Heikki Pulkkinen

Improving Energy Efficiency with Occupant Tracking

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, October 5, 2016

Thesis supervisor:

Prof. Peter Lund

Thesis advisor:

D.Sc. (Tech.) Laura Sepponen

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Heikki Pulkkinen

Title: Improving Energy Efficiency with Occupant Tracking

Date: October 5, 2016 Language: English Number of pages: 9+70

Degree Programme in Engineering Physics and Mathematics Code: F3002

Department of Applied Physics

Supervisor: Prof. Peter Lund

Advisor: D.Sc. (Tech.) Laura Sepponen

Detecting occupancy with motion detectors is the basis of many building automation systems for heating, ventilation, air conditioning or lighting. The conventional time delay method used in these automation systems has received many attempts to improve it due to a difficulty in maintaining a balance between user comfort and energy efficiency. These attempts have included more advanced statistical and machine learning based approaches.

This thesis proposes that data from multiple motion detectors can be handled with novel discrete target tracking. Multiple Hypothesis Tracking (MHT) can be used to model occupant movements in the area and ultimately to detect occupancy. This model uses the adjacency relationships as well as division between border and interior nodes, whereas the exact positions of the sensors are irrelevant. Targets which have been tracked to the border of the area (i.e. to the vicinity of the exits) can be deleted sooner than those previously detected by the interior sensors.

Simulated motion detector data was used to optimise the model parameters. The approach was also tested with two different setups with real sensors. The algorithm was successfully able to remember the presence of occupants which are not currently in the line of sight of the sensors. In a challenging test setup, accuracy of the predictions was 89 %, thus demonstrating a potential for 15 % energy savings compared to conventional time delay method.

Keywords: Energy efficiency, Occupant detection, PIR, Target tracking, MHT, BSN, HVAC

AALTO-YLIOPISTO	DIPLOMITYÖN
PERUSTIETEIDEN KORKEAKOULU	TIIVISTELMÄ

Tekijä: Heikki Pulkkinen

Työn nimi: Energiatehokkuuden parantaminen tilan käyttäjien seurannalla

Päivämäärä: 5.10.2016 Kieli: Englanti Sivumäärä: 9+70

Teknillisen fysiikan ja matematiikan koulutusohjelma Koodi: F3002

Teknillisen fysiikan laitos

Valvoja: Prof. Peter Lund

Ohjaaja: TkT Laura Sepponen

Useat valaistuksen, lämmityksen, ilmastoinnin ja jäähdytyksen taloautomaatiojärjestelmät perustuvat läsnäolon tunnistamiseen liikesensoreilla aikakatkaisumenetelmää käyttäen. Kirjallisuudesta löytyy paljon ehdotuksia aikakatkaisumenetelmän korvaamisesta kehittyneemmillä tilastotieteeseen tai koneoppimiseen perustuvilla lähestymistavoilla. Tarkemmalla menetelmällä olisi mahdollista parantaa järjestelmän energiatehokkuutta heikentämättä käyttäjän tyytyväisyyttä. Tässä työssä esitetään, että useiden liikesensoreiden antama tieto voidaan yhdistää uudella diskreettiin sijaintiin perustuvalla kohteiden seurantamenetelmällä.

Monihypoteesiseurannalla on mahdollista mallintaa tilan käyttäjien liikkeitä ja päätellä onko tila kyseisellä hetkellä käytössä vai ei. Malli hyödyntää tietoa siitä, mitkä sensorit ovat vierekkäisiä sekä jakoa reunoilla ja alueen sisällä olevien sensorien välillä. Tarkan sensorien sijainnin selvittäminen ei kuitenkaan ole tarpeellista. Alueen reunoille (eli uloskäyntien läheisyyteen) kulkeneet kohteet voidaan poistaa mallista aiemmin kuin ne, joista viimeiset havainnot on tehty alueen sisältä.

Tarvittavat parametrit optimoitiin simuloitujen liiketunnistusten avulla. Lähestymistapaa testattiin myös oikeilla sensoreilla kahdella eri koeasettelua. Siten oli mahdollista säilyttää muistissa sellaisten läsnäolijoidenkin paikat jotka eivät olleet sillä hetkellä sensoreiden havaintoalueen sisällä. Haastavassa koeasettelussa menetelmän tarkkuus oli 89 % ja mahdollinen energiansäästöpotentiaali aikakatkaisumenetelmään verrattuna oli 15 %.

Avainsanat: Energiatehokkuus, läsnäolon tunnistus, passiivinen infrapunatunnistin, kohteiden seuranta, monihypoteesiseuranta, binäärisensoriverkko, talotekniikka

Preface

It has been said that adding an equation halves the popularity of a book [1]. In other words, the number of readers for a similar study without any equations is

$$R' = 2^n \cdot R, \tag{1}$$

where n is the number of equations and R is the actual number of readers. Since the total number of equations in this thesis is 40, I could have reached around one trillion (10^{12}) times larger audience had I left out the math. I have to thank you, one of the few readers this study has left, on coming this far. I hope this thesis is more helpful to you than to all those readers I have lost.

Writing this thesis would not have been possible with the support of a broad group of friends and colleagues. Professor Aki Vehtari gave me valuable insights into the issue when I was about to begin my research, and explained to me why my initial approach was not the best possible. The most challenging part of this thesis was the development of a tailored target tracking algorithm and Professor Simo Särkkä's support has been irreplaceable for that matter. Thank you to you both!

I believe that this project has also taught me something about writing thanks to the feedback I have received from Laura Sepponen, Anya Siddiqi, Niko Ferm, Ukko Liukkonen, and Heli Vainio. Juulia Suvilehto deserves special thanks for the extraordinarily detailed feedback I got from her. I just hope that this section does not have any spellign errors.

I also want to thank supervisor Prof. Peter Lund and instructor Laura Sepponen. Completing this thesis would have certainly taken longer without Laura's encouragements to concentrate on the major study subjects. Jukka Ahola and Pauli Korhonen have been a great help to me with the sensors and other practical stuff. Max Björkgren has taught me how to keep an open mind for new ideas and test as many of them as quickly as possible. Sometimes one also has to think about the issues from the customer's point of view, as I have learned from Henri Juslén and Lars Hellström. Thank you to everyone else at Helvar too for all your support and the great work environment.

Espoo, September 2016

Heikki Pulkkinen

Contents

Symbols and Abbreviations

Symbols

C_{CO2}	Indoor CO_2 concentration
C'_{CO2}	CO_2 concentration of the air supplied by ventilation
$\mathbf{E}_{\Delta t}$	Emission matrix for Δt. See Equation (30).
E_t	Estimated occupancy status at time t. See Equation (2).
$\mathbf{F}$	State transition model for a Kalman filter
G_t	Ground truth, actual occupancy status
H	Hypothesis consisting of earlier decisions. See Equation (15).
$\hat{H}$	Parent hypothesis. See Equation (15).
K	Number of occupants
k	PIR triggering rate ratio between closest and adjacent nodes
L_B	Lifetime of the targets in border nodes
L_I	Lifetime of the targets in interior nodes
M	Maximum number of hypotheses
N_m	mth sensor node
N_F	Estimated amount of lost messages. See Section 4.1.2.
N_S	Amount of successful messages. See Section 4.1.2.
Q	Ventilation flow rate
R	Success rate of the wireless connection. See Section 4.1.2.
S	CO_2 generation rate per occupant
s_{T_n}	Position of target T_n
T_n	nth tracked target
$\mathbf{T}_{\Delta t}$	Transition matrix for timestep Δt. See Equation (34).
t	Time
V	Volume of the space
$\mathbf{w}_t$	White noise at time t
$\mathbf{x}_t$	State vector at time t. See Equation (10).
y_t^i	ith observation which has arrived at time t
$\mathbf{Y}_t$	All observations up to time t
$\mathbf{Y}_t^n$	Observations up to time t associated with target T_n
Δt	Length of the time step
Θ	Association and movement hypothesis. See Equation (14).
$\mathbf{\Lambda}_E$	Emission rate matrix
$\mathbf{\Lambda}_T$	Transition rate matrix. See Equation (33).
λ_{FA}	False alarm rate
λ_E	PIR sensor triggering rate in current node. See Equation (39).
λ_{NT}	New track rate
λ_T	Target transition rate to adjacent nodes. See Equation (38).
ω_t	Partition of Y_t. See Equation (13).

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative in respect to time t
$\sum_i$	Sum over index i
$\prod_i$	Product over index i
$A \mapsto B$	Mapping from A to B
$\mathbf{A}[i,j]$	Matrix $\mathbf{A}$ element in row i and column j
$\log(x)$	Logarithm of x to base 10
$O(n)$	Complexity of an algorithm with the big O notation.
$P(A\|B)$	Conditional probability of A given B
$P_{\Delta t}(A)$	Probability of an event A happening within time Δt

Abbreviations

ANN	Artificial Neural Network
BBN	Bayesian Belief Network
BSN	Binary Sensor Network
CO_2	Carbon dioxide
ECF	Energy Conservation Factor. See Equation (4).
FA	False Alarm (clutter)
GNN	Global Nearest Neighbour
HMM	Hidden Markov Model
HVAC	Heating, Ventilation and Air Conditioning
LNN	Local Nearest Neighbour
MCMC	Monte Carlo Markov Chain
MDS	Multidimensional Scaling
MHT	Multiple Hypothesis Tracking
NT	New Track
PAF	Prediction Accuracy Factor. See Equation (3).
PIR	Passive infrared
RMS	Root Mean Square
UCF	User Comfort Factor. See Equation (5).

1 Introduction

Buildings account for one-third of energy consumption in the European Union [2, 3]. The majority of this is consumed by heating, ventilation, and air conditioning (HVAC) and lighting. Since the need to operate these systems depends highly on the presence of occupants, accurate detection of occupancy has been an ongoing research topic. Even a slight improvement in this area can quickly bring major energy savings.

Passive infrared (PIR) sensors have been a common way to detect occupancy since their introduction 40 years ago. The output of the sensor can directly be set to control lighting or sometimes even HVAC systems. This is usually done using the time delay method: When the sensor detects motion, a timer starts. Any subsequent observations will reset the timer. When the timer ends, the area is considered unoccupied.

The weakness of the time delay is well known to many users of automatic lighting control. If the occupant is somewhere outside of the sensor's line of sight, the lights will at some point turn off too early. Sometimes that can also happen if the occupant is not moving enough. Improving the sensor quality can sometimes resolve these problems. In this thesis, however, we explore the possibility of solving the issue with more advanced data analysis.

If a human sees someone entering a room, it is easy to understand that the room is occupied until the occupant is seen leaving the area. It does not matter if the occupant is not detected directly during that time. In a similar way, a computer program can detect occupancy indirectly. Examples of these kinds of situations are shown in Figure 1. If the previous observation was made by one of the sensors close to these positions, the space can safely be assumed to be occupied.

This can be accomplished by having a computer model the occupants within the building. A framework for this kind of data analysis already exists in the form of target tracking. The approach was initially developed for radars where measurements

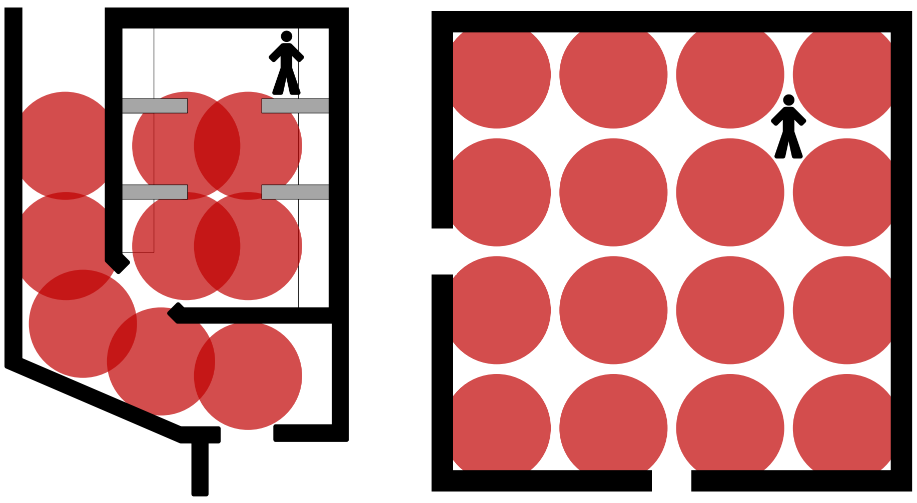


Figure 1: Two examples of layouts where an occupant is not in the line of sight of any of the sensors (red circles), but cannot leave the area without being detected.

are also explained by associating them to one or more of the modeled objects [4].

In our application, we have access to wireless motion detectors with a dense layout (see Figure 1). Our goal is to make the installation and configuration of the system as effortless as possible, and for this reason, the exact positions of the sensors are not known. As we discuss in Section 7.2, it is possible to determine which sensors are adjacent to each other only by using the sensor data [5, 6]. This information about the structure is the basis of our model.

Traditionally occupancy is measured with only a few motion detectors per room with a large line of sight. The large number and tight spacing of the sensors in our setup enable us to reach a much better accuracy than has previously been possible. Complex patterns and events can be detected by combining the outputs of multiple simple binary sensors.

In this thesis, we introduce a novel method of occupancy detection in a discrete sensor network. The type of the sensors accounts for the major difference of our method compared to the traditional target tracking. Whereas radars can measure target's position on a continuous scale, a motion detector can only give discrete information.

This kind of sensor setup is called Binary Sensor Network (BSN) [7]. Unfortunately, target tracking in such an environment is usually based on the knowledge of exact positions of the sensors. Furthermore, a standard BSN approach requires a high density of sensors.

In addition to knowledge on which sensors are adjacent, we utilise information about which sensors are on the border of the area, such as close to doors or other exits. Target tracking is then used to determine whether the occupants are in the vicinity of the interior sensors or if they have approached the exits. With this method, we can apply long time delays when occupants have previously been detected within the interior and switch to unoccupied state sooner when they have been tracked to the border of the area.

The goal of this thesis is to determine if target tracking is an accurate method for occupancy detection. To do this, we first adjust the earlier approaches so that they can be used with our devices in Chapter 3. In Chapter 5 the method is tested with simulations and real sensor data.

The accuracy of occupant detection is measured in two different ways. Highest priority is given to user comfort, which is related to the accuracy of correctly detecting occupancy when the area is occupied. In these cases, the accuracy requirement was set to 90 % or 95 %.

The second factor is the utilisation of potential energy savings when the area is unoccupied. For example with the time delay method the area is incorrectly considered to be occupied even some time after the occupants have left. If two methods can achieve the same level of user comfort, the one with the higher amount of energy savings is considered superior. In Chapter 6 target tracker and the time delay method are compared.

This thesis does not go into the details of automatically configuring the sensor network. This area has already been investigated earlier, and a satisfactory solution already exists [5, 6]. Improving the current algorithm is a possible research area in the future.

Integration between the occupancy detection to HVAC, lighting or some other system is also outside of the scope of this thesis. The aim of this thesis is limited to determining the current occupancy of a space as accurately as possible.

2 Background

There are numerous approaches for occupant detection and occupant counting. Motion sensors such as passive infrared (PIR) sensors are the most common equipment and are also used as the main data source in this thesis. Other possible sources of information include carbon dioxide (CO_2) sensors, microphones, and video cameras. Occupancy information is usually divided into the following subcategories [8]:

1. Presence of occupants anywhere in the building
2. Presence of occupants in each room
3. Count of people present
4. Activity: What are the occupants doing?
5. Identification of the occupants

In this thesis we focus on the first three categories since these types of information can be acquired with motion detectors. To identify the occupant one would need additional sources of information. Possible approaches are facial recognition, integration with access control or a mobile application. Combining motion detector data with one of these is a potential topic for future research.

First two categories are also called occupant detection while the third category is sometimes referred as occupant counting [9]. Occupant detection aims at telling the difference between time periods of the room being occupied or unoccupied. In some cases, such as controlling heating and lighting, occupant detection is enough, and attention should be focused on making the detection as accurate as possible. These methods are discussed further in Section 2.1.

Having an approximation of the number of people is useful for some applications. These include ventilation, usage statistics, evacuation, and security related issues. Section 2.2 describes possible approaches to this problem.

Sensor data can also be processed with tracking algorithms. In the case of binary sensors, such as motion detectors, the approach is called binary sensor network. Review of previous work on this matter is provided in Section 2.3.

2.1 Occupant Detection

This section summarises earlier work with occupant detection. We focus on research where PIR sensors are the main information source.

In Section 2.1.3 we discuss the most common way to detect occupancy with a simple time delay method. It has been suggested that a decision tree (Section 2.1.4) would be an accurate method to combine data from multiple sensors. Utilising history data with a Bayesian Belief Network (Section 2.1.5) or a Hidden Markov Model (Section 2.1.6) is another possible approach. Before that we discuss how to measure accuracy (Section 2.1.1) and classify errors (Section 2.1.2).

The goal of occupant detection is to calculate estimated occupancy status

$$E_t = \begin{cases} 1, & \text{if space is occupied at time } t \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

There are two main types of occupancy detection. Sometimes the analysis has to be done live, which is the case with heating, ventilation, and air conditioning (HVAC) automation. In a live analysis, only the data obtained up to that point in time can be used.

The second type can be called analysis of history data. In this case, the result is not required immediately, and also measurements after the time of interest can be utilised. For example, if a motion sensor sees movement soon after time t the area was probably occupied also at time t. An example of this kind of situation would be the estimation of usage patterns. This approach is likely to achieve more accurate results. In this thesis, we do not use future data in this way, but this is a potential topic for future research. [10]

2.1.1 Measuring of Accuracy

In this section we discuss how occupancy detection accuracy should be measured. Determining the percentage of correct predictions compared to all predictions is the most intuitive approach. That is exactly how Prediction Accuracy Factor (PAF) is calculated since

$$\text{PAF} = \frac{\sum_{t=1}^{k} E_t = G_t}{k}, \tag{3}$$

where G_t is the ground truth defined like E_t was defined in Equation (2) and k is the total number of time bins [11]. Meaning of E_t and G_t are illustrated in Figure 2.

When the ratio is close to 1, the predictions are accurate. A ratio of 0.5 means that the prediction method performs at chance level. PAF is commonly used in the literature to evaluate occupancy detection methods.

PAF has its limitations since not all errors in the occupancy detection are equal. Misinterpreting unoccupied space as an occupied can lead to extra energy consumption. For this reason, Energy Conservation Factor (ECF) is defined as the ratio between number correctly estimated states where the area is unoccupied and the number of all cases when the area is unoccupied [11]. In other words

$$\text{ECF} = \frac{\sum_{t=1}^{k} E_t = G_t = 0}{\sum_{t=1}^{k} G_t = 0}. \tag{4}$$

It is much worse to misinterpret occupied state as unoccupied which would lead to bad user experience. A good example is lights turning off when the occupant is still present but not moving. User Comfort Factor (UCF) has been introduced to

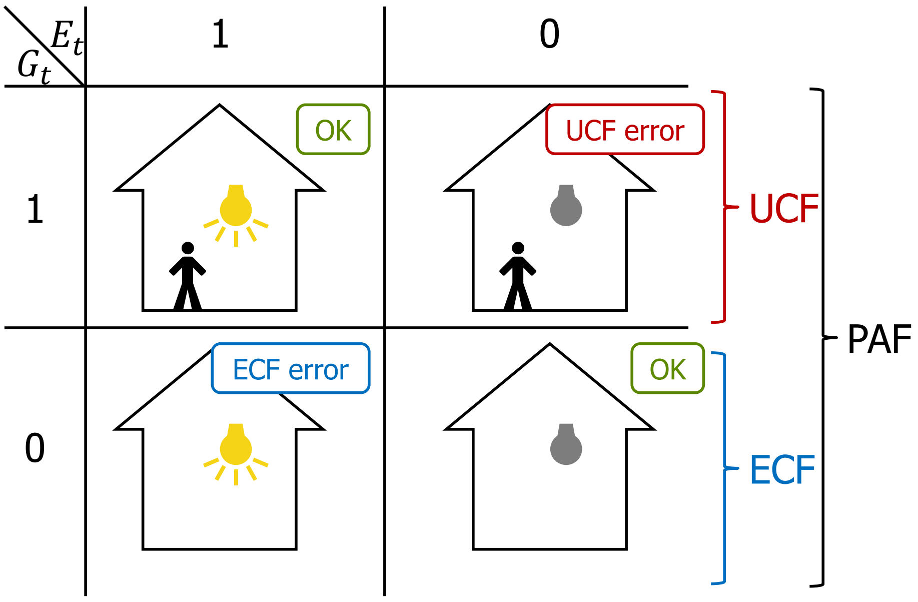


Figure 2: Different error types in a setup where predicted occupancy is used to control the lighting.

account for the issue [11]. UCF is the ratio of correctly determined time of occupancy and the total time of occupancy [11]. In mathematical form that is

$$\mathrm{UCF} = \frac{\sum_{t=1}^{k} E_t = G_t = 1}{\sum_{t=1}^{k} G_t = 1}. \tag{5}$$

In our application, a sufficient UCF is a fundamental requirement. Excellent energy efficiency is worth nothing if the user does not want to use the automation because of comfort issues. [12]

2.1.2 Classification of Errors

Occupant detection errors are divided into two sets: transition errors and spurious errors. These are illustrated in Figure 3.

Transition errors are errors related to state transitions between occupied and unoccupied. Transition errors can be further divided into two subcategories. Because of *sensor lag*, it might take some time for the system to notice a change in the environment. The second subcategory is the *errors in the ground truth* in the form of the exact timing of the occupancy state updates. [9]

Spurious errors consist of all other error sources. A sensor being triggered by something else than an occupant (these are later called false alarms) can cause to this type of mistake. It is also possible that sensors do not provide enough information. This can occur when occupants are asleep or out of range. [9]

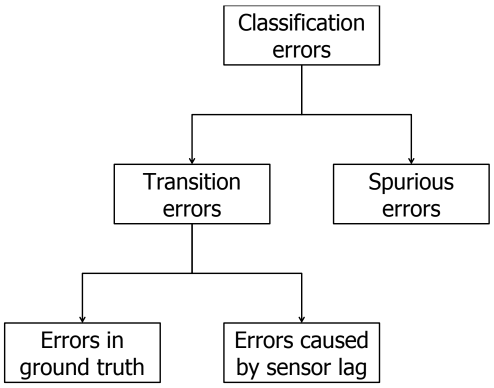


Figure 3: Possible errors with occupant detection.

2.1.3 Time Delay Method

The time delay method is the simplest and most common way to utilize motion detectors. The basic idea is, that if motion is detected, the area is assumed to be occupied for given amount of time. Longer time delays bring higher energy savings, but it also reduces the user comfort when lights are switched off prematurely.

Von Neida et al. [13] have done comprehensive tests with this method. They used 5 min and 20 min time delays and were able to demonstrate energy savings between 6 % and 13 % compared to manual control. It was possible to calculate PAF, UCF, and ECF from the number given by them and these values can be found in Table 1.

Table 1: Key figures and number of test areas with the time delay method. The values are based on research done by Von Neida et al. [13].

Space type	**PAF**	**UCF**	**ECF**	**N**
Break room	83 %	88 %	82 %	11
Classroom	78 %	81 %	77 %	35
Conference	87 %	82 %	88 %	33
Private office	83 %	94 %	80 %	37
Restroom	51 %	95 %	40 %	42

The weighted averages are 75 %, 89 %, and 71 % for PAF, UCF, and ECF, respectively. During the two-week monitoring period, the area was unoccupied for most of the time, especially during nights and weekends. Since estimating occupancy during nights is easy, ECF was much higher during nights than during day time.

Other authors have reported energy savings of 38 % [14], 40 % [15] and 20-26 % [16]. It should be noted that these studies only demonstrate the ideal potential of the new method. Automatic control has been found to decrease the users willingness to manually turn off the lights when exiting the area. This phenomena is called the rebound effect and it can reduce energy savings by 30 % [17].

There have been multiple suggestions on how to improve the time delay method. It is possible to learn optimal values for time delays directly from the motion detector data [18, 19]. This approach can greatly reduce the work required for configuring the devices and also save up to 37.9 % of energy usage [18, 12].

2.1.4 Decision Trees

The idea behind a decision tree is to divide the problem ("Is the space occupied?") to simpler questions ("Has the motion sensor detected movement within last 5 minutes?" or "What is the value of CO_2 sensor?"). These questions are then used to build a decision tree, i.e. a flowchart which gives an answer to the initial problem as the output. There are many alternative methods to find the optimal decision tree to give accurate results with as few questions as possible.

The output of decision trees is easy to understand which is a definite advantage compared to support vector machines or artificial neural networks (ANNs). These methods tend to be more like a black box between the algorithm inputs and outputs. [9]

Hailemariam et al. [9] used decision trees to detect occupancy in an office cubicle. They collected data from motion, CO_2, sound, light, and current sensors at regular scanning intervals. Two methods were used to calculate inputs for the decision tree: average and root mean square (RMS). The time interval for the average and RMS varied between 1 min and 64 min. [9]

Surprisingly the best accuracy of 98.4 % was achieved by using only 2 min RMS of the motion sensor [9]. Since only one input was used the use of decisions trees was unnecessary.

2.1.5 Bayesian Belief Networks

Bayesian Belief Networks (BBN) model the relationships between the hidden variables and the observed measurements. For example, detected motion can be explained by an occupant in the area or sensor malfunctioning, which are the two hidden variables in this case. The model can also be built to depend on earlier occupancy states.

The structure of the network is usually configured by the user, although some work has been done to configure it automatically [20]. It is possible for the model to learn the conditional probabilities of the network from the ground truth.

BBNs are also easy to comprehend. BBNs can handle sensor malfunction and can be used to give Bayesian estimates of the sensor status based on the collected data. However, this method can be computationally expensive. [21]

It is possible to use a BBN to model sensor reading caused by the occupants. Dodier et al. [21] used a setup of three PIR sensors and a telephone sensor which indicated whether the phone was "off-hook" or not. The network was constructed so that occupancy status affected all four sensor readings and the occupancy of the next time step. Sensor readings were also configured to depend on the sensor status (OK or malfunctioning) to account for faulty devices. The accuracy of the method was not reported in the article but according to the figures a BBN seems to give better results than a simple time delay method. [21]

2.1.6 Hidden Markov Models

Markov models are an approach to model dynamic systems which move randomly between different states. They have been used to occupant detection directly [11] and are applied in Section 3.5 to model occupant movements.

The key assumption is that future states depend only on the current state. It implies that transition probabilities are constant, and they can be summarised in a transition matrix [22]. An example of a Markov model is shown in Figure 4.

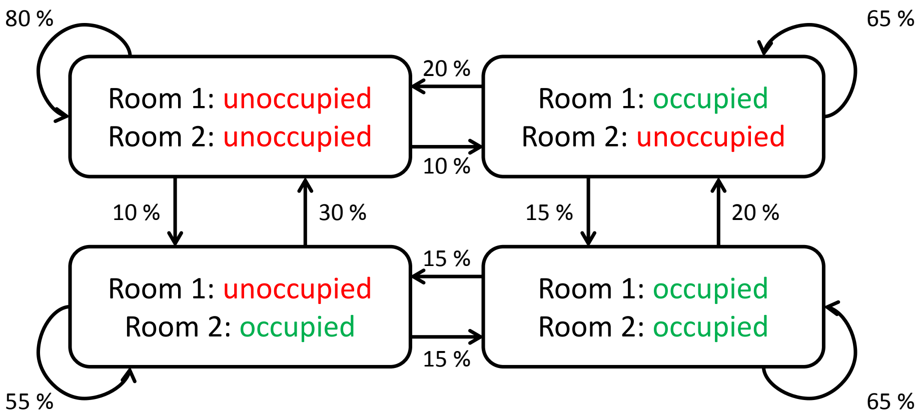


Figure 4: An example Markov model for occupancy status of two adjacent rooms. Percentages are transition probabilities between the states.

One way to apply such a model is to simulate the system. The next state is chosen from the current state based on the transition probabilities. If the simulation started from both rooms being unoccupied, the probabilities would be 80 % for staying in the same state, 10 % for room 1 to become occupied and 10 % for room 2 to become occupied. This kind of simulation is called Monte Carlo Markov Chain (MCMC).

In a Hidden Markov Model (HMM) the states cannot be measured directly. Instead, the model has emissions which are visible to the observer. Emissions depend on the state, and it is possible to estimate the actual state if the conditional probabilities are known. Viterbi algorithm can be used to find out the most probable sequence of states for given emission sequence.

In Figure 4 the possible sources of emissions would be motion detectors in rooms 1 and 2. Occupants usually trigger the sensors when they are present, but sometimes they are also triggered completely randomly. Possible emissions in this case are

1. No motion detected
2. Motion detected in room 1 but not in room 2
3. Motion detected in room 2 but not in room 1
4. Motion detected in both rooms

Each state in Figure 4 would have fixed emission probabilities for these four alternative emissions. For example, if room 1 is unoccupied and room 2 is occupied the most common emission would be *2. Motion detected in room 1 but not in room 2.*

If the number of rooms is increased this approach becomes computationally expensive. For just 30 rooms there would be more than one billion (10^9) possible emissions. To efficiently use an HMM and the Viterbi algorithm one has to keep the size of the model relatively small.

Batra et al. [11] have successfully used Hidden Markov Models and Viterbi algorithm to detect occupancy. In their model, there were just two hidden states: occupied and unoccupied. The transition and emission probabilities were approximated from the ground truth. The accuracy of 97 % was achieved when the window for data processing was between 10 s and 30 s. However, similar results can be reached with a much simpler time delay method. [11]

2.2 Occupant Counting

The goal of occupant counting is to determine the number of occupants in a given area from sensor data. It is usually necessary to use a sensor capable of measuring people count such as a CO_2 sensor or a video camera.

Possible algorithms for using just a CO_2 sensor are explained in Section 2.2.1 and Section 2.2.2. ANNs can be used to combine data from different types of sensors and estimate people count [23]. That is discussed in further detail in Section 2.2.3.

Video cameras are another possible source of occupant count information. Cameras have been used e.g. to estimate people flow rates in corridors [24]. Kalman filters are a good way to process this information (Section 2.2.4).

2.2.1 Steady State Algorithm with Carbon Dioxide Sensor

Average adult generates around 0,0052 litres of CO_2 per second [25]. CO_2 itself does not affect Indoor Air Quality. However, CO_2 level correlates well with the level of other occupant-generated contaminants. A CO_2 concentration of more than 1000 ppm signifies that the ventilation is not high enough [25].

In Demand Controlled Ventilation, the CO_2 level is used directly to control ventilation. The CO_2 level can also be used to approximate the number of occupants. CO_2 balance can be expressed as

$$\frac{dC_{\mathrm{CO2}}}{dt} = \frac{K \cdot S}{V} + Q\frac{C'_{\mathrm{CO2}}}{V} - Q\frac{C_{\mathrm{CO2}}}{V}, \tag{6}$$

where C_{CO2} is the CO_2 concentration of the space, K is the number of occupants, S is CO_2 generation rate per occupant, t is time, Q is ventilation flow rate, C_s is the CO_2 concentration of outdoor air and V is the volume of the space [26, 27]. In other words, the change in CO_2 concentration $\frac{dC_{\mathrm{CO2}}}{dt}$ equals the generation rate of the occupants $\frac{K \cdot S}{V}$ plus amount of CO_2 in replacement air $Q\frac{C'_{\mathrm{CO2}}}{V}$ minus amount of CO_2 in the air flowing out $Q\frac{C'_{\mathrm{CO2}}}{V}$. The equation assumes perfect mixing of the air.

Equation (6) can be derived to form

$$V\frac{dC_{\mathrm{CO2}}}{dt} = K \cdot S - Q\left(C_{\mathrm{CO2}} - C'_{\mathrm{CO2}}\right). \tag{7}$$

Derivative term is zero with the assumption that CO_2 level is in a steady state. With this assumption the number of occupants is given by

$$K = \frac{Q}{S} \left(C_{\mathrm{CO2}} - C'_{\mathrm{CO2}} \right) . \tag{8}$$

In other words, CO_2 produced by the occupants is calculated from the amount of CO_2 removed by the ventilation, which is further used to estimate occupant count. [26]

The steady state system is simple to use, but it has a noticeable delay. Labeodan et al. [8] made a study in a meeting room and found out that the lag was about 1.5 h. The duration of the lag is highly influenced by the shape and dimensions of the space [25].

2.2.2 Dynamic Algorithm with Carbon Dioxide Sensor

If CO_2 level begins to increase rapidly, it is immediately possible to deduce that there must be multiple persons in the room. A scientific way to use this information is to measure the derivative of C_{CO2}.

Equation (7) can be written in form

$$K = \frac{Q}{S} \left(C_{\mathrm{CO2}} - C'_{\mathrm{CO2}} \right) + \frac{V}{S} \frac{dC_{\mathrm{CO2}}}{dt} . \tag{9}$$

The derivative of the concentration $\frac{dC_{\mathrm{CO2}}}{dt}$ can be estimated for example with the difference quotient. [26]

Wang et al. [26] have tested dynamic and static algorithms with simulations. The dynamic algorithm was found to give volatile results because of large relative error in the derivative of CO_2 concentration. They successfully used a second order filter to smooth the results. [26]

2.2.3 Artificial Neural Networks

ANNs mimic how human brain processes information with simple neurons. The inputs, such as the sensor reading or earlier outputs, are processed with one or more hidden layers to give an output value. A simple ANN is illustrated in Figure 5.

The inputs are used to calculate values of the hidden nodes. This is usually done with a linear combination of the inputs with some constant weights and a non-linear activation function, such as hyperbolic tangent. Connections between the nodes represent the neurons—the weight of a connection determines if activation in the previous layer should activate or deactivate a node in the current layer. The weights are optimised based on the learning data to give results similar to the ground truth.

This kind of simple network can be improved in many different ways. There can be more than just one hidden layer to make the calculations iterative. Alternatively, one could simplify the model radically by having just one hidden layer and randomizing the weights between the hidden layer and the inputs. This method is called extreme learning machine, and it has the advantage of exceptionally fast and computationally cheap learning [28].

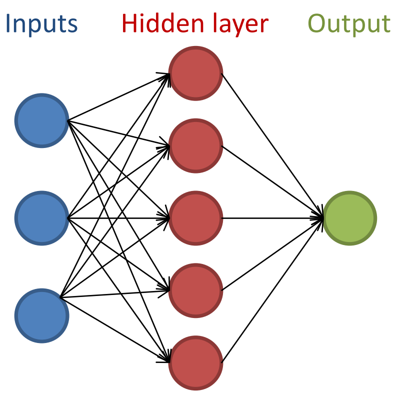


Figure 5: ANN with three input variables and one hidden layer with five nodes. Strength of the links between the nodes, which correspond to the neurons in human brain, are calculated from the teaching data.

It is possible to combine information from multiple different sensors with neural networks [29, 23]. Yang et al. [23] used a large set of sensor inputs: lighting, sound, motion, CO_2 concentration, temperature, relative humidity, reflector (infrared), door status, 1 min motion count, 1 min reflector count, 1 min door count, 5 s sound average, and 5 min sound average. According to a principal component analysis, the least significant inputs were 5 min sound average, door status, and temperature. These were excluded from the analysis. [23]

The remaining nine inputs were analysed with a back propagation ANN. Ground truth data was collected with a mobile device to which the occupants had continuously updated the number of people in the room. The ground truth was used to teach the ANN. A multi-layer ANN resulted to the best accuracy of 92 %.[23]

One should note that the light sensors were one of the key sensors used by the ANN. The lights were operated manually by the users. If the goal is to use the occupancy information to control lights automatically, it is not possible to use the light sensor data in this way. Furthermore, it is challenging to collect enough ground truth data from an actual installation site. Configuring an ideal system should not require any manual input from the end users.

In a test environment, it is possible to collect an adequate amount of ground truth data to teach an ANN. In our application, this is not the case since it is not feasible to gather ground truth data from all new sites.

2.2.4 Kalman Filters

In an area with a large number of rooms, or other areas of interest, one approach is to base the estimates on people flow. If one can measure the amount of occupants moving from one zone to another, the amount of people in each zone can also be estimated. Kalman filters are a good method to estimate the occupancy count from people flow measurements.

Kalman filter is an elegant method for combining earlier information with new measurements [30]. The goal is to approximate state vector $\mathbf{x}_t$ at time t. In target tracking, described in Section 2.3, a common choice of variables in $\mathbf{x}$ are location, speed, and possibly acceleration [4]. The system is assumed to follow

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{w}_t, \tag{10}$$

where $\mathbf{F}$ is the state transition model and $\mathbf{w}_t$ is white noise. In other words, the next state of the system is a linear function of the previous state plus a random variable. Kalman filter also estimates the error of the state variables in the form of a covariance matrix. When the state vector is updated based on a new measurement more weight is given to the previous state estimates with a small error.

CO_2 measurements have been successfully integrated with people flow measurements [24] with this method. Meyn et al. [24] used closed-circuit television video cameras to measure the people flow rates at the corridors.

State vector $\mathbf{x}_t$ consisted of the number of occupants in areas of interest and rate of occupants moving from one area to another. Even though the number of people is a vector and moving rate is a matrix, they both consist of a fixed number of variables which can be listed into a single vector. State transition model $\mathbf{F}$ was built based on the fact that the total number of people is conserved.

The method was able to estimate the current occupancy information with live data. Even better results were achieved by analysing the history data of a whole day. With this method, the prediction accuracy was 89 % [24].

2.3 Target Tracking

In this section, we go over relevant earlier research in target tracking. The tracking algorithm we developed in this thesis (described in Chapter 3) is based on some of the topics introduced in this section. For this reason, the reader should pay extra attention to global nearest neighbour (Section 2.3.2) and multiple hypothesis tracking (Section 2.3.3).

Target tracking has been initially developed for radars. The key problem is how to associate the measurements with the targets. When the association is done, each location for each target can be updated with its measurements. Radars usually do this with a Kalman filter. Principal components of a typical target tracker are illustrated in Figure 6. [4]

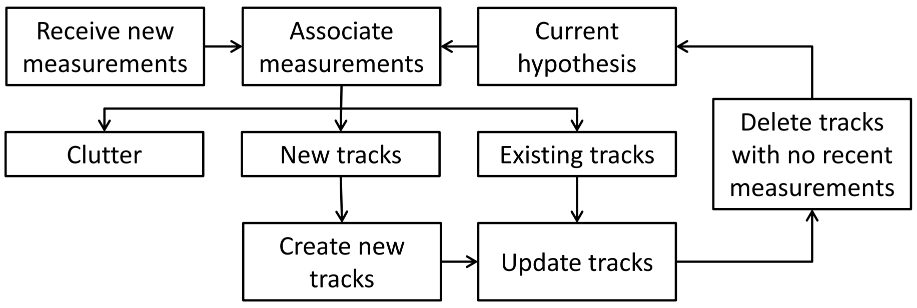


Figure 6: Outline of a typical single hypothesis tracking algorithm.

When new measurements arrive they are first associated with clutter (false alarms), new tracks or existing tracks. Association can be done with multiple methods some of which are described in sections 2.3.1, 2.3.2, 2.3.3, and 2.3.5.

New track associations are processed with the initiator logic which creates new tentative tracks. Since a single false alarm could have caused the tentative track, they are not shown to the user. Most simple method to confirm tentative tracks is to require a certain amount of measurements within the first seconds (initiator lifetime). Alternatively this can be accomplished with score based approaches [4].

Measurements associated with one of the current tracks are used to update the track. In a typical radar solution, Kalman filter is used for this purpose [4]. If the speed or acceleration of the target has been estimated, these can be used to update the position of the target even without an associated measurement.

In the final step tracks without enough recent updates are deleted. One possibility is to check if there has been a required amount of measurements within a predefined track lifetime. For tentative tracks, the lifetime usually is shorter or required amount of measurements is higher compared to confirmed tracks.

The algorithm has now computed an updated list of tracks (current hypothesis) which can be used to explain next set of measurements.

To speed up the computations, gating of the measurements can also be used. The idea of the gating is to eliminate impossible associations as soon as possible. One way to do this is to consider only associations to targets which are within a gating radius from the measurement.

2.3.1 Local Nearest Neighbour

Local Nearest Neighbour (LNN) is a method suitable for mainly single target tracking. Targets are simply associated with the nearest available measurement. The same measurement can be associated with multiple targets. [4]

Figure 7 shows an example of the association problem. LNN solution would be to associate $T_1 \mapsto y_k^1$, $T_2 \mapsto y_k^2$ and $T_3 \mapsto y_k^2$.

Measurements, which are outside of the gating of all currently tracked objects, are usually used to initiate new tracks. Best practice to handle unused measurements within the gating depends on the application. Using them to initiate new tracks is a good idea if targets usually cause just one measurement. In the case of PIR sensors, it is common that targets trigger multiple sensors simultaneously. Using the other measurements to initiate new tracks would lead to having multiple tracked targets for a single actual target. Discarding all unused measurements within the gating would thus be a good idea.

At some points, multiple tracked objects start to overlap. Overlapping targets are always associated with same measurements. Merging similar tracks could solve this issue. However, it is more convenient to use a more advanced algorithm such as Global Nearest Neighbour (GNN).

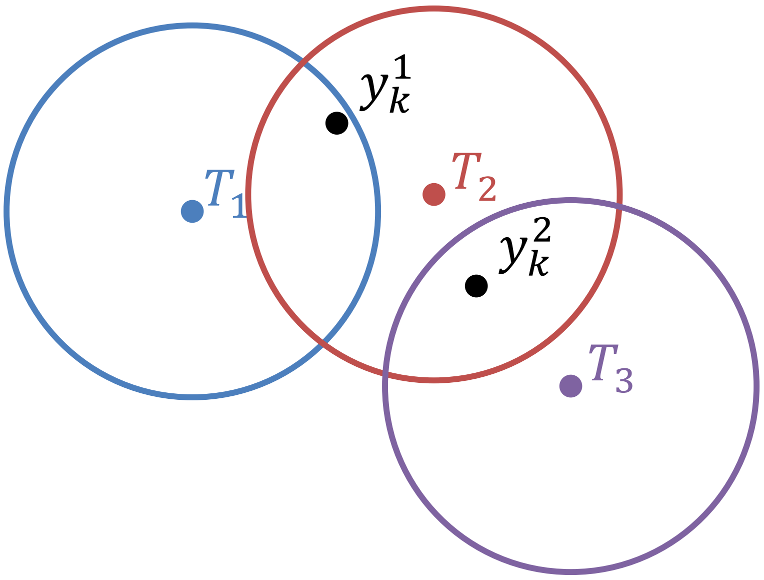


Figure 7: An example of association problem with three tracked targets (T_1, T_2, and T_3) and two measurements (y_k^1 and y_k^2). Circles represent the gating windows.

2.3.2 Global Nearest Neighbour

With GNN each measurement is associated with only one target, and each target has a maximum of one associated measurement. This method is superior to Local Nearest Neighbour.

Association is usually done by constructing an association matrix. Rows correspond to different measurements while columns are targets, new targets (NT), and false alarms (FA). An association matrix for situation in Figure 7 would be

$$\begin{array}{c|ccccccc} & T_1 & T_2 & T_3 & FA_1 & FA_2 & NT_1 & NT_2 \\ \hline y_k^1 & 0.015 & 0.3 & 0 & 0.01 & 0 & 0.02 & 0 \\ y_k^2 & 0 & 0.4 & 0.03 & 0 & 0.01 & 0 & 0.02 \end{array}. \qquad (11)$$

This matrix has three currently tracked targets (T_1, T_2, and T_3) and two measurements (y_k^1 and y_k^2). The probabilities can be calculated with Bayes' theorem similar to our approach in Section 3.4. According to the matrix probability of association $y_k^1 \mapsto T_1$ is 0.015, $y_k^1 \mapsto T_2$ is 0.3 and so on. The probability of false alarm is 0.01 and probability of a new track is 0.02. If there had been just a single measurement y_k^1, we would choose association $y_k^1 \mapsto T_2$ because of the maximum likelihood.

The most likely explanation for the second measurement is also T_2. That would violate the basic principle of global nearest neighbour that each target can have one associated measurement. The alternative possibilities are $y_k^1 \mapsto T_2$, $y_k^2 \mapsto T_3$ and $y_k^1 \mapsto NT_1$, $y_k^2 \mapsto T_2$. The combined probabilities of these examples are $0.3 \cdot 0.03 = 0.009$ and $0.4 \cdot 0.02 = 0.008$, respectively. In this case, $y_k^1 \mapsto T_2$, $y_k^2 \mapsto T_3$ would be the most likely association.

The general assignment problem is to choose one cell from each row so that the sum of the cells is maximised. In our case, we need to maximise the product of the elements. The matrix can be transformed to the correct form with logarithm function. Instead of maximising the product, one gets the same result by maximising the logarithm of the product. The logarithm of the product is equal to the sum of the logarithms. With the numbers of the previous example, this means that the log probability (which is to be maximised) is

$$\log(0.3 \cdot 0.03) = \log(0.3) + \log(0.03)\,. \tag{12}$$

There are many ways to solve the assignment problem. Auction algorithm mimics an auction where each customer (measurement) wants to buy exactly one of the multiple items (tracks, FA, NT). Blackman [4] has given a good description of the algorithm.

2.3.3 Multiple Hypothesis Tracking

Multiple Hypothesis Tracking (MHT) was first proposed by Reid et al. [31]. The association decisions are in one sense approximations which lead to loss of information. The core idea of multiple hypothesis tracking is to delay the making of association decisions. That enables the use of the association which best corresponds to the future measurements.

These different association decisions are formulated as multiple hypotheses with different weights. The weights are calculated from the probabilities of the hypotheses. Probability is calculated from probability of the parent hypothesis times probability of the association. It is generally useful to normalize the sum of the probabilities to one.

Each measurement is used to generate new hypotheses from all current hypotheses. There has to be a process to delete some hypotheses since the number of hypotheses would otherwise grow exponentially. Possible methods include low probability hypothesis pruning, N-scan pruning, and hypothesis merging [4]. In the method described in Chapter 3, only low probability hypothesis pruning and hypothesis merging are used.

2.3.4 Binary Sensor Network

Binary Sensor Network (BSN) is a collection of binary sensors, such as motion detectors. When the number of sensors is high enough, a BSN can be used to approximate the location of a target. Target tracking is suitable for this purpose because of its ability to combine previous hypotheses with new measurements. [10, 7]

Aslam et al. [7] have developed an algorithm for BSNs based on particle filtering. With their simulations, they were able to show some fundamental limitations of a BSN such as the existence of indistinguishable tracks.

2.3.5 Monte Carlo Markov Chain Data Association

Monte Carlo Markov Chains Data Association is an approach for data association designed especially for BSNs. Association is built iteratively from an initial guess.

MCMC is used to generate new associations from the current one. This algorithm is presented only for completeness on a theoretical level and is not used in the practical part of this thesis.

Let Y_t be all measurements up to time t. Y_t^1, Y_t^2 ... Y_t^K are the sets of measurements associated with targets 1, 2... K. Y^0 is the set of false alarms. Partition ω_t is the set of vectors Y_t^i

$$\omega_t = \left\{Y_t^1, Y_t^2, ...Y_t^K\right\}. \tag{13}$$

New partitions are generated in MCMC with the goal of finding a partition which explains the data as well as possible. The algorithm starts with an initial guess for ω_t^0. A new partition is generated from the current partition with some simple alternations, such as creating new targets or extending the current targets to account for more measurements. The probability of the new partition is evaluated. If this probability is larger than a random number between 0 and 1 the algorithm moves to the new partition. The new ω_t is then used to a generate a new generation of possible partitions. [10]

The probability of a partition can be calculated with Maximum a Posteriori method. Partition with maximum posterior probability $P(\omega|Y_t)$ is considered the best solution. [10]

Chen et al. [10] have used this method to track people in an open field with a binary sensor network. They used 144 PIR sensors in a 12x12 grid with 5 meter spacing between the sensors. Since each iteration of the algorithm processes all measurements up to time t it is not feasible to use this method on a live system. The method is designed for processing history data. [10]

2.4 Summary of Earlier Research

Summary of the research described in this chapter be found in Table 2. PAF has been calculated with Equation (3) in the case of occupant detection. For occupant counting the table lists the reported accuracy. Unfortunately, the method to calculate this varies from article to article and the numbers are not always comparable.

It is surprising that the time delay method gives much worse results than other occupant detection methods [13]. A possible explanation is that the test was done with a less than optimal setup.

Decision trees [9], BBNs, [21] and HMMs [11] have been successfully used for occupant detection. PAF of these methods is 97 % - 98 % and based on the published information it is not possible to tell if any of these methods is superior to the others. It is noteworthy that similar results have been achieved with much simpler algorithms [9, 11].

The CO_2 sensor is a natural choice for occupant counting. The dynamic algorithm can give reliable results in the right algorithm [26]. ANN is a good candidate to combine information from multiple different sensors but also requires plenty of teaching data [23, 29]. If one is willing to use video cameras, it is possible to use them to measure people flow rates in the entrances and exits of the area of interest [24].

Table 2: Overview of relevant earlier work in the fields of Occupant Detection (OD), Occupant Counting (OC) and Target Tracking (TT)

Author	**PIR**	**Other sensors**	**Goal**	**Methods**	**Acc.**
Von Neida [13]	Yes	-	OD	Time Delay	75 %
Hailemariam [9]	Yes	Multiple	OD	Decision tree	98 %
Dodier [21]	Yes	Telephone	OD	BBN	98 %
Batra [11]	Yes	-	OD	HMM	97 %
Labeodan [8]	No	CO_2	OC	Steady state CO_2	35 %
Wang[26]	No	CO_2	OC	Dynamic CO_2	-
Yang [23]	Yes	Multiple	OC	ANN	92 %
Meyn [24]	Yes	CO_2, cameras	OC	Kalman filter	89 %
Chen [10]	Yes	-	TT	MCMC	-

Target tracking has been successfully used with binary sensors [10]. It is possible to track multiple targets and even calculate their position based on only PIR sensors. To the author's knowledge, target tracking has not been applied to occupancy detection before.

3 Target Tracking in a Discrete Network

In this chapter, we describe our new proposal for tracking algorithm designed to handle discrete positions. The goal of our occupancy algorithm is to do occupancy detection as accurately as possible. Sensor adjacency information cannot easily be utilised with any other method but it gives valuable information about the structure of the space.

Position is typically considered to be continuous in target tracking. This approach is natural since a radar can measure position on a continuous scale. A binary sensor network is not able to provide this kind of information directly. It is still possible to calculate the continuous position from multiple observations [10, 7]. That requires the locations of the sensors and their ranges to be well known.

In our case, the exact positions of the sensors are not known. It is possible to estimate the location and use them for target tracking in a Binary Sensor Network (BSN) [6]. The efficiency of this approach depends highly on how accurate the positioning algorithm is. If the locations have a lot of error, the tracking algorithm cannot be expected to give good results. Discrete positions make the algorithm simpler to implement while still being accurate enough for our application.

We use only the minimal information about the structure of the space, which is modelled with a Hidden Markov Model (HMM) described in Section 3.5. PIR sensor signals correspond to HMM emissions (see Section 2.1.6). Initially, the transition and emission probabilities are given to the model as parameters, but it is also possible to calculate them from passive Infrared (PIR) sensor data (see Section 7.2).

Conventional tracking algorithms process the measurements in static time steps. In other words, time is treated as a discrete variable. It is often helpful to handle multiple measurements simultaneously since every time the algorithm abandons a hypothesis, information is lost permanently. A single hypothesis model discards all but the most likely hypothesis. Multiple hypotheses model can keep a limited amount of alternative hypotheses in memory, but some information is still lost in the hypotheses which are pruned off. [4]

Processing multiple observations at the same time complicates some issues. Global Nearest Neighbour (GNN) uses auction algorithm to decide how observations are associated with targets [4] as was explained in Section 2.3.2. There is no need for the auction algorithm since there is only one observation. In Section 3.1, we show how new hypotheses are generated from all possible movements. This kind of functionality might be challenging to implement for multiple observations.

Another issue to consider is sensor triggering frequency. Discrete time models are useful if the targets trigger the sensors constantly. Our sensors trigger a few times in a minute, so processing observations individually is more viable.

3.1 Generation of hypotheses

The algorithm maintains multiple hypotheses about the possible state of the system. In practice, a hypothesis consists of tracked targets (their states and some information about their history) and a probability of the hypothesis. This information is sufficient to calculate a new generation of hypotheses efficiently. In a mathematical sense, a hypothesis can be viewed as a set of previous association assumptions.

In a typical Multiple Hypothesis Tracking (MHT) system new hypotheses are generated from different associations of the measurements [4]. In the discrete case, there is also the possibility of generating new hypotheses from separate movements of the tracked targets.

Let Θ be an association and movement hypothesis for measurement y_k

$$\Theta : y_k \mapsto (T, N). \tag{14}$$

In other words, Θ is a mapping from the observation y_k to targets T and nodes N. A given Θ means that target T_n has moved to N_m and caused PIR measurement y_k from there. In the framework of an HMM, we can also say that T_n emitted y_k (see Section 2.1.6). To account for new tracks and false alarms, indexes $n = 0$ and $n = -1$ are used respectively.

Hypothesis H is defined as

$$H = \{\Theta, \hat{H}\}, \tag{15}$$

where $\hat{H}$ is the parent hypothesis from the previous time step. This recursive formula implies that H is the set of all association and movement hypotheses Θ up to that time.

Generating new hypotheses based on different movements is extremely useful in the case of a still target which frequently triggers sensor in an adjacent node. This phenomenon is illustrated in Figure 8.

The first diagram (a) in Figure 8 is the current hypothesis with one tracked target. Motion is detected at node N_2. Possible associations are

1. Target did not move and triggered N_2 from its current position. This leads to scenario (b).

2. Target moved to the adjacent node and triggered N_2 from there. This leads to scenario (c).

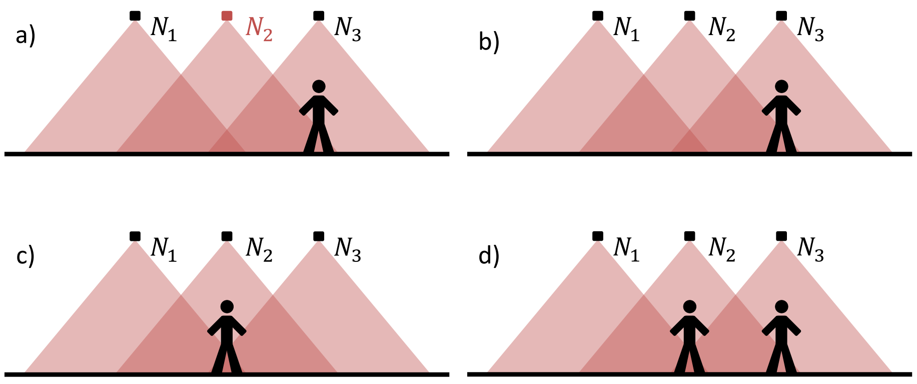


Figure 8: Current hypothesis (a) and three possible new scenarios (b, c, and d) due to a measurement from motion detector N_2.

3. y_k was a new track alarm. This leads to scenario (d).

4. y_k was a false alarm. This leads to scenario (b).

Since associations 1 and 4 result in the same outcome (the scenario in Figure 8 b), they would be merged as is described in Section 3.2.

Based on this one observation, the first two scenarios are the two most likely ones. If the next observation comes from N_1 the MHT algorithm would adapt hypothesis 2 as the most likely scenario. The next iteration could also be based on hypothesis 1 in the case the next observation is from N_3. If there are future measurements from both N_1 and N_3 then hypothesis 3 with a new target can become the most likely option.

3.2 Hypothesis Merging and Pruning

It is common that different associations lead to very similar hypotheses after a few time steps. In the case of discrete positions, this becomes even more important as it is possible that the two hypotheses have exactly same tracked target positions. In these cases, it is a good idea to merge the two hypotheses. Figure 9 shows how hypotheses 1 and 2 have identical tracked target positions and can be merged to hypothesis 3.

The probability of the merged hypothesis is simply the sum of the two prior probabilities. Last update time of the tracks can be merged with different approaches. We simply take the weighted average based on the probabilities of the hypotheses.

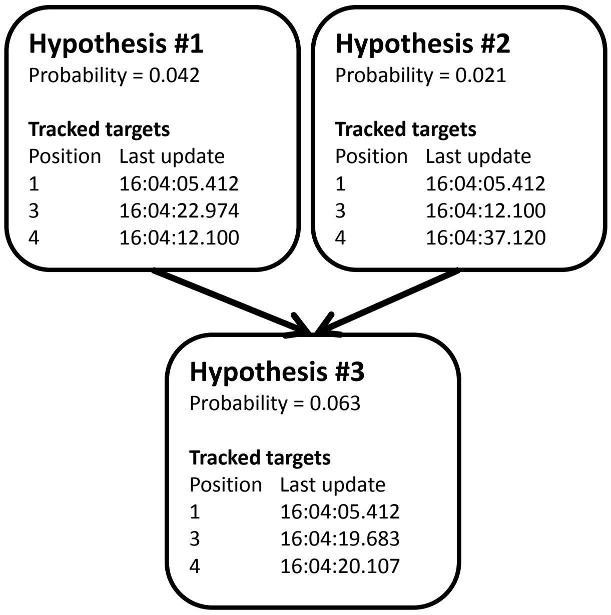


Figure 9: Merging of Hypothesis 1 and Hypothesis 2 into Hypothesis 3. The hypotheses can be merged because the target positions are identical.

In the example of Figure 9, last update time of track 3 is calculated from

$$\frac{0.042 \cdot 16{:}04{:}22.974 + 0.021 \cdot 16{:}04{:}12.100}{0.042 + 0.021} = 16{:}04{:}19.683. \tag{16}$$

The field of last update only affects the deletion of the tracks. The exact value of that variable is not crucial since it is used only to tell the difference between active and inactive tracks.

Merging is a good start for limiting the number of hypotheses, but alone it is not sufficient. Our approach also uses pruning. At the end of each iteration loop all but M most probable hypotheses are deleted.

3.3 Track Initiation and Deletion

In MHT, new track initiation is done with alternative hypotheses. Therefore, it is not necessary to use initialisation logic described in Section 2.3.

For every new measurement, a hypothesis is created to account for this measurement being caused by a new track. Possible nearby tracked targets are usually more probable explanations for the observation. In these cases, the hypothesis with a new target will soon get pruned off.

If there are no nearby tracked objects, possible explanations are a false alarm and a new track. It is usually preferable to choose parameters so that false alarm is a more probable explanation for a single observation than a new track [4]. In our case, we decided to choose $\lambda_{NT} > \lambda_{FA}$ because of the low rate of false alarms with PIR sensors. Thus a single new observation creates a new confirmed target and an alternative false alarm hypothesis with a smaller probability.

Tracked objects are deleted if there have not been updates to them in a given time period, which is also called lifetime of the targets. This lifetime can depend on the location of the target. In a border node from where targets frequently leave the monitored area, it is beneficial to have a relatively short lifetime, i.e. to delete the targets quickly after they have exited.

There might be interior areas from which exiting is not possible but which are outside the line of sight of the sensors, such as the locations of the occupants in Figure 1. Deleting the targets too early would result in errors in these cases. That can be avoided by setting the lifetime of the targets longer in the interior nodes.

In Chapter 4 we introduce two lifetime parameters: one for interior nodes and one for border nodes. Nodes in the vicinity of exits of the monitored area should be configured as border nodes. Interior nodes can include exits to some rooms as long as occupants always have to return to the monitored area before leaving entirely. That allows the tracker to keep them in the model even when they cannot be directly detected by the sensors.

Deletion of tracks could also be done by generating potential hypotheses to account for them being deleted and not deleted. However, this is not computationally efficient since deletion hypotheses would have to be created at every time step for every tracked target. The number of hypotheses would grow too high, and the deletion hypotheses would end up being pruned off. [32, 33]

3.4 Probability of a Hypothesis

Equations given in this Section were derived by us for this specific model. We used Bayes' theorem to reformulate the probability of hypothesis H for given observations up to time k. When the theorem is practiced with only to the most recent observation y_k

$$P(H|Y_k) = P(H|y, Y_{k-1}) = \frac{P(y_k|H, Y_{k-1}) \cdot P(H|Y_{k-1})}{P(y_k|Y_{k-1})} \tag{17}$$

Since $P(y_k|Y_{k-1})$ does not depend on H, it can be considered constant. Equation (15) can be applied to term $P(H|Y_{k-1})$ to get a linear relationship

$$P(H|Y_k) \propto P(y_k|H, Y_{k-1}) \cdot P(\Theta|\hat{H}, Y_{k-1}) \cdot P(\hat{H}|Y_{k-1}). \tag{18}$$

The probability of y_k and Θ depend only on the new hypothesis and the parent hypothesis, respectively. Y_{k-1} does not affect their conditional probability so it can be removed from these terms.

$$P(H|Y_k) \propto P(y_k|H) \cdot P(\Theta|\hat{H}) \cdot P(\hat{H}|Y_{k-1}) \tag{19}$$

Equation (19) corresponds to the fundamental theorem of target tracking [4].

The **second term** in Equation (19) is the probability of target T_n moving to node N_m and other targets not moving. In the case of false alarm or new track association, none of the targets move. For false alarm association ($n = -1$) one needs to compute the probability of targets staying at their current locations. For a new track association ($n = 0$) this has to be multiplied by the probability of a new track emerging in node N_m. All these cases can be combined to equation

$$P(\Theta|\hat{H}) = \begin{cases} 1 & \text{if } n = -1 \\ P_{\Delta t}(\text{a new track at } N_m) & \text{if } n = 0 \\ P_{\Delta t}(T_n \text{ moved to } N_m) & \text{otherwise.} \end{cases} \tag{20}$$

To be mathematically exact we should also include $P_{\Delta t}(\text{no new track at } N_m)$ to the first and last line. In practice, this is not necessary since the probability of a new track is tiny. Thus, the complement probability can be approximated as one.

Furthermore, this formula is missing the terms for targets $j \neq n$ not moving. The reason for this is, that if these probabilities were included it would also be required to create new hypotheses for all the possible movements. For a single target this could be done, but for a larger number of targets, the complexity of the algorithm grows exponentially.

The **first term** in Equation (19) is the probability of T_n emitting y_k and other targets not emitting anything. Because the probability is conditional for given H, T_n is assumed to be located at node N_m. When $n = -1$, the term corresponds to a false alarm at N_m and no emissions from the targets. In mathematical from this is

$$P(y_k|H) = \begin{cases} P_{\Delta t}(\text{false alarm at } y_k) \cdot \prod_j P_{\Delta t}(T_j \text{ didn't emit}) & \text{if } n = -1 \\ P_{\Delta t}(y_k \text{ emitted from } N_m) \cdot \prod_j P_{\Delta t}(T_j \text{ didn't emit}) & \text{if } n = 0 \\ P_{\Delta t}(y_k \text{ emitted from } N_m) \cdot \prod_{j \neq n} P_{\Delta t}(T_j \text{ didn't emit}) & \text{otherwise.} \end{cases} \tag{21}$$

In these equations, products over j are computed only for the previously recognised targets. For $n = 0$ the product does not include the new target. The same notation is be used for the rest of this chapter.

The **last term** $P(\hat{H}|Y_{k-1})$ in Equation (19) is the probability of the parent hypothesis $\hat{H}$. The probability of a hypothesis can be calculated recursively using Equation (19), Equation (21), and Equation (20).

Since same parent hypothesis is used generate multiple new hypotheses it is possible to speed up the calculation by defining

$$C_{\hat{H}} = \prod_j \cdot P_{\Delta t}(T_j \text{ didn't emit}). \tag{22}$$

Equation (19) can now be written in form

$$P(H|Y_k) \propto A_H \cdot C_{\hat{H}} \cdot P(\Theta|\hat{H}) \cdot P(\hat{H}|Y_{k-1}), \tag{23}$$

where A_H is similar to Equation (21)

$$A_H = \begin{cases} P_{\Delta t}(\text{false alarm at } y_k) & \text{if } n = -1 \\ P_{\Delta t}(y_k \text{ emitted from } N_m) & \text{if } n = 0 \\ \frac{P_{\Delta t}(y_k \text{ emitted from } N_m)}{P_{\Delta t}(T_n \text{ didn't emit})} & \text{otherwise.} \end{cases} \tag{24}$$

Most of the time this algorithm should be run only when a new observation arrives. There should also be a way to handle extended periods of not getting any measurements. Regular updates can be run without any new observation to keep the list of hypotheses up to date. In this case, each current hypothesis $\hat{H}$ is used to form only one new hypothesis H which are given relative probabilities according to Equation (23). Because there is no observations $A_H = 1$ and since none of the targets is moving $P(\Theta|\hat{H}) = 1$.

$$P(H|Y_k) \propto C_{\hat{H}} \cdot P(\hat{H}|Y_{k-1}), \tag{25}$$

It is possible that this calculation changes the most likely hypothesis. $C_{\hat{H}}$ is larger for hypotheses with a smaller number of targets.

Because constant $P(y_k|Y_{k-1})$ from Equation (17) was not calculated, the number we get here is only proportional to the actual probability. As the final step of the calculation, the probabilities should be normalized so that their sum is equal to one to keep their order of magnitude practical [4]. In our method, we normalize the sum of probabilities over all hypothesis to one.

It could also be possible to normalize the probability of all new hypotheses for the same parent hypothesis into 1 [33]. The disadvantage of this method is, that if there is only one child hypothesis (e.g. in the case of no new observations), it gets the same probability as its parent hypothesis. That is undesired behaviour since the lack of new observations does not decrease the probabilities of hypotheses with multiple targets.

3.5 Calculating Probabilities from the Markov Model

In this thesis, movements and emissions of tracked targets are modelled with a continuous time multi-agent Markov model. Sensor nodes correspond to Markov model states. Markov model agents represent the tracked targets which move between the states. Only a list of neighbours for each node is required to build a Markov model; it is not necessary to know the exact positions of the nodes.

In this section, we use false alarm rate Λ_{FA}, new track rate Λ_{NT}, emission rate matrix $\mathbf{\Lambda}_E$ and transition rate matrix $\mathbf{\Lambda}_T$ to compute all probabilities required in Section 3.4. These four parameters contain all the information we need about the sensor type and placement. They can be configured manually or possibly computed with some of the methods discussed later in Section 7.2.

We can now proceed to calculate probabilities in Equation (20) and Equation (24). Let us assume that false alarms are Poisson distributed and let λ_{FA} be the false alarm rate. Exponential distribution estimates the distribution for time interval t ($t \geq 0$) between the beginning of the observation and a false alarm. Probability density function for exponential distribution is

$$f(t) = \lambda_{FA} e^{-\lambda_{FA} t}. \tag{26}$$

Cumulative distribution corresponds to the probability that the false alarm occurred before given time Δt. Cumulative distribution for exponential distribution is the integral of the probability density

$$\int_0^{\Delta t} f(t)dt = \int_0^{\Delta t} \lambda_{FA} e^{-\lambda_{FA} t} dt = 1 - e^{-\lambda_{FA} \cdot \Delta t}. \tag{27}$$

That is also the probability of a false alarm in Equation (24)

$$P_{\Delta t}(\text{false alarm at } y_k) = 1 - e^{-\lambda_{FA} \cdot \Delta t}. \tag{28}$$

Respectively, the probability of a new track can be calculated from new track rate λ_{NT}

$$P_{\Delta t}(\text{a new track at } N_m) = 1 - e^{-\lambda_{NT} \cdot \Delta t}. \tag{29}$$

The first matrix to describe the relationships between the nodes is emission rate matrix $\mathbf{\Lambda}_E$. In the most simple case, the matrix has elements only in the main diagonal. These elements are rates at which targets trigger the motion sensor of their current node. PIR sensors often have overlapping lines of sight. They get easily triggered by targets which are closer to adjacent sensors than them. The element $\mathbf{\Lambda}_E[i,j]$ is the rate at which target at node i triggers a sensor at node j.

$\mathbf{E}_{\Delta t}$ is used to denote the emission matrix for time step Δt. The probability of a sensor at node j being triggered by a target at node j within the time period Δt can be calculated with a similar logic we used for Equation (28).

$$\mathbf{E}_{\Delta t}[i,j] = 1 - e^{-\Delta t \cdot \mathbf{\Lambda}_E[i,j]}. \tag{30}$$

Probability in Equation (24) is an element in the emission matrix

$$P_{\Delta t}(y_k \text{ emitted from } N_m) = \mathbf{E}_{\Delta t}[N_m, y_k]. \tag{31}$$

The probability of tracked target not emitting at all is used in Equation (22) and Equation (24). Target T_n at node N_m not triggering a sensor in node i has probability of $1 - \mathbf{E}_{\Delta t}[N_m, i]$. The probability T_n of not triggering a sensor anywhere is product of these probabilities or in other words

$$P_{\Delta t}(T_n \text{ didn't emit}) = \prod_j \left(1 - \mathbf{E}_{\Delta t}[N_m, j]\right). \tag{32}$$

Transition rate matrix $\mathbf{\Lambda}_T$ is also linked to the relationships between the nodes. Element $\mathbf{\Lambda}_T[i, j]$ (where $i \neq j$) represents the flow rate of tracked targets from node i to node j [22]. The diagonal elements are defined so that sum of each row is zero:

$$\mathbf{\Lambda}_T[i, i] = -\sum_{j \neq i} \mathbf{\Lambda}_T[i, j]. \tag{33}$$

Transition rate matrix can be used to compute transition matrix. Theory of continuous time Markov models is outside the scope of this thesis, but those interested can find more information from Norris' book [22]. The transition matrix for given time step Δt is given by exponential function

$$\mathbf{T}_{\Delta t} = e^{\Delta t \cdot \mathbf{\Lambda}_T}. \tag{34}$$

Note how this equation is different from (30). Instead of taking the exponential function for scalars, one has to use the exponential function for matrices. [22]

Element $\mathbf{T}_{\Delta t}[i, j]$ is the probability of tracked object moving from node i to j within time period of Δt. That is equal to the probability in Equation (20)

$$P_{\Delta t}(T_n \text{ moved to } N_m) = \mathbf{T}_{\Delta t}[s_{T_n}, N_m], \tag{35}$$

where s_{T_n} is the position of target T_n.

When $\Delta t \to 0$, transition matrix approaches to the identity matrix. That is intuitive since when the time step is small enough, it is unlikely that target has moved away from any of the nodes. [22]

3.6 Adding of an Initialisation Hypotheses

A hypothesis without any targets can increase its probability rapidly according to Equation 25 when all occupants have left the area. The existence of this kind of hypothesis can have a large impact on the outcome of the tracker. If the maximum number of hypotheses is small, empty hypothesis is not usually included in the list of hypotheses. If something goes wrong and the occupants manage to leave without the tracker noticing, it has to wait L_I to be able to delete the targets and acquire a hypothesis without any targets.

This method can be considered as a way to initialise the tracker with a small probability. The probability of the new hypothesis was set to one tenth of the probability of previously least likely hypothesis.

3.7 Algorithm Outline

In this chapter we went through the details of an applied target tracking algorithm. The steps can be summarised as following:

1. Receive a new observation y_k or receive the trigger to run the algorithm without an observation.
2. Generate all possible association hypotheses for each parent hypothesis $\hat{H}$.
3. Generate all possible movement hypotheses for each association hypothesis. This and the previous step correspond to Equation (14). Calculate probabilities of the hypotheses with Equation (23) or (25).
4. Delete tracks which have no recent observations as described in Section 3.3.
5. Combine hypotheses from different parent hypotheses to a single list. Merge similar hypotheses as described in Section 3.2.
6. Limit the number of hypotheses to M most probable hypotheses as described in Section 3.2.
7. Add initialisation hypothesis as Section 3.6 describes.
8. Normalize the sum of probabilities of the hypotheses to one.

The algorithm begins with just one hypothesis, which has no tracked targets and has a probability of 1.

4 Materials & Methods

Let us now focus on the methods for testing these approaches in practice. Section 4.1.1 describes the characteristics of passive infrared (PIR) sensors, which are be used as the primary data source. Different test setups are explained in Section 4.2. Finally, in Section 4.3 we return to target tracking and decide how the parameters mentioned in Chapter 3 should be chosen.

4.1 Test Equipment

The test equipment consists of three parts

1. Motion sensors with a binary output (see Section 4.1.1)
2. Node modules which read the value of the motion sensors and send a wireless message if motion has been detected. They also listen to the wireless messages sent by other nodes and change their states according to their internal logic.
3. Gateway, which listens to all wireless messages, writes them into a log file and (optionally) uploads the data to the Internet.

Wireless messages are sent not only because of the detected motion but also from state transitions of the nodes. The exact nature of these states is not relevant to this work. It is only necessary to know that there are five states and that the devices broadcast all their state transitions. That allows us to estimate the success rate of the wireless connection with the method described in Section 4.1.2.

4.1.1 Motion Sensor

We used Helvar Active+ Sense sensors [34] in all test setups. They consist of a PIR sensor and a photodetector. The photodetector was not utilised in this thesis, but we discuss it in Chapter 7. Picture of the sensor can be seen in Figure 10.

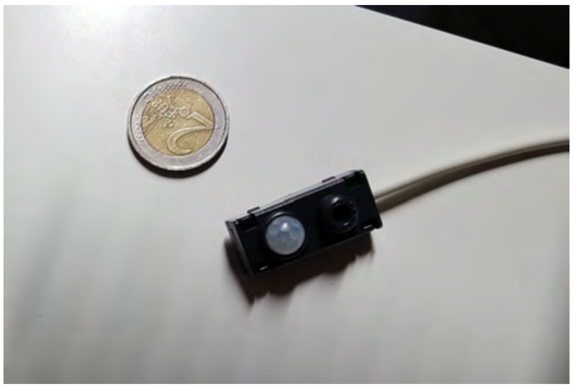


Figure 10: Helvar Active+ Sense and a coin for scale. PIR sensor is on the left and light sensor is on the right.

The angle of coverage for PIR sensor is 80°. When the sensors are placed at the height of 2 meters they cover an area with a radius of approximately 2 meters. Our method for generation of hypotheses (Section 3.1) is based on the assumption that targets trigger sensors also in the adjacent nodes. If the distance between the sensors is less than 4 meters, this assumption is justified.

4.1.2 Estimation of Success Rate

A state transition message contains both the previous state and the next state. This information allows us to approximate the probability of the gateway successfully receiving a message from a given node. First, the number of messages where the previous state differs from the next state of the previous message is calculated and labeled as N_F. It is guaranteed that one state transition message has been lost between these two messages. Thus N_F is the lower limit of lost messages.

We can detect a lost message only if that message has contained an actual state transition. Therefore, N_F should be compared to the total amount of messages containing a state transition. If the count of successfully received state transitions is N_S, the total number of this kind of messages is $N_F + N_S$. So the lower limit of error rate is

$$R = \frac{N_F}{N_S + N_F} \tag{36}$$

Note that this equation should only be used if R is small. If the error rate is high, it is more likely that two lost messages cancel each other out, and the error is left undetected.

4.2 Test Setups

In this thesis, we used four different test setups for different purposes. The first setup generated simulated data on a single computer and used that to test different types target tracking in a corridor. The test installation described in chapter 4.2.2 is also ultimately a simulation, but in this case, only motion detectors (Section 4.1.1) were simulated. Remote nodes sent the motion sensor data to the wireless network, which was first collected and logged by a Gateway and only after that analyzed with the target tracking algorithms.

In the controlled test environment case study real PIR sensors were attached to the remote nodes in a small office area. The output of these sensors was saved to the Gateway and later analysed with the target tracker. The movement of the occupant was documented manually and then compared to the output of the tracker.

In an actual open office, it is hard to collect accurate ground truth data. In the last test setup, we only tested if the method gives plausible results and how they compare to the time delay method.

4.2.1 Simulated Corridor

A single corridor is an uncomplicated environment for initial target tracker tests. Since position has only one dimension, it is easy to visualise position and time in a two-dimensional chart. We generated simulated data from a corridor to test how the algorithm can handle two targets which move past each other. Such test is common for the testing of the core functionality of a target tracker since it is simple, provides real challenges to the algorithm, and is easy to visualise in two dimensions.

The simulation consisted of 30 sensor nodes. Simulated targets could only move to the adjacent nodes: they stay still or move one step up or down. The targets were simulated to trigger measurements in both their current node and the adjacent nodes. Clutter was also generated by sending completely random observations to the tracker.

4.2.2 Test Cabinet

More advanced tests were done with a test cabinet. The test cabinet consisted of 25 sensor nodes simulating open office area with a 5 by 5 grid of PIR sensors. A computer was used to simulate the occupants and send motion detector signals to the nodes. The central goal of this setup was to test the wireless connections and software of the nodes, but the data was also found useful in the development of the target tracker.

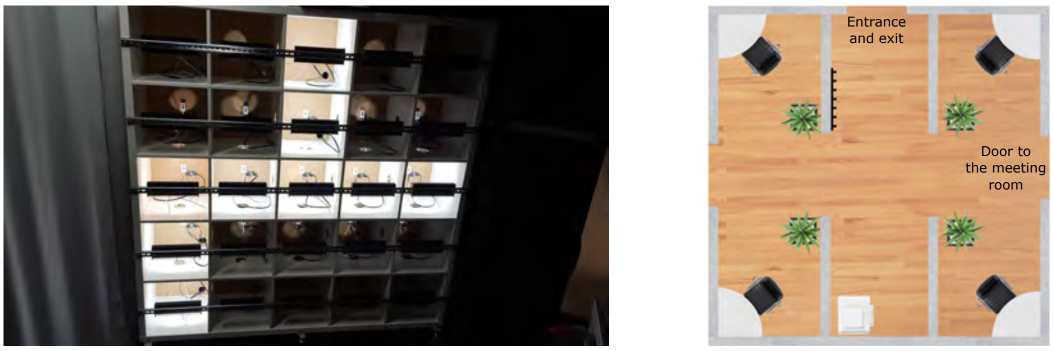


Figure 11: Left: Paths of two occupants have been illuminated in the test cabinet. Right: Layout of the simulated office.

The simulation consisted of four occupants who came to the office in the morning, had some periods of absence during the day (meetings, lunch) and left in the evening.

The targets triggered nodes while moving. When they were sitting at their desks, they triggered the nearest sensor once every 30 s to 35 s. Of course, this is not very realistic since the interval between the observations is always over 30 s. Additionally, there was no motion simulated to the adjacent sensors. However, the advantage of this setup is that the ground truth is very accurate, and the data are thus suitable for our parameter optimisation.

4.2.3 Controlled Office Environment

These tests were performed in an empty office building. The test setup consisted of eight nodes with PIR sensors. Sensor placement is shown in Figure 12. Sensor numbers also shown in the figure are used in Section 5.3.2.

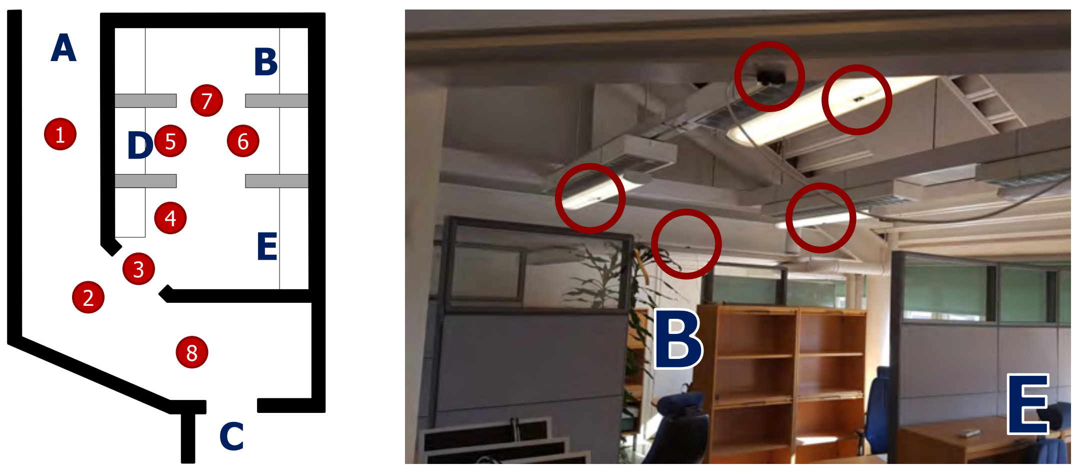


Figure 12: Test layout (left) and a picture of the area (right). PIR-sensors are marked with red.

On the left, we can see the layout of the office area, sensor positions (red) and areas where the occupant spent time (A-E). The picture on the right is a from the same area. Five sensors inside of the room are highlighted with red circles. Areas B and E are also visible in the picture. Area D is behind of the gray screen on the left. Test arrangement had two exits (A and C). The three nodes in the corridor can only be seen on the layout but not in the picture. These were defined as border nodes in the target tracker.

If the occupant was spending time at position D, the motion sensor was able to detect him sporadically. At positions B and E the sensors were so far away that constant measurements were not possible. This setup was designed to demonstrate the capabilities of the target tracker to "remember" that an occupant had entered these areas and to keep the area in occupied state until the occupant is detected leaving the area entirely.

During the test, accurate notes about the positions of the occupants were made. They were used to generate a ground truth data set which indicated exactly when the area became occupied and unoccupied. This dataset is utilised in Section 5.3 and Chapter 6 to calculate the key ratios for real sensors.

4.2.4 Case Study in an Open Office

Eight test sensors were placed into a small open office area. Figure 13 shows the layout of the room and sensor placements.

In this test case, the gateway was not only recording the PIR observations and writing them into a log file but also sending all measurements to a cloud server. Target tracking was done on the server with the real time data.

This setup enables extending the system to a wider area where it is not possible to establish a connection between all of the nodes and the gateway. Limitations of the wireless network in our use are not known, but this would certainly require dozens of nodes. In this case, there could be multiple gateways collecting the data.

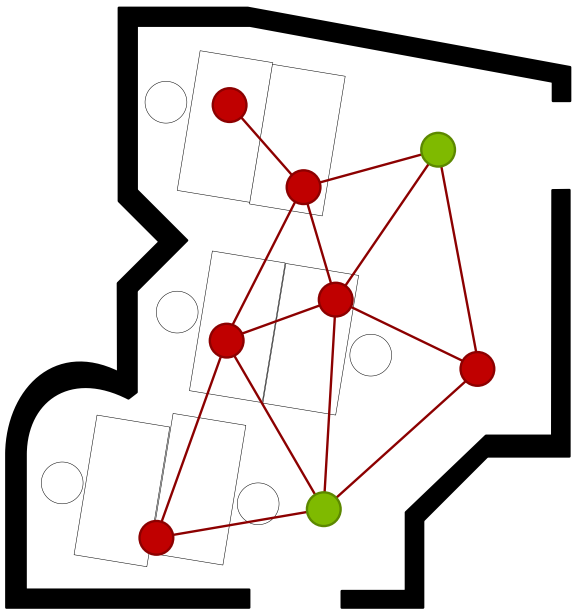


Figure 13: Office floorplan, sensor nodes and used internode adjacency relationships. Green and red represent border and interior nodes, respectively.

It also ensures that there is always enough computational resources for the target tracking.

Unfortunately, we were not able to collect the ground truth data from this test installation. Thus in the analysis of results in Section 5.4 we are limited to evaluating the plausibility of the results and comparing the energy usage to that of the time delay method.

4.3 Target Tracker Parameters

In Section 3.5 we discussed how the required probabilities can be calculated from model parameters. The parameters were false alarm rate, new track rate, emission rate matrix, and transition rate matrix.

In a mathematical sense, this list of four parameters is already very simple. However, in the practical sense, the matrices are complex since they consist of multiple elements. To eliminate the need for these matrices, we introduce the adjacency matrix

$$\mathbf{A}[i,j] = \begin{cases} 1, & \text{if nodes } i \text{ and } j \text{ are adjacent} \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

We can generate a simple transition rate matrix using the adjacency matrix

$$\mathbf{\Lambda}_T[i,j] = \begin{cases} \lambda_T \cdot \mathbf{A}[i,j], & \text{if } i \neq j \\ -\lambda_T \sum\limits_{j \neq i} \mathbf{A}[i,j], & \text{if } i = j, \end{cases} \tag{38}$$

where λ_T is the new transition rate parameter. Diagonal elements ($i = j$) are calculated with a Equation (33) to ensure that the sum of each row is zero.

With the sensor triggering rate parameter λ_E we can use the adjacency matrix to generate emission rate matrix as

$$\mathbf{\Lambda}_E[i,j] = \begin{cases} k \cdot \lambda_E \cdot \mathbf{A}[i,j], & \text{if } i \neq j \\ \lambda_E, & \text{if } i = j, \end{cases} \tag{39}$$

where k is the ratio between the probability of triggering the nearest PIR sensor and of triggering an adjacent PIR sensor.

At this point, some readers might wonder why we want to give all node pairs the same adjacent node triggering rate. Certainly, if the adjacent nodes are very close to each other measurements in the other node are far more likely than for two nodes which are farther apart from each other. That is the case if some knowledge about the distance between the nodes is available. This thesis focuses on a more simple alternative, where the topology of the node network is the only information source.

False alarm rate λ_{FA} and new track rate λ_{NT} were also described in Section 3.5. Since these are already scalar values, they can be included in the parameter list in Table 3.

Table 3: List of target tracking parameters and their default values.

Parameter	**Default value**	**Description**
λ_T	100 mHz	Target transition rate to adjacent nodes
λ_E	100 mHz	Rate at which targets trigger the nearest sensor
k	0.1	Adjacent sensor triggering rate is $k \cdot \lambda_E$
λ_{FA}	10 nHz	False alarm rate
λ_{NT}	100 µHz	New track rate
L_I	20 min	Lifetime of the targets in interior nodes
L_B	10 s	Lifetime of the targets in border nodes (e.g. building entrances)
M	10	Maximum number of hypotheses

Table 3 also includes two lifetime parameters: one for interior nodes and one for border nodes. These are used for track deletion (see Section 3.3). If they are given a different value, it is important to consider how the division between border nodes and interior nodes are done.

To run the tracker we need values for eight scalar parameters, a true/false adjacency matrix and a true/false vector indicating which nodes are border nodes and which are not. In Chapter 5 our goal is to optimise the scalar parameters so that the same values could be used in all real life situations. After that only the structure of the network is required in the form of adjacency matrix and border node vector.

5 Results

In this chapter, we test the algorithm from Chapter 3 in simulated and real life environments. We first examine the fundamental properties of the approach in Section 5.1. These results are based on a computer simulation.

In Section 5.2 we go through parameters in Table 3 one by one and check how they affect the results. The data for these tests was collected with the test cabinet: actual wireless devices which received simulated motion detector input data.

In Section 5.3 we use real sensors recording human behaviour in a controlled office environment. The status of occupancy is determined with both target tracker and the time delay method so that comparison between the two methods can be made in Chapter 6.

Finally, in Section 5.4 the data are extracted from a real test installation in a normal open office area described in Section 4.2.4. Unfortunately, ground truth data similar to the controlled test environment is not available. Therefore the analysis is limited to evaluating the plausibility of the results.

5.1 Preliminary Tests

First Multiple Hypothesis Tracker (MHT) is compared to Local Nearest Neighbour (LNN) and Global Nearest Neighbour (GNN) in Section 5.1.1. The results in a simple simulated environment show that MHT is the most promising approach of these three when working with discrete sensor data, but a GNN might also be sufficient.

In Section 5.1.2 we test how sensitive the results are to the length of the time step Δt. The initial hypothesis was that the value of the time step is not significant, and the algorithm could be simplified by eliminating the extra variable.

In Section 5.1.3 we test hypothesis merging and see how it affects the hypothesis trees. Initially merging was discovered to be computationally expensive but this obstacle was solved with an improved algorithm.

5.1.1 Simulated Corridor

Setup for these simulations was described in Section 4.2.1. Raw sensor data are shown in the top left graph of Figure 14.

We begin by inspecting the **raw data** on the top left of Figure 14. These measurements are caused by two targets walking in a corridor to opposite directions, meeting in the middle and then continuing in their own directions. Note that it is also possible that the targets turn around when they meet and return in the direction they came from. With the given data both of these two scenarios are possible explanations.

There was plenty measurements in this simulation. The results in Section 5.3 indicate that with real sensors the measurements are get sparser. Another inaccuracy compared to the real life scenario is that real PIR sensors do not have completely random clutter. There can be some unexpected measurements, but there is almost always an occupant somewhere near to the sensor.

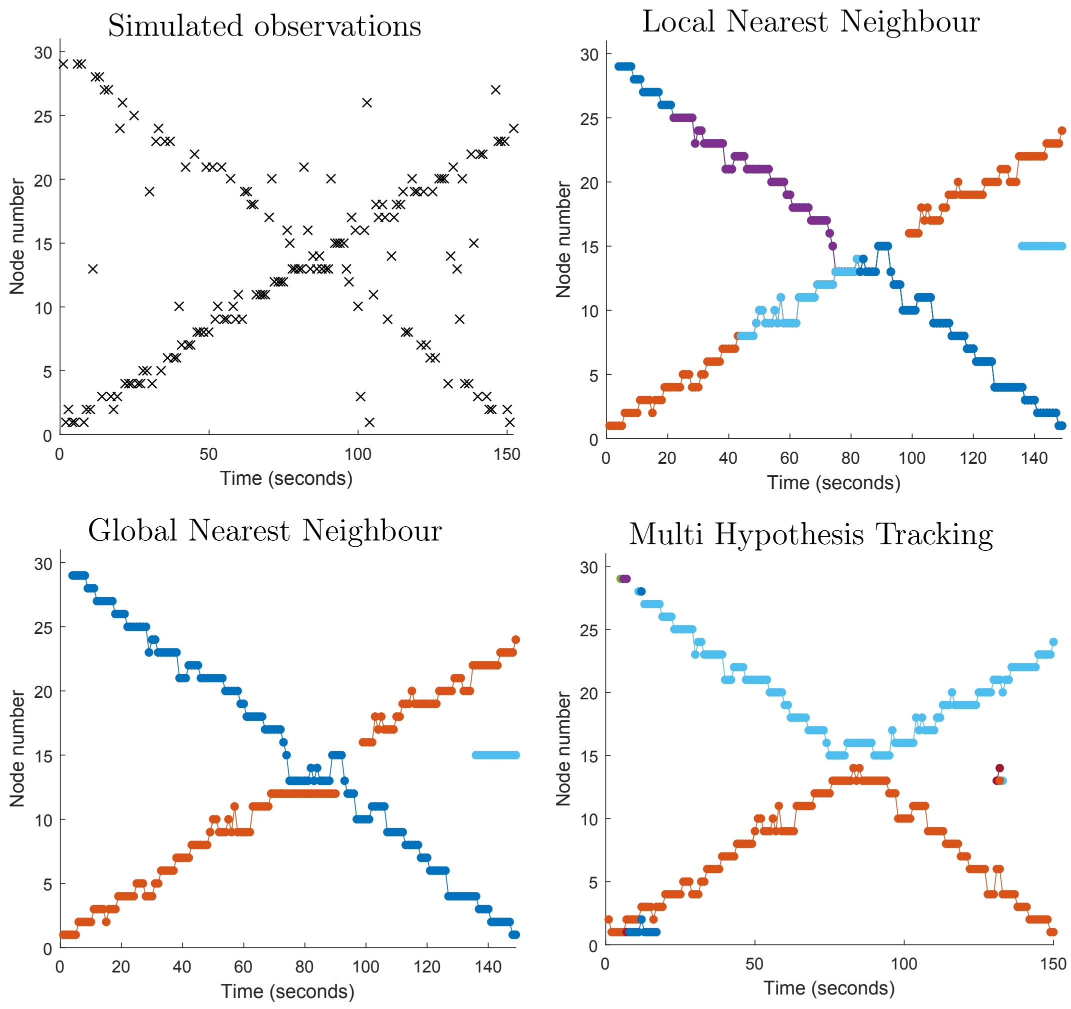


Figure 14: Raster plot of the simulated binary detections is presented at top left. The other graphs are the positions calculated with LNN, GNN, and MHT. Different colours indicate different tracked targets.

LNN algorithm, shown at top right in Figure 14, can capture the general idea of the tracked objects. First, there is correctly only two targets (orange and dark blue). Around 25 seconds from the start, a purple target appears. It is not visible in the graph, but in reality, the purple target is overlapping with the dark blue which is not visible after the appearance of the purple target. Similarly, light blue target starts to overlap with the orange after around 45 seconds. Close to the end, another light blue target appears in an area with no real targets. It was caused by coinciding false alarms around that area.

Results with **GNN** are more accurate as there are just two notable tracked targets (orange and dark blue). Algorithm is not flawless when the two targets are close to each other, but it catches on after a couple of observations. As previously, a third target (light blue) appears close to the end of the simulation, due to multiple coincident false alarms.

The most accurate interpretation for this data set is given by the **MHT**. There are two tracked targets (orange and light blue), and their tracks are continuous through the whole data set. MHT assumes that after the targets meet in the middle, they turn around and walk back to where they came from. As mentioned earlier, this outcome can be justified since that kind of simulation would produce a similar data set.

5.1.2 Sensitivity to the Length of the Time Step

Preliminary tests revealed that the tracker is not very sensitive to the length of the time step. Time step Δt is used in Equation (28), Equation (29), Equation (30), and Equation (34) to calculate Markov model related probabilities. In this section, we test an alternative algorithm where variable Δt is set to one second regardless of the actual time difference.

To examine this in more detail, we take a closer look at the first 20 seconds of simulation test case from Section 5.1.1. Figure 15 illustrates the results in the form of hypothesis trees. Hypothesis merging was not used in this test.

In Figure 15 the horizontal axis represents time. Arrows between the hypotheses indicate which hypothesis from the previous generation was used for generating of each hypothesis. All hypotheses are used to generate child hypotheses but because of pruning the child hypotheses might not be visible on this chart.

On each column, the most likely hypothesis is the one with the largest marker. The most likely hypotheses are also highlighted with red. The most likely hypothesis

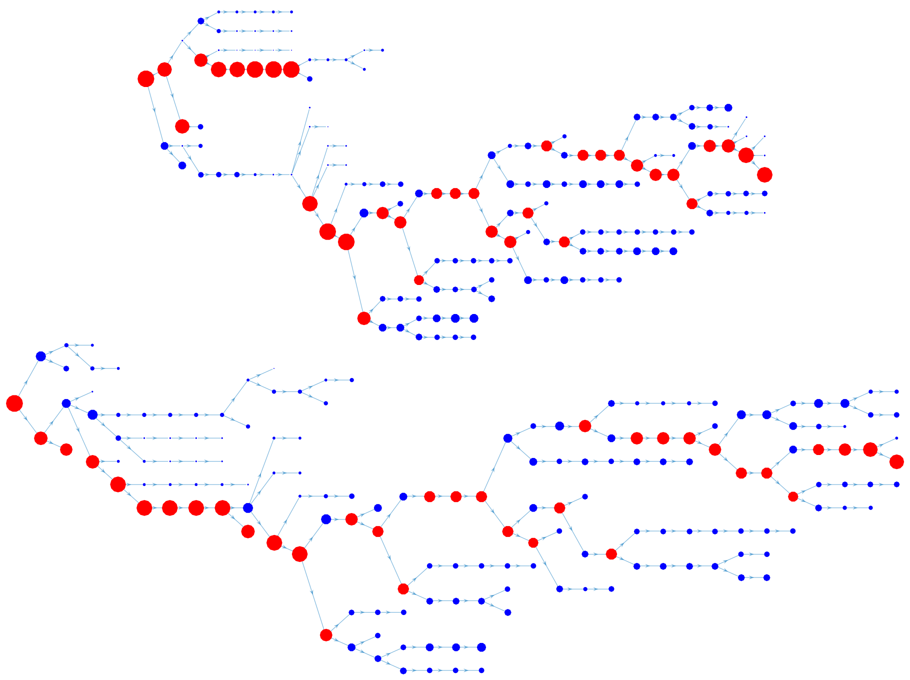


Figure 15: Hypothesis tree with the normal algorithm (top) and a constant time step of $\Delta t = 1$ (bottom). The probability of a hypothesis is proportional to the area of the circle and most likely hypotheses are highlighted with red. Time is on the x-axis and arrows represent parent-child relationships.

frequently jumps to another branch of the tree. This indicates that GNN, which is unable to do that kind of corrections, would not perform as we as MHT.

The two graphs in Figure 15 are very similar but not identical. While the general shape of the graph and the path of the most likely hypothesis are the same, there are some differences in the probabilities of the less likely hypotheses. To confirm this, we also compared the outputs (the tracked target locations for the most likely hypothesis) of the two algorithms. They are identical except for one small difference. The standard algorithm recognised a second tracked object at time step 9 while constant time step method confirmed it at time step 10.

For the rest of this chapter, we use the standard method with variable time step.

5.1.3 Hypothesis Merging

Next, we implemented merging of hypotheses. The hypothesis tree looks very different after merging is implemented. Multiple hypotheses from the previous generation can now be used to generate a new hypothesis because of the merging of the child hypotheses. Hypothesis tree in Figure 16 shows analysis of the same dataset as in Section 5.1.2 shown without merging in Figure 15, but with a maximum number of hypotheses limited to 5. This choice was made to keep the graph readable since merging has radically increased the number of child-parent relationships. In practice, it is beneficial to choose a larger maximum number of hypotheses than 5.

In Section 3.2 we anticipated that merging might be computationally expensive since the number of hypotheses is limited to M only after the merging. To test this, we compared run time between algorithm which made use of merging and a simplified version where merging was skipped. The merging was initially done by comparing all hypothesis pairs to see if they are equal. As expected, this resulted in $O(n^2)$ complexity, which can also be seen on the right in Figure 16. The second-degree

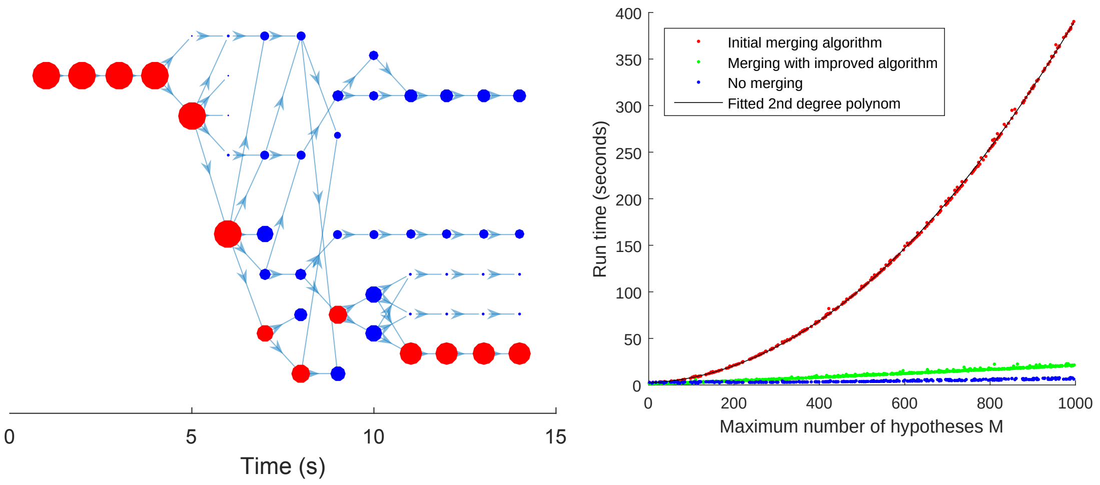


Figure 16: Hypothesis tree with merging in effect (left) and run time of different algorithms (right)

polynomial fits perfectly to the runtime of the initial merging algorithm.

A more advanced method is to sort the hypotheses in a way which makes similar hypotheses subsequent. This kind of sorting can be done in $O(n \log n)$. After that step, similar hypotheses can be combined efficiently in $O(n)$. With this improvement, run times with and without merging had the same order of magnitude. Source code for the two algorithms used in this section can be found from Appendix B.

5.2 Parameter Optimisation in a Simulated Environment

In this section, we use the test cabinet (see Section 4.2.2) to find out how parameters (see Section 4.3) affect the tracker behaviour.

We start examining of lifetime of the targets parameters in Section 5.2.1 with interior node lifetime. In Section 5.2.2 the model is further improved by adding a different lifetime to the border nodes. However, this addition reveals some issues which required the introduction of initialisation hypothesis and extending of the borders.

In Section 5.2.3 and Section 5.2.4 we effectively change the modeled triggering rate of closest sensor and those adjacent to it, respectively. Since observations in adjacent nodes were not simulated one should exercise caution when applying the results in this section. In these tests the value of k did not have any significant effect on the tracker.

Finally, in Section 5.2.5, Section 5.2.6, and Section 5.2.7 we test the three remaining rate parameters λ_T, λ_{FA}, and λ_{NT}, respectively.

The implications of these results are be discussed in Section 6.2.

5.2.1 Lifetime in Interior Nodes

In this initial test, we did not define any interior nodes, and all nodes were treated as if they were on the border. Thus we only have one lifetime parameter to optimise. Results with four different parameter values are shown in Figure 17.

Each graph in Figure 17 has the ground truth (G_t) on the left and estimated occupancy (E_t) on the right. The maximum number of hypotheses M was set to 5 to speed up the calculations. Simulation time is presented on the y-axis. Black lines represent the time periods when the area was occupied (1) while white line means that it was unoccupied (0). Errors in the predictions are highlighted with red if $G_t = 1$ and blue if $G_t = 0$.

The same analysis was run with 2000 values for the lifetime between 0 s and 50 s. Each of the error plots in Figure 17 is described as a single column of pixels. These have been combined for the graph on the left in Figure 18.

As Figure 17 shows, when lifetime of targets is under 5 s, the algorithm regularly fails to detect occupancy. Once the lifetime is increased, the errors gradually decrease until most of them disappear around a lifetime of 35 s. Beyond 35 s, the lifetime does not have a large impact on the results except for increased time of estimated occupancy after actual occupancy has ceased.

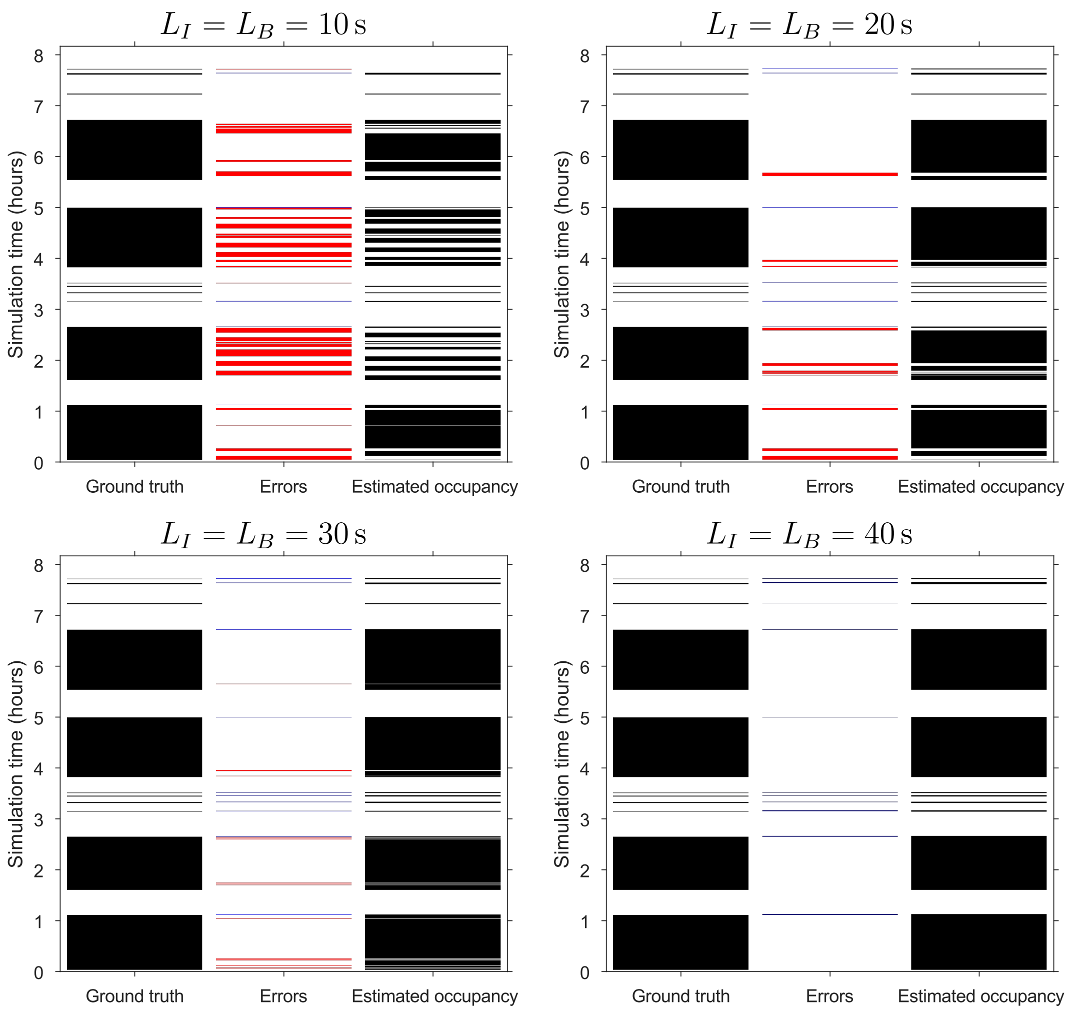


Figure 17: Real occupancy status of a room and status estimated with target tracker using different lifetimes of the targets. Errors are marked with red when the space was occupied (UCF error) and blue when unoccupied (ECF error).

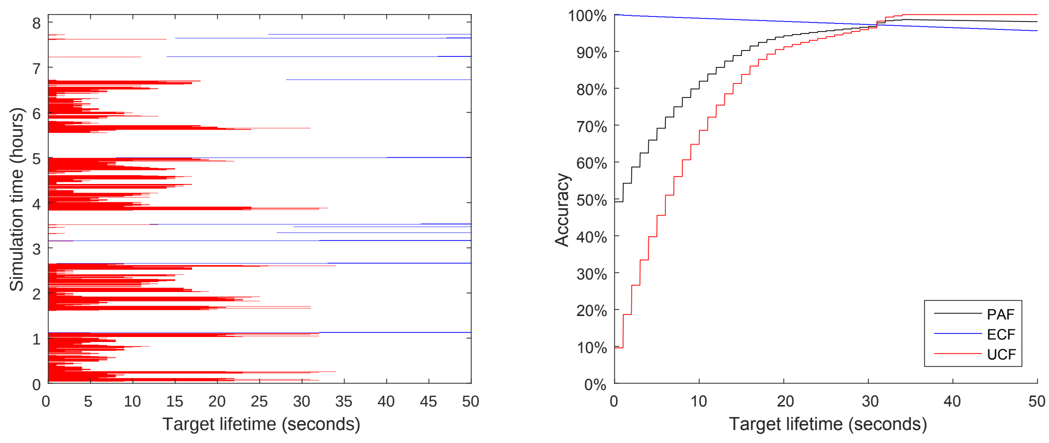


Figure 18: Error barcodes for different interior node lifetimes (left) and key ratios as a function of lifetime of the targets (right).

Duration of red and blue error affect to the User Comfort Factor (UCF) and Energy Conservation Factor (ECF), respectively. These and the Prediction Accuracy Factor (PAF) calculated with Equation (5), Equation (4), and Equation (3) and are shown in graph on the right in Figure 18.

5.2.2 Border Lifetime and Adding of an Initialisation Hypothesis

We now proceed to use a different lifetime value for border and interior nodes. That allows us to have the best of both worlds: user comfort from long-lasting targets in interior nodes and energy savings from short lifetimes on the border nodes.

Until this point, we have used the same value for lifetimes of the targets in interior nodes L_I and border nodes L_B. The parameters should be given different values to get the real benefit from a target tracking model. This way targets, which have been successfully tracked to the interior nodes, become more significant and stable than the targets which have been tracked to exits.

This approach was first tested by making only the nodes directly adjacent to entrances border nodes. The top left graph in Figure 19 shows the results with this method when other parameters were given standard values (see Table 3). The maximum number of hypotheses M is 10 with the standard settings. As can be seen from the bar code plot, the targets sometimes persist in the model long after all simulated occupants have left the office. Thus, this method alone is not able to give satisfactory results.

A closer look at the tracking results reveals that this was caused by multiple targets leaving the area simultaneously. The border node detected motion only once, which could be used to move only one of the targets to the border node. Rest of the targets were left behind to the nodes adjacent to the border and thus were not deleted within the short deletion time.

One way to deal with this issue is to increase the maximum number of hypotheses. If there are no new signals from PIR sensor, the hypotheses with fewer targets become more probable over time. We can see that that with 100 hypotheses this issue is solved on the top right graph of Figure 19.

The algorithm was updated to add an initialisation hypothesis after every time step, as described in Section 3.6, to help to remove this kind of instability. These results are shown in Figure 19 bottom left. With a maximum of 10 hypotheses, the problem persists but is solved with a maximum of about 20 hypotheses.

A third successful improvement was to extend the border to adjacent nodes. In practice, this means that lifetime was 10 s in the two nodes next to exits and the six other nodes adjacent to them. These results are shown in Figure 19 bottom right. There are almost no errors even when the maximum number of hypotheses is only 10.

In Figure 20 we can see how an increasing number of hypotheses affects these two new approaches. Lifetime at interior nodes L_I was set to 100 min to enhance the effect further. All other parameters were kept at default values.

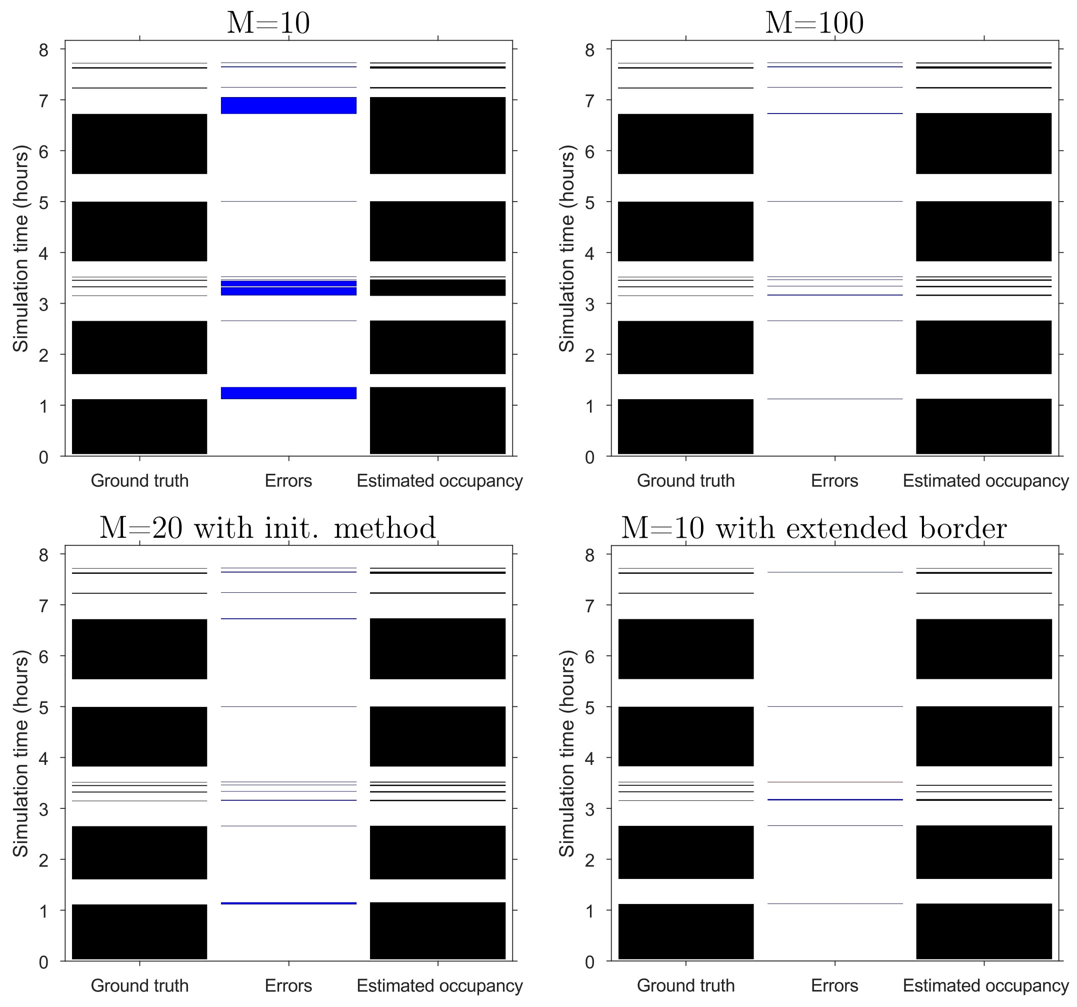


Figure 19: Results when lifetime is set to 10 s at border nodes and 20 min at interior nodes. Init. method means adding of the initialisation hypothesis with a small probability.

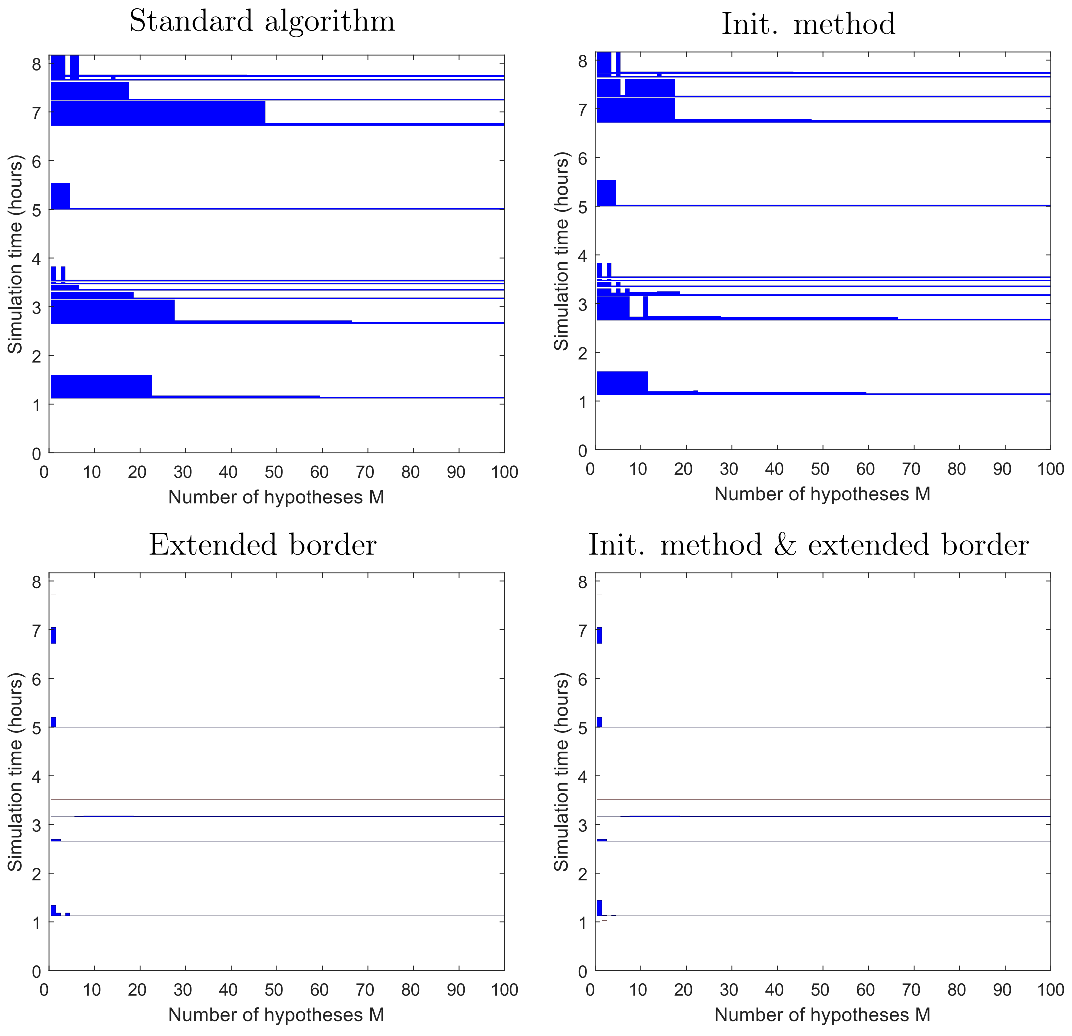


Figure 20: Errors with different number of hypotheses (M) using alternative methods. Init. method means adding an initialisation hypothesis after every time step.

5.2.3 Sensor Triggering Rate

We now return to defining all nodes as border nodes again, which allows us to examine parameter λ_E closer. As explained in Section 4.3, sensor triggering rate λ_E denotes the rate at which targets trigger their nearest PIR sensor. In this section, we assume that triggering a neighbouring PIR sensor is one tenth as likely, or in other words $k = 0.1$ (see Table 3).

In a preliminary test, the same node was repeatedly triggered by one-second intervals. In real life, this would correspond to an occupant staying close to a single sensor, but moving and regularly triggering the PIR sensor. The correct conclusion from that kind of data would be that there is only one tracked target. However, with a bad choice of tracker parameters and a high rate of observations more than one target can be created in the model.

When λ_E is set to have smaller values than 1 Hz the algorithm correctly deduces that there should be more than one occupant in that node. The deduction is not made instantly but requires a sufficient number of observations from the node. These results are shown in Figure 21.

Within the first 100 s, a single-target-hypothesis is always most likely when $\lambda_E > 450\,\mathrm{mHz}$ (cyan area). One target is the most likely explanation for all values of λ_E up to around 30 Hz. After that rounding errors start to interfere with the

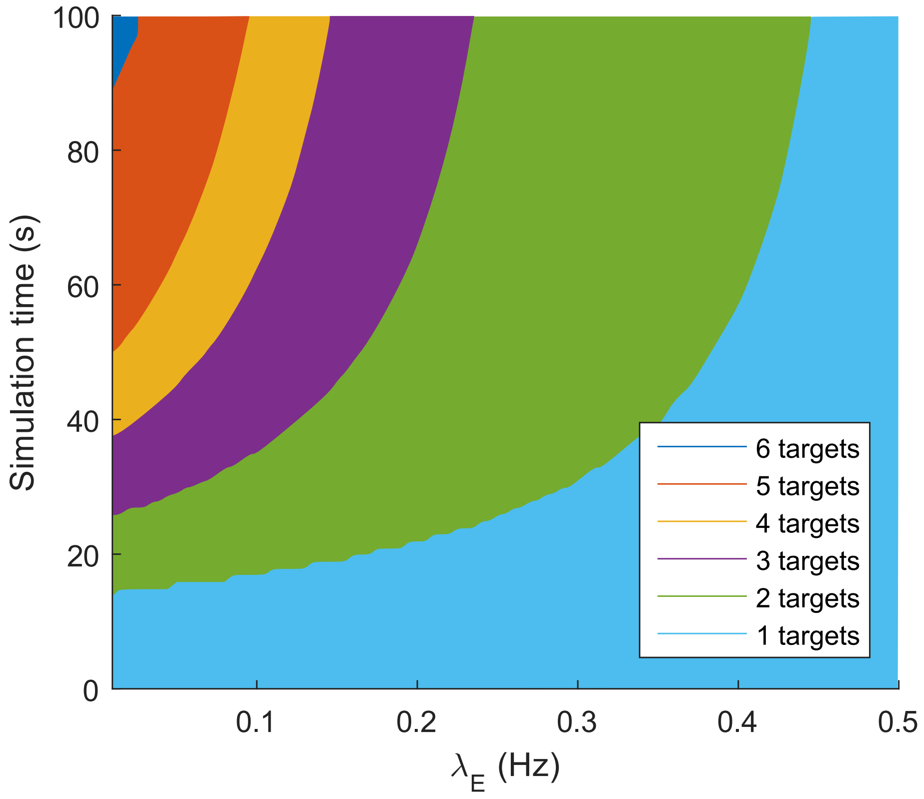


Figure 21: The modeled number of targets to explain constant measurements at a rate of 1 Hz as the function of sensor triggering rate λ_{E1}. The expected result is one target.

computations and the algorithm no longer works.

Another test cabinet simulation was used to test this parameter with more realistic data. Barcode graphs with values $\lambda_E = 500\,\text{mHz}$ and $\lambda_E = 5\,\text{Hz}$ are shown in Figure 22. Both lifetime parameters were set to 100 s in this test. Default values in Table 3 were used for other parameters.

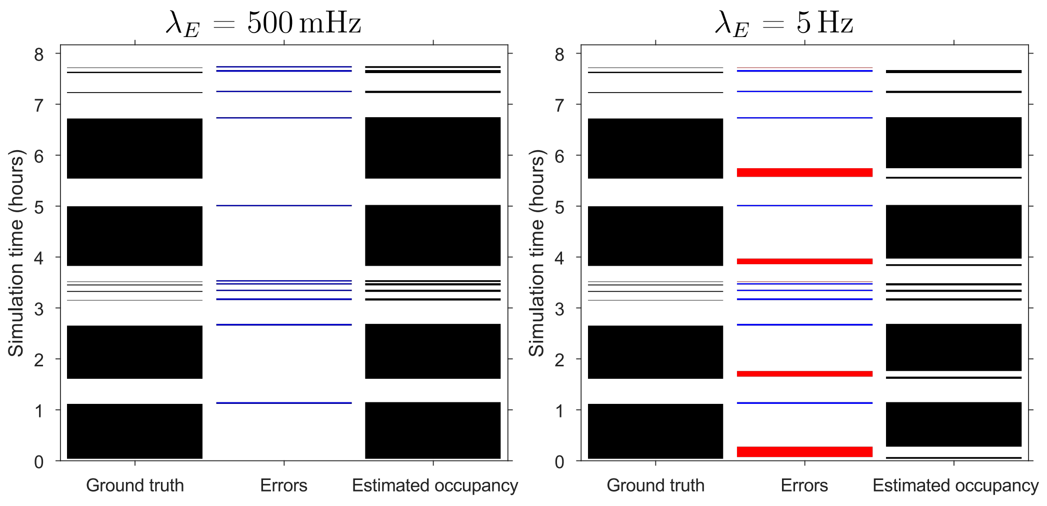


Figure 22: Increasing λ_E to 5 Hz introduces unwanted UCF errors (red).

In the graph on the left, we can see that the predictions are very accurate. Errors are present only after periods of occupancy in the form of transition errors.

When $\lambda_E = 5\,\text{Hz}$ the results are no longer that accurate. We notice some new error clusters where the prediction has failed to detect occupancy. Based on the ground truth data these occur when only one occupant was present in the office. The pattern of these errors can be seen in Figure 23.

As can be seen in Figure 23, if λ_E is too large a seldomly observed target cannot be reliably detected. First errors seem to appear around $\lambda_E = 1.35\,\text{Hz}$. The graph

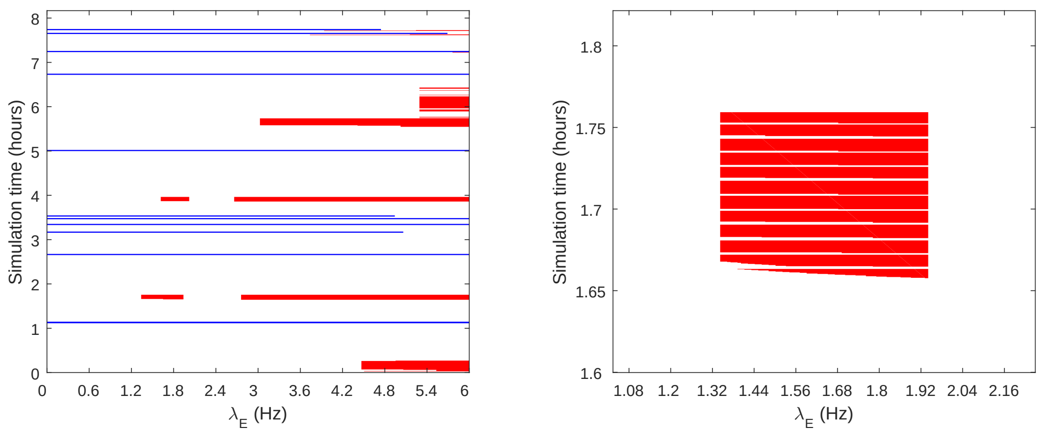


Figure 23: Error barcodes as the function of λ_E (left) and a close-up of one of the error clusters (right).

on the right shows a close-up of the first error cluster in the bottom left. Note how the error consists of stripes. In between the model correctly noticed the target, but changed to the wrong hypothesis after only a few time steps.

5.2.4 Adjacent Node Triggering Rate

Adjacent node triggering rate k is a useful parameter to explain an occupant triggering multiple sensors from a single location. Although test cabinet simulation was not designed to include this kind of behaviour, it was used to check the effect of this parameter nevertheless. Results can be found in Figure 24.

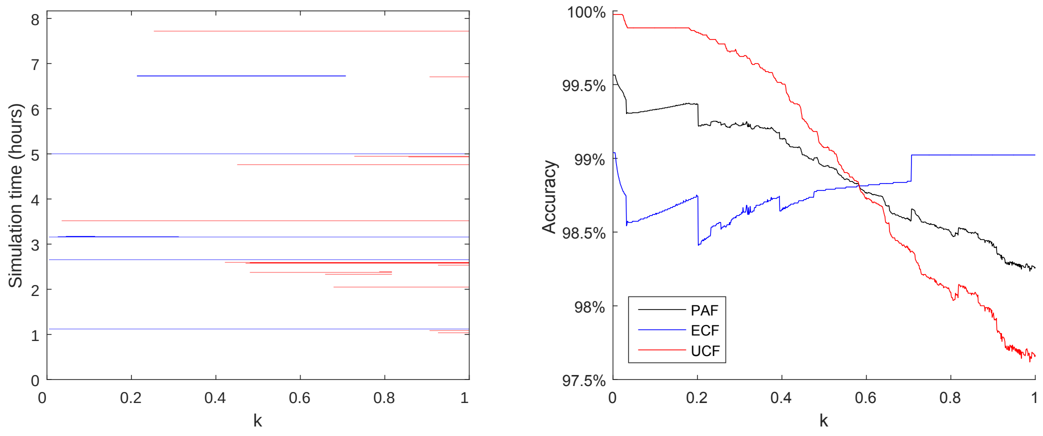


Figure 24: Errors with different values for k (left) and key ratios (right).

In Section 5.2.4 values of k between 0 and 1 were tested. Since k is defined as the ratio between the likelihood of observation in the most likely sensor and the adjacent sensors, the values of k have to be between 0 and 1. As can be expected from the design of the simulation mentioned earlier, modeling detected motion in adjacent nodes is not beneficial for test cabinet data analysis. $k = 0$ gave the best results in this case, but results were not significantly different with any values $k < 1$.

Thus it is unfortunately not possible to give any recommendations for the value of k based on this simulation.

5.2.5 Transition Rate

Transition rate is the rate at which occupants move from one node to adjacent nodes. It can be used to calculate the transition probability (see Equation (34)).

With the default value of 100 mHz, the probability of transition to an adjacent node is 13 % for a time step of one second. The actual values for transition rates are usually somewhere around this number. For a moving target the probability of moving to an adjacent node within one second is somewhere around 50 % while a still target can stay in the same node for hours and thus have a transition rate of less than 100 µHz.

Transition rates between 1 µHz and 100 Hz were tested, but there was no apparent effect on the results. It is unclear why this parameter had such a negligible effect. Possibly a small transition rate decreases the probability of all hypotheses equally, and after normalization, the output is same as before.

5.2.6 False Alarm Rate

In some applications of target tracking, such as radars, false alarms can be very frequent. However, for motion detectors, they are almost non-existent. A properly functioning PIR sensor never makes observations in an empty building. In Section 5.3 and data shown in Appendix C measurements were never made when an occupant was not present. Apart from other moving objects, such as pets or maybe a curtain moving with the wind, we can safely say that false alarm rate is zero.

However, this parameter can have another role in a target tracking model. If something has gone wrong, a very unlikely false alarm can be used as an explanation for the events. To bring this kind of flexibility to unexpected combinations to the model, we used a small value of λ_{FA} as the false alarm rate. [33]

A high false alarm rate favors hypotheses with fewer targets. If only a few observations have been made, false alarms can be more likely explanation than an actual target. This effect is visible in Figure 25.

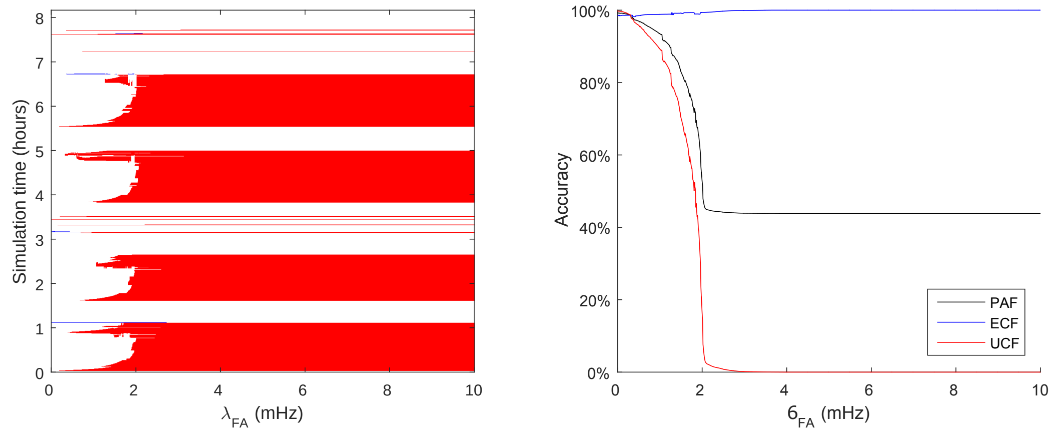


Figure 25: Errors with different values for λ_{FA} (left) and key ratios (right).

Even $\lambda_{FA} = 0$ mHz seems to be working well in the simulated environment. It is understandable since this simulation does not include clutter. At $\lambda_{FA} = 1$ mHz first errors emerge and after $\lambda_{FA} = 2$ µHz the tracker completely fails to detect any targets. Based on these results it can be recommended that λ_{FA} should be less than 100 µHz. The default value of 10 nHz is smaller than any of these values with a large safety margin.

5.2.7 New Track Rate

There is one parameter left to optimise: new track rate. It can be interpreted as the rate at which new occupants enter the area. Tested values varied between 0 mHz and 10 Hz and figure 26 illustrates the results. Note that these graphs have logarithmic x-axes.

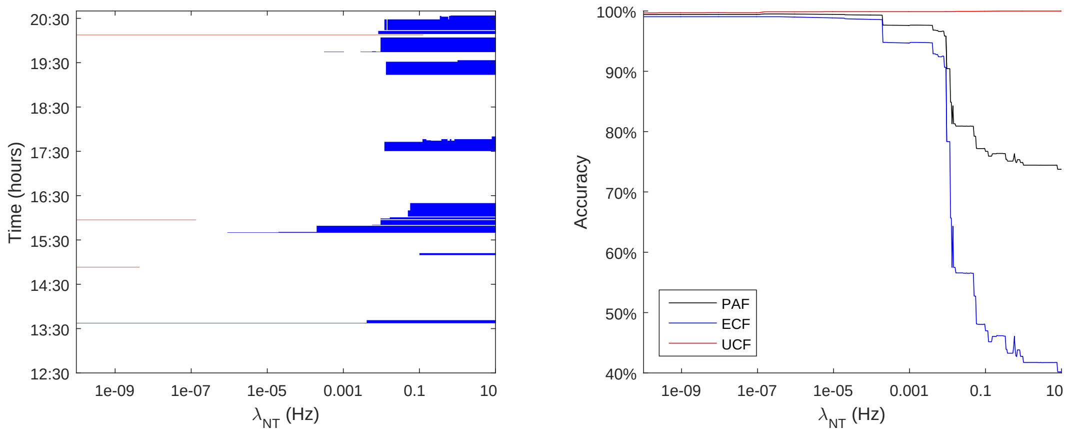


Figure 26: Errors with different values for λ_{NT} (left) and key ratios (right). Note that the horizontal axis has a logarithmic scale.

The target tracker seems to work well even with minuscule values of λ_{NT}. A minor increase in performance can be noticed after 10 nHz. Explanation clearly is that λ_{FA} is 10 nHz. When $\lambda_{NT} < \lambda_{FA}$ new occupants are first associated with false alarms, and only after a second observation a new tracked target is initialised. Generally in target tracking, this would be used to filter clutter and ignore single measurements. In the case of PIR sensors false alarms are almost nonexistent, and a new tracked target should be initiated immediately after the first observation.

When new target rate is too high, more than 10 mHz, ECF starts to decrease. At that point, there is clearly incorrectly many targets within the model. To summarise, new track rate should be slightly larger than false alarm rate but always less than 10 mHz.

5.3 Controlled Office Environment

Now that we have a general understanding of the tracker parameters, it is time to verify the algorithm with proper sensor data. The challenge with real sensors is to determine the actual occupancy status, i.e. the ground truth. In this section, occupancy status is determined by documenting all occupant movements through the whole test period. This setup enables accurate calculations of the relevant key ratios.

These tests were performed according to the description from Section 4.2.3. During the four and half hours, 874 messages were received from the wireless network of which 587 reported detected motion. Additionally, 428 messages contained a state transition and 11 times it was noticed that at least one state transition message had been lost.

With these values, the error rate can be calculated with Equation (36) to be 2,5 %. Since the probability of error is small, Equation (36) can be used reliably. According to the more detailed data, these errors were distributed evenly among all the sensor nodes.

In Section 6.2.3 the conclusion is that λ_E should be set somewhere between half and 50 times the actual sensor triggering rate. The real triggering rate was measured from the dataset to compare the parameter values with actual conditions. The values for five different occupant positions are shown in Table 4.

Table 4: Motion detector triggering rates measured at while the occupant was sitting different locations from Figure 12.

Location	**Observation rate of the closest sensor (mHz)**	**Total Observation rate (mHz)**	**Sample time (seconds)**
A	0	0	1276
B	0	0	1103
C	0	0	230
D	80	140	376
E	1	5	1555

Positions A and C are outside of the test are and have correctly the rate of 0 observations per second. The same also holds for position B (see Figure 12), which was designed to be outside of the sensor line of sight. Position D is directly under one sensor, and this value of 80 mHz can be used as an estimate of parameter λ_E.

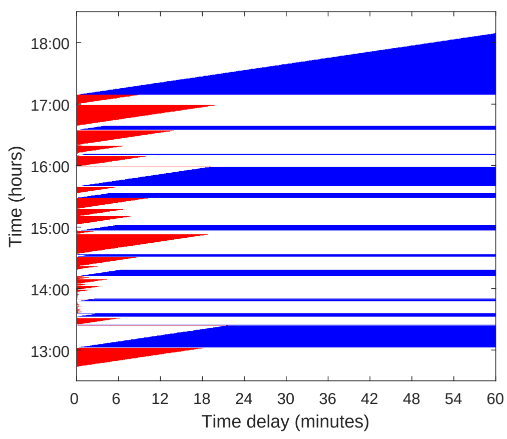
For position D a significant number of observations were also done in the adjacent nodes. Average value of k for the three adjacent sensors was 0.2. No observations were done outside of the adjacent nodes during the test period.

5.3.1 Time Delay Method

Time delay method was applied to the dataset to get a comparison for the target tracker. An observation always set the area to occupied until the time delay had run out. The delay was given values between 1 min and 60 min, and the results can be seen in 27.

The error plot is much easier to comprehend than for target tracking. During long periods without any measurements there was a red user comfort related error if the time delay was not sufficiently long.

The optimal Prediction Accuracy Factor (PAF) of 77 % was archived with a time delay of 19 min. This result corresponds to the longest time of occupancy when the occupant was not seen by the sensors.



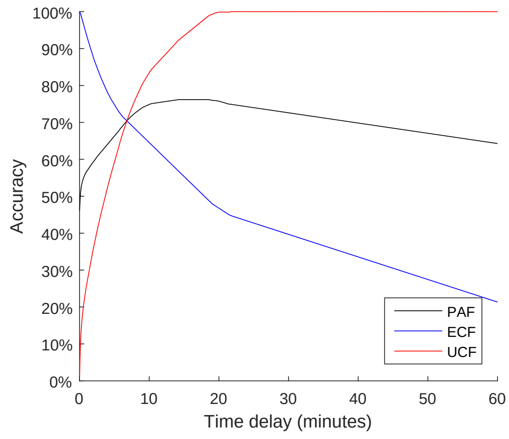


Figure 27: Errors with different lifetime values (left) and key ratios as a function of lifetime of the targets (right).

5.3.2 Target Tracking

In the following test, we had $\lambda_{NT} = 100\,\mathrm{nHz}$, $L_I = 60\,\mathrm{min}$, $L_B = 30\,\mathrm{s}$, and $M = 100$. Other parameters had the default values in Table 3 with except λ_E which was used as a variable.

Measurements and the target tracking output for the first 20 minutes with $\lambda_E = 18\,\mathrm{mHz}$ can be seen in Appendix C. Since a raw data table can be difficult to comprehend, the same values are also shown in Figure 28. It has been made in a similar fashion to Figure 14. The sensor IDs on the y-axis can be found from Figure 12. Since sensor 8 did not make any observations within the test period and sensors 1-7 are placed almost in a one-dimensional arrangement, we can use the sensor ID number as an axis in the figure.

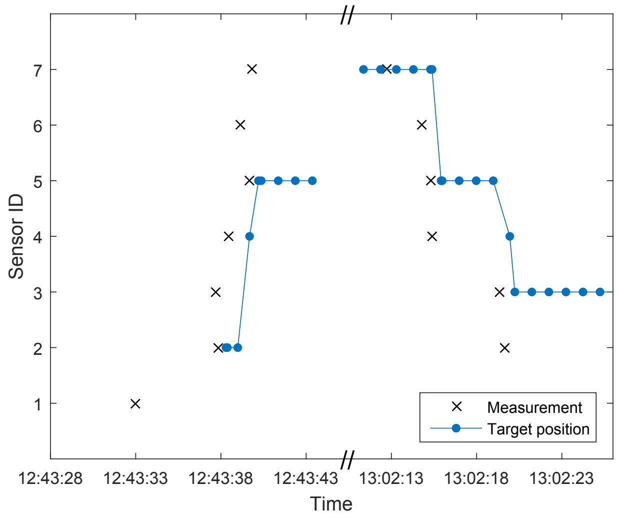


Figure 28: Raw motion sensor data and target tracking output. Note that the x-axis has been broken.

One can immediately notice that the simulated and real measurements are very different from each other. Most importantly: there is much less real measurements. Each sensor gave only one measurement when the occupant walked past them.

Blue line shows the output of the target tracker. During the long break in time axis the target moved was moved from sensor 5 to sensor 7, which is not visible in Figure 28. However, it can be seen well in Appendix C.

The target seems to always one step behind of the measurements. This was due to the fact that measurements in adjacent nodes were given such high likelihood. Certainly there was also a hypothesis where the target had immediately moved to the location of a new measurement, but it had a smaller probability than hypothesis where the measurement was explained by a target in the adjacent node.

The results with four different values of λ_E can be seen in Figure 29.

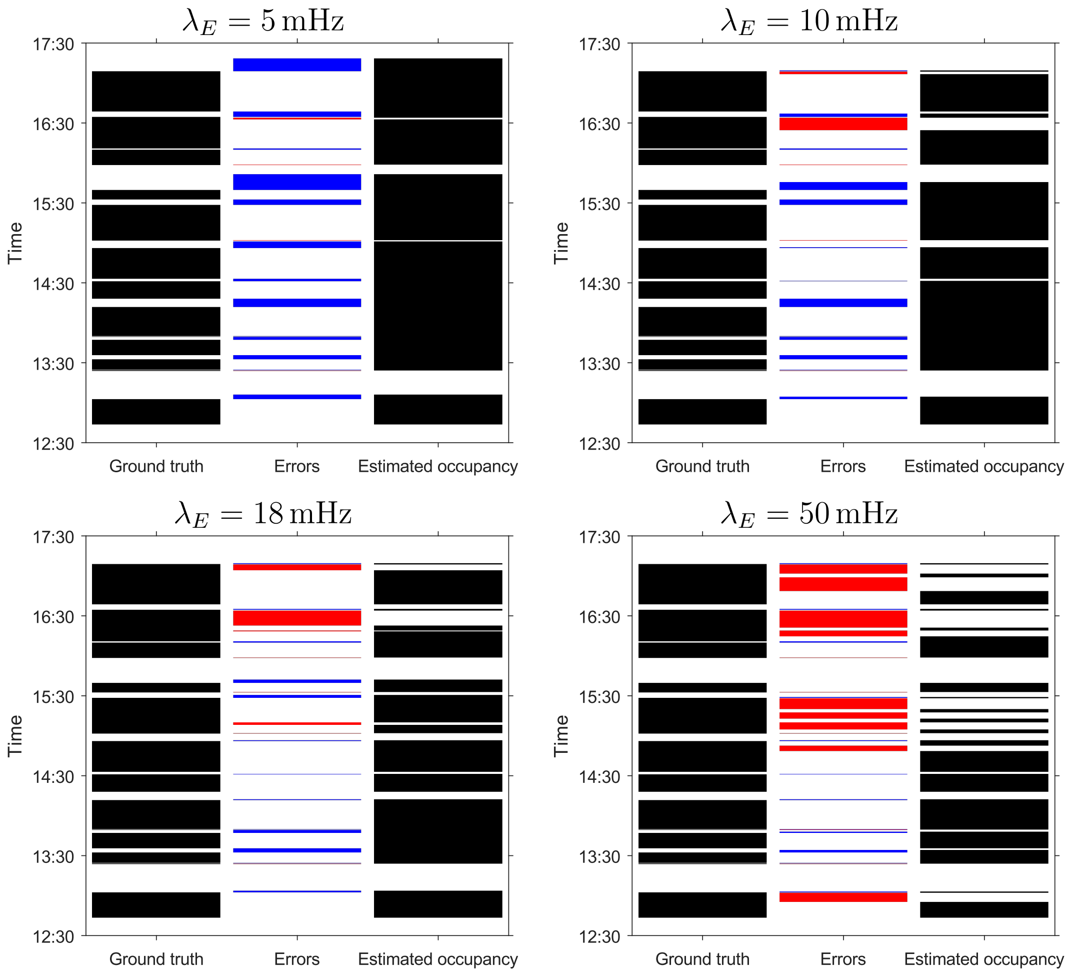


Figure 29: Real and estimated occupancy with varying values for λ_E. The optimum was at 18 mHz.

When λ_E is close to zero ($\lambda_E = 5\,\text{mHz}$, see Figure 29 top left) the area is almost always calculated to be occupied, which causes a significant loss in the energy conservation ratio.

When $\lambda_E \leq 29\,\mathrm{mHz}$, the tracker correctly detects occupancy during the 19 min period that the occupant was hiding at position B (see Figure 12). The same also happened between 12:43 and 13:02 and there were no observations during that time. These periods were not handled correctly when $\lambda_E = 50\,\mathrm{mHz}$.

Most difficult period of occupancy to detect was just before 16:30. When $\lambda_E > 10\,\mathrm{mHz}$ the tracker fails to detect occupancy there (Figure 29 top right, bottom left and bottom right). During that time the occupant was at position E of Figure 12. This phenomenon is understandable since there is no motion detector with a direct line of sight to that position. Additionally, a short distance to the exit makes position E more complicated to analyse than position B.

Initially, area E was also supposed to have a sensor. After performing the test, this sensor was found out to be malfunctioning and not reporting any motion. Position E was thus left without sensor and became a similar hiding place as position B. Adjacent sensors were still sometimes able to detect the occupant at position E.

Results with all values of λ_E between $100\,\mu\mathrm{Hz}$ and $1\,\mathrm{Hz}$ can be seen in Figure 30. PAF over 85 % was reached when $6\,\mathrm{mHz} < \lambda_E < 29\,\mathrm{mHz}$. Maximum PAF of 89 % is achieved at $\lambda_E = 18\,\mathrm{mHz}$ (Figure 30 bottom left).

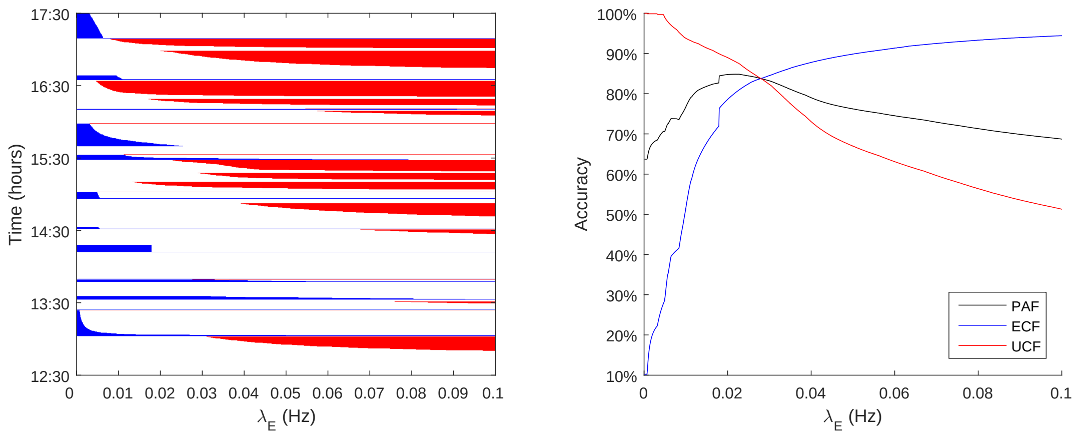


Figure 30: Errors with different values for λ_E (left) and key ratios (right) with real sensor data.

In Section 5.2.3 it we discussed that small values of λ_E can causes the tracker to create multiple targets to account for a single occupant. That also happened in this test. For $\lambda_E = 18\,\mathrm{mHz}$ the tracker was handling multiple targets more than once through the analysis. In some cases, such as when the occupant was triggering multiple sensors from position E, there were multiple targets in the same position with one another.

5.4 Tracker Behaviour in an Open Office

Using the feedback from results in Section 5.2 and Section 5.3 we made some updates to the default parameter values in Table 3. These updates are discussed in more detail in Section 6.2 and Section 6.3. The parameter values used in this section can be found from Table 2 in column "Recommendation". The only exception is that λ_E was set to 100 mHz.

The tracker was run live in an open office. Short lifetime at border nodes (2 min) made target tracker react quickly to leaving occupants while long lifetime at interior nodes (60 min) enabled it to detect extended periods of occupancy outside of the line of sight. Results and their comparison to the time delay method are shown in Figure 31. The results seem plausible. Unfortunately, ground truth data was not available in this test.

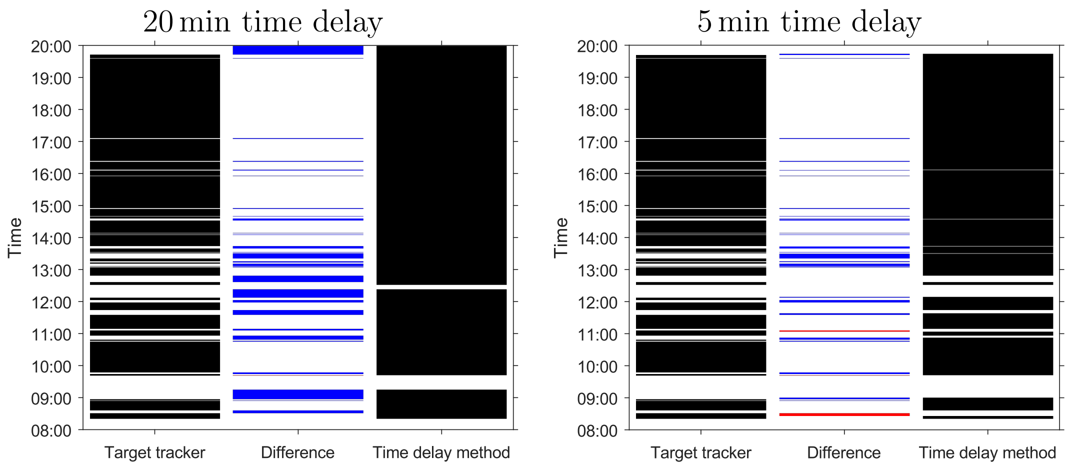


Figure 31: Comparison between target tracking and time delay method with two different values for the time delay.

6 Discussion

We now proceed to address the implications of the results in Chapter 5. In Section 6.1 the preliminary tests are discussed. Best results are given by a Multi Hypothesis Tracking utilising the continuous length of the time step and hypothesis merging. However, the difference to more simple methods, Global Nearest Neighbour (GNN) and fixed length of the time step, is not remarkable.

In Section 6.2 the results from simulations with the test cabinet are be used to optimise target tracking parameters in Table 3. These results are summarised in Section 6.5 together with some recommendations based on tests with real sensors.

In Section 6.3 we compare the accuracy of target tracking to that of the time delay method. These are the most significant results in this thesis since according to Figure 33 target tracking gives superior results.

Section 6.4 also compares the time delay method and target tracking. We discuss how to combine time delay and target tracking into a hybrid algorithm, and how large the energy savings are compared to a long time delay of 20 min.

6.1 Preliminary Tests

Results in Section 5.1 can be used to justify major design choices of the target tracking algorithm.

In Section 5.1.1 Local Nearest Neighbour (LNN), Global Nearest Neighbour (GNN) and Multi Hypothesis Tracking (MHT) were tested in a simulated environment. With LNN the model created an excessive amount of tracked targets. As noted in Section 2.3.1 there is no mechanism in LNN to delete or merge overlapping tracked targets. If this algorithm was run for a longer period of time, it is possible that more and more tracked targets keep stacking up eventually slowing down the computations.

GNN gave very similar results to MHT. Neither of these methods was correctly able to identify that there were only two targets with constant velocities. For our application, the exact identity of the targets is not necessary. In some other application where the identity of the targets also matters, a good approach could be to estimate not only the position but also the velocity of the targets [33]. Since the velocity vector includes the information about the direction of the targets, this kind of advanced model should give the correct analysis in this test setup.

A look into the more detailed output of the application reveals, that the most likely hypothesis was generated from something else than the previous most likely hypothesis only a few times in this test. It did not occur at all in the most interesting part of the calculation, i.e. when the two targets met. Consequently, it should be possible to achieve similar results with a well-tuned GNN.

In Section 5.1.2, sensitivity to the length of the time step was tested by comparing results with the standard algorithm and an alternation where the length of the time step was fixed to 1 s. Although the time step is not a particularly important variable, it was shown to have a minor role. For this reason, variable time steps were used in all other tests.

As can be seen in Figure 16, merging with the improved algorithm is computationally effective. Merging similar hypotheses frees space from the list of hypotheses to less probable hypotheses, which would normally require a much higher maximum number of hypotheses to get detected. Overall, merging can be seen as a method to decrease the need for computational resources.

6.2 Parameter Optimisation in a Simulated Environment

Results in Section 5.2 give good data to base target tracking parameter values on. In Section 6.2.1 we first figure out the minimum value for the lifetime of the targets. As expected, the value should be larger than the maximum expected interval between two observations. In Section 6.2.2 we divide the nodes into border nodes and interior nodes. Initially, this created some issues with tracker stability, but these were later solved by adding an initialisation hypothesis and increasing the number of border nodes.

Parameter λ_E is discussed further in Section 6.2.3. It is limited by two factors. Too small a value will result into the tracker creating multiple targets to account for a single occupant. Since this conclusion is incorrect, this kind of behaviour is undesired. However, in Section 6.3, we see that the wrong model can sometimes give better results. The second factor limiting values for λ_E is that if the value is too large, the tracker might ignore targets which are detected too rarely by the passive infrared (PIR) sensors.

The test cabinet did not model the PIR sensors perfectly. The main difference was that motion in adjacent nodes was not simulated, i.e. a still occupant was modeled with measurements from a single sensor. Results with real sensors in Table 4 indicate that at location D almost half of the observations were made by adjacent sensors. Furthermore, the distribution of the time intervals between two measurements were very different from each other. This is illustrated in Figure 32.

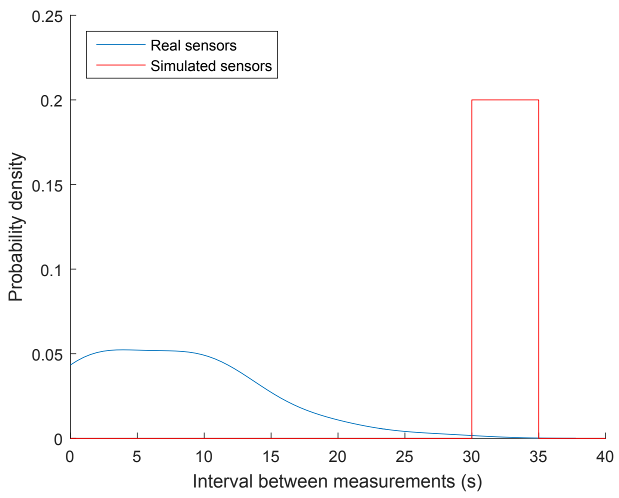


Figure 32: The estimated distribution for time intervals between two measurements (one or both of which can be from an adjacent sensor) for location D compared to the distribution used in the simulation.

The simulated measurements occurred on 30 s to 35 s intervals while the actual distribution had many intervals under 10 s. This difference explains well why lifetime values under 30 s gave such inaccurate resuls in Section 5.2.1.

Because of these inaccuracies, parameter k, the ratio of observation rate of a nearby sensor and the closest sensor, did not affect the results as can be seen in Section 5.2.4. Likewise, transition, false alarm, and new track rate parameters are not discussed in this chapter since none of these played a significant role in the fine tuning of the tracker. Results are completely indifferent to any reasonable values of transition rate. As long as λ_{NT} and λ_{FA} rates are measured in nHz and $\lambda_{FA} < \lambda_{NT}$, the algorithm performs as expected.

6.2.1 Lifetime in Interior Nodes

Results which examine the impact of the lifetime in interior nodes are presented in Section 5.2.1. When lifetime is 40 s (Figure 17, bottom right), the results of the prediction correspond to ground truth remarkably well. Some transition errors are visible on the graph in the proximity of changes to the state of the ground truth. It seems that there are no spurious errors (see Figure 3) with a lifetime of 40 s.

The shorter the lifetime the more errors appear. This can be explained by the fact that the still targets are simulated to trigger PIR sensors every 30 s to 35 s as noted in Section 4.2.2.

The stairway-like shape of the curve in Figure 18 is somewhat surprising. There are no such approximations in the tracker that could cause this kind of discrete improvements to the results. It seems that the rounding has been done within the PIR sensor simulator. A possible explanation is that the interval between two observations is rounded to the closest integer (in seconds) after the randomization.

The accuracy seems to improve substantially when the lifespan passes certain thresholds. That could be caused by rounding in the simulation software and it is not expected that this phenomenon would appear in real data.

In the case of this simulation, lifetime of 35 s gives the best results. This does not generalise to real life situations since occupants are not guaranteed to trigger a sensor at least every 35 s. One might be willing to compromise the ECF by increasing lifetime in order to maximise user comfort.

In a more advanced target tracker, the suitable lifetime parameter could be learned from the data as has been done previously with the time delay method [18, 19]. However, such approach is outside of the scope of this thesis.

6.2.2 Border Lifetime and Adding of an initialisation Hypothesis

In Section 6.2.2 we applied a different lifetime for interior and border nodes. Initially, the accuracy was poor (see top left graph in Figure 19), but the issue was solved with three alternative approaches: increasing maximum number of hypotheses, adding of an initialisation hypothesis (see Section 3.6), and extending the borders to account for nodes adjacent to exit nodes.

In Figure 20 it is clearly visible how the use of initialisation hypothesis and extending border decrease the required amount of hypotheses in the model. When the extended border is used, there are virtually no errors when $M \geq 5$. Both of these methods are beneficial to the stability of the tracker and should be utilised when L_I and L_B are given different values.

6.2.3 Sensor Triggering Rate

Sensor triggering rate λ_E was found out to be one of the most important parameters. As can be seen in Figure 21, too small a values of sensor triggering rate λ_E result in a single occupant being modeled with multiple targets. In this test, the sensor was simulated to detect motion once every second. For the default value of $\lambda_E = 100\,\text{mHz}$ four targets is the most likely explanation after 100 s since that part of the graph is yellow. After 622 s the number of targets reaches its presumable maximum value of 7 targets. This kind of behaviour is mathematically understandable but practically undesirable. Based on this test, λ_E should be set larger than half of the actual sensor triggering rate.

The theoretical maximum rate of the sensors used in this thesis is approximately 300 mHz. In practice, the maximum rate is much lower. Our measurements indicated that the sensors rarely detect motion at a higher frequency than 130 mHz and even that required careful movement which was designed to trigger an observation as often as possible. This behaviour can thus be avoided by setting $\lambda_E \geq 50\,\text{mHz}$.

This simulation based result agrees with the test with real sensors. Results in Section 5.3 indicate that with $\lambda_E = 18\,\text{mHz}$ a single occupant was modeled with multiple targets.

In Figure 23 it can be seen that too large values for λ_E interfere with the detection of targets which are seldom observed. In that test, the simulated triggering rate was around 30 mHz for still targets. Therefore λ_E should be smaller than 50 times the actual triggering rate.

Since λ_E is such an important parameter, it can be worthwhile to set the optimal value individually for each sensor. In a more advanced algorithm, it would be possible to learn optimal value directly from the data in a similar way as how optimal time delay can be learned (see Section 2.1.3). [18].

In summary, modeled sensor triggering rate should be set between 0.5 and 50 times the actual triggering rate.

6.3 Controlled Office Environment

In chapter 5.3 the same dataset was analysed with both time delay method and a target tracker. Results with the time delay method (Figure 27) are clearly worse than those with target tracker (Figure 30).

Von Neida et al. [13] have also been able to get PAF of 74 % with the time delay method. It is surprising that the accuracy is that close to our results with the time delay. Test in Section 5.3.1 was performed only during daytime while Von Neida et al. had two weeks of continuous data. That should give them an advantage since nights are easier to estimate (always unoccupied) than days. That means that overall our estimations seem to have been more accurate.

Occupant detection is a two-objective optimisation problem between user comfort and energy savings. For this reason, comparing the target tracker and the time delay method is not a straightforward task. The comparison can still be made by selecting a goal value for the User Comfort Factor (UCF) and then determining the best possible Energy Conservation Factor (ECF) for each method. This comparison can be seen in Figure 33.

The datasets of this figure are the same as in figures 30 and 27. For the time delay method, different values of UCF were obtained by changing the length of the time delay between 3 min and 18 min. Values of the time delay are not visible in the figure.

Similarly, different key ratios for target tracker were calculated by using the λ_E as the variable. It was given values between 5 mHz and 106 mHz. Note that only one of the parameters was varied in this test. By also changing the values of the other parameters, the target tracker could reach even higher values of ECF with the same UCF.

As demonstrated in Figure 33, target tracking gave superior results to the traditional model. The difference was most significant if UCF of 90 % or less is sufficient. For example, for UCF = 80 % the time delay method made more than triple the amount of mistakes when the room was unoccupied compared to the target tracker. For UCF = 95 % this value dropped to 1.5 and when UCF = 99 % the two methods give the same result.

If UCF of around 90 % can be accepted, the target tracker can reach an ECF of at least 86 %. In other words, 86 % of the time when the area is unoccupied the target tracker has been able to detect it successfully. In the test run, the area was unoccupied 34 % of the time. This means that the energy savings for lighting could potentially be 29 % compared to a system which was on during the whole test period.

A time delay method with a UCF of 90 % would only reach ECF of 59 %, which would result in 20 % energy savings. In other words, the target tracker has the

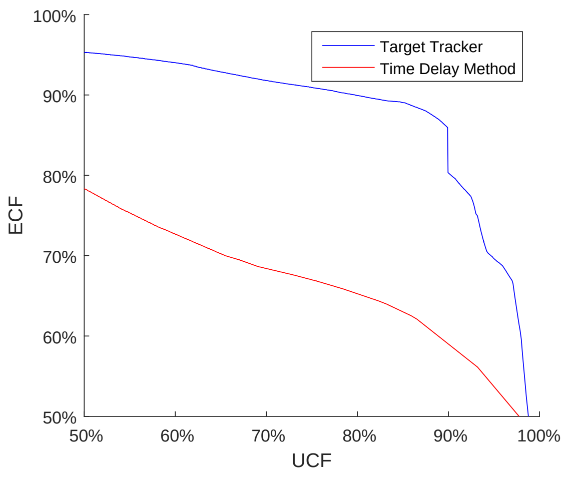


Figure 33: Comparison between the target tracker and a time delay method.

potential to save 15 % of energy compared to the current solution.

These results can be applied to all occupancy-dependent energy use which can be turned off instantly after occupancy has ended. Lighting is a good example of this kind of application. Ventilation can also be treated this way to some extent, while heating and cooling would require entirely different approach.

As mentioned in Section 2.1 there have been previous improvements to the time delay method with various algorithms. For example, Bayesian Belief Networks have given promising results [21]. Comparing the strengths and weaknesses of the target tracker and these methods is outside of the scope of this research.

6.4 Tracker Behaviour in an Open Office

As can be seen in Figure 31, 20 min time delay method is much more conservative than the target tracker. If a lighting or heating, ventilation, and air conditioning (HVAC) system were controlled by target tracker alone, it would save 20 % of total energy usage. The lack of ground truth means that we are unable to know whether this kind of systems would decrease user comfort or not.

The output of the target tracker and 5 min time delay method seems similar to target tracker. During the test period, time delay method estimated twice the area to have become unoccupied slightly earlier than the target tracker. These could potentially have been UCF errors caused by the short delay. In this case, target tracker would decrease energy usage only 8 % compared to the time delay method.

The target tracking could also be used together with the time delay method. A short delay can be reinforced with a target tracker, which is specialised in detecting still targets. Therefore one should use OR operator: if one of the methods detects occupancy the room is kept in occupied state. In this test, a combination of a target tracker and the 5 min time delay would save 12 % of energy compared to estimation with a 20 min delay. Although real occupancy is not available, it is unlikely that both target tracker and 5 min time delay method would have made a mistake at the same time.

6.5 Summary recommended parameter values

We have performed basic tests to all relevant parameters of the target tracker. Recommendations for parameter values can be found in Table 5.

Table 5 also includes the default values, which were just an initial guess. The difference between the final recommendation and the default value was most significant for sensor triggering rate λ_E. Minor adjustments were also made for new track rate λ_{NT}, lifetime at interior nodes L_I, lifetime at border nodes L_B, and the maximum number of hypotheses M. For the other parameters λ_T, k and λ_{FA} the tests were not conclusive, and default values were kept in effect.

λ_E was found out to be one most important and complicated parameters. On one hand, if the value is too large, User Comfort Factor (UCF) drops. On the other hand, too small values for this parameters makes the method incorrectly explain a

Table 5: List of target tracker parameters, allowed intervals, and recommended values.

Parameter	**Default value**	**Minimum**	**Maximum**	**Recommendation**
λ_T	100 mHz	1 µHz	100 Hz	100 mHz
λ_E	100 mHz	10 mHz	100 Hz	20 mHz
k	0.1	0	1	0.1
λ_{FA}	10 nHz	0 Hz	100 µHz	10 nHz
λ_{NT}	100 µHz	λ_{FA}	10 mHz	100 nHz
L_I	20 min	30 s	60 min	60 min
L_B	10 s	10 s	L_I	2 min
M	10	5	-	100

single occupant with multiple tracked targets. That sometimes happens with the recommended value of 20 mHz, but this is compensated with an increased UCF.

We also tested advanced methods: hypothesis merging (Section 5.1.3), adding initialisation hypotheses (Section 5.2.2), and extending border nodes to adjacent nodes (Section 5.2.2). All of these were discovered beneficial to accuracy and stability of the tracker and using them can be recommended.

7 Recommendations for Future Research

[redacted]

7.1 Improvements to the Current Tracker

[redacted]

7.2 Automatic Configuration of Node Layout

[redacted]

[redacted]

7.3 Predictive Models

[redacted]

[redacted]

8 Conclusions

In this thesis, we present and test a novel method of analysing data from multiple motion detectors. The test results indicate that target tracking successfully detects occupancy with a higher accuracy than a conventional time delay method. In a controlled test environment target tracking could potentially save 15 % of energy compared to simple occupancy based control. These energy savings can be archived without compromising with the user comfort. Configuring such system can be partly automatised, but requires still slightly more work than the time delay method.

The new approach is exceptionally suitable for detecting occupants who are outside of the line of sight of the sensors but unable to leave the area without being noticed by a sensor. This property is useful in areas with high density of simple passive infrared (PIR) sensors, and where the occupants have only a few exits.

These results show that target tracking can be used to increase energy efficiency without compromising the user comfort. It can be used to collect accurate statistics about the usage patterns, which in turn can be used to configure a heating, ventilation, and air conditioning (HVAC) system manually.

In an improved system, these two systems can be integrated to control HVAC directly with the occupancy information. In an integrated system, the proportion of the energy savings can be different from the 15 % mentioned earlier depending on the occupancy pattern. That is especially the case with systems which require energy usage even when the area is not occupied, such as heating. More research and tests with an implemented system integration are needed to test this more accurately.

Finally, a target tracker could be used in parallel with another method. When using it together with a 5 min time delay, we were able to demonstrate 12 % energy savings compared to a 20 min delay. Unfortunately, we were not able to measure the user comfort of this approach in the scope of this thesis, but it can be assumed to be relatively accurate.

We have identified areas for future research in multiple areas. The target tracking algorithm can still be simplified without interfering with the quality of the results. It is also possible to reduce the amount of configuration required by learning the structure sensor network from raw data. Furthermore, the motion sensor provides plenty of data to forecast occupancy, which would provide valuable information for HVAC systems.

The algorithm described in this thesis should only be applied to dense sensor networks. There should be at least two sensors between the area of interest and the exit for the target tracker to detect longer periods of occupancy reliably. It is also critical to process the observations in the order that they appeared. A lag in the wireless connection could thus be a great challenge.

This work demonstrates that model based sensor data analysis can be done efficiently and accurately. No teaching data are required with a high-level model, and the algorithm can be easily applied to any new sensor sites.

In Finland, yearly energy consumption of residential and commercial lighting alone is approximately 5 TWh [35, 36]. The potential of 15 % energy savings demonstrated in this thesis translates into around 20 million euros per year.

References

[1] S. Hawking. A brief history of time, 1998.

[2] L. Pérez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[3] I. E. Agency. Technology roadmap: energy efficient building envelopes, 2013.

[4] S. Blackman and R. Popoli. *Design and analysis of modern tracking systems. 1999.*

[5] J. Ahola. Estimation of movement information in wireless local area networks. Master's thesis in Aalto University. 2016.

[6] C. R. Wren, D. C. Minnen, and S. G. Rao. Similarity-based analysis for large networks of ultra-low resolution sensors. *Pattern recognition*, 39(10):1918–1931, 2006.

[7] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proceedings of the 1st international conference on embedded networked sensor systems*. ACM, 2003, pp. 150–161.

[8] T. Labeodan, W. Zeiler, G. Boxem, and Y. Zhao. Occupancy measurement in commercial office buildings for demand-driven control applications—a survey and detection system evaluation. *Energy and buildings*, 93:303–314, 2015.

[9] E. Hailemariam, R. Goldstein, R. Attar, and A. Khan. Real-time occupancy detection using decision trees with multiple sensor types. In *Proceedings of the 2011 symposium on simulation for architecture and urban design*. Society for Computer Simulation International, 2011, pp. 141–148.

[10] P. Chen, S. Oh, M. Manzo, B. Sinopoli, C. Sharp, K. Whitehouse, O. Tolle, J. Jeong, P. Dutta, J. Hui, et al. Instrumenting wireless sensor networks for real-time surveillance. In *Robotics and automation, 2006. icra 2006. proceedings 2006 IEEE international conference on*. IEEE, 2006, pp. 3128–3133.

[11] N. Batra, P. Arjunan, A. Singh, and P. Singh. Experiences with occupancy based building management systems. In *Intelligent sensors, sensor networks and information processing, 2013 IEEE eighth international conference on*. IEEE, 2013, pp. 153–158.

[12] Z. Nagy, F. Y. Yong, and A. Schlueter. Occupant centered lighting control: a user study on balancing comfort, acceptance, and energy consumption. *Energy and buildings*, 126:310–322, 2016.

[13] B. Von Neida, D. Manicria, and A. Tweed. An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems. *Journal of the illuminating engineering society*, 30(2):111–125, 2001.

[14] C. F. Reinhart. Lightswitch-2002: a model for manual and automated control of electric lighting and blinds. *Solar energy*, (1):15–28, 2004.

[15] J. S. Han, Y. K. Jeong, and I. W. Lee. Analysis of electric energy consumption patterns: a case study of a real life office building. In *Applied mechanics and materials*. Vol. 330. Trans Tech Publ, 2013, pp. 158–162.

[16] J. D. Jennings, F. M. Rubinstein, D. DiBartolomeo, and S. L. Blanc. Comparison of control options in private offices in an advanced lighting controls testbed. *Journal of the illuminating engineering society*, 29(2):39–60, 2000.

[17] M. Sunikka-Blank and R. Galvin. Introducing the prebound effect: the gap between performance and actual energy consumption. *Building research & information*, 40(3):260–273, 2012.

[18] Z. Nagy, F. Y. Yong, M. Frei, and A. Schlueter. Occupant centered lighting control for comfort and energy efficient building operation. *Energy and buildings*, 94:100–108, 2015.

[19] V. Garg and N. Bansal. Smart occupancy sensors to reduce energy consumption. *Energy and buildings*, 32(1):81–87, 2000.

[20] G. Rebane and J. Pearl. The recovery of causal poly-trees from statistical data. *Arxiv preprint arxiv:1304.2736*, 2013.

[21] R. H. Dodier, G. P. Henze, D. K. Tiller, and X. Guo. Building occupancy detection through sensor belief networks. *Energy and buildings*, 38(9):1033–1043, 2006.

[22] J. R. Norris. *Markov chains.* (2008). Cambridge university press, 1998.

[23] Z. Yang, N. Li, B. Becerik-Gerber, and M. Orosz. A non-intrusive occupancy monitoring system for demand driven hvac operations. In *Construction research congress 2012*, 2012.

[24] S. Meyn, A. Surana, Y. Lin, S. M. Oggianu, S. Narayanan, and T. A. Frewen. A sensor-utility-network method for estimation of occupancy in buildings. In *Decision and control, 2009 held jointly with the 2009 28th chinese control conference. cdc/ccc 2009. proceedings of the 48th IEEE conference on*. IEEE, 2009, pp. 1494–1500.

[25] S. Emmerich and A. Persily. *State-of-the-art review of co2 demand controlled ventilation technology and application*. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2001.

[26] S. Wang, J. Burnett, and H. Chong. Experimental validation of co2-based occupancy detection for demand-controlled ventilation. *Indoor and built environment*, 8(6):377–391, 1999.

[27] A. Kusiak and M. Li. Optimal decision making in ventilation control. *Energy*, 34(11):1835–1845, 2009.

[28] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[29] Z. Yang, N. Li, B. Becerik-Gerber, and M. Orosz. A multi-sensor based occupancy estimation model for supporting demand driven hvac operations. In *Proceedings of the 2012 symposium on simulation for architecture and urban design*. Society for Computer Simulation International, 2012, p. 2.

[30] S. Särkkä. *Bayesian filtering and smoothing.* (3). Cambridge University Press, 2013.

[31] D. B. Reid. An algorithm for tracking multiple targets. *Automatic control, IEEE transactions on*, 24(6):843–854, 1979.

[32] S. Särkkä, A. Vehtari, and J. Lampinen. Rao-blackwellized particle filter for multiple target tracking. *Information fusion*, 8(1):2–15, 2007.

[33] S. Särkkä. Personal communication. Feb. 19, 2016.

[34] *Active+ sense.* (T3020). Issue 2. Helvar Oy Ab, Jan. 2016.

[35] Tilastokeskus. Asumisen energiankulutus. *Suomen virallinen tilasto (SVT)*, 2014.

[36] A. Korhonen, H. Pihala, A. Ranne, V. Ahponen, and L. Sillanpää. Electricity saving possibilities in household and office appliances including lighting. *Työtehoseuran julkaisuja 384*, 2002.

A Java method for generation of hypotheses

The following Java code was used to generate new hypotheses. Methods generateFA, generateNT, and generateMV were used to generate false alarm, new track and movement hypotheses, respectively.

Listing A1: Method in the Hypothesis class for generation of child hypotheses.

```java
/**
 * Generates child hypothesis based on false alarms, new track or
 * a movement association. If parameter m is set to -1, the
 * iteration is run without a measurement. Only one child hypothesis
 * with will be returned in this case.
 *
 * @param m number of the sensor which made the new measurement.
 * @param time Time of the measurement as unix time in milliseconds.
 * @return ArrayList containing all new child hypotheses.
 */
public ArrayList<Hypothesis> getChildHypotheses(int m, long time){
    ArrayList<Hypothesis> children = new ArrayList<Hypothesis>();
    double C = calculateC(this.probability);

    // Clone track list and delete inactive tracks
    ArrayList<Track> temp_tracks = cloneTracks(this.tracks);
    temp_tracks = deleteTracks(time, temp_tracks);

    if(m == -1){
        // There is no current measurement, A = 1 (Equation 25)
        children.add(generateFA(cloneTracks(temp_tracks), C, 1, time));
    }
    else {
        // Accounts for FA association, A = P(FA) (Equation 22)
        children.add(generateFA(cloneTracks(temp_tracks), C, tracker.FA, time));
        // Accounts for NT association (n=0)
        children.add(generateNT(cloneTracks(temp_tracks), C, time, m));

        // Loop over possible target positions for the measurement
        for (int x = 0; x<tracker.EMIS.length; x++){
            // Skip x if the emission probability is 0.
            if (tracker.EMIS[x][m] <= 0) continue;

            // Loop over current targets
            for (int targetIndex = 0; targetIndex < temp_tracks.size(); targetIndex++){
                // Accounts for association to current track
                children.add(generateMV(cloneTracks(temp_tracks), C, time, m, targetIndex, x));
            }
        }
    }
    return children;
}
```

B Merging algorithms

This appendix has the source code for the merging algorithms used in Section 5.1.3 in Java. Hypothesis.equals(Object o) was overwritten to return true if o is also a Hypothesis object and the two object have same amount of targets with same locations. Hypothesis.comp sorts the hypotheses list so that similar hypotheses are adjacent. In practice, it is done by constructing a string with all the target positions. Hypotheses are then sorted so that these state strings are in alphabetical order.

Listing B1: Initial mering algorithm.

```java
ArrayList<Hypothesis> merge(ArrayList<Hypothesis> hypotheses){
    ArrayList<Hypothesis> newList = new ArrayList<Hypothesis>();

    // Loop over hypotheses. O(N^2) because of a nested loop.
    for (Hypothesis current : hypotheses){
        // ArrayList.indexOf(Object o) has to do a second loop.
        int index = newList.indexOf(current);
        if(index != -1){
            // A similar hypothesis was found.
            newList.get(index).merge(current);
        } else {
            newList.add(current);
        }
    }
    return newList;
}
```

Listing B2: Efficient mering algorithm.

```java
ArrayList<Hypothesis> merge(ArrayList<Hypothesis> hypotheses){
    ArrayList<Hypothesis> newList = new ArrayList<Hypothesis>();
    Hypothesis previous = null;

    // Sort by strings of target states. O(N*log(N))
    hypotheses.sort(Hypothesis.comp);

    // Loop over hypotheses. O(N)
    for (Hypothesis current : hypotheses){
        if (current.equals(previous)){
            // The adjacent hypotheses are similar.
            previous = previous.merge(current);
        } else {
            // The two hypotheses are different.
            if (previous != null){
                newList.add(previous);
            }
            previous = current;
        }
    }
    newList.add(previous);

    return newList;
}
```

C Example data from the controlled environment

In this appendix we provide raw data tables related to results in Section 5.3. We will use the sensor ID numbers from Figure 12 for measurements and target tracking output. The notes, which were done manually, use area names which can also be found in Figure 12.

Table C1 consists of example dataset where the occupant started from location A, hid to location B for 19 minutes and returned in location A. The table combines three different datasets: raw motion detections, target tracker output, and notes about the location or movement of the occupant. The two first datasets were accurately synchronised based on the unix time stamps, which had the accuracy of 1 ms. The notes did not have that accurate time stamp in the first place, and those have been added to an approximately correct row.

For target tracking λ_E was set to 18 mHz. Each time a measurement was made a target tracking iteration was ran immediately. Some of the lines had no measurement since these were just periodic updates to the hypothesis list which were done on 1 s intervals when there was no new measurements. Some of these lines have been omitted to save space. The omitted lines occured on one second intervals, had no measurements or notes and had the same target tracking output as the line above them.

This dataset demonstrates well the power of target tracking. When the occupant was in location B, the target tracker recognised that the room is occupied even with the lack of new measurements. But when the occupant left the area and went to location A, the target was deleted quickly i.e. the status of the room was set unoccupied.

Table C1: Example dataset from controlled office environment.

Time	**Measurement**	**Target location**	**Notes**
12:43:26			Left location A
12:43:34	1		
12:43:35			
12:43:36			
12:43:37			
12:43:38			
12:43:38	3	2	
12:43:38	2	2	
12:43:39	4	2	Enter room
12:43:40	6	4	
12:43:40	5	5	
12:43:40	7	5	
12:43:41		5	
12:43:42		5	
12:43:43		5	
12:43:44		5	
12:43:45		5	Enter location B
12:43:46		5	
330 similar rows omitted			
12:49:18		7	
768 similar rows omitted			
13:02:08		7	Left location B
13:02:09		7	
13:02:10	7	7	
13:02:11		7	
13:02:12		7	
13:02:12	6	7	
13:02:13	5	5	
13:02:13	4	5	
13:02:14		5	
13:02:15		5	
13:02:16		5	
13:02:17	3	4	
13:02:17	2	3	
13:02:18		3	
13 similar rows omitted			
13:02:31		3	Enter location A
13:02:31		3	
72 similar rows omitted			
13:03:45			