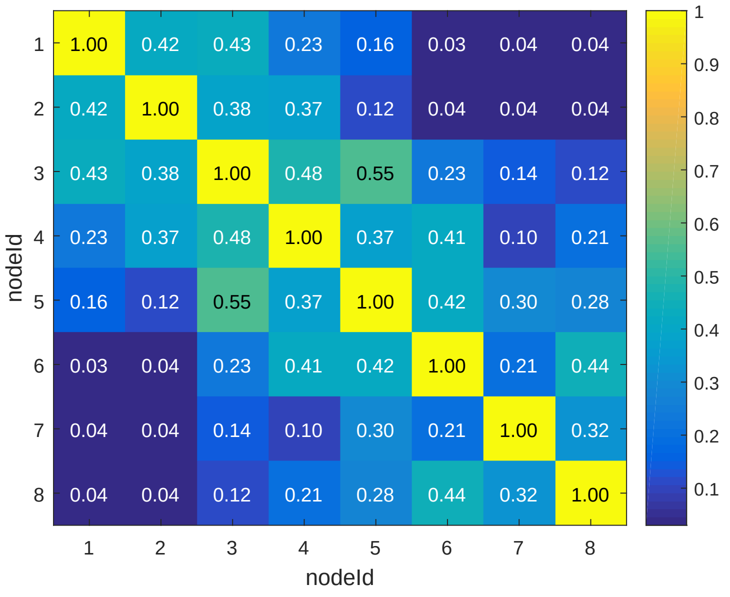


Figure 5.3: Correlation matrix R

Date	Time	sensor	state
2008-01-02	08:02:19.58542	M39	ON
2008-01-02	08:02:19.884461	M41	OFF
2008-01-02	08:02:20.297468	M32	ON
2008-01-02	08:02:20.297468	M32	ON
2008-01-02	08:02:20.689425	M31	ON
2008-01-02	08:02:21.957307	M44	OFF
2008-01-02	08:02:22.517275	M39	OFF

Table 5.2: My caption

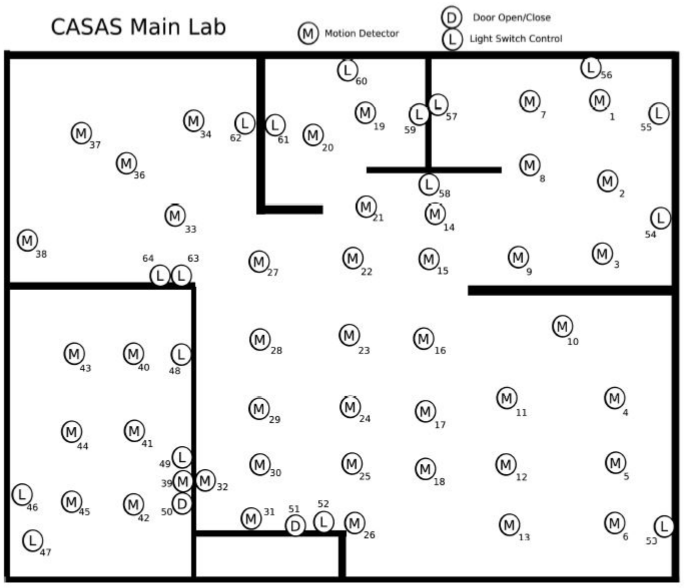
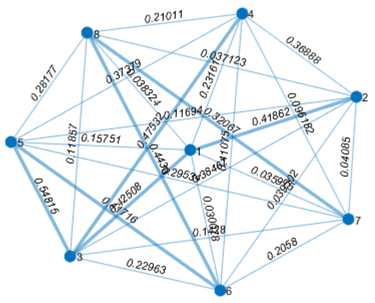


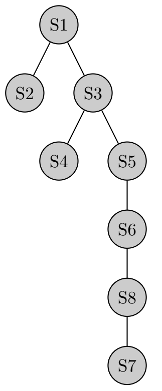
Figure 5.4: Floor plan for WSU CASAS office testbed, named Tokyo

was chosen as it is mentioned in [17] that the region covered by these sensors is more active. The data from the WSU testbed is represented in the form of a four tuple as shown in table reftab:WSUDATA. We re-sample the data at 100ms and use to validate our method.

We extract 10 dataset of 20 hrs each. Tes results obtained are as shown in the table **??**



(a) Correaltion matrix R represented as Graph G with maximum spanning tree highlighted



(b) Maximum spanning tree of R

Chapter 6

Conclusions and Future Work

6.1 Conclusions

TODO CONCLUSIONS

6.2 Future Work

TODO FUTURE WORK

Bibliography

[1] Xuesong Liu and Burcu Akinci. Requirements and evaluation of standards for integration of sensor data with building information models. *Computing in Civil Engineering*, 2009:10, 2009.

[2] Burcu Akinci, Mario Berges, and Alejandro Gomez Rivera. *Exploratory Study Towards Streamlining the Identification of Sensor Locations Within a Facility*, chapter 226, pages 1820–1827. doi: 10.1061/9780784413616.226. URL http://ascelibrary.org/doi/abs/10.1061/9780784413616.226.

[3] Jing Wang, Ratan K Ghosh, and Sajal K Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010.

[4] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, pages 13:1–13:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2431-1. doi: 10.1145/2528282.2528302. URL http://doi.acm.org/10.1145/2528282.2528302.

[5] Merthan Koc, Burcu Akinci, and Mario Bergés. Comparison of linear correlation and a statistical dependency measure for inferring spatial relation of temperature sensors in buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, pages 152–155, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3144-9. doi: 10.1145/2674061.2674075. URL http://doi.acm.org/10.1145/2674061.2674075.

[6] Jiakang Lu, Yamina Taskin Shams, and Kamin Whitehouse. Smart blueprints: How simple sensors can collaboratively map out their own locations in the home. *ACM Trans. Sen. Netw.*, 11(1):19:1–19:23, August 2014. ISSN 1550-4859. doi: 10.1145/2629441. URL http://doi.acm.org/10.1145/2629441.

[7] Carl Ellis, James Scott, Ionut Constandache, and Mike Hazas. Creating a room connectivity graph of a building from per-room sensor

units. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 177–183. ACM, 2012.

[8] Sebastian Müller, Axel Helmer, Enno-Edzard Steen, Melina Frenken, and Andreas Hein. Automated clustering of home sensor networks to functional re-gions for the deduction of presence information for medical applica-tions. *Wohnen–Pflege–Teilhabe–Besser leben durch Technik* , 2014.

[9] Dimitri Marinakis, Gregory Dudek, and David J Fleet. Learning sensor network topology through monte carlo expectation maximization. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4581–4587. IEEE, 2005.

[10] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957. ISSN 1538-7305. doi: 10.1002/j.1538-7305.1957.tb01515.x. URL http://dx.doi.org/10.1002/j.1538-7305.1957.tb01515.x.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[12] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23 (1):31–42, January 1976. ISSN 0004-5411. doi: 10.1145/321921.321925. URL http://doi.acm.org/10.1145/321921.321925.

[13] D. E. Ghahraman, A. K. C. Wong, and T. Au. Graph optimal monomorphism algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(4):181–188, April 1980. ISSN 0018-9472. doi: 10.1109/TSMC.1980.4308468.

[14] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Fast graph matching for detecting cad image components. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 1034–1037 vol.2, 2000. doi: 10.1109/ICPR.2000.906251.

[15] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs.

[16] Diane J Cook and Maureen Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5):480, 2009.

[17] Diane J Cook, Aaron Crandall, Geetika Singla, and Brian Thomas. Detection of social interaction in smart spaces. *Cybernetics and Systems: An International Journal*, 41(2):90–104, 2010.