Multiple Occupant Indoor Localization

by

Masoud Vatanpour Azghandi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Recognizing the indoor activities of people is a key functionality of smart homes, since it is a prerequisite for any supportive action in service of the occupants. In this thesis, we investigate the multiple occupant localization problem in indoor environments. We developed a method based on the use of inexpensive passive infrared motion sensors together with Radio Frequency IDentification (RFID) readers. In our method, the former type of sensors, placed throughout the space, recognize movement and RFID readers, placed selectively at key locations, unambiguously recognize individuals' locations (some or all of the occupants are assumed to be wearing passive RFID tags) as they pass through their coverage area. Due to their high cost and generally cumbersome placement requirements, RFID readers must be judiciously placed. Thus, we study the placement of the readers such that the occupants trajectory ambiguity is reduced. We rely on a heat-map representing the frequency with which individuals visit locations as they move through the indoor space and on models of coverage for the Passive Infra-Red (PIR) sensors and RFID readers and develop a heuristic for the RFID reader placement.

We also revisit the problem of cost-efficient PIR sensor placement for high-quality indoor localization, extending it for sensors with diverse coverage footprints, and the occlusion effects due to obstructions typically found in indoor environments. The objective is the placement of the smallest number of sensors with the right combination of footprints. Given the vast search space of possible placement and footprint combinations, we adopt an evolutionary technique. We demonstrate that our technique performs faster and/or produces more accurate results when compared to previously proposed greedy methods. Furthermore, our technique is flexible in that adding new sensor footprints can be trivially accomplished. We evaluate the effectiveness of our method in both RFID reader and PIR sensor placement under different occupancy conditions with simulations.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Ioanis Nikolaidis and co-supervisor Eleni Stroulia for the continuous support of my Masters study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Hong Zhang and Prof. Zac Friggstad for their encouragement, insightful comments, and challenging questions.

Last but not least, I would like to thank my beloved wife, Shahrzad, and my family for supporting me along the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Smart-Condo™ [3, 4] is a comprehensive platform, targeting the delivery of improved healthcare through a variety of services. This fully equipped one bedroom condo has functional appliances, moveable counters, removable floors, and is also integrated with a Wireless Sensor Network (WSN) for monitoring the occupants. The setup allows the collection of health-related events and potentially using this data to enhance the quality of life for older adults.

One important purpose of smart homes (*i.e.,* the Smart-Condo™) is to provide information regarding inhabitant activities to facilitate the task of the caregivers who are in charge with the occupants' well-being. Of the information collected by, a possibly large, number of sensors, the information regarding the location of the individuals stands out as crucial to determining the tasks performed and the disposition of the inhabitants. While simple "anonymous" sensors can be used to localize a single individual, e.g., low-cost PIR motion sensors, such solution is clearly inadequate when more then one individual are to be tracked and their trajectories need to be separated and labeled. A benefit of using exclusively PIR sensors is that the occupant does not need to wear any special tag or device. However, in order to track multiple individuals, we are expanding the requirements to include RFID tag(s) worn by each individual. Technically, given the low-cost of passive RFID tags, an individual can "wear" more than one tag, but for the sake of ease of presentation we assume that each individual corresponds to a single tag. The understanding is that, in real settings, it is enough for at least one of the tags worn by the individual to be read to localize the person as being close to the RFID reader.

One concern in using PIR sensors for localization is their placement. Suboptimal placement may result in inefficient usage of the equipment, and correspondingly, waste of equipment cost and deployment effort. Pre-deployment simulation to evaluate the anticipated performance of potential deployment alternatives is a useful methodology for systematically obtaining high-quality sensor placements. The question then becomes how to decide which placements to simulate, *i.e.,* how many sensors should the potential deployment include, of what type, and where exactly these sensors should be placed. In principle, a desirable placement should have the smallest number of contributing sensors (in order to minimize equipment cost, deployment effort, and operating energy consumption) at locations such that the overall space is sufficiently covered and the target's location can be inferred with a desirable degree of accuracy and precision.

The PIR sensor placement optimization presented in this thesis builds on our previous work, in the context of the Smart-Condo™ project, where we examined placement of same-type sensors, under a cardinality constraint (*i.e.,* a limited budget of sensors) [1] for the purpose of recognizing the location of an individual in a home environment. We adopt a similar formulation in this thesis. Namely, the formulation includes the representation of space as a line drawing (floorplan), and the possible locations for the sensor place are from a set which is expressed as a (fine) grid of points over the floorplan. The placement is assumed to take place on the ceiling (facing "down"). While alternative placements can be accommodated, experience from practical deployments has reinforced that ceiling placement is the most convenient for indoor deployments. Our set of motion sensors includes a variety of different volumetric shapes, namely a cone, a square based pyramid, and a rectangular based pyramid. Considering only orthogonal placement with respect to the floor the projection of each of these shapes becomes a disk, a square and a rectangle, respectively. Moreover, the projection of any sensor might be effected by walls, doors or obstacles around the house depending on where the sensor is placed (explained more in Section 3.3).

One question addressed in this thesis is whether a combination of PIR sensor deployment in an indoor space, coupled with the judicious use of RFID readers deployed at certain points in the space, and assuming the individuals wear passive RFID tags, is a viable solu-

tion for multi-occupant localization. Specifically, we would like to derive figures of merit for the combination of PIR+RFID localization and strategies for RFID reader deployment. The latter, RFID reader deployment, is a significant component of the total cost because RFID readers tend to radiate enough power that their deployment away from a continuous power source is problematic, i.e., powering them from batteries is not a viable long-term solution. Additionally, the use of relatively large RFID reader antennas to produce reliable readings from the tags increases the per-reader cost and results in cumbersome deployment. In short, the deployment of the readers introduces costs and constraints that are not present in the case of the wireless PIR sensors. Hence, we are interested to reduce the number of RFID readers and deploy them in locations that are as effective as possible, i.e., where they can add the most in terms of improving the accuracy of localization given that a PIR-based sensors already exist. Note that for the same reasons (power source needs, antenna sizing) we adopt the, reasonable, model that the individuals wear passive tags and the environment is equipped with readers, rather than the other way round (individuals carrying readers) found typically in inventory-management applications of RFID where workers carry readers.

One of the guiding assumptions is that in indoor spaces, the use of Global Positioning System (GPS) [5] for localization is problematic due to poor signal coverage and increased error that renders it useless when seeking localization error no more than 1-2 meters which is generally needed for any meaningful indoor application. As noted, we forego the use of *active* RFID tags, and similarly, the use of tags with wireless transmitters regardless of wireless technology used. The primary reason is that active devices require batteries resulting in less inconspicuous packaging than simple peel-on passive RFID tags. Battery powered tags are also a hindrance (or even a risk) if the object on which such tags are affixed is subjected to common abuse, e.g., when the fabric in which it is embedded is washed. For those reasons, and given current available technologies, passive RFID tags are still an elegant and flexible solution. Their use comes at the, asymmetrically, higher cost and special placement considerations of the RFID readers.

Abstractly, the problem at hand is one of sensor fusion for the purposes of tracking individual trajectories, in a mixed environment of anonymous (PIR) and identity (RFID)

sensors. The solution advocated here is a two step process. First, the motion sensors are used to determine paths for (possibly groups of) individuals roaming the indoor space. Clearly, their paths mix and soon become ambiguous even if we knew the original locations of each individual. A second stage, the RFID-based disambiguation of the paths helps mitigate the ambiguity of the first step but is limited because the RFID readers are present only at certain locations and have limited coverage. This leads us to develop a model that can assist the placement of RFID readers using a "skeletal" tree of the paths of motion of individuals in the indoor space, extending previous work that had introduced a heat-map representation [1] of the most/less visited locations by a single individual in the space. Throughout the thesis we assume that the floorplan of the space is known and available to our algorithms.

1.1 Thesis Contributions

This thesis contributes in the deployment of an automated planning methodology for multi-occupant localization in a smart house environment. More specifically, the contributions are as follows:

- We propose an evolutionary methodology to address the problem of optimized PIR placement for PIR sensors with diverse coverage footprints. The resulting placements will be used to mount sensors in locations that helps gather sensing information more efficiently. Furthermore, we consider the objective of reducing the number of sensors in order to minimize system cost, but delivering a reasonable degree of accuracy.

- We construct a skeleton representation of the existing heat-map in order to spot important travel corridors in the space. Using the formed skeleton, we propose a heuristic to determine the proper location for RFID readers, and improve localization of the occupants.

- We define a framework, and the process used to extract tracks for multiple people. With the information retrieved form the PIR sensors we construct, possibly ambiguous, trajectories for all occupants involved. We then disambiguate the trajectories by assigning people to paths, according to RFID readings.

- We propose a data structure that captures the sequence of events happening in an indoor environment, i.e., collisions, identifications, etc. This so called EM Graph provides the basis to calculate our performance metrics, namely, the ambiguity error metric and the tracking error metric.

- We consider multiple scenarios and thoroughly evaluated with simulations, cases where the patient or other residents in the smart house, refrain from wearing their tag. Our PIR placement optimization is compared against a sequential methodology, and the placement heuristic for RFID readers is compared with a random and a manual placement.

In summary, the major contribution of this thesis, is automated sensor deployment planning for indoor environments. We organize various experiments through both a simulator and the real-world network, to evaluate our system. The developed software is ready for operation on the main platform in the Smart-Condo™ and was written with a focus on modularity and scalability.

1.2 Thesis Organization

This thesis is organized as follows. Chapter 2 reviews related work. In Chapter 3 we provide the problem formulation and present details of the mobility and coverage models. Chapter 4 comprehensively presents and evaluates the proposed methodology to optimistically place motion sensors for the purpose of indoor localization. In Chapter 5 we focus on a systematic approach for RFID reader placement and illustrate results for certain system settings and scenarios. Finally we conclude with Chapter 6, reviewing the lessons learned by carrying out this research and note of possible future directions.

Chapter 2

Review of Indoor Localization Technologies

2.1 Placement Strategies

According to [6], sensor placement methodologies can be categorized into static and dynamic. With static approaches, the optimization process is pursued before the actual network deployment and it is assumed that various network parameters will remain the same. In that way, the desired properties that were achieved (e.g. overall coverage and/or network topology) will not change. The second type of placements are the dynamic approaches. In some applications dynamic repositioning of the sensor nodes may be an reasonable way to increase system performance. This is useful when the initial placement leads to undesired usage of system resources. Although we realize the importance of dynamic placement in network maintenance and adaptability, the thesis is concerned exclusively with static placement.

In a static placement the deployment can be done either randomly (by scattering a large population of low cost sensors in the target environment), or deterministically (controlled placement for maximizing certain desired properties). Choosing which placement best suits the application depends on the scale of the application. Randomized distribution of sensors is acceptable for large-scale networks where controlled placement is not a viable solution, e.g. in reconnaissance missions during combat or disaster recovery. In an accessible environment, a deterministic approach is acceptable, especially if a small number of sensors are to be deployed in a rather small network. Such is the case of smart home applications

and hence we do not investigate randomized and redundant sensor placement, but rather put emphasis on deploying a small number of sensors.

2.2 Sensor Placement Optimization

2.2.1 Optimization Objectives

Typical optimization objectives for sensors include coverage, connectivity, longevity and data fidelity [6]. The objective that has received most of the attention in the literature is coverage which deals with maximizing the covered area under surveillance. Coverage objectives depend on the underlying coverage model of each individual sensor. Previous work adopts models of disk coverage of the sensors [7], whereas, more recently focus has changed to irregular polygons caused by objects distorting the sensor's coverage range [1, 8, 9]. In the main framework of the Smart-Condo™ , we too, consider irregular shapes due to sensor projections being impacted by permanent obstacles, walls and doors.

Data fidelity, is an optimization objective through which a sensor network ensures the credibility of the gathered data, typically by fusing readings from various sensing sources. Data fusion reduces the probability of false reports and will surely boost the overall accuracy of the system with the cost of increased node density.

We would like to remind that, the current sensor infrastructure that is used in the Smart-Condo™ is a single hop model. This is mainly because: a) the area of interest that we need to cover is relatively small (i.e. a one bed room suite, approximately 66 square meters in area) and b) we focus on real time updates as well as off-line processes, so we desire minimal amount of communication delays, typically introduced in a multi-hop network topology. These two reasons makes us narrow down the focus of this thesis on coverage and data fidelity (i.e. localization accuracy) rather than worrying about network connectivity and longevity.

2.2.2 Optimization Techniques

Many research activities have been conducted around the Optimized Sensor Placement (OSP) problem. The OSP problem has proven to be NP-hard for many different proposed

formulations [6], thus computationally intractable. An extensive survey, [10], studies OSP in Structural Health Monitoring (SHM). The authors narrow down on one problem formulation and choose specific evaluation criteria for their sensor optimization and based on this, compare various pre-existing methodologies. The methodologies compared are namely: deterministic, sequential and combinatorial optimization methods (including GAs, simulated annealing algorithm, Particle Swarm Optimization (PSO), monkey algorithm, and the ant colony optimization algorithm). For a better understanding we focus on combinatorial and sequential approaches in the next few paragraphs.

Various OSP solutions benefit from combinatorial methodologies (e.g. PSO in [11–14] and GA in [15, 16]). Kang, Li, and Xu [17] use a Virus Coevolutionary Partheno-Genetic Algorithm (VEPGA) to optimize sensor placement in large space structures (*i.e.,* portal frames and concrete dams) for damage identification based on data that describes the dynamic behavior of a structure system. They concluded that their method outperforms the sequential reduction procedure. Poe and Schmitt adopt a GA approach to sensor placement for worst-case communication delay minimization [18]. Comparing their results against an exhaustive and a Monte-Carlo method, they found out that these methods serve as an upper and lower bound, respectively. Their method is a fast and near-optimum solution for optimized placement. Yi, Li and Gu [19] compared evolutionary methods for sensor placement and described a Generalized Genetic Algorithm (GGA) approach for a predefined number of sensors. According to them the GGA can get better results than the simple version of the GA. They also describe a number of different exhaustive and evolutionary methods to sensor placement. These papers show that evolutionary methods are preferred over the exhaustive search approaches.

The sequential approach proposed in [1] (which is conceptually closest to the work done here) is also conducted in the context of the Smart-Condo™ and aiming at inexpensive placements for high-accuracy localization of a individual in a home environment. The greedy (or sequential) approach of [1] identified sensor locations one at a time, resulting in exploring a large number of potential locations and, in some cases, requiring a large amount of time. Some of the work on PIR placement adopts evolution-based techniques to address issues of complexity (see Chapter 4).

An additional complexity is that in this thesis we consider two different types of technologies that collect different types of information (namely, PIR sensors and RFID readers), to be used at different points in the localization process. While (weighted) coverage maximization appears to be a suitable objective for PIR sensor placement, the objective of RFID readers is to disambiguate paths produced based on the output of the localization derived from PIR data. Essentially, the input to the RFID reader disambiguation process is not only dependent on the PIR placement but also on the (dynamic) performance of the PIR-based localization algorithm, and hence on the performance of the PIR-based localization scheme. Therefore, the placement of RFID readers is judged against a backdrop of both an (off-line) PIR placement and of a (usually on-line) PIR localization algorithm. In our PIR sensor placement, we address essentially two objectives: both covering the space with PIR sensors for the purpose of maximized information gain, and at the same time minimizing the number of sensors.

2.3 Indoor Localization Techniques

2.3.1 RFID-assisted Methods

Many RFID-assisted localization systems have been proposed, and some rely on a pre-deployed grid of location-aware tags, *i.e.,* each tag carries its own spacial coordinates and relays that information upon request. The grid of tags does not necessarily need to follow a conventional square arrangement pattern, as Han et. al. [20] have concluded. They suggest that a triangular pattern will help reduce the error of positional and orientational estimations. They claim that this allows inexpensive estimation of user location. Also benefitting from the grid of tags, Yeh et. al. [21] use ultrasonic, orientation, and force sensors coupled with an accelerometer, and an RFID reader, all embedded in a pair of sandals to perform location determination for the wearer. Their calculation is based on Dead Reckoning (DR) information, corrected with a location-aware passive-tag grid. Similarly, [22] use passive tags at reference points which are read by a moving RFID-reader paired with an Internal Measuring Unit (IMU) as a DR device. Also, [23], fuse data from an IMU with Received Signal Strength (RSS) from multiple active RFID tags, producing highly accurate location

estimates by eliminating the drift of an IMU-only system. Willis et al. [24] propose a system that embeds an RFID reader in a walking cane for blind people. The cane is able to pick up location estimates by reading information off the already deployed RFID tags.

2.3.2 Probabilistic and State Estimation Methods

Many works in the literature focus on target tracking systems that utilize conventional state estimation techniques. One of the more broadly used methods is various versions of the Kalman Filter, as in [25–27]. In [28] the authors use the Extended Kalman Filter (EKF) to estimate the position, velocity and phase off-set of the target. They track a tagged target by installing RFID readers at fixed locations and extract localization information which is then fed to an extended Kalman Filter and a "smoother" component to provide accurate results. Compared to an Received Signal Strength Indication (RSSI) based method, they claim to have reconstructed trajectories much closer to the ground truth. Within a similar setup, Nick et. al. [29] compare two different extensions of the Kalman Filter for RFID tag localization purposes, namely the Extended and the Unscented Kalman Filters (EKF and UKF). They conclude that UKF performs better than EKF.

In [30] the authors propose the well known SLAM (Simultaneous Localization And Mapping) approach, to perform 2D target tracking based on an RFID tag grid. In [31] a laser-based FastSLAM algorithm is used to create a map of the environment which is used to determine the locations of the 100 tags attached in different locations. Then, a Monte-Carlo localization algorithm is used to estimate the pose (orientation and location) of a robot or a person using the location information from the tags. The Monte-Carlo localization method works by considering a set of random samples of the believed pose of the robot and a weighing factor for each sample (indicating its importance). In each iteration new samples are collected and weighted according to the observation likelihood.

Although these techniques show improved localization accuracy, most of them assume the collection of very detailed information from their available sensor setting (*i.e.,* RSSI) and we do not explore that direction in this thesis.

2.3.3 Alternative Technologies

Vision-based techniques, [32, 33], are very appealing for Multiple Target Tracking (MTT) in indoor environments. However, they tend to suffer from (a) concerns about privacy of the images captured (some occupants are reluctant to accept such technologies), (b) sensitivity to occlusion effected by various objects, and (c) generally higher computational and/or bandwidth (and hence energy) needs for on-site signal processing. Techniques based on ultrasound and WLAN fingerprinting have been proposed as well. iLoc+ [34] utilizes fixed reference nodes to receive ultrasound pulses emitted by interactive badges attached to each person. Although very accurate in terms of localization, it suffers from time-consuming installation of the large number of reference nodes that are required. The LOCOSmotion system [35], combines WLAN fingerprinting and speed information obtained from an embedded accelerometer in an off-the-self smart-phone. It is easy to install and to use, but an on-site training phase has to be carried out, by manually collecting signal fingerprints at various locations. The training process needs to be repeated if the environment changes. Another interesting study is the tracking system presented in [36]. The authors fuse tracking data from floor sensors with that of the accelerometer embedded in a cellular phone to accurately track a known number of people. In their experimental evaluation, they only investigate two very naive trajectory scenarios for two people. In the present thesis we do not restrict people's paths, and they are allowed to roam around in space, resulting in, often, paths that are very complex and full of collisions. Apart from that, the sensor infrastructure that we have selected due to simplicity, cannot consistently sense the occupants, like in [36]. In particular, individuals are only sensed when in the proximity of a PIR sensor or an RFID reader.

In all of the above mentioned methods, wearing bulky equipment is often required for the sake of accurate system operation. In contrast, our intention is to keep wearable equipment as minimal and inconspicuous as possible, hence only requiring that users wear lightweight passive tags, e.g., embedded in their articles of clothing.

Chapter 3

Mobility and Coverage Models

3.1 Mobility Model

3.1.1 Trajectory Model

Within their environment, people go through their activities, moving through open spaces from one point to another. Intuitively, there are some locations within the environment, i.e., kitchen counter, one's own seat at the dinner table, chairs, one's own bed, that tend to be *destinations* of the people's movement. These destinations are potential starting/ending point for people's paths within the sensed environment. We call these points Areas of Interest (AoI).

Considering the set $\mathcal{P}$ representing the k individuals ($\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$), the sequence of destinations chosen by each occupant, i.e, their trajectory T^{P_i} is composed. Each trajectory is a sequence of a person's locations-at-time, $l_{P_i,t_n} = (x_{P_i,t_n}, y_{P_i,t_n})$ in t_n order.

A collision between P_i and P_j means that there is a timestamp t_{col} when the distance between $(x_{P_i,t_{col}}, y_{P_i,t_{col}})$ and $(x_{P_j,t_{col}}, y_{P_j,t_{col}})$ is less than δ. Here, the distance function is Euclidean and δ is assumed to be two times the radius of the circle that represents a person's body: $\delta = 2 \times R$. Collision may also occur between three or more people at the same time if the above conditions apply for any pair of occupants among the colliding ones.

3.1.2 Pause Model

Since individuals spend time to preform an activity, in order to make individual movement models as realistic as possible *pause times* are introduced. A pause is a time period with a duration chosen from an exponential distribution. The probability function for this random variable ($\mathcal{T}$) is defined as follows:

$$f_t(\mathcal{T}) = \begin{cases} \lambda e^{-\lambda t}, & t \in \mathbb{R}_{\geq 0} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

In Equation 3.1, the mean value for $\mathcal{T}$ is equal to $\frac{1}{\lambda}$.

3.2 A Heat-map Representation of Occupant Mobility

In this research, we adopt the model developed in [1]. In this model, given the floorplan of the target smart home, relevant information is extracted from the environment: (a) location of areas of interest (i.e. locations in the environment that form the origin/destination of an occupant's path), (b) movables (temporary obstacles that may be moved to another location), (c) permanent obstacles and walls, and (d) doors. Considering that movable objects can be moved (within a confined area) if necessary, and doors can be opened to pass, *walkable* points in the space are recognized.

Also, to model an occupants daily routine a sequence of different AoIs are randomly chosen and the occupant traverses the walkable points creating trajectories (as discussed in the previous section). The paths are chosen by a path finding algorithm (*i.e.,* a generic implementation of A*) with some degree of randomness to mimic a path that a human would choose. After that, each point in space is assigned a certain color intensity based on the frequency of visitations, resulting in a heat-map. Figures 3.1 and 3.2 show such heat-map. The heat-map values of the covered points will be used to define the information utility of sensors placed at the various locations.

The heat-map is produced only once, at the pre-deployment phase. It is at this phase that all the decisions regarding sensor deployment have to be made. After the sensors are mounted on the ceiling, redeployment is considerably expensive and time consuming.

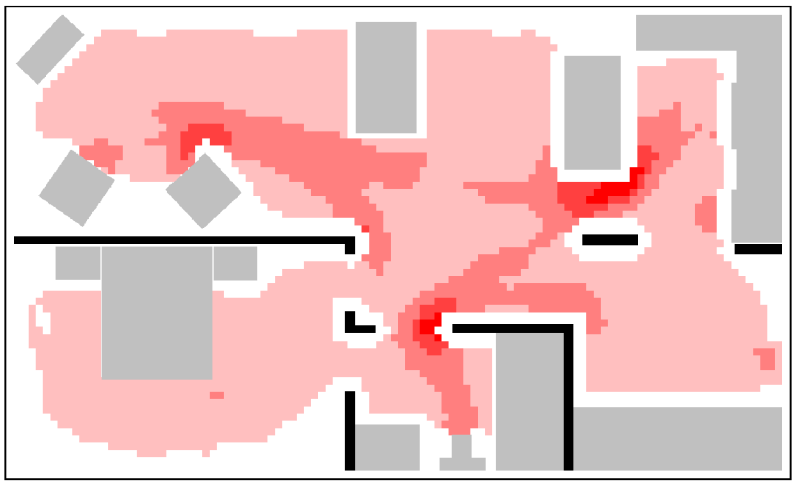


Figure 3.1: Coverage utility heat-map of a person walking in the Smart-Condo™ . The intensity of color at any point indicates that the point was used more frequently.

Figure 3.2: Coverage utility heat-map of a person walking in the Independent Living Suit (ILS).

Thus, both the quantity of sensor we need to achieve our cost/accuracy trade-off, and the best placement is determined.

Using the two-dimensional heat-map that contains N points, (x_1, y_1), ..., (x_N, y_N), where each point has an information utility of v_i $(i = 1, 2, \ldots, N)$ we construct a coverage utility map (with the same dimensions) which is the mapping of intensities from the heat-map into numerical values, *i.e.,* a normalization step. This is done using a configurable application-specific parameter called c_{max} that indicates the upper bound of values in the coverage utility map. Equations 3.2 and 3.3 show how the translation between heat-map values (v_i) and coverage utility values (c_i) are calculated.

$$c_i = \lceil \frac{v_i}{t(c_{max})} \rceil \tag{3.2}$$

Here, c_{max} is the maximum desired coverage score.

$$t(c_{max}) = \frac{max_{i \in \{1,\ldots,N\}} v_i}{c_{max}} \tag{3.3}$$

The main purpose of mapping heat-map values to coverage utility scores, is to flexibly normalize the range of numbers as the user desires, while ensuring that the balance between the lowest and highest values is retained. An interpretation of this mapping is that, the user only desires a maximum number of c_{max} sensors covering the most visited points.

The coverage utility map will be used throughout the rest of the study, mainly during the PIR placement.

3.3 Motion-Sensor Coverage Model

Based on our experience in the Smart-Condo™ project (and while alternative placements can be accommodated), experience from practical deployments has reinforced that ceiling placement is the most convenient for indoor deployment of PIR sensors. The set of PIR sensors we use includes a variety of different volumetric shapes, namely a cone, a square based pyramid, and a rectangular based pyramid. Considering only orthogonal placement with respect to the floor the projection of each of these shapes becomes a disk (e.g. the "Slight

motion detection" type from the NaPiOn series, [37]), a square (e.g. the "ATU1000C" Series, [38]) and a rectangle (e.g. the "spot detection" type from the NaPiOn, and the "Standard" type from the WL series VZ series [37]), respectively. Moreover, the projection of any sensor might be affected by walls, doors or permanent obstacles around the house depending on where the sensor is placed. Figure 3.3 presents an example where two sensors (marked with green) with, originally, rectangular footprints (depicted with blue lines) are restricted because of obstruction by walls (thick black lines) and doors (painted in yellow). Some parts of the polygon fall "behind" a door and the sensor may cover these areas if the door is open, otherwise the door restricts the sensor's projection even further. We use the convention of denoting the probability that a door is open by $p_{door\ open}$. $p_{door\ open}$ translates into probability of detection of motion in the aforementioned parts of the polygon *i.e.,* if the door is open. However in order to avoid complexity we set $p_{door\ open}$ to zero, and our coverage model simplifies into a boolean [1] coverage model, as in Equation 3.4.

$$p_{s\to o} = \begin{cases} 1, & \text{if } o \in A_s \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

In this definition o is a point in space and A_s is the set of points sensor s covers.

If many sensors in the set of sensors, Z, cover o, the joint sensing probability at that specific point is [39]:

$$P_o^Z = 1 - \prod_{s \in Z} (1 - p_{s\to o}) \tag{3.5}$$

3.4 RFID Reader Coverage Model

The RFID reader coverage model we use is a directed boolean sector model (details described in [40]). According to this model (illustrated in Figure 3.4), Φ_0 is called the *orientational angle*, ω is the *angle of view*, r is the *sensing range* and its coverage function is given by Equation 3.6 [40]. All points lying within the sector (for example, the point

[1] In most cases, as in our evaluation section, this will be a "sharp" probability, i.e., either 1 or 0, but the formulation is applicable to general coverage probabilities.

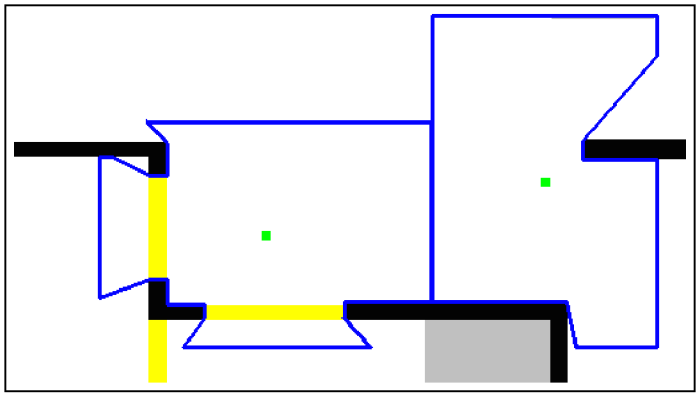


Figure 3.3: Sensor coverage projections in the presence of permanent obstructions [1].

marked by a star in Figure 3.4) are *covered* by the RFID reader, and the points outside the sector are said to be not covered. The coverage function ($g_{o\to z}$) for the boolean sector model is given by Equation 3.6, where o is the RFID reader, z is a point in space, and $d(o,z)$ is the Euclidean distance between them. Note that, in our work, we assume that the RFID reader's are mounted on the walls, and we adopt Euclidean distance for d.

$$g_{o\to z} = \begin{cases} 1, & \text{if } d(o,z) \leq r \textbf{ and } \Phi_0 \leq \Phi_z \leq \Phi_0 + \omega \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

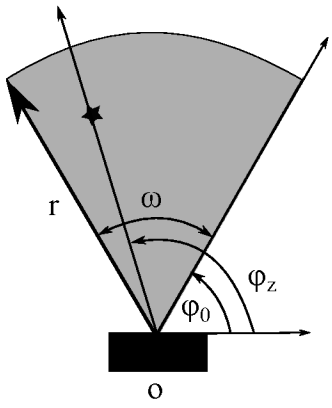


Figure 3.4: RFID coverage model.

When a person enters the coverage region of an RFID reader, the person's unique *ID* (corresponding to the person's RFID tag) and the location of the sensing RFID reader are

emitted and a new *unambiguous* location is recognized for the person in question. In effect, the sensing RFID reader unambiguously recognizes the person within its coverage area and infers the corresponding person's location. These unambiguous location inferences will be used to disambiguate the ambiguous location inferences produced based on the PIR sensors.

3.5 Combining PIR and RFID Localization

From the PIR motion-sensor infrastructure, the system receives a *sensor event* (which we notate by $\mathcal{SE}$), *i.e.,* a binary bit-string, where each bit corresponds to a motion sensor with value 0 when no movement is detected within the sensor's coverage area and 1 otherwise. Each sensor event can be translated to one (or more) *active polygons*, defined by the intersection of the coverage areas of the sensors with value 1 in the bit-string. Assuming a single occupant, the original Smart-Condo™ location-inference method recognizes the location of the occupant as the "center" of the smallest polygon in the overlap of the footprints of all firing ("1") sensors. However, when multiple different people move in the same space, the same event may be caused by one of them moving (and the scenario is reduced to the original single-occupant case); or all of them congregating in the same polygon; or each of them moving in different non-overlapping areas of the active polygons causing different subsets of sensors to fire. For a simple example, Fig. 3.5 illustrates two scenarios that produce the same sensor event, $\{11\}$. This ambiguity can be, to some degree, resolved, as the occupants start again moving separate from each other, adding new points to their trajectories - some of them disambiguated by RFIDs, as we will see in the following.

Moreover, we define the *possible future locations*, as a circle of radius r around each person, which covers the points that a person can reach within the next timestamp. In this manner, the circle is a prediction of all adjacent sensors that can possibly fire because of this individual's movement in the next time instant, thus creating a set of plausible future sensor readings. If the sensor readings at the next timestamp belong to the union of the plausible next readings, the sensor event can be interpreted to recognize the locations of all individuals based on their locations at the last timestamp; thus, for each person, the

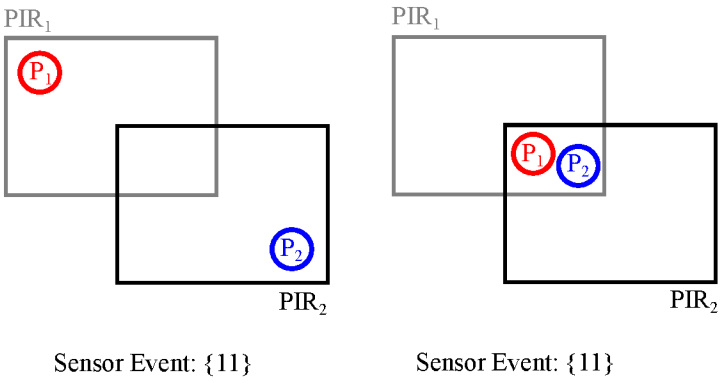


Figure 3.5: Location combinations of two individuals resulting in the same PIR sensor event.

geometric centre of the corresponding polygon is added to the person's trajectory. If the next reading is outside the union of plausible next readings, it could be due to a conservative (small) r which is easy to adjust by increasing r by Δr. Note that the same effect (same sensor event for different individual locations) could also be caused because the sensors do not sense continuously but duty cycle the motion sensor every so often, *i.e.,* their sensing rate is limited.

Algorithm 1 describes this process. Lines 1 through 3 sets the initial points in the trajectory of each individual; we assume we know $|\mathcal{P}|$, the number of individuals involved. $|\mathcal{P}|$ also indicates the maximum number of separate tracks we can have at any point in time. Line 5 initializes r to a conservative value r_{init} based on the rate of sensing and the typical person's speed. Lines 8 through 10 determines the plausible sensor events that each person could have traversed to (and hence cause) from their previous stand point; more specifically the function "findCandidateSensorEvent" receives the previous trajectory location of the j-th occupant and the radius r, and it determines which subset of sensor events could be triggered by this j-th occupant in the next time slot due to any movement of the j-th occupant (this includes the possibility that it stays in the same location as well). Hence $F_{i,j}$ is the set of possible bit-strings (similar to SE entries) that could be caused by the j-th individual's movement. Line 11 adds the bit-wise OR of the bit-strings present in the Cartesian product of the $F_{i,k}$ sets ($k \in 1 \ldots |\mathcal{P}|$) to the set C_i. Line 15 converts the index j to matrix subscripts and in Line 17 the corresponding sensor event is used to infer

Algorithm 1 Extract $|\mathcal{P}|$ trajectories from motion sensor events

Require: $|\mathcal{P}|$ // number of occupants
$\mathcal{SE} = \{SE_j\}$ // sensor event trace
Ensure: T // trajectories for all individuals

for $i = 1$ **to** $|\mathcal{P}|$ **do**
  $T_0^i \leftarrow initialTrackPoints(i)$
end for
for $i = 1$ **to** $|\mathcal{SE}|$ **do**
  $r \leftarrow r_{init}$
  $foundMatchingCombination \leftarrow$ false
  while !$foundMatchingCombination$ **do**
    for $j = 1$ **to** $|\mathcal{P}|$ **do**
      $\mathcal{F}_{i,j} \leftarrow findCandidateSensorEvent(T_{i-1}^j, r)$
    end for
    $C_i \leftarrow LogicalOR(\mathsf{X}_{k=1}^{|\mathcal{P}|} \mathcal{F}_{i,k})$
    for $j = 1$ **to** $|C_i|$ **do**
      if $C_{i,j} == SE_i$ **then**
        $foundMatchingCombination \leftarrow$ true
        $subScripts = indexToSubscript(j, |\mathcal{F}_{i,1}|, |\mathcal{F}_{i,2}|, \ldots, |\mathcal{F}_{i,|\mathcal{P}|}|)$
        for $k = 1$ **to** $|\mathcal{P}|$ **do**
          $T_i^k \leftarrow sensorEventToLocation(\mathcal{F}_{i,k_{subScripts_k}})$
        end for
        Break
      end if
    end for
    if !foundMatchingCombination **then**
      $r \leftarrow r + \Delta r$
    end if
  end while
end for

each individual's actual location in space. The *break* statement on Line 19 considers the first match between members of C_i and the SE_i being considered, neglecting the rest of the discovered combinations. It should be noted that this process deals with the question of dealing with the speed of movement of the occupants in the space, assuming that we start with a small ("conservative") initial speed.

3.5.1 An Example

To illustrate the trajectory-extraction and error-calculation functions, consider a scenario (see Appendix A for a detailed explanation) of two people walking in the environment. According to Figure 3.6(a,b) occupants P_1 and P_2 start from two different starting points (at t_0) and walk towards each other. After they meet, each takes a separate route diverging from each other and finally stop. Suppose that P_1 is identified by an RFID reader at the end of the path. The ground truth for the trajectories of occupant P_1 and P_2 is shown in green and blue circles respectively. As they walk, they trigger a number of motion sensors and a sequence of combined sensor events (Figure 3.7) are produced. The sensor events are then fed to the path-extraction algorithm (described in Algorithm 1) in order to produce unlabeled trajectories for the two individuals. The individual trajectories are shown as connected black squares in Figure 3.6(c).

3.5.2 The Environment Model Graph

The information from RFID readers is used to annotate the occupants' trajectories (inferred based on PIR sensors) with the IDs of the occupants who have "likely" caused these sensors to fire. The process is based on the directed *EM Graph* inspired by the "tracklet" graph of Ivanov et al. [41]. Assume an EM Graph $G(V,E)$; nodes in G correspond to certain events in time: they are either (a) a starting point of an occupant (denoted as s_{P_i,t_0}, indicating the initial position of person P_i); or (b) identifications, indicating that a person has been unambiguously identified through RFID events (denoted as l_{P_i,t_m} showing that at time t_m, P_i was detected by the RFID that triggered the event); or (c) collisions, *i.e.,* points not covered by RFIDs where, based on PIR observations, a number of potential occupants may congre-

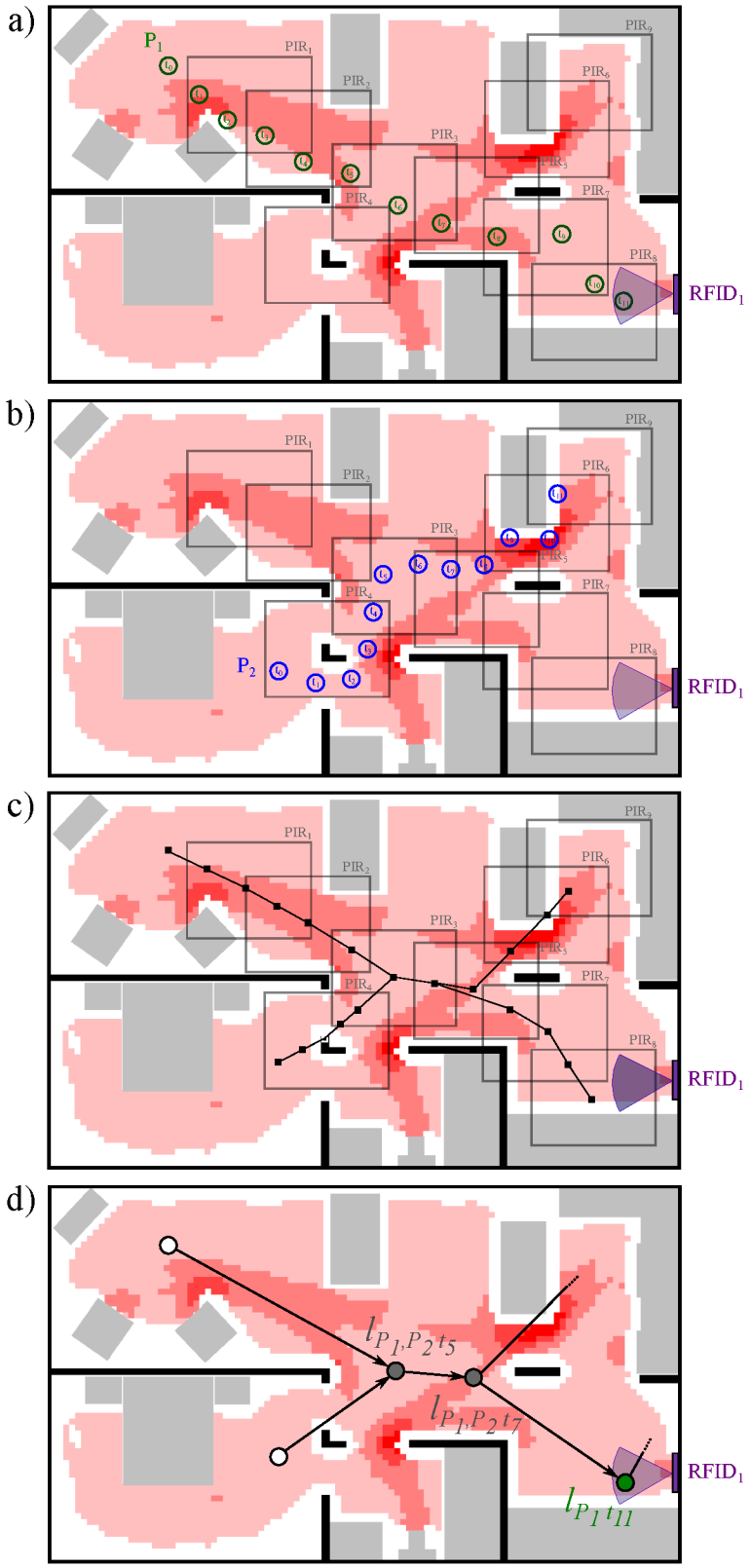


Figure 3.6: Example: (a) raw traces for occupant P_1, (b) raw traces for occupant P_2, (c) unlabeled trajectories of the two individuals, and (d) the derived EM Graph.

Sensor Events:
t_0: 00010000000
t_1: 10010000000
t_2: 10010000000
t_3: 11010000000
t_4: 01110000000
t_5: 01100000000
t_6: 00101000000
t_7: 00101000000
t_8: 00001010000
t_9: 00000110000
t_{10}:00000111000
t_{11}:00000101100

Figure 3.7: Sensor events for occupants P_1 and P_2.

gate ($l_{P_i,P_j,\ldots,t_n}$). In Figure 3.6(d), nodes corresponding to starting points are white, those corresponding to identifications are green, and those corresponding to collisions are shown in grey. In this graph, an edge is a maximal length sequence of locations in time that are confined between two nodes of the graph (including identifications by RFID readers) and represents the trajectory of at least one person (edges between two collision nodes represent the trajectory of more than one person). Each edge e is annotated with an occupant set by X_e which carries the IDs of occupants who have been (likely) located at the source and the destination of the edge. The cardinality of these sets, in effect, are a proxy measure for the degree of ambiguity in the occupant-locations' inferences: on average, higher cardinality implies more (undesirable) "confusion". Therefore, our objective in deciding the RFID reader placement is to reduce the cardinality of those sets.

For each $e \in E$, we initialize its occupant set X_e to have all members in $\mathcal{P}$. Let us assume we have a reading from an RFID reader (hence authoritative) identifying P_i at t_m. Note that the process repeats as long as we get new P_i readings, and until no changes can be made to the occupant set of any edge. Then, we proceed to reduce the X_e sets through a heuristic process involving the four steps below, applied to each identification node, l_{P_i,t_m}, in the EM Graph. (**step 1:**) All IDs other than P_i are removed from the edges adjacent to l_{P_i,t_m}.

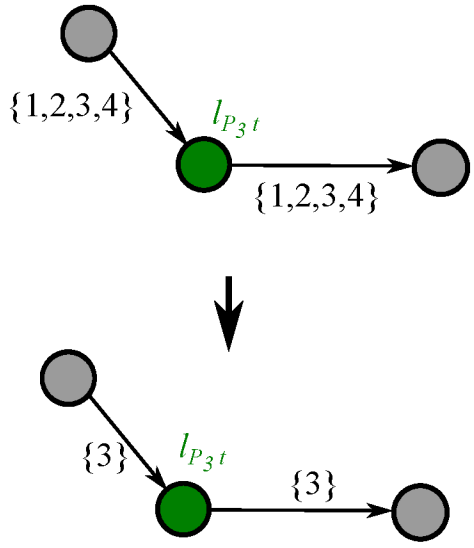


Figure 3.8: Reducing X_e sets, step 1.

If P_i was unambiguously identified at a location in some point in time, and no collision with another trajectory has occurred, there is no other individual that could have walked in (away from) this location at this point in time. (**step 2:**) P_i is removed from all other components[2] of the graph that do not contain the identified P_i, since P_i can only belong to a single component of the graph. (**step 3:**) P_i can be removed from paths that diverge from collision nodes in which P_i has participated prior to the current identification, *i.e.,* if P_i has separated at a collision node, and later authoritatively identified, all other paths from that collision node cannot possibly contain P_i. (**step 4:**) Similarly to (step 3) but in reverse temporal order, P_i can be removed from paths that, after the current identification, join at future collision nodes but cannot contain the identified P_i. An example for each of the above mentioned steps are presented through Figures 3.8 - 3.11.

[2]A component is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph. A graph may contain multiple components.

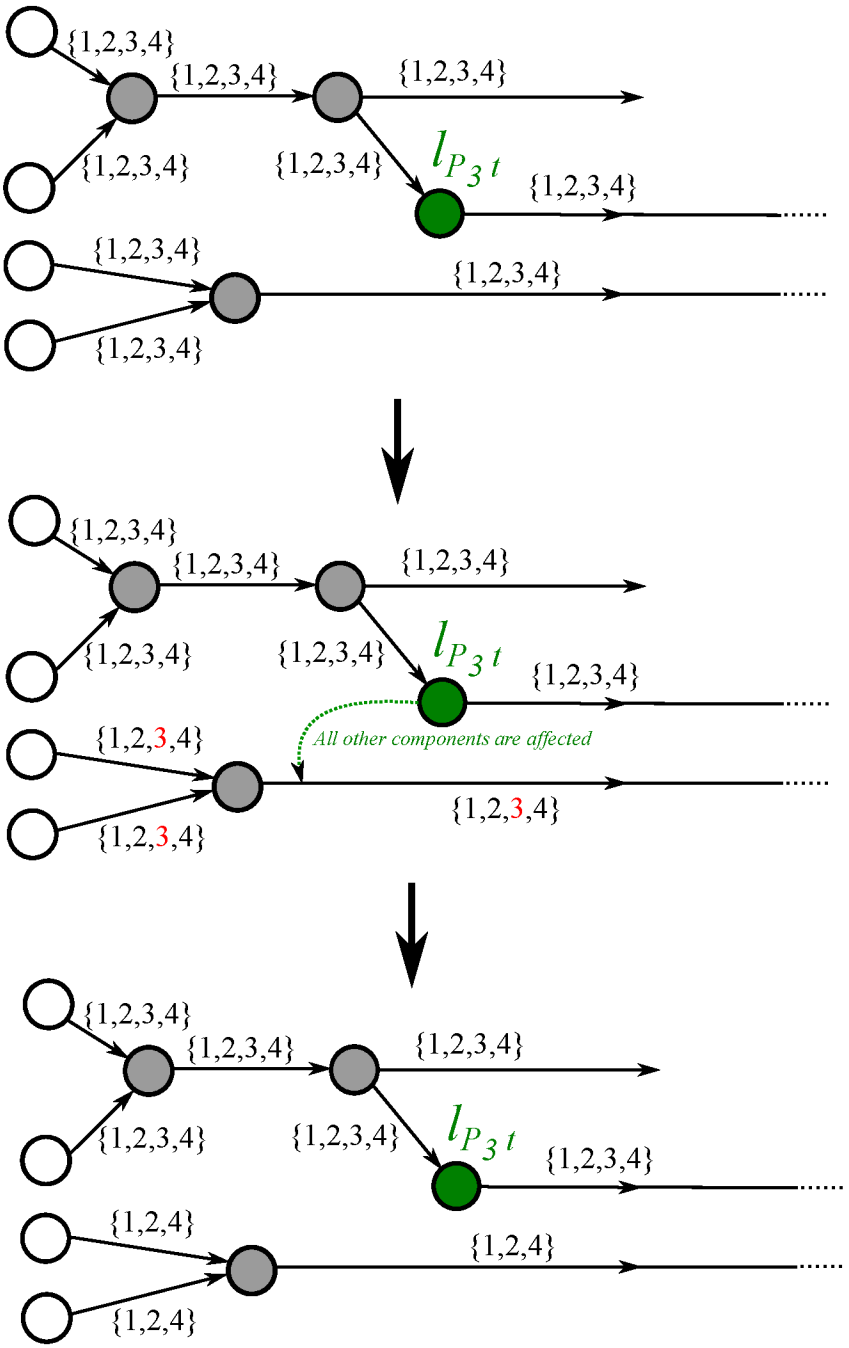


Figure 3.9: Reducing X_e sets, step 2.

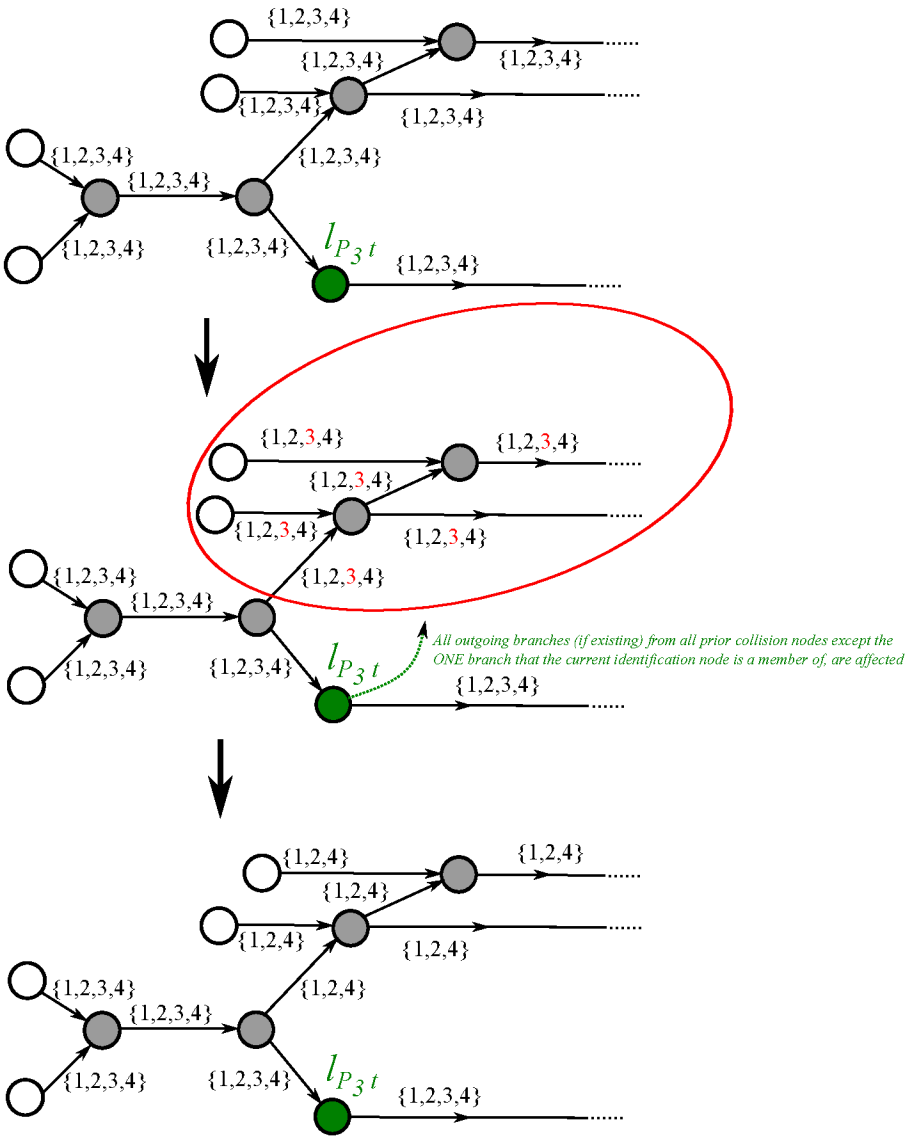


Figure 3.10: Reducing X_e sets, step 3.

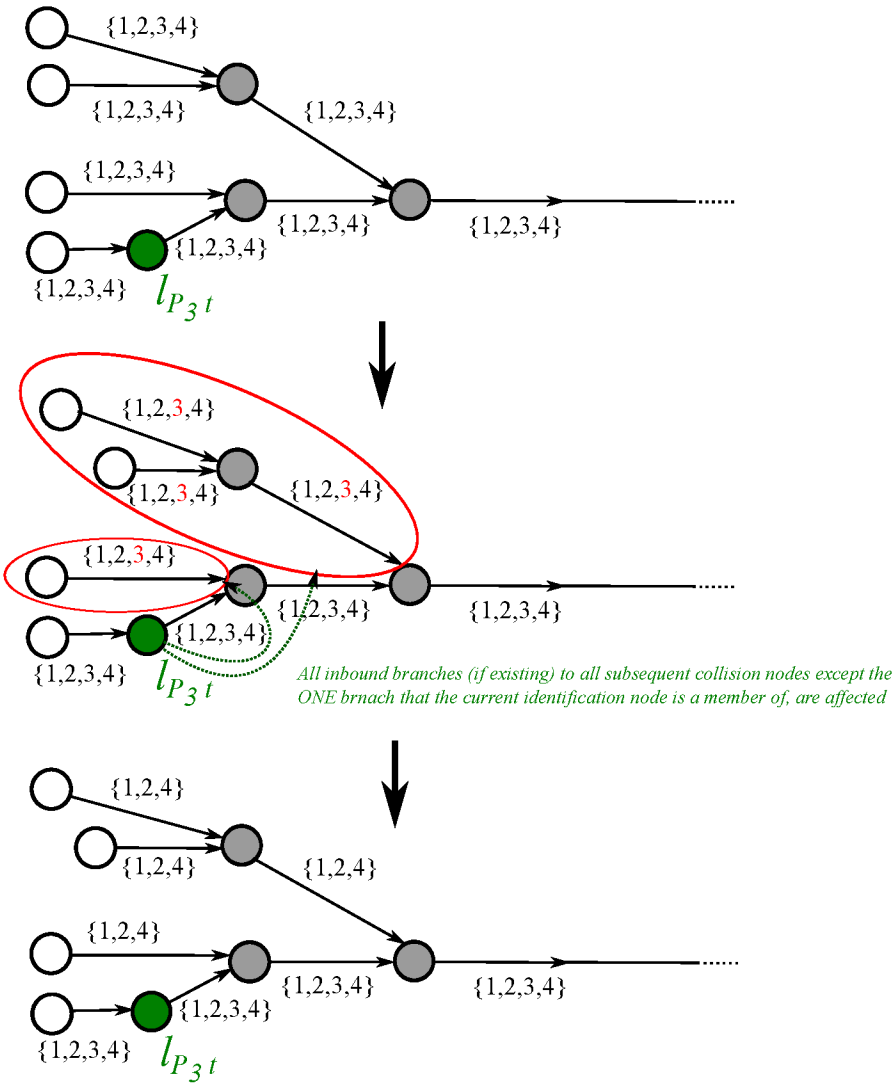


Figure 3.11: Reducing X_e sets, step 4.

3.5.3 Performance Metrics

The *ambiguity* metric indicates the extent to which a particular occupant's trajectory has been incorrectly inferred. It is defined as the ratio of the sum of the durations of the collision-graph edges with non-singleton occupant sets of size ℓ, that include the individual's *ID*, over the sum of the durations of all the edges whose occupant sets include the individual's *ID*. Therefore, each individual's ambiguity is represented by the set $A_{P_i} = \{a_2^{P_i}, a_3^{P_i}, \ldots, a_{|\mathcal{P}|}^{P_i}\}$ where $a_\ell^{P_i}$ is calculated according to Equation 3.7.

$$a_\ell^{P_i} = \frac{\sum_{e \in E, P_i \in X_e, |X_e| = \ell} \bar{d}_e}{\sum_{e \in E, P_i \in X_e} \bar{d}_e} * 100\% \tag{3.7}$$

Here $\bar{d}_e$ is the duration of the edge e. The overall (across all occupants) ambiguity is then calculated as the average of all individual ambiguities:

$$A^* = \{a_\ell^* : \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} a_\ell^P | \ell = 2, 3, \ldots, |\mathcal{P}|\} \tag{3.8}$$

The *tracking-error* metric consists of a lower and an upper bound for the localization error. Given an individual, all the segments that this individual may have traversed based on the occupant sets of the collision-graph edges, are collected. Of those, the two paths with largest and smallest Euclidean distance from the person's actual path are used, respectively, to compute the upper bound (Tracking Error Upper-bound, TEU) and the lower bound (Tracking Error Lower-bound, TEL) error. TEU and TEL are calculated per-individual and averaged across time.

In the example with two individuals of Figure 3.6, the EM Graph produced (Figure 3.6(d)) captures the fact that the two occupants collide with each other between t_5 and t_7. Using the disambiguation based on the RFID reader information, some parts of the trajectories are disambiguated and some remain ambiguous. Specifically, the trajectory between times t_0 and t_7 is ambiguous for both individuals. The ambiguity error for each person is (according to Equation 3.7): $A_{P_1} = A_{P_2} = \{\frac{t_7 - t_0}{t_{11} - t_0} * 100\%\}$. A^*, which is the average ambiguity and is also equal to A_{P_1} (see Equation 3.8). According to the definition

in Section 3.1.1, TEL and TEU are the sum of the Euclidean distances (orange lines in Figure 3.12(a) and Figure 3.12(b), respectively) averaged across time and divided by the number of individuals.

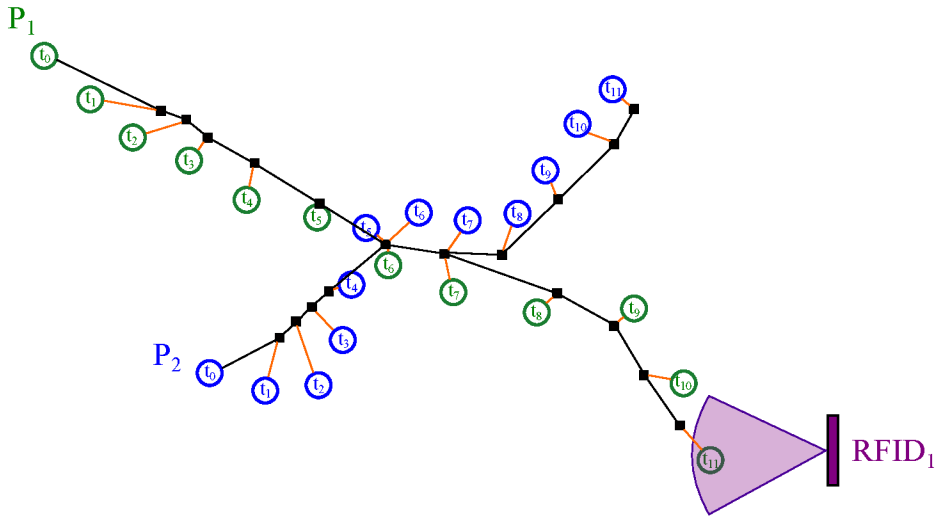
3.6 Problem Formulation

Our work is driven by the geometry of the indoor space and the geometry of the areas covered by each sensor variety. We assume the existence of (and use) a heat-map of the *visitation frequency* of each point in the sensed environment, as in Vlasenko et al [1]. This two-dimensional map, includes the location of the walls (W) and obstacles (O). The remaining points (I) have an information utility which is the probability of a person being present at that point. Overall, the heat-map contains N points, $(x_1, y_1, l_1), \ldots, (x_N, y_N, l_N)$. l_i indicates which group, W, O or I the point belongs to.

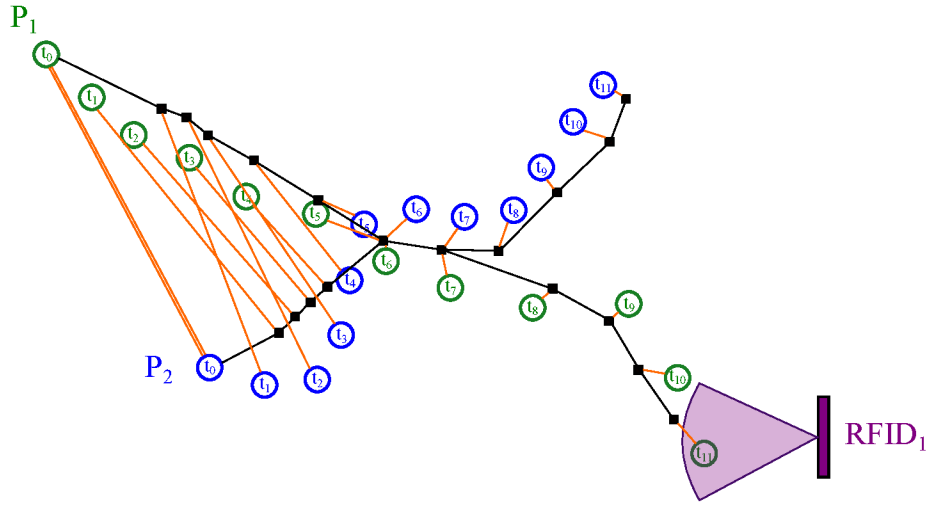
The objective of our method is, given k ($k > 1$) individuals present in and moving around the environment, to reduce the error in inferring the location of each individual in the space at any point in time. The error we aim to minimize is TEU (details described in Section 3.5.3), or in mathematical terms:

$$min\ \mathbb{E}(TEU) \tag{3.9}$$

Although minimizing the overall ambiguity error (*i.e.,* $min\ \mathbb{E}(A^*)$) could also be pursued as an objective function, we do not think that it is an explicit expression for RFID+PIR placement determination. This is because, $\mathbb{E}(A^*)$ (unlike $\mathbb{E}(TEU)$ which is expressed as Euclidean distance) is sensitive to the number of tracked individuals (as will be shown in Section 5.3) and even if it is large it could respond to a small, practically speaking, error in terms of locations.



(a) TEL



(b) TEU

Figure 3.12: Example TEL and TEU calculations.

Chapter 4

PIR Sensor Placement

4.1 Introductory Background

In this chapter we focus on an orthogonal aspect of the problem, expanding the problem formulation presented in [1] to include the various sensor footprints (from a finite set of footprints) and to determine the optimal number of sensors (not just their placement). The immediate implication of this more general problem formulation is a significantly enlarged search space, due to increased number of possible combinations of placements and footprints. We address the increase in complexity using GAs. Evolutionary methods, based on GAs, are frequently employed to explore large problem spaces in order to identify high-quality solutions. Our sensor placement problem naturally belongs to this category. The main reasons behind choosing the GA over other traditional search methods are: (a) in this problem formulation there are multiple local optima, (b) the number of parameters is large and (c) it supports multiple objectives. The GA technique elaborated in this thesis outperforms the greedy algorithm in [1] in two different aspects. First it efficiently delivers placements with acceptable coverage accuracy. For example, we have noted that it reaches 94% coverage after just 50 seconds of execution on an off-the-shelf personal computer. Second, it delivers placements with higher-accuracy when efficiency is not a factor. It is able to eventually reach coverage of 100, whereas the greedy algorithm can do no better that 98%.

4.2 PIR Sensor Placement Methodology

GA solutions are represented as a *population* of *chromosomes*, where each chromosome consists of *genes*. In our model, the genes are a sensor's type and its (x, y) location. In each evolution, new, and hopefully improved, solutions are developed by applying a number of different *operators* on the population. The criterion for deciding whether a particular chromosome is better than another is defined through a *fitness function*, also known as the *cost function*. The classical GA methodology that we use defines the following steps:

- **Start:** Generate an initial random population of chromosomes representing potential problem solutions, which in our case is sensor combinations. Section 4.2.1 discusses the encoding of sensor placements as chromosomes and the construction of the initial population.
- **Loop:** Create a new population by:
  - selecting a percentage of the current population as parents, and performing crossover to produce new offsprings,
  - performing mutation of the new offsprings,
  - applying elitism, i.e., including the new offsprings in the new population but reducing the population to its original size, keeping only the best solutions for the next iteration.

  Section 4.2.2 discusses the process of population renewal.
- **Termination Check:** If an end condition has been reached,
  - return the best solution in the current population,
  - else, go to *Loop*.

  Section 4.2.3 discusses the termination conditions of our method.

We note that the GA methodology requires a number of parameters for its configuration, which must be designed taking into account the specifics of the problem domain. A list of these parameters is given in Table 4.1.

Table 4.1: GA parameters.

Parameter	Description
m	Population size
u	Number of children produced in each iteration
n	Number of sensors in each initial chromosome
max	Maximum number of sensors that each solution can have ($\|\| K \|\|$)
p_c	The probability of crossover for the parents
p_m	The probability of mutation on each child
l	Number of iterations for the whole process

4.2.1 Chromosome Encoding

Each chromosome in the population contains a number of *genes*. In the context of our sensor placement problem, genes are sensors with coverage models introduced in Section 3.3. Each gene also carries information about the corresponding sensor's type (thus its footprint) and its location in space. Upon initialization, the type of the gene is selected randomly, with a uniform distribution, between all available sensor types (*i.e.,* circle, square or rectangle) considering various orientations as different types. The location of each gene is also chosen according to a uniform distribution amongst all points in space.

A chromosome can be represented by a set of sensors as follows:

$$C : \{s_1, s_2, \ldots, s_{\|C\|}\} \tag{4.1}$$

By combining equation 3.5 and 3.4, we can conclude that $P_o^C = 1$ whenever there is at least one sensor in C that covers o.

In order to evaluate the performance of a solution found by the genetic algorithm, we divide the chromosome's collective information gain (calculated according to Equation 4.2) by the summation of positive coverage utility values. This will give a metric that we call

the *coverage percentage metric*, formulated as follows:

$$\lambda^C = \frac{\sum_{i=1}^{N}(P_i^C.c_i)}{\sum_{\substack{i=1 \\ c_i \geq 0}}^{N}(c_i)} * 100\% \tag{4.2}$$

In the *Start* step, we have to create the initial population for the algorithm to use as its first evolution. This is done by randomly creating *m* chromosomes, *i.e.,* sensor placements. The initial length of the chromosomes must be set to a value less than or equal to a user-specified upper bound called *max i.e.,* ($n \leq max$).

4.2.2 Selection, Crossover, Mutation and Elitism

To produce children two parents must be selected for a crossover procedure. The parents are chosen randomly from the population, according to a selection percentage also known as the *crossover probability*, (p_c). For example, if the population consists of 100 chromosomes, and $p_c = 0.2$, then approximately 20 parents are randomly chosen to participate. From the chosen parents, two parents are randomly selected to create two children. A second parameter, *u*, decides the total number of children that will be produced in each iteration. We continue the procedure until we have all *u* children.

We use the "cut and splice" crossover method [42]. In this method, the crossover point for each parent is chosen separately and randomly. As shown in Figure 4.1, the first part of the first parent and the second part of the second parent are used to construct the genes in the first child, and the rest construct the second child. Figure 4.1 shows how cut and splice is performed to produce two children from two parents. This crossover method leads to offsprings with different chromosome lengths, which in our case implies different numbers of sensors placed. As we are seeking the number of sensors to use, as well as the sensor locations, this method helps achieve this. In addition, throughout the whole process of the GA, chromosome lengths must remain less than or equal to *max*.

Once *u* children chromosomes have been produced through the parent-population crossover, according to the mutation probability (p_m), they might be mutated. The chromosome mutation operation that we use in our experiments is called a uniform mutation [42], which

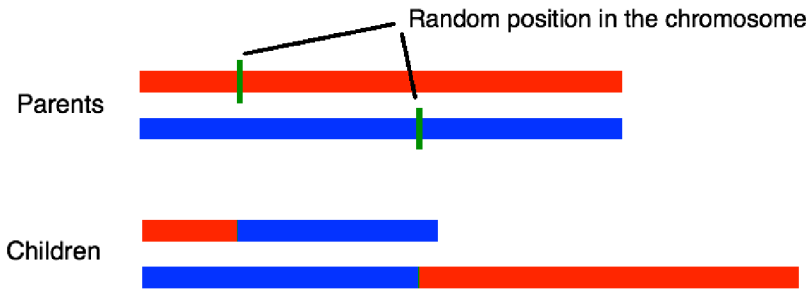


Figure 4.1: "Cut and splice" crossover method for producing children chromosomes.

involves randomly choosing a gene from the chromosome and performing the mutation operation on that gene. For this, we either use relocating the sensor to a new random location or we change its type, for example, turn its coverage footprint from a disk to a square.

The last step of this phase involves adding the new children in the population, and selecting the best chromosomes from the combined population for the next iteration so that the population cardinality always remains stable. This selection is accomplished through the fitness function.

The fitness of each chromosome is evaluated according to the following criteria: (a) Sensors should cover as much *heat* as possible; at the same time they are penalized for covering "restricted" areas, *i.e.,* areas known to not require coverage (b) the number of contributing sensors should be minimized, therefore shorter chromosomes are preferable to longer chromosomes. In order to capture these criteria, we have formulated the fitness function to be:

$$F^C = w * \sum_{i=1}^{N} (P_i^C.c_i) - \sum_{i=1}^{\|C\|} \| A_{s_i} \| \tag{4.3}$$

Where P_i^C can be computed using equation 3.5 and w and is a fixed values. The value of w shows the user's preference between better coverage of the space ($w > 1$) and using fewer sensors in the solution ($w < 1$). $w = 1$ will indicate no significant preference between the two factors.

4.2.3 The Termination Criterion

In principle, the evolutionary process terminates if (a) the maximum number of iterations or execution time has been reached, or (b) the average fitness of the population does not change over a certain number of evolutions, or (c) 100% accuracy is achieved. When the genetic algorithm terminates, the best solution in the current population is returned as the final solution.

4.3 PIR Sensor Placement Evaluation

The evolutionary process described above is controlled by the parameters listed in Table 4.1. As it is a convention for users of evolutionary algorithms, parameter tuning needs to be performed based on experimental comparisons on a limited scale [43]. We follow this requirement by fine-tuning the GA model parameters on a simple artificial scenario. In the heat-map of Figure 4.2, the overlapping footprints of four sensor types are shown. The different footprint shapes correspond to different sensor types; the first type projects a square footprint ($edge = 175$ points); the second type projects a disk ($radius = 100$ points); and the third and fourth type are two different oriental (0° and 90°) footprints of a rectangle ($length = 200$ and $width = 150$ points). The darker the colour, the higher the significance of the area, *i.e.,* the higher the utility of the area. The environment shown in this simple heat-map consists of 6 different regions to be covered by the sensors, and the ideal solution should place exactly 6 sensors of the right types in exactly the right locations. Given this desired solution, we proceeded to identify the parameter configuration that results in the solution at hand.

Because of the implicit randomness of the evolutionary GA process, we cannot predict the exact resulting amounts for coverage percentage. So, the experiments [1] conducted in this section are tested multiple times (*i.e.,* 50 times), and the average is taken.

To configure the algorithm parameters, we started by optimizing p_c, with the following initial settings: $m = 50$, $n = 10$, $u = 50$, $l = 50$ and $p_m = 0.2$. Figure 4.3 shows how the coverage percentage changes for different values of p_c. Lower p_c values result in lower

[1] All experiments are run on Mac OS X, Processor: 1.7 GHz Intel Core i5, Memory: 4 GB 1600 MHz DDR3, platform: Java.

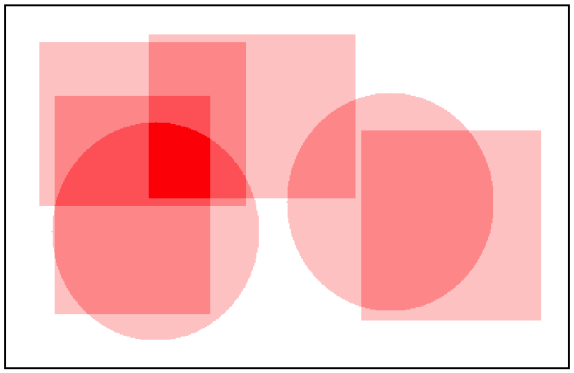


Figure 4.2: Artificial input with 6 overlapping regions to cover.

accuracy. This is because, when there is little crossover in the evolutionary process, the potential parents are less likely to pair with each other to produce better solutions. On the other hand, a high crossover probability also threatens accuracy, because good subsets of sensors inside the chromosomes have a high risk of getting involved in a crossover procedure and being separated from each other; in these cases, the good subset, which should have been passed to the next generation of chromosomes, is lost. We notice that the coverage percentage at $p_c = 0.4$ yields the best result. We used this value for the remainder of the experiments and proceeded to select p_m through the experiments shown in Figure 4.4.

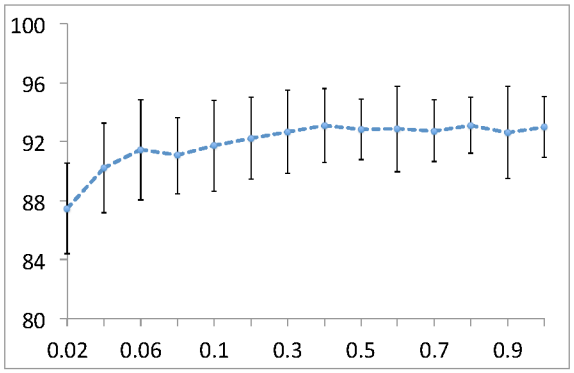


Figure 4.3: Coverage percentage for different values of p_c.

As p_m increases, more mutations occur and, as a result, the diversity of the chromosomes increases. However, too much mutation may result in losing good chromosomes generated through crossover. Based on the data of Figure 4.4, p_m was set to 0.5 for the rest

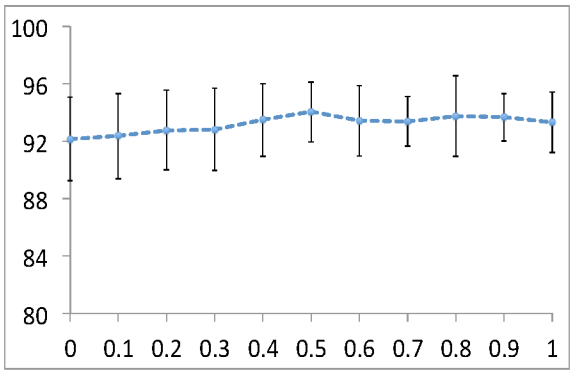


Figure 4.4: Coverage percentage for different values of p_m.

of the experiments.

Figures 4.5, 4.6 and 4.7 show the impact of l, m and u, respectively. According to Figure 4.5, higher number of iterations lead to better coverage percentages. Also, as Figure 4.6 suggests the value of m should be set to be the same as u, which in this case is 50. In Figure 4.7, we note that increasing the value of m (which is equal to u) will result in higher accuracy. This is because, more children are produced in each iteration, increasing the likelihood of finding a better solution. Further, as the number of iterations increases, the likelihood of identifying a better solution increases since more of the solution space is explored. However, the downside to increasing m, u and l is that execution time will also increase. So the question becomes how to decide the tradeoff between m, l and accuracy versus time consumption.

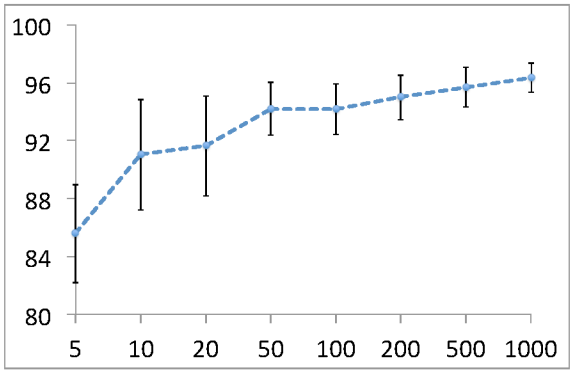


Figure 4.5: Coverage percentage for different values of l.

To answer this question we experimented with different configurations of m during

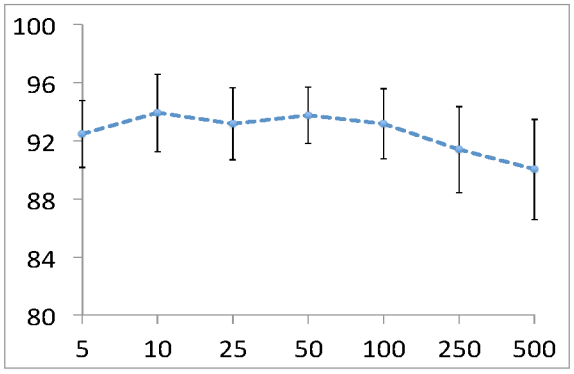


Figure 4.6: Coverage percentage for different values of m.

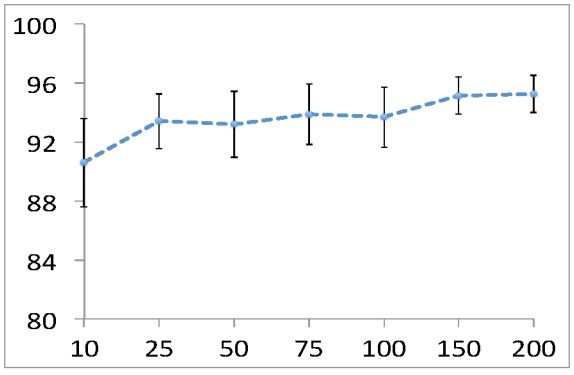


Figure 4.7: Coverage percentage for different values of u.

different time periods. As shown in Figure 4.8 a larger population size results in better accuracy, given the same amount of execution time. Note that, because more execution time is consumed to produce children through crossover in each iteration, there will be fewer iterations in total.

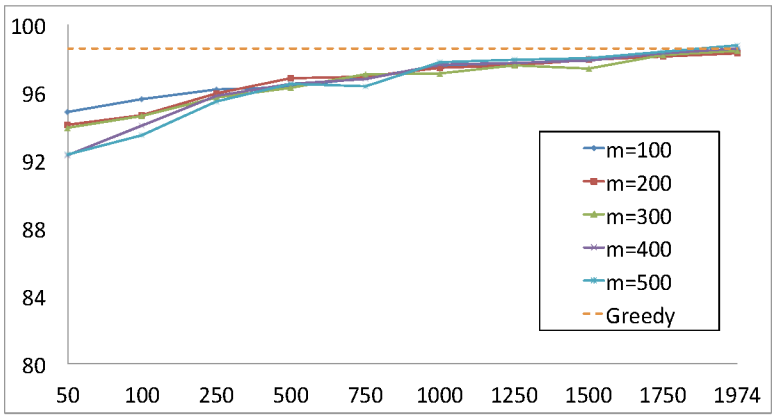


Figure 4.8: Results from GA for different values of m.

In addition to running the same algorithm multiple times, we also illustrate a logarithmic trend line function in order to visualize how different configurations of the GA are performing. The trends in Figure 4.9 indicate that although $m = 500$ performs slightly worse than $m = 100$ at the beginning, it catches up and finally exceeds the accuracy achieved with $m = 100$. In conclusion, the process of parameter fine-tuning reported in this section produced the following parameter configuration: $m = 500$, $u = 500$, $p_c = 0.4$ and $p_m = 0.5$. When m is higher than 500, results deteriorate because the time spent in each iteration on producing children increases and in a given amount of time the algorithm cannot perform enough iteration to achieve acceptable results.

Comparison Against the Greedy Method: Let us now compare our GA method against the greedy algorithm by [1]. In the greedy method, sensors are placed one after the other, in a location which optimizes the additional "heat", i.e., utility, covered. Once a sensor is placed, the heat in the area covered by the sensor is reduced, since gaining more localization accuracy in a given area is becoming increasingly less useful as the area gets covered. This procedure continues until the desired number of sensors has been reached. It is important to note here that, in the greedy method [1,39], the number of sensors to be placed is assumed

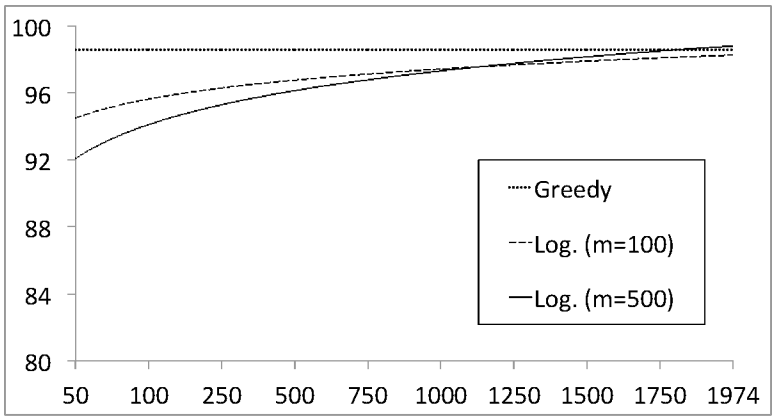


Figure 4.9: Trend lines shown for $m = 100$ and $m = 500$.

to be decided a-priori, based on the cost that can be afforded. Near-optimal results are guaranteed in this method when certain types of sensors are used [1]. Although realistic, this problem formulation does not address the cases where the budget is flexible and the users desire to explore the cost/accuracy tradeoff. Another fundamental shortcoming of the greedy method is that it the time consumes grows rapidly as new types of sensors are added.

Running the greedy algorithm on the simple heat-map of Figure 4.2 always returns the same result of 98.58 percent coverage, after an average time of 1974 seconds. Figure 4.9 shows that after approximately 1800 seconds the GA surpasses the greedy method. The greedy does not reach 100 percent accuracy with the simple case heat-map, because it fails to exactly match every area with the right sensor (Figure 4.10); it always starts by covering the highest-utility area, i.e., the darkest area in the heat-map, and gets stuck in a local optimum. This phenomenon does not happen with the genetic algorithm. The GA can get even 100 percent accuracy thanks to its ability to find the best combination of sensors and not focusing on one at a time (Figure 4.11). Although the process may take time for very high accuracy (*i.e.,* 37333.55 seconds for 99.81%) but in time critical applications, it can achieve very high accuracy in short time periods. Accuracy as high as 94.81 % can be reached within the first 50 seconds of execution time.

In order to compare the two methods more realistically, we use the two complicated and larger scale heat-maps, shown in Figures 3.1 and 3.2. The scale of the Smart-Condo™

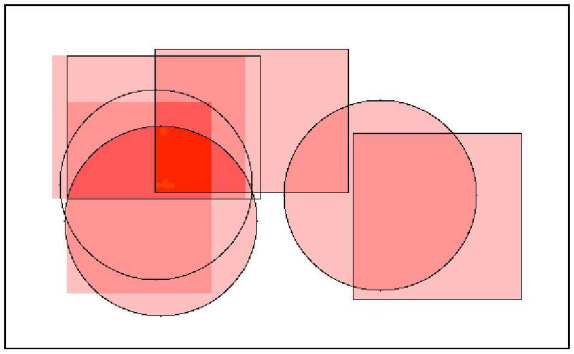


Figure 4.10: Sensor placement on a simple case heat-map with the greedy method.

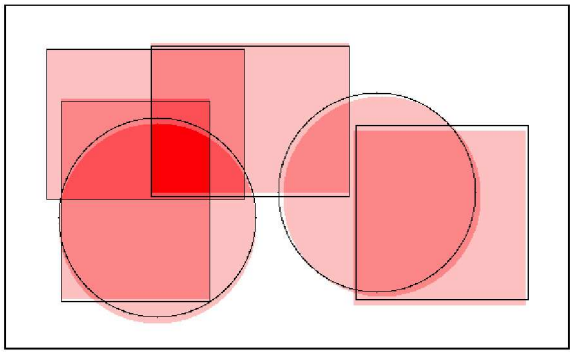


Figure 4.11: Sensor placement on a simple case heat-map with the GA method.

heat-map is 629×1060 points and for the ILS it is 573×651 points. Moreover, the value of c_{max} is set to 4 while producing these heat-maps. It is noteworthy that the data reported throughout the remainder of this section for the GA is the average of 10 runs.

The comparison methodology here is to first see how the greedy method is preforming with a designated budget of sensors and mark its coverage percentage (calculated using equation 4.2) for that budget as the "desired accuracy". After that, we give the GA the same budget and see how fast we can achieve coverage percentages higher than or equal to the desired accuracy.

In the first experiment conducted, we only use one type of sensor footprint, the rectangle. Say the budget for covering the ILS is 15 sensors, we would like to know how much coverage each method achieves having this budget. It turns out that greedy can get 81.48% coverage percentage in 466.7 seconds. This value becomes our desired value, that the GA aims to reach. The GA can achieve this accuracy in just 41.5 seconds on average.

Now, let us use a sensor set consisting of more than one type of sensor. In the second experiment we will use a budget of 5 squares, 5 disks, and 5 rectangles (all with the same sizes as before). The same procedure as in the first experiment is adopted to fill tables 4.2 and 4.3 which show the results for both methods when dealing with heat-maps 3.1 and 3.2, respectively. In addition, we record the fitness (calculated according to Equation 4.3) of the solution as well as the average and standard deviation of chromosome fitness in the population in which that solution was found (namely solution cost, average cost, and cost STD).

Table 4.2: Comparison of different methods in the Smart-Condo™ layout.

Desired	Greedy		GA				
Accuracy(%)	nos[2]	time(s)	nos	time(s)	Average Cost	Best Cost	σ of Cost
74.29	10	3725	10	185	11933.41	12104.76	49.73
76.82	11	4231	11	275	12744.75	12873.20	44.36
79.93	12	4622	12	391	13313.95	13450.34	53.05
83.69	13	5023	13	387	14401.83	14488.54	47.51
84.50	14	5380	14	372	14389.32	14467.30	45.90
85.25	15	5572	15	348	14453.41	14595.90	51.05

Table 4.3: Comparison of different methods in the ILS layout.

Desired	Greedy		GA				
Accuracy(%)	nos	time(s)	nos	time(s)	Average Cost	Best Cost	σ of Cost
72.45	10	643	10	56	2442.19	2487.82	16.55
73.77	11	741	11	35	2313.08	2432.68	34.65
76.80	12	837	12	52	2433.45	2504.76	22.44
77.60	13	912	13	62	2461.45	2529.08	26.86
79.43	14	967	14	58	2442.21	2557.78	31.65
80.80	15	996	15	50	2406.02	2524.48	28.05

According to these tables, the GA reached the desired accuracy checkpoint with the same budget of sensors in considerably less time. Comparing the results from experiments one and two, we notice that increasing sensor types affects the greedy's time consumption substantially, while increasing the GA's only slightly.

The GA can continue beyond this point and find even better solutions which brings us to our third and final experiment. In this experiment, again we get back to only having 15 rectangles and want to see how much accuracy we can achieve if we let the GA consume the same amount of time that the greedy has used. The GA reaches an average coverage percentage of 83.55% and shows its superiority when it comes to hitting high accuracy, too.

[2]Number of sensors.

Chapter 5

RFID Reader Placement

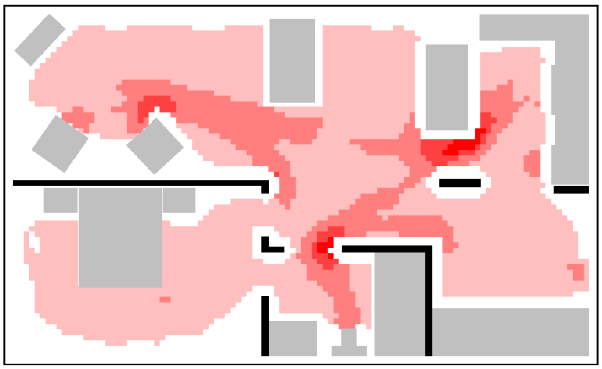
In the previous chapter we have explained the work in [1] via a GA formulation to generalize the placement for PIR sensors with a variety of footprints. Next we determine the RFID placement driven by our *min* $\mathbb{E}(TEU)$ objective introduced in Chapter 3.

5.1 The Skeletonization Process

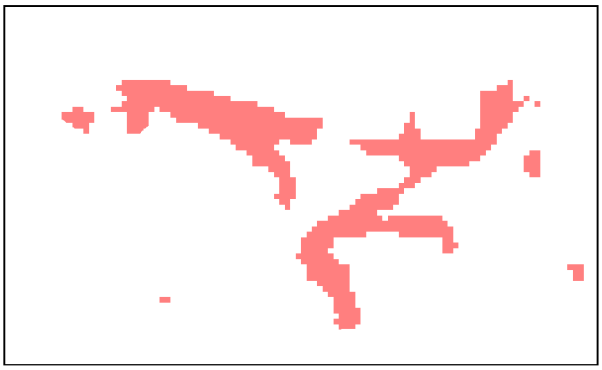
The RFID readers have to be judiciously placed in order to provide ample (frequent) opportunities for RFID reader data to be acquired. At the same time, the number of RFID readers is typically limited due to their cost and awkward placement requirements. Poor placements may lead to undesirably high ambiguity (defined in 3.1.1). To avoid this situation, we use the *skeleton* of the heat-map (Figure 5.1(c)). The heat-map skeleton is derived through a two-step process, involving (a) thresholding, and (b) iteratively thinning the binary values created by the thresholding process. The first step uses a configurable threshold value τ_{hm} to separate significant intensities (*i.e.,* really frequently traversed areas) from less important information in the heat-map (0 for less than τ_{hm}, and 1 otherwise). The newly created binary map is subsequently iteratively run though a thinning process until thinning no longer changes the result (more details on the algorithm can be found in [44–47]). The output of this skeletonization process represents a useful insight about the most likely paths chosen by the occupants in a long run. It is a means of capturing the "thoroughfares" of the occupants' movement through the space.

Having the skeleton, a pruning step is performed to remove short branches and cy-

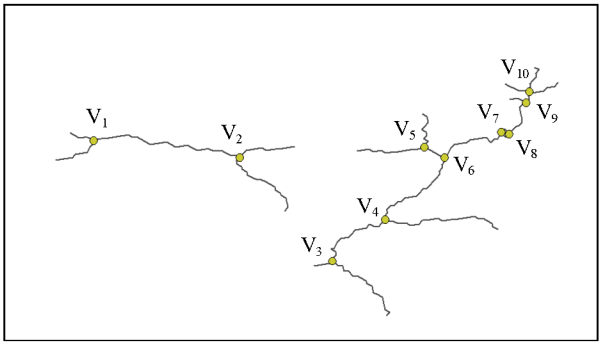
cles (see Figure 5.2 for example). Removing branches is controlled by a user-specified parameter, τ_{prune}; cycle elimination is accomplished by removing the least significant branch within each cycle. The final result is represented by an non-directional n-ary tree, $T = (V_t, E_t)$. The vertices of T are also called the branching points and the edges are the actual paths (*i.e.,* they correspond to a sequence of real spatial points, and are not just links between vertices). Figure 5.1 shows each step of the conversion from heat-map to a usable tree of paths.



(a)Coverage utility heat-map of a person walking in the Smart-Condo™

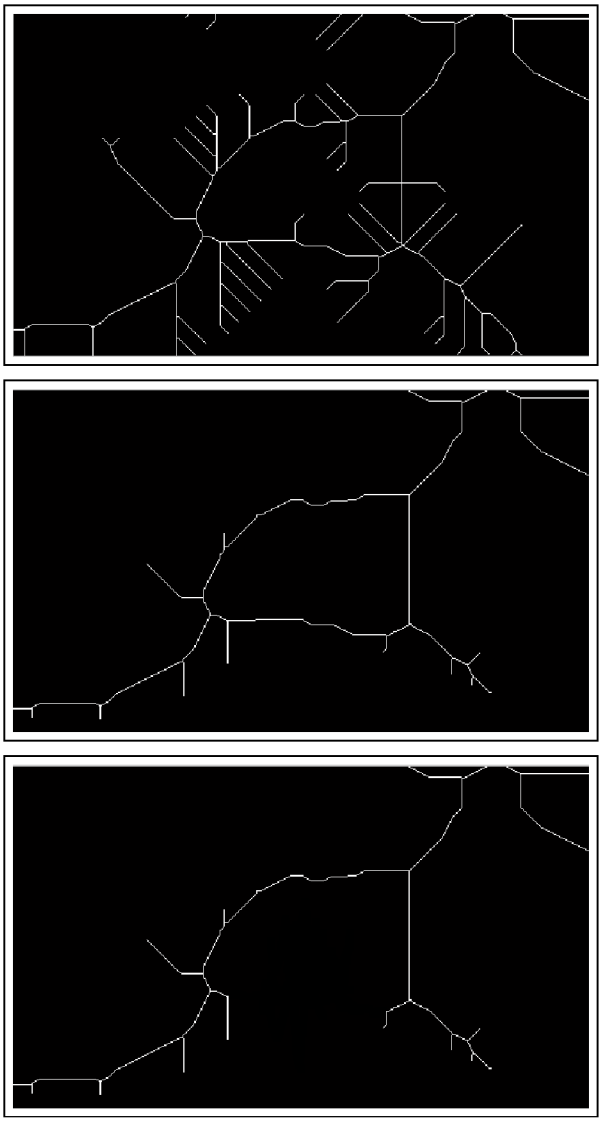


(b)Thresholding with $\tau_{hm} = 2$.



(c)Tree resulted from skeletonizing the heat-map.

Figure 5.1: Skeletonization process



(a) original skeleton ($\tau_{hm} = 1$).

(b) removing short branches ($\tau_{prune} = 45$).

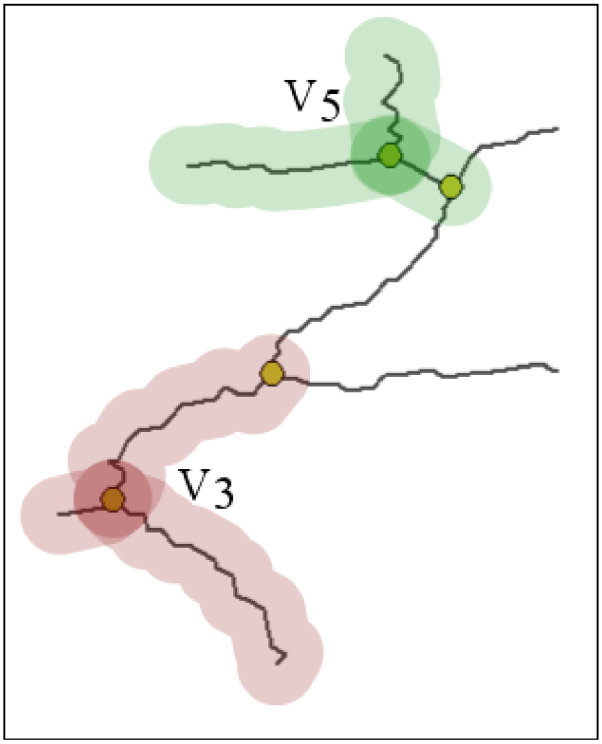
(c) removing cycles

Figure 5.2: Skeleton Pruning

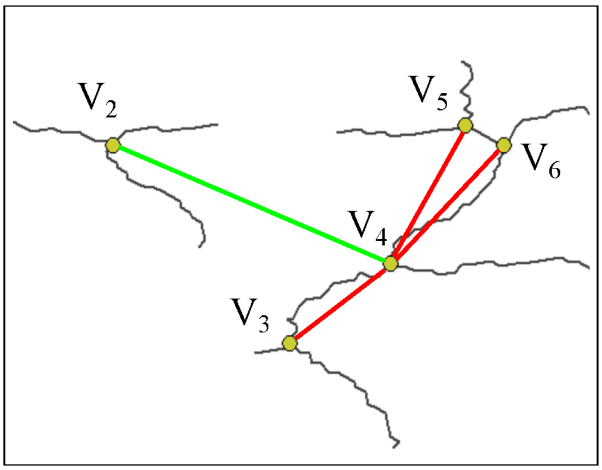
5.2 Proposed Heuristic

The Score Function: The heuristic for the RFID reader placement is based on a score function ($\mathcal{F} = H_v^{\alpha} \times D_v^{\beta}$) that captures intuitively appealing features of a good placement. Specifically, the score favours placing the readers at points expressing a high "volume" (a power of H_v) $\times$ "distance" (a power of D_v) product. The volume (see Figure 5.3 (a) for example) is captured by the sum of the weights of the edges that are attached to v, where an edge weight represents the traversal frequency of that edge and is calculated as the

cumulative heat within a certain proximity with all spatial points on that edge. The distance (see Figure 5.3 (b) for example) is the sum of distances to already selected placement locations (given that the scheme is iterative). The distance metric attempts to avoid placing the readers near each other as they would then cover shorter trajectory segments than they would have covered if placed further apart.



(a) Volume factor - between points V_3 and V_5 which ever node has the most cumulative heat score underneath the red and green shades, respectively, will be favoured by the Volume factor.



(b) Distance factor - between points V_2, V_3, V_5 and V_6 the Distance factor favours V_2 since it is most distant to the already chosen V_4 point

Figure 5.3: Distance and Volume Factors

In the score function, parameters α and β are user specified parameters showing preference between the volume and distance factors. Larger values of α in comparison to β puts emphasis on the volume factor rather than the distance factor, and vice versa. We conducted simulation experiments to determine the right mix of α and β exponents, which we report in the simulations section.

If h is the number of RFID readers available for placement, our process iterates h times, placing one reader at a time on the basis of the highest value of $\mathcal{F}$. For the first location to be selected, the score function is considered to only be H_v.

A technicality of the solution derived is that it selects points that can be "unnatural" for mounting an RFID reader; i.e. far from a power source or not on a wall. Hence a post-processing step is used to satisfy such practical constraints. Namely, the mounting point for each RFID reader is a point in W that is closest to the point chosen by the score-based heuristic (considering normal orientation against the wall). Based on the distance between the mounting point and the chosen location, the reader's minimum transmission power may have to be adjusted to ensure proper coverage.

5.3 RFID Reader Placement Evaluation

We produced a number of test cases simulating different numbers of occupants moving in the Smart-Condo™ . Different activities in the environment can be taken into account as daily routines. Using the oven, going to the bathroom, sleeping on the bed etc. are examples of daily routines that can be performed by any of the occupants involving movement to particular AoIs. In total, we considered 14 such activities, and we simulated the movement of the occupants assuming that each occupant randomly chooses three activities out of the 14 possible. Upon reaching a destination the occupant pauses by an average time of 10 seconds and then continues moving towards the next destination. On average, for each occupant to complete their tasks, a time span of approximately 45 seconds is considered. It is note worthy that the initial location of the occupants are completely random and there may be cases where multiple occupants start from the same starting point. Although determining a fixed destination (*i.e.,* the entrance door) as every occupant's starting point can

be accommodated in our system (a realistic scenario where all occupants enter the house at the same time and then the rest of their activities are pursued), we decided to look at events happening during a specific period of time while all occupants are present and active in the environment. Also, we do not have any control over the number of collisions that may occur, and this too, is random in nature.

Within the simulated Smart-Condo™ we conducted experiments for $k = 2$, $k = 3$ and $k = 5$ occupants. First, the motion sensors were placed according to [1], which provided an optimal solution consisting of 11 PIR sensors. Using PIR sensors alone results in ambiguous trajectories when used for more than one inhabitant. Over all the generated test cases, 20%, 3%, 0% and 0% included no collisions when, respectively, 2, 3, 4 and 5 occupants were simulated. The small fraction of collision free tracks for 2 (and to a lesser extent for 3) individuals is because in small enough groups, even though the paths of the individuals intersect, they are not necessarily at the same point at the same time (what we term a "collision") hence the problem is less complicated, but this fraction quickly diminishes as k increases.

Next, and in order to evaluate our RFID reader-placement method, we compare it against an "expert" manual placement and a random one. The manual placement could be the product of someone with basic knowledge on how RFID sensors work and their role in location inference, but designed "naively" aiming to cover (to the degree possible) the entire space with readers. With a limited budget of RFID readers, the manual placement will first try to place at least one reader in every room (although the point of attachment can be random as long as it's facing "inside" the area it covers). The placement for the manual placement is depicted in Figure 5.4. For the randomized placement for the number of RFID readers in budget, points within with a non-zero utility score in the heat-map were randomly chosen. Then, the closest wall to each point is found and the RFID reader's mounting point is determined (similar to finding mounting points for the tree-based method). We simulated and report on the average of 10 randomized placements.

We compare all three methods in terms of their tracking and ambiguity errors. We set $\tau_{hm} = 2$, $\alpha = 1$, $\beta = 2$, and $r = 100cm$ ($\Delta r = 1cm$). In the first set of results the number of RFID readers is set to 5. The results are averages of 100 simulation runs for each

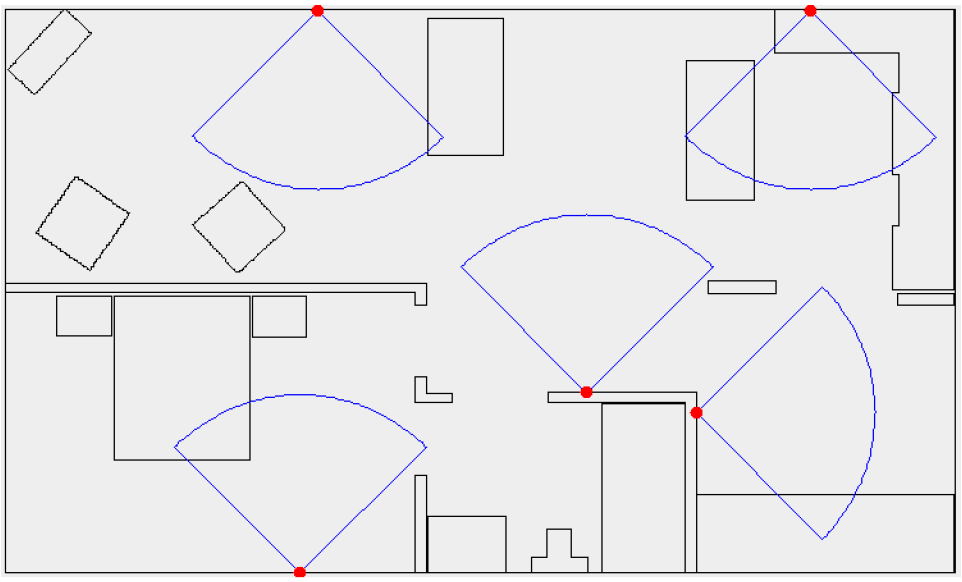


Figure 5.4: Manual placement for 5 RFID readers.

reader placement. Table 5.1 compares the random, manual and tree-based methods. In each subtable, $k-1$ rows show the members of set A^* (calculated according to Equation 3.8), and two rows present the tracking error's upper-bound (TEU) and lower-bound (TEL) (described in Section 3.1.1) for each tested method. In all cases the tree-based method surpasses the other two, exhibiting smaller tracking error and smaller ambiguity. Furthermore, the relative merits of the tree-based method become increasingly apparent as k increases. Admittedly, the tracking error deteriorates fast with the number of occupants: from a tight range of about 60cm for two occupants, the range widens to between 70cm and 75cm for three occupants, and it expands to an upper bound of more than 1.69m in the case of five occupants. Clearly substantial work is needed to design placement that suffer less deterioration as the number of occupants increases.

Finally, Table 5.2 shows that increasing the number of RFID readers, despite its ability to monotonically reduce TEU and TEL, results in diminishing returns. Additionally, given the total coverage possible with more readers, the error of the manual and random placements is also reduced as h increases. The manual process produces the best results for 3 readers based on visual inspection of the floorplan/heat-map and based on an understanding of how the spaces involved are used; however, it is difficult to generalize its good performance to larger h because it is challenged by the numerous deployment possibilities

that arise then.

Table 5.1: Comparison of RFID-placement methods ($h = 5$)

		Tree-based	**Manual**	**Random**
k=2	$a_2^*(\%)$	2.959	3.127	3.302
	TEU(m)	0.610	0.611	0.614
	TEL(m)	0.608	0.608	0.608
k=3	$a_2^*(\%)$	7.224	7.540	8.155
	$a_3^*(\%)$	0.379	0.681	0.836
	TEU(m)	0.755	0.780	0.825
	TEL(m)	0.702	0.706	0.715
k=5	a_2^* (%)	9.144	9.382	10.015
	a_3^* (%)	5.048	6.234	9.860
	a_4^* (%)	3.206	3.039	3.599
	a_5^* (%)	1.186	1.516	1.643
	TEU(m)	1.691	1.769	2.202
	TEL(m)	0.847	0.882	0.920

Table 5.2: Impact of different number of readers (h) in the presence of 3 people.

h	**Tree-Based**		**Manual**		**Random**	
	TEU(m)	TEL(m)	TEU(m)	TEL(m)	TEU(m)	TEL(m)
3	1.098	0.763	0.865	0.720	1.228	0.780
4	0.795	0.709	0.818	0.712	0.922	0.728
5	0.755	0.702	0.780	0.706	0.825	0.715
6	0.732	0.698	0.732	0.698	0.769	0.706
7	0.731	0.698	0.697	0.695	0.734	0.699

5.4 Tagging and Marking

Depending on the user's requirements, tracking information for one person (usually the patient) or all individuals in the environment may be needed. In the former case we say that the patient is *marked*, and in the latter everybody is marked. Based on who is marked

and who is tagged we will consider four interesting scenarios that are most likely to happen in an assistive ambient living setting.

All individuals are tagged and the patient is marked: In this case, everybody in the smart environment is wearing their tag and tracking information for the patient is desired. Here, the RFID reading from all tags contribute in resolving ambiguity and reducing tracking error.

The patient is the only person un-tagged and marked: This case may happen when the patient is reluctant to wear his/her tag (because of personal reasons). In order to achieve a decent amount of tracking information about the patient, care providers (or other individuals present in the environment except the patient) are asked to ensure they carry their own tag.

The patient is the only person tagged and marked: In this scenario, for practical reasons people coming in and going out for the smart space may not be able to wear tags. This means that only the patient's tag reading are used for tracking purposes.

No one tagged or marked: For comparison, we also consider a case where no information is provided from the RFID infrastructure. In the event of a system failure (*i.e.,* tags not communicating with the readers) an estimate of the best case and worst case tracking errors are reported. In this case it is trivial that the ambiguity for all paths at any time remains equal to the maximum value possible (which is k).

In the experiment regarding different tagging scenarios we utilize a similar setting but fix the number of RFID readers to 5.

Table 5.3: Comparison of different tagging scenarios.

k	**No one**		**Everyone**		**Only Patient**		**All but Patient**	
	TEU(m)	TEL(m)	TEU(m)	TEL(m)	TEU(m)	TEL(m)	TEU(m)	TEL(m)
2	3.650	0.579	0.580	0.579	0.624	0.579	0.619	0.579
3	4.809	1.217	0.769	0.710	1.882	0.846	2.236	0.938
5	5.737	1.217	2.226	0.932	5.230	1.171	4.534	1.213

It is noteworthy that in Table 5.3 the only marked person is the patient since most of the time only the patient's tracking information in needed. As expected, the case where all occupants are wearing their tags results in much less error than any other scenario. On the contrary, if the occupants do not wear their tags the amount of error will increase rapidly to the point that localization is useless. The result for cases where only the patient is wearing their tag, and when all but the patient are tagged are close to each other for the number of occupants we have chosen here, but we suspect as this number increases, tracking error would be more in the former case. This is because the number of collisions with un-tagged people increases rapidly in the former causes increased confusion, whereas in the latter, the abundance of useful information coming from the tagged $k-1$ occupants helps reduce the tracking error.

5.5 Effect of Realistic RFID Coverage Models

Although the coverage model presented in Equation 3.6, is a standard coverage model for RFIDs used in the literature, in reality this shape is quite different. Practically, when using and RFID reader, coverage patterns similar to Figure 5.5 is expected. Because of this dissimilarity in shape, false negatives (presence in the ideal, *i.e.,* excepted, coverage region and not being detected due to a non-ideal coverage pattern) and false positives (presence outside of the ideal coverage area and unexpectedly begin detected), may occur. We would like to investigate how non-ideal coverages can affect the localization and ambiguity error.

In order to accommodate this in our simulations we will replace the Boolean Sector model presented in Section 3.4 with a model that has a *main lobe*, two *side lobes* and a *back lobe*, ensuring that the overall area of the two models are equal. In the form of an experiment we study the difference in TEL and TEU when this replacement is done. The realistic coverage model is depicted in Figure 5.6.

In our experimental setup we use the discussed tree-based method and set $k = 2$ and $h = 5$. For the Realistic model the r is set to the same radius as the Boolean Sector, and $\omega = 40^{\circ}$.

According to Table 5.4, since the Realistic model has lost some coverage area in the

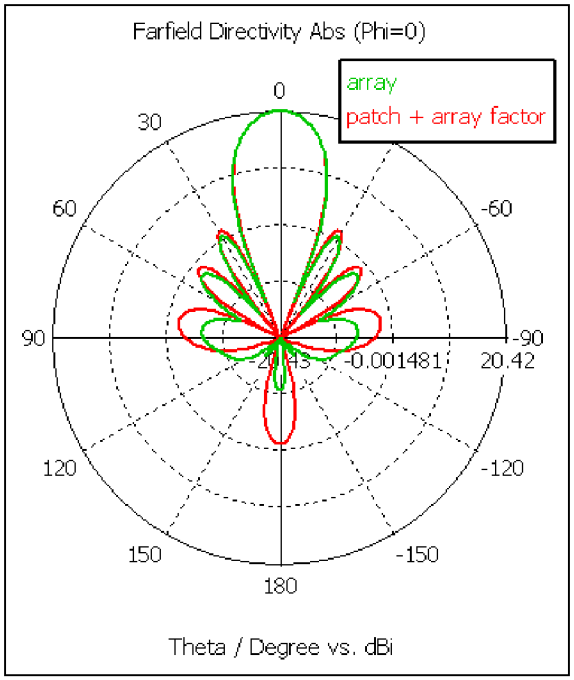


Figure 5.5: Realistic RFID coverage model. [2]

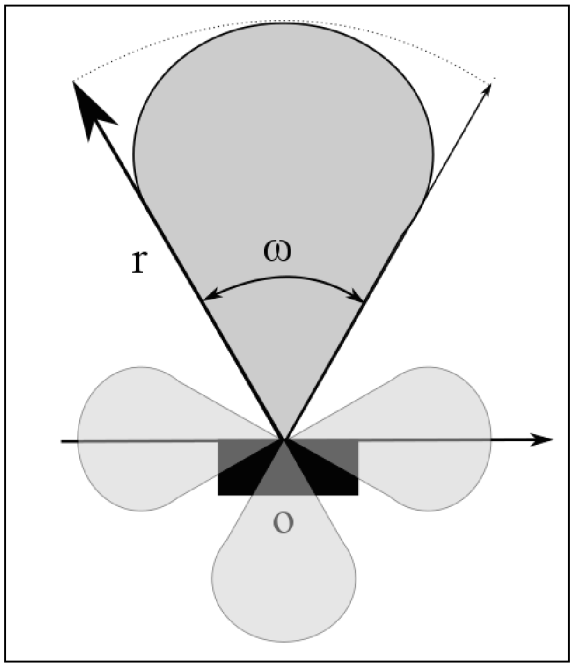


Figure 5.6: The Realistic model with main and side lobes.

Table 5.4: Comparing the Boolean Sector and Realistic coverage models

Model	$a_2^*(\%)$	TEU(m)	TLU(m)
Boolean Sector Model	3.002	0.580	0.579
Realistic Model	3.068	0.611	0.608

side and back lobes, the average localization error for this model has increased, but by little. The reason for this is because useful coverage which normally is expected in front of the RFID reader, is diminished, and is being replaced by areas that are less likely to cover a person and benefit the localization process (*i.e.,* occupants do not tend to walk too close to the walls). We expect that as the number of side-lobes increases and the area of the main lobe is reduced, the localization error with increase.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this thesis we described, simulated, and evaluated an indoor multiple-person tracking system. Using passive infra-red motion sensors and RFID based technology we were able to improve localization accuracy for a number of occupants in a smart environment.

In order to get the most out of the PIR sensor, we first discovered the best placement for a subset of the available sensors through an optimization process. Because of the vast searching space introduced by multiple shapes, and a numerous possible locations, we adopted a genetic algorithm. Compared to a sequential design with a similar problem formulation, we showed that the proposed method works more efficiently and considers minimal number of sensors used. After that, PIR sensors are utilized to infer the locations (and trajectories over time) where the occupants are likely to be, but the inevitable collision of two or more people does not allow proper disambiguation of these paths. For that, we have used RFID readers along with tags that are worn by the occupants indicating their unique ID. As soon as an occupant is identified by a reader, our knowledge of "who-is-where" is improved, and path labelling (indicating which resident is currently on which path) is performed.

We also, proposed a heuristic for RFID reader placement aiming to cover areas of the space that are frequently visited, central and therefore likely to be the locations of multiple individuals' colliding - resulting in ambiguous location inferences. Our simulations demonstrate that our RFID placement method favorably compares against random placements as

well naive manual placements aiming to simply cover the whole space. Furthermore, conducting simulations for scenarios with certain practical limitations reveals the flexibility of our method upon implementation.

Overall, experimental results suggest that our placement methodology and localization algorithm may eventually replace the time consuming and hard work of manually selecting locations for sensors (and selecting their appropriate type) in an indoor environment setting.

6.2 Future Work

In future work we will focus on more comprehensive disambiguation objectives. More specifically, we will emphasis on the coverage of AoIs; since these locations are where occupants spend most of their time.

An interesting direction for future could be to study scenarios where one or more sensor simply stop working. This failure may be due to battery outage, disconnection, chip malfunction, etc. and although very rare, but might happen. We would like to investigate how well our system performs under such circumstances, and whether it is possible to improve fault-tolerance by considering it as a factor in our placement optimization.

A known limitation of our system is that the deployment planning for PIR sensors and RFID readers are two consecutive steps. Ideally, our system should be able to optimize both placements at the same time. In that way, RFID reader footprints can be placed to cover high traffic areas (where they prove to be more efficient for disambiguation and localization purposes), allowing the PIR sensors to focus more on the blind spots outside of the daily routines, rather than overlapping redundantly.

In future we will further expand our experiments to include real-life trails over a short-period of time. We would like to compare our simulator localization error with that of the real network when a real patient is using the smart house. Apart from testing and comparing the localization error we may look into differences between the developed system and the real world: are the mobility models similar? Have we correctly simulated pause times? Are there any problems associated with mounting the sensors; do we have to consider them in future placement solutions?

Chapter 7

Glossary and Acronyms

Acronyms

DR Dead Reckoning. 9

EM Graph Environment Model Graph. 5, 10, 11, 21–23, 28, 72, 73

GA Genetic Algorithm. 8, 11, 12, 31–34, 36, 40–45

GGA Generalized Genetic Algorithm. 8

GPS Global Positioning System. 3

IMU Internal Measuring Unit. 9

OSP Optimized Sensor Placement. 7, 8

PIR Passive Infra-Red. 1–6, 8–11, 15, 18, 19, 22, 23, 30–32, 36, 45, 50, 58, 59

PSO Particle Swarm Optimization. 8

RFID Radio Frequency IDentification. 1–6, 8–12, 16–18, 22, 23, 28, 30, 45, 47, 49–55, 57–59, 62, 63

RSS Received Signal Strength. 9

SHM Structural Health Monitoring. 8

WSN Wireless Sensor Network. 1

Glossary

c_i Coverage utility value at point i. 15

c_{max} Maximum desired overage score. 15

δ Minimum distance to consider two occupants collided. 12

$d(o,z)$ Euclidean distance between RFID reader o and point z. 17

$g_{o \to z}$ Coverage function of RFID reader o at point z. 17

H_v Sum of the weights of the edges that are attached to v, where an edge weight represents the traversal frequency of that edge and is calculated as the cumulative heat within a certain proximity with all spatial points on that edge. 47

I Points in the heat-map that are not walls or permanent obstacles. 28, 30

k Number of occupants. 12

N Number of spacial points in the heat-map. 15, 30

O Points in the heat-map representing permanent obstacles. 28, 30

$\mathcal{P}$ The set containing all occupants. 12

Φ_0 Orientational angle of an RFID reader. 16, 17

Φ_z Orientational angle between the spacial point z and an RFID reader that is covering it. 17

$p_{door\,open}$ Probability of a door begin open. 16

P_o^Z Probability of detecting an individual at point o by the motion sensors in the set Z. 16

$p_{s \to o}$ Coverage function of motion sensor s at point o. 16

r Sensing range of an RFID reader. 16

T^{P_i} Trajectory of person P_i, which is a sequence of l_{P_i,t_m}. 12

v_i Heat-map value at point i. 15

W Points in the heat-map representing walls. 28, 30, 49

ω Angle of view of an RFID reader. 16, 17, 55

Z A set of motion sensors. 16

Bibliography

[1] I. Vlasenko, "Deployment planning for location recognition in the smart-condotm: Simulation, empirical studies and sensor placement optimization," Master's thesis, University of Alberta, 2013.

[2] Computer Simulation Technology, "Microstrip Patch Array Design," [Online; accessed 30-August-2015]. [Online]. Available: https://www.cst.com/Applications/Article/Microstrip-Patch-Array-Design

[3] N. M. Boers, D. Chodos, J. Huang, P. Gburzynski, I. Nikolaidis, and E. Stroulia, "The Smart Condo: Visualizing independent living environments in a virtual world," in *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*. IEEE, 2009, pp. 1–8.

[4] V. Ganev, D. Chodos, I. Nikolaidis, and E. Stroulia, "The Smart Condo: integrating sensor networks and virtual worlds," in *Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications*. ACM, 2011, pp. 49–54.

[5] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, "Global positioning system. theory and practice." *Global Positioning System. Theory and practice., by Hofmann-Wellenhof, B.; Lichtenegger, H.; Collins, J.. Springer, Wien (Austria), 1993, 347 p., ISBN 3-211-82477-4, Price DM 79.00. ISBN 0-387-82477-4 (USA).*, vol. 1, 1993.

[6] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621–655, 2008.

[7] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[8] S. S. Dhillon and K. Chakrabarty, *Sensor placement for effective coverage and surveillance in distributed sensor networks*. IEEE, 2003, vol. 3.

[9] P. David, V. Idasiak, and F. Kratz, "A sensor placement approach for the monitoring of indoor scenes," in *Smart Sensing and Context*. Springer, 2007, pp. 110–125.

[10] T.-H. Yi and H.-N. Li, "Methodology developments in sensor placement for health monitoring of civil infrastructures," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[11] A. R. M. Rao and G. Anandakumar, "Optimal placement of sensors for structural system identification and health monitoring using a hybrid swarm intelligence technique," *Smart materials and Structures*, vol. 16, no. 6, p. 2658, 2007.

[12] H. Chen, Y. Zhu, K. Hu, and T. Ku, "RFID network planning using a multi-swarm optimizer," *Journal of Network and Computer Applications*, vol. 34, no. 3, pp. 888–901, 2011.

[13] H. M. Elkamchouchi, H. M. Elragal, and M. A. Makar, "Cellular radio network planning using particle swarm optimization," in *Radio science conference, 2007. NRSC 2007. National*. IEEE, 2007, pp. 1–8.

[14] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 3, pp. 281–295, 2006.

[15] Q. Guan, Y. Liu, Y. Yang, and W. Yu, "Genetic approach for network planning in the RFID systems," in *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, vol. 2. IEEE, 2006, pp. 567–572.

[16] D. Jourdan and O. L. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," in *Vehicular technology conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 5. IEEE, 2004, pp. 2466–2470.

[17] F. Kang, J.-j. Li, and Q. Xu, "Virus coevolution partheno-genetic algorithms for optimal sensor placement," *Advanced Engineering Informatics*, vol. 22, no. 3, pp. 362–370, 2008.

[18] W. Y. Poe and J. B. Schmitt, "Placing multiple sinks in time-sensitive wireless sensor networks using a genetic algorithm," in *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference-*. VDE, 2008, pp. 1–15.

[19] T.-H. Yi, H.-N. Li, and M. Gu, "Optimal sensor placement for structural health monitoring based on multiple optimization strategies," *The Structural Design of Tall and Special Buildings*, vol. 20, no. 7, pp. 881–900, 2011.

[20] S. Han, H. Lim, and J. Lee, "An efficient localization scheme for a differential-driving mobile robot based on RFID system," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 6, pp. 3362–3369, 2007.

[21] S.-Y. Yeh, K.-H. Chang, C.-I. Wu, H.-H. Chu, and J. Y.-J. Hsu, "Geta sandals: a footstep location tracking system," *Personal and Ubiquitous Computing*, vol. 11, no. 6, pp. 451–463, 2007.

[22] S. House, S. Connell, I. Milligan, D. Austin, T. L. Hayes, and P. Chiang, "Indoor localization using pedestrian dead reckoning updated with RFID-based fiducials," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE, 2011, pp. 7598–7601.

[23] A. R. J. Ruiz, F. S. Granja, J. C. Prieto Honorato, and J. I. G. Rosas, "Accurate pedestrian indoor navigation by tightly coupling foot-mounted IMU and RFID measurements," *Instrumentation and Measurement, IEEE Transactions on*, vol. 61, no. 1, pp. 178–189, 2012.

[24] S. Willis and S. Helal, "A passive RFID information grid for location and proximity sensing for the blind user," *University of Florida Technical Report*, pp. 1–20, 2004.

[25] R.-C. Chen and Y.-H. Lin, "Apply Kalman filter to RFID received signal strength processing for indoor location," in *Awareness Science and Technology (iCAST), 2012 4th International Conference on*. IEEE, 2012, pp. 73–77.

[26] A. Bekkali, H. Sanson, and M. Matsumoto, "RFID indoor positioning based on probabilistic RFID map and Kalman filtering," in *Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on*. IEEE, 2007, pp. 21–21.

[27] C.-H. Huang, L.-H. Lee, C. C. Ho, L.-L. Wu, and Z.-H. Lai, "Real-time RFID indoor positioning system based on Kalman-filter drift removal and heron-bilateration location estimation," *Instrumentation and Measurement, IEEE Transactions on*, vol. 64, no. 3, pp. 728–739, 2015.

[28] S. Särkkä, V. V. Viikari, M. Huusko, and K. Jaakkola, "Phase-based UHF RFID tracking with nonlinear Kalman filtering and smoothing," *Sensors Journal, IEEE*, vol. 12, no. 5, pp. 904–910, 2012.

[29] T. Nick, J. Goetze, W. John, and G. Stoenner, "Comparison of extended and unscented Kalman filter for localization of passive UHF RFID labels," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*. IEEE, 2011, pp. 1–4.

[30] P. Yang, W. Wu, M. Moniri, and C. C. Chibelushi, "SLAM algorithm for 2d object trajectory tracking based on RFID passive tags," in *RFID, 2008 IEEE International Conference on*. IEEE, 2008, pp. 165–172.

[31] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 1015–1020.

[32] A. Milan, K. Schindler, and S. Roth, "Detection-and trajectory-level exclusion in multiple object tracking," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 3682–3689.

[33] C.-R. Yu, C.-L. Wu, C.-H. Lu, and L.-C. Fu, "Human localization via multi-cameras and floor sensors in smart home," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 5. IEEE, 2006, pp. 3822–3827.

[34] S. Knauth, A. Andrushevich, L. Kaufmann, R. Kistler, and A. Klapproth, "The iLoc+ ultrasound indoor localization system for aal applications at evaal 2012," in *Evaluating AAL Systems Through Competitive Benchmarking*. Springer, 2013, pp. 83–94.

[35] N. Fet, M. Handte, S. Wagner, and P. J. Marrón, "Locosmotion: An acceleration-assisted person tracking system based on wireless lan," in *Evaluating AAL Systems Through Competitive Benchmarking*. Springer, 2013, pp. 17–31.

[36] T. Ikeda, H. Ishiguro, and T. Nishimura, "People tracking by fusing different kinds of sensors, floor sensors and acceleration sensors," in *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*. IEEE, 2006, pp. 530–535.

[37] Panasonic Electric Works, "Mp motion sensor NaPIOn, datasheet," [Online; accessed 30-August-2015]. [Online]. Available: http://www3.panasonic.biz/ac/e_download/control/sensor/human/catalog/bltn_eng_pir.pdf

[38] Hobbell Motion Sensing Switches, "Sensors for an Energy Conscious World," [Online; accessed 30-August-2015]. [Online]. Available: http://ecatalog.hubbell-wiring.com/press/pdfs/WLBHM002.pdf

[39] Q. Wang, K. Xu, G. Takahara, and H. Hassanein, "Wsn04-1: deployment for information oriented sensing coverage in wireless sensor networks," in *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*. IEEE, 2006, pp. 1–5.

[40] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 32, 2011.

[41] Y. Ivanov, A. Sorokin, C. Wren, and I. Kaur, "Tracking people in mixed modality systems," in *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007, pp. 65 080L–65 080L.

[42] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[43] S. K. Smit and A. E. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 399–406.

[44] L. Shapiro and R. Haralick, "Computer and robot vision," *Reading: Addison-Wesley*, vol. 8, 1992.

[45] T. Y. Kong and A. Rosenfeld, *Topological algorithms for digital image processing*. Elsevier, 1996.

[46] L. Lam, S.-W. Lee, and C. Y. Suen, "Thinning methodologies-a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 9, pp. 869–885, 1992.

[47] W. K. Pratt, "Image segmentation," *Digital Image Processing: PIKS Inside, Third Edition*, pp. 551–587, 1991.

Appendix A

Extended Example

In this section we will explain the details of the example brought in Section 3.5.1. As the two occupants in this example move in the environment they are detected by the PIR sensors and sensor events are generated. During system runtime a sequence of these sensor events are produced, and as a post-processing step the extraction algorithm (Algorithm 1) attempts to derive exactly $|\mathcal{P}| = 2$ trajectories for the two present individuals. Let us go through each timestamp in the system and present important parameters: (we consider one-based indexing for the sets)

At time t_0 (see Figure A.1):
At the beginning of the process the known locations of each occupant is added to their trajectory: $T_0^1 = loc_{P_1,t_0}$ and $T_0^2 = loc_{P_2,t_0}$. Where loc contains the ground truth locations.

At time t_1 (see Figure A.2):
sensor event: $SE_1 = 100100000$
candidate sensor events: $\mathcal{F}_{1,1} = \{000000000, 100000000\}$, $\mathcal{F}_{1,2} = \{000000000, 000100000\}$
logical OR of the Cartesian product: $C_1 = \{000000000, 000100000, 100000000, 100100000\}$
matching combination: $C_{1,4}$, $subScripts : \{2,2\}$
point added to trajectory: $T_1^1 = centerOfMass(100000000)$,
$T_1^2 = centerOfMass(000100000)$. The $centerOfMass$ function returns the central coordinates of the smallest intersecting polygon driven from the sensor event.

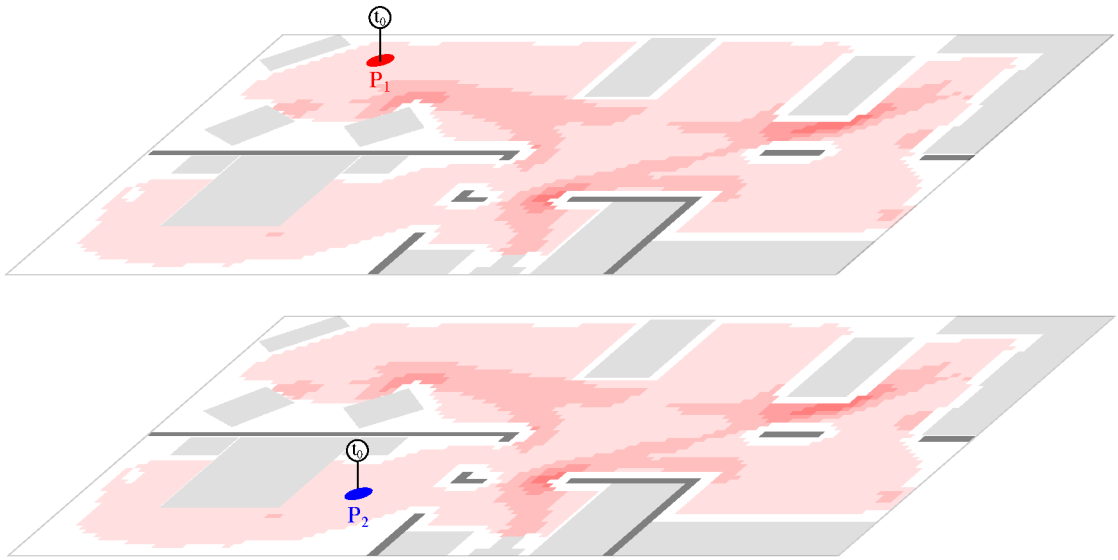


Figure A.1: Two occupants at time t_0.

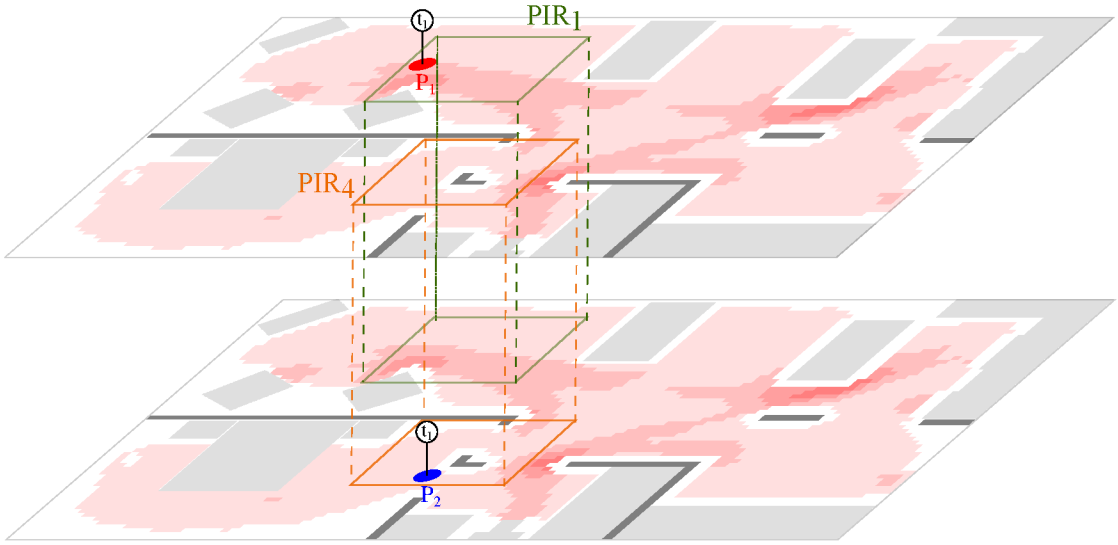


Figure A.2: Two occupants at time t_1.

At time t_2 (see Figure A.3):

sensor event: $SE_2 = 100100000$

candidate sensor events: $\mathcal{F}_{2,1} = \{100000000\}$, $\mathcal{F}_{2,2} = \{000000000, 000100000\}$

logical OR of the Cartesian product: $C_2 = \{100000000, 100100000\}$

matching combination: $C_{2,2}$, $subScripts : \{1,2\}$

point added to trajectory: $T_2^1 = centerOfMass(100000000)$,

$T_2^2 = centerOfMass(000100000)$.

At time t_3 (see Figure A.4):

sensor event: $SE_3 = 110100000$

candidate sensor events: $\mathcal{F}_{3,1} = \{110000000\}$, $\mathcal{F}_{3,2} = \{000100000\}$

logical OR of the Cartesian product: $C_3 = \{110100000\}$

matching combination: $C_{3,1}$, $subScripts : \{1,1\}$

point added to trajectory: $T_3^1 = centerOfMass(110000000)$,

$T_3^2 = centerOfMass(000100000)$.

At time t_4 (see Figure A.5):

sensor event: $SE_4 = 011100000$

candidate sensor events: $\mathcal{F}_{4,1} = \{010000000, 110000000\}$,

$\mathcal{F}_{4,2} = \{001000000, 001100000\}$

logical OR of the Cartesian product: $C_4 = \{011000000, 011100000, 111000000, 111100000\}$

matching combination: $C_{4,2}$, $subScripts : \{1,2\}$

point added to trajectory: $T_4^1 = centerOfMass(010000000)$,

$T_4^2 = centerOfMass(001100000)$.

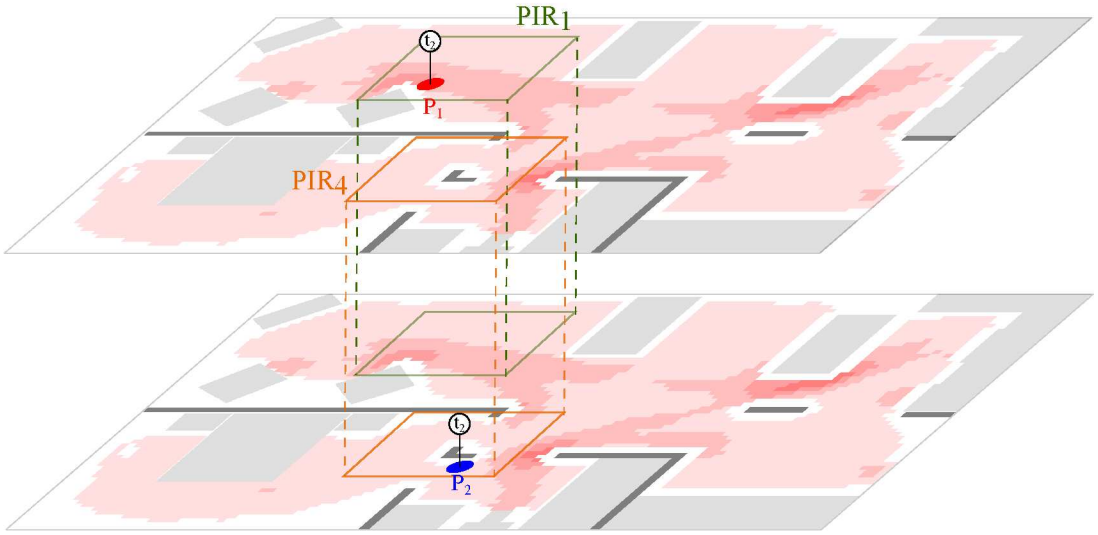


Figure A.3: Two occupants at time t_2.

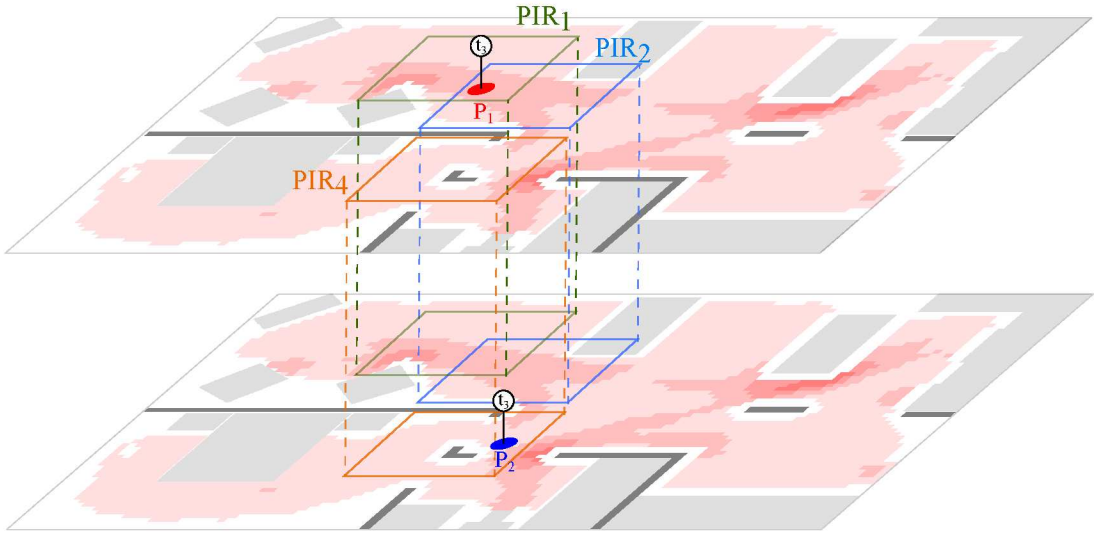


Figure A.4: Two occupants at time t_3.

At time t_5 (see Figure A.6):

sensor event: $SE_5 = 011000000$

candidate sensor events: $\mathcal{F}_{5,1} = \{011000000\}$,

$\mathcal{F}_{5,2} = \{001000000, 011000000\}$

logical OR of the Cartesian product: $C_5 = \{011000000, 011000000\}$

matching combination: $C_{5,1}$, $subScripts : \{1,1\}$

point added to trajectory: $T_5^1 = centerOfMass(011000000)$,

$T_5^2 = centerOfMass(001000000)$.

At time t_6 (see Figure A.7):

sensor event: $SE_6 = 001010000$

candidate sensor events: $\mathcal{F}_{6,1} = \{001000000\}$,

$\mathcal{F}_{6,2} = \{001000000, 001010000\}$

logical OR of the Cartesian product: $C_6 = \{001000000, 001010000\}$

matching combination: $C_{6,2}$, $subScripts : \{1,2\}$

point added to trajectory: $T_6^1 = centerOfMass(001000000)$,

$T_6^2 = centerOfMass(001010000)$.

At time t_7 (see Figure A.8):

sensor event: $SE_7 = 001010000$

candidate sensor events: $\mathcal{F}_{7,1} = \{001010000\}$,

$\mathcal{F}_{7,2} = \{000010000, 001010000\}$

logical OR of the Cartesian product: $C_7 = \{001010000, 001010000\}$

matching combination: $C_{7,1}$, $subScripts : \{1,1\}$

point added to trajectory: $T_7^1 = centerOfMass(001010000)$,

$T_7^2 = centerOfMass(001010000)$.

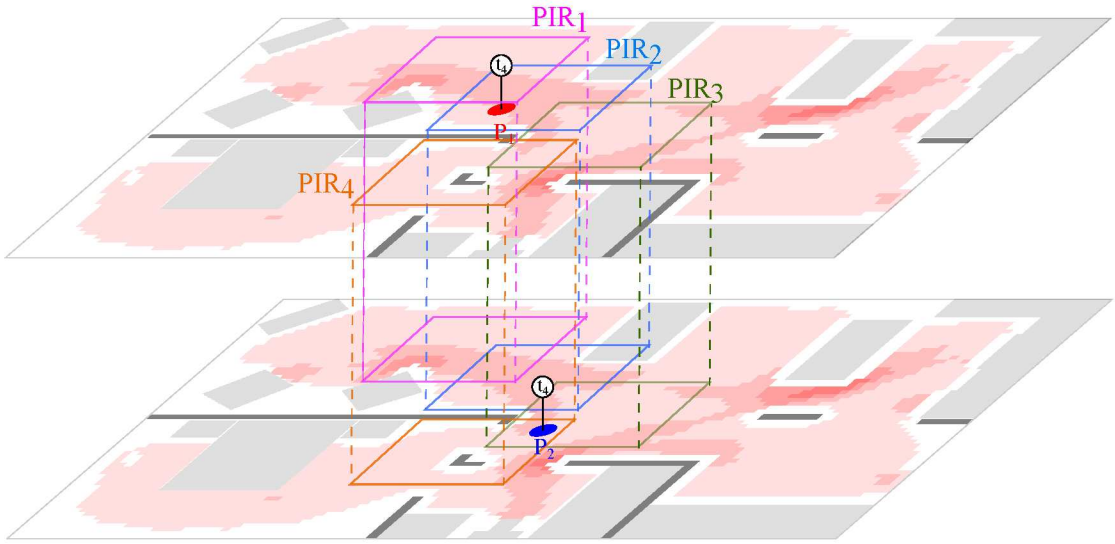


Figure A.5: Two occupants at time t_4.

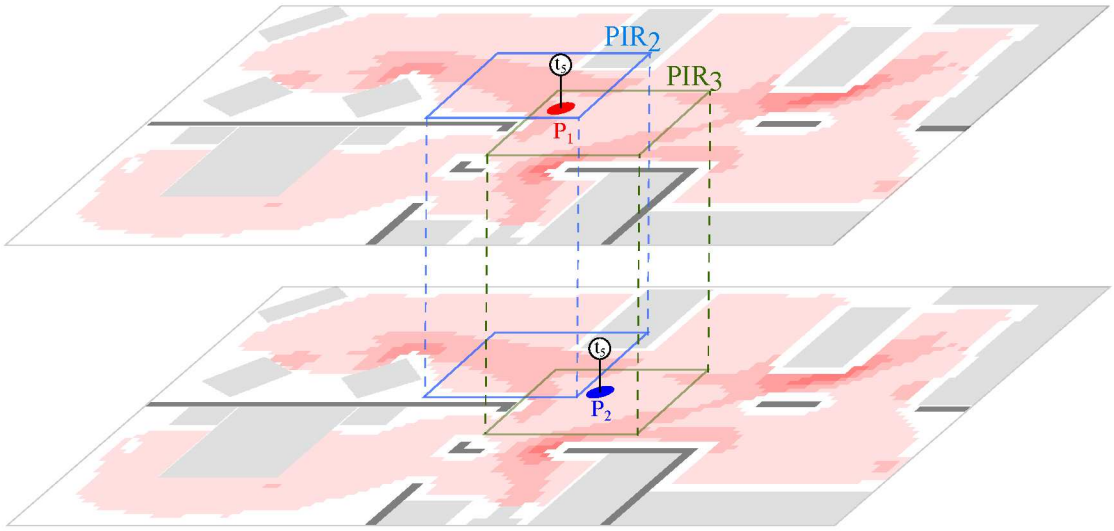


Figure A.6: Two occupants at time t_5.

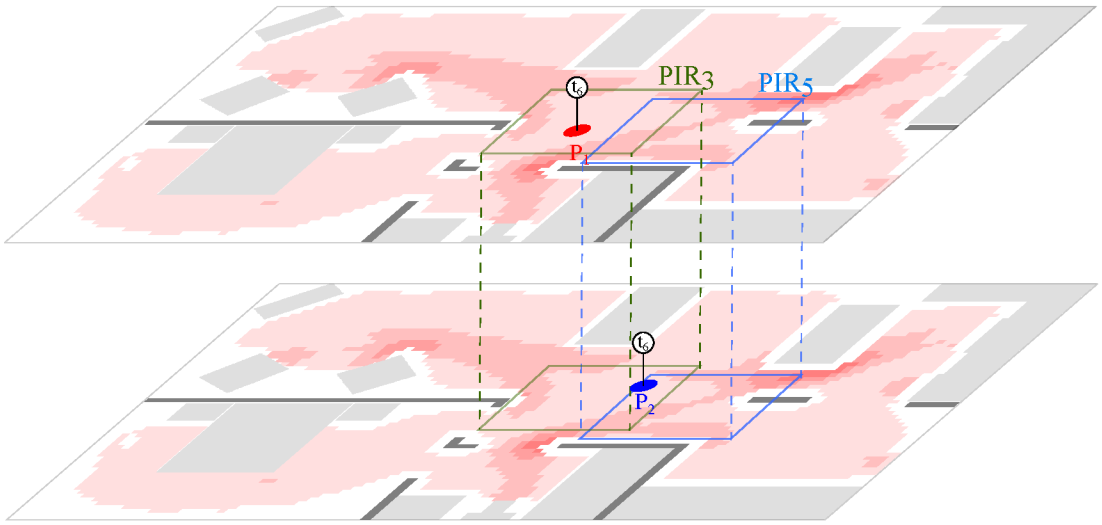


Figure A.7: Two occupants at time t_6.

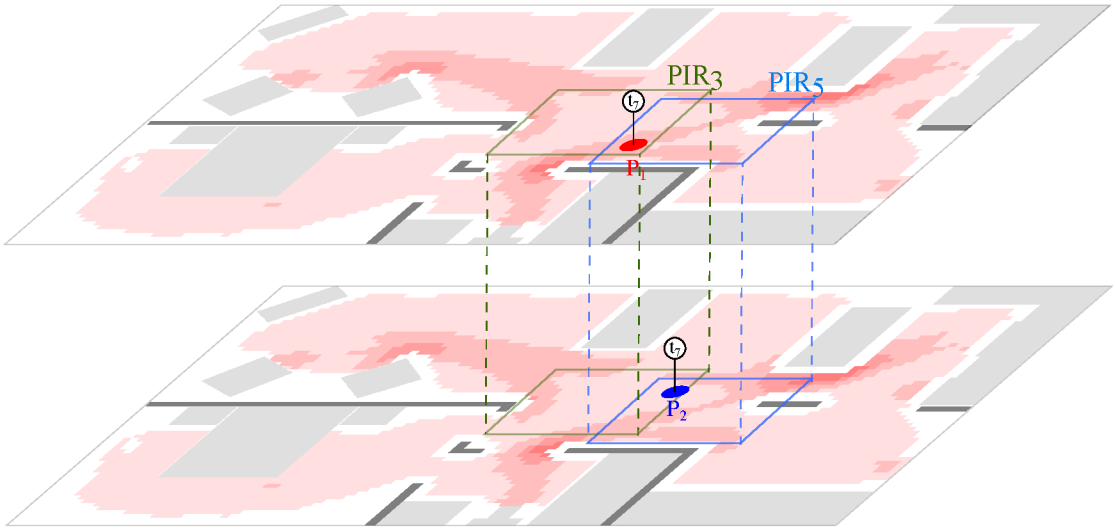


Figure A.8: Two occupants at time t_7.

At time t_8 (see Figure A.9):

sensor event: $SE_8 = 000010100$

candidate sensor events: $\mathcal{F}_{8,1} = \{000010000, 000010100\}$,

$\mathcal{F}_{8,2} = \{000010000\}$

logical OR of the Cartesian product: $C_8 = \{000010000, 000010100\}$

matching combination: $C_{8,2}$, $subScripts : \{2,1\}$

point added to trajectory: $T_8^1 = centerOfMass(000010100)$,

$T_8^2 = centerOfMass(000010000)$.

At time t_9 (see Figure A.10):

sensor event: $SE_9 = 000001100$

candidate sensor events: $\mathcal{F}_{9,1} = \{000000100\}$,

$\mathcal{F}_{9,2} = \{000001000, 000011000\}$

logical OR of the Cartesian product: $C_9 = \{000001100, 000011100\}$

matching combination: $C_{9,1}$, $subScripts : \{1,1\}$

point added to trajectory: $T_9^1 = centerOfMass(000000100)$,

$T_9^2 = centerOfMass(000001000)$.

At time t_{10} (see Figure A.11):

sensor event: $SE_{10} = 000001110$

candidate sensor events: $\mathcal{F}_{10,1} = \{000000010, 000000110\}$,

$\mathcal{F}_{10,2} = \{000001000\}$

logical OR of the Cartesian product: $C_{10} = \{000001010, 000001110\}$

matching combination: $C_{10,2}$, $subScripts : \{2,1\}$

point added to trajectory: $T_{10}^1 = centerOfMass(000000110)$,

$T_{10}^2 = centerOfMass(000001000)$.

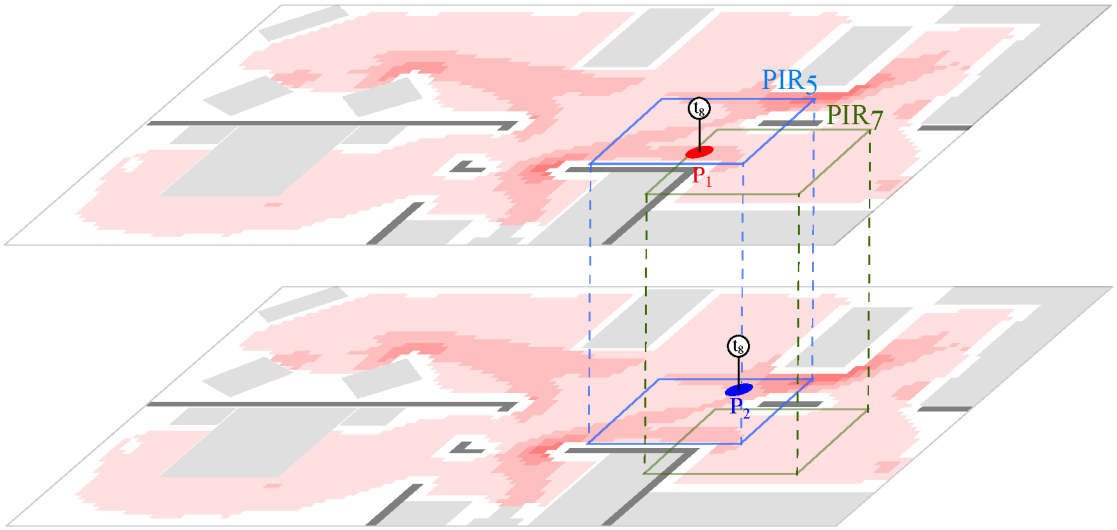


Figure A.9: Two occupants at time t_8.

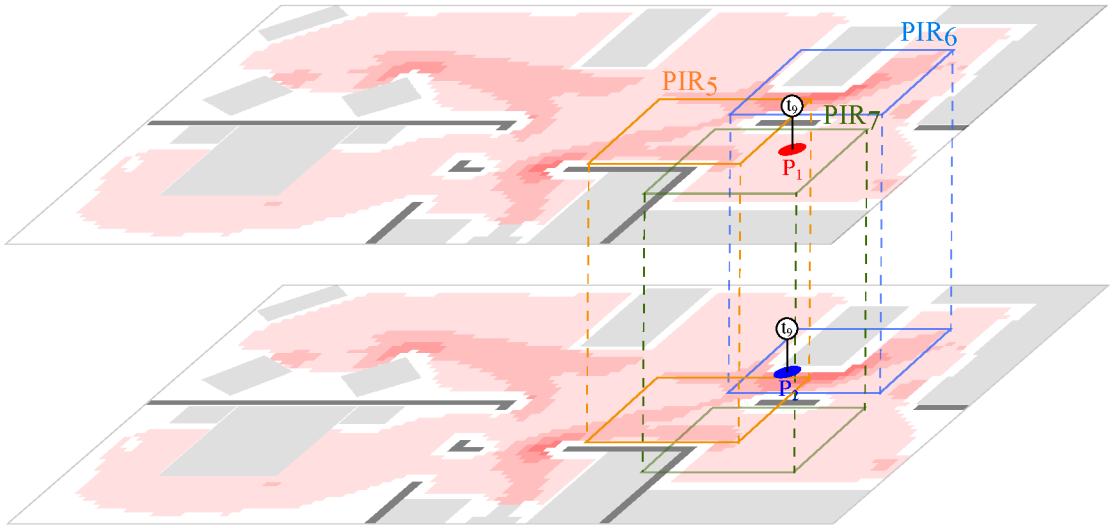


Figure A.10: Two occupants at time t_9.

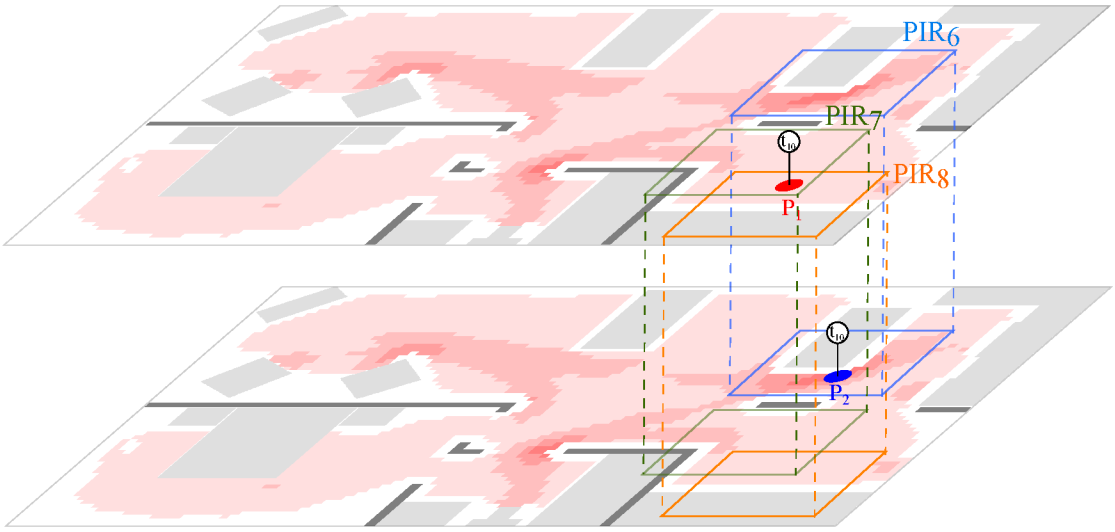


Figure A.11: Two occupants at time t_{10}.

At time t_{11} (see Figure A.12):

sensor event: $SE_{11} = 000001011$

candidate sensor events: $\mathcal{F}_{11,1} = \{000000010\}$,

$\mathcal{F}_{11,2} = \{000001001\}$

logical OR of the Cartesian product: $C_{11} = \{000001011\}$

matching combination: $C_{11,1}$, $subScripts : \{1,1\}$

point added to trajectory: $T^1_{11} = centerOfMass(000000010)$,

$T^2_{11} = centerOfMass(000001001)$.

After all 11 time stamps the trajectories for the two occupants would be as follows. Here, $c_{\{PIR_1,PIR_2,...,PIR_q\}}$ means the center of the intersection between all PIR sensors from PIR_1 to PIR_q. Also, T^1 represents "a trajectory" and does not necessarily mean that the trajectory belongs to individual number one. (likewise for T^2)

$T^1 : \{c_{\{PIR_1\}}, c_{\{PIR_1\}}, c_{\{PIR_1,PIR_2\}}, c_{\{PIR_2\}}, c_{\{PIR_2,PIR_3\}}, c_{\{PIR_3\}}, c_{\{PIR_3,PIR_5\}}, c_{\{PIR_5,PIR_7\}}, c_{\{PIR_7\}}, c_{\{PIR_7,PIR_8\}}, c_{\{PIR_8\}}\}$

$T^2 : \{c_{\{PIR_4\}}, c_{\{PIR_4\}}, c_{\{PIR_4\}}, c_{\{PIR_3,PIR_4\}}, c_{\{PIR_3\}}, c_{\{PIR_3,PIR_5\}}, c_{\{PIR_3,PIR_5\}}, c_{\{PIR_5\}}, c_{\{PIR_6\}}, c_{\{PIR_6\}}, c_{\{PIR_6,PIR_9\}}\}$

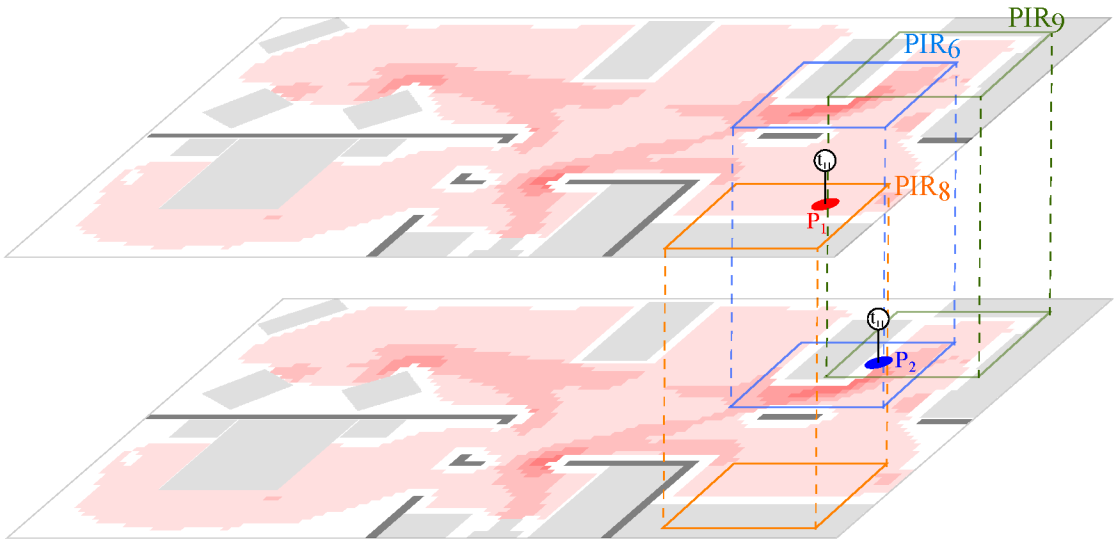


Figure A.12: Two occupants at time t_{11}.

After this step a smoothing process is performed to make trajectories more realistic. In this process for all subsequent points in the trajectory that have the same coordinates, we stretch the points along the trajectory. For example in T^2, the trajectory points T_1^2, T_2^2 and T_3^2 are the same. For smoothing, we replace T_1^2, T_3^2 with $\frac{T_0^2+c_{\{PIR_4\}}}{2}$ and $\frac{c_{\{PIR_3,PIR_4\}}+c_{\{PIR_4\}}}{2}$. Figure A.13 shows the trajectory points (illustrated as black squares) resulted from the extraction algorithm, and Figure A.14 shows the trajectories after smoothing.

The estimated and smoothed trajectories are considered to create the EM Graph. Having the starting points of each occupant, the corresponding nodes are added to an empty graph (white nodes). As we have two occupants in our example, two starting nodes are made at locations loc_{P_1,t_0} and loc_{P_2,t_0}. Next, we follow the estimated trajectories with $|\mathcal{P}|$ different pointers, marking collision points (and building a grey collision node in the graph) where occupants have collided, and making an identification node (in green) where a person has been detected by an RFID reader. In our example, the two occupants collide at t_5 and remain collided until t_8. Although we have collision at t_6 and t_7, but because the occupants remain collided at these timestamps, we do not need to create new collision nodes. Also in our example P_1 is identified by an RFID reader so an identification node is created as well.

A new edge is created between each newly created node and the last pointer positions,

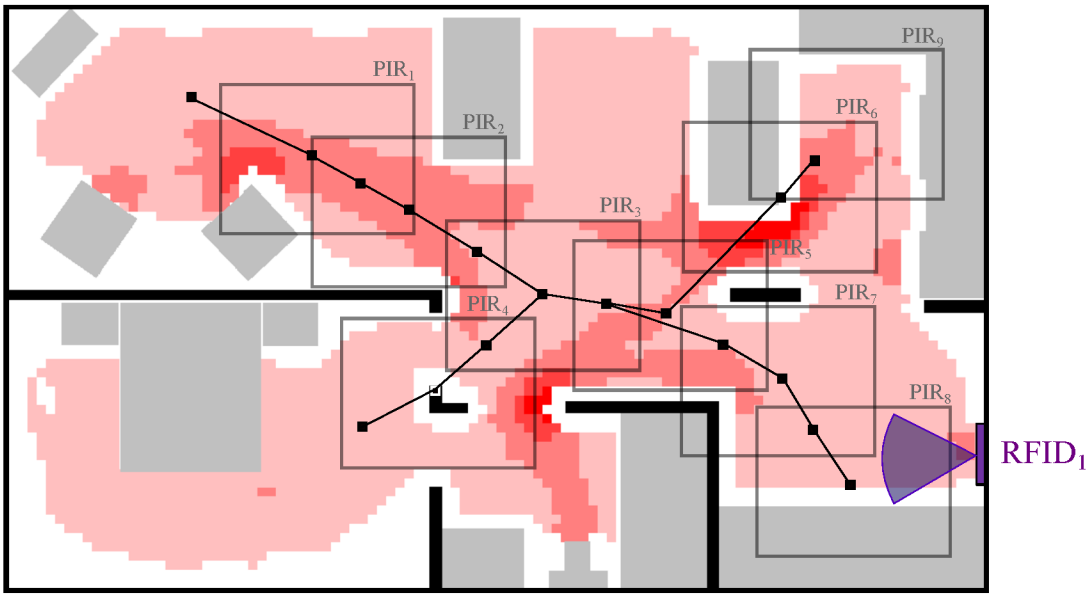


Figure A.13: Non-smoothed trajectory points.

and the trajectory points are stored in the edge as its "tracklet". In the example, T_0^1 to T_4^1 becomes the tracklet for e_1 (refer to Figure A.19 for edges ID numbers), T_0^2 to T_4^2 becomes the tracklet for e_2, T_5^1 to T_8^1 and T_5^2 to T_8^2 are two different tracklets stored in e_3, T_9^1 to T_{11}^1 becomes the tracklet for e_4, and finally T_9^2 to T_{11}^2 is assigned as the tracklet for e_5.

After building the entire EM Graph, the occupant set of all edges is set to all occupant IDs, and the set reduction process is performed. In the EM Graph built for this example (see Figure A.19) the only reductions are e_4 ($X_{e_4} = \{2\}$) and e_5 ($X_{e_5} = \{1\}$) because of the RFID reading at t_{11}.

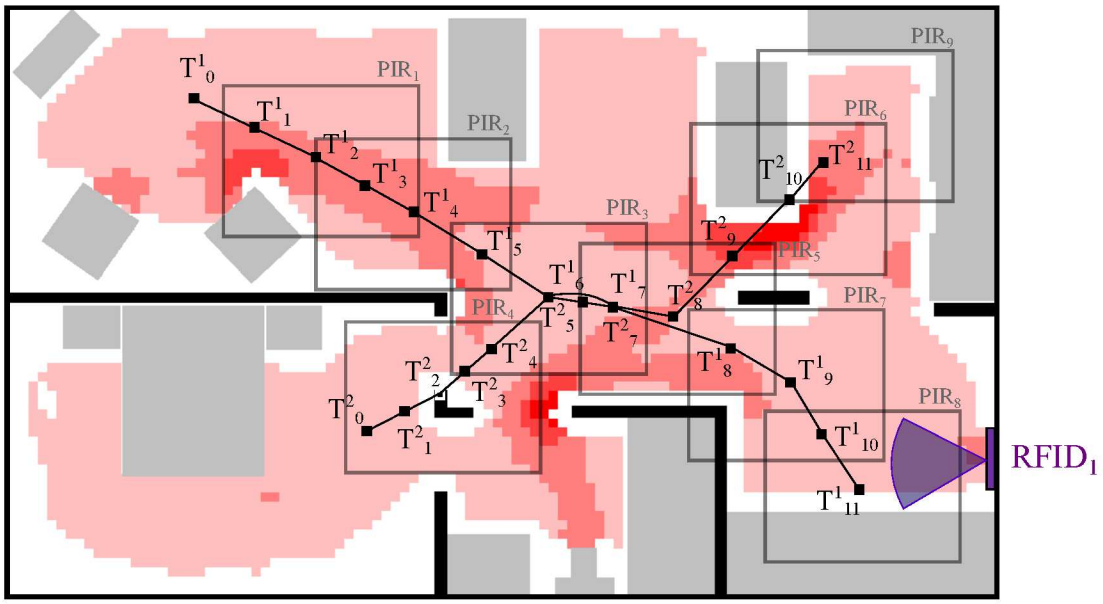


Figure A.14: Smoothed trajectory points.

Figure A.15: Produced EM Graph for the two occupants up to t_0.

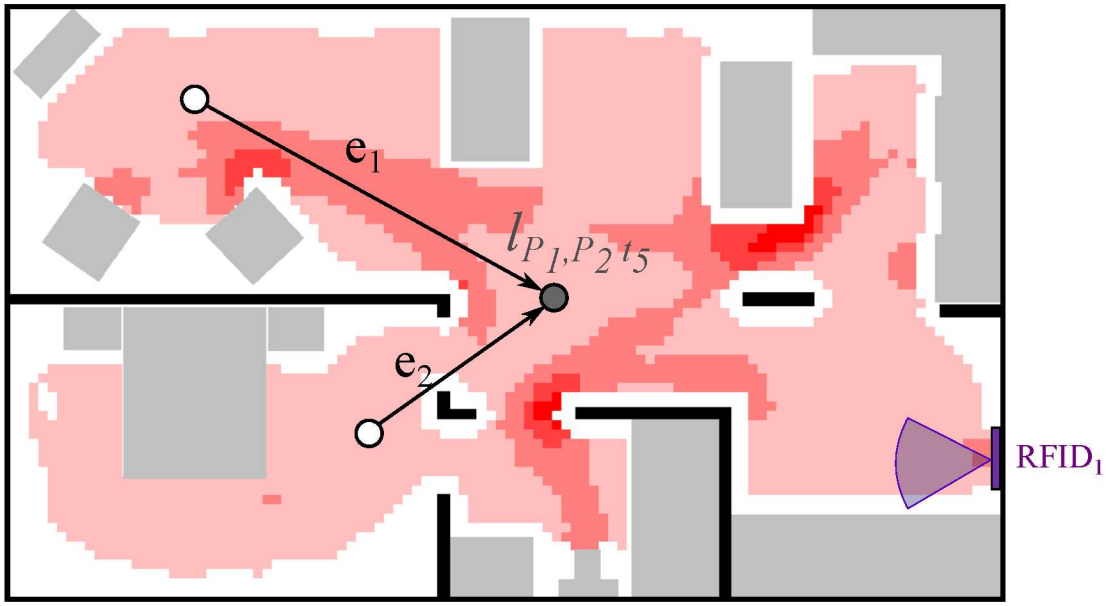


Figure A.16: Produced EM Graph for the two occupants up to t_5.

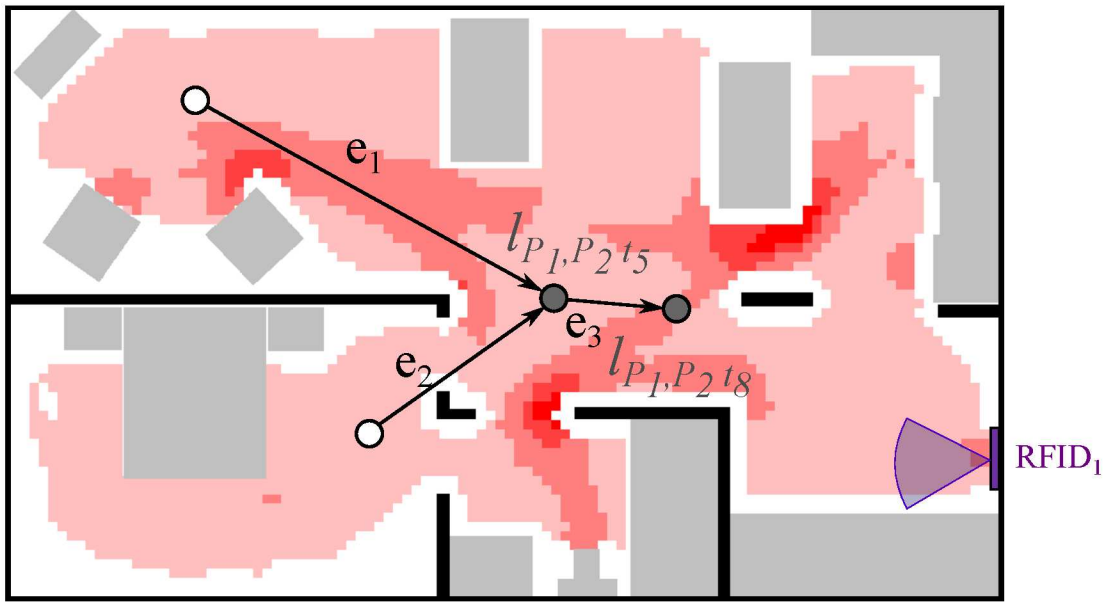


Figure A.17: Produced EM Graph for the two occupants up to t_8.

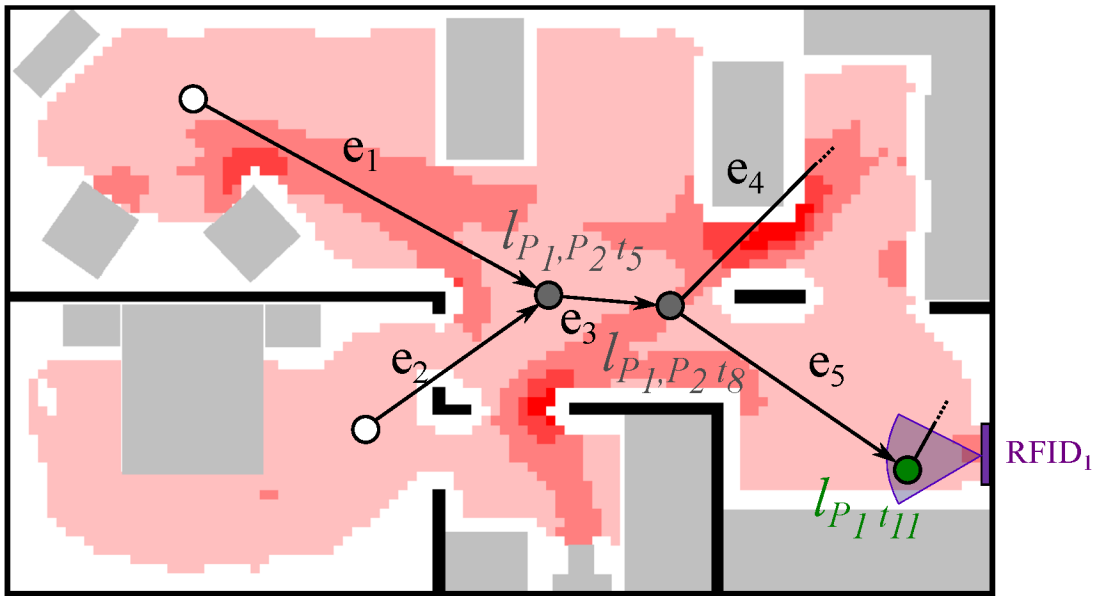


Figure A.18: Produced EM Graph for the two occupants up to t_{11}.

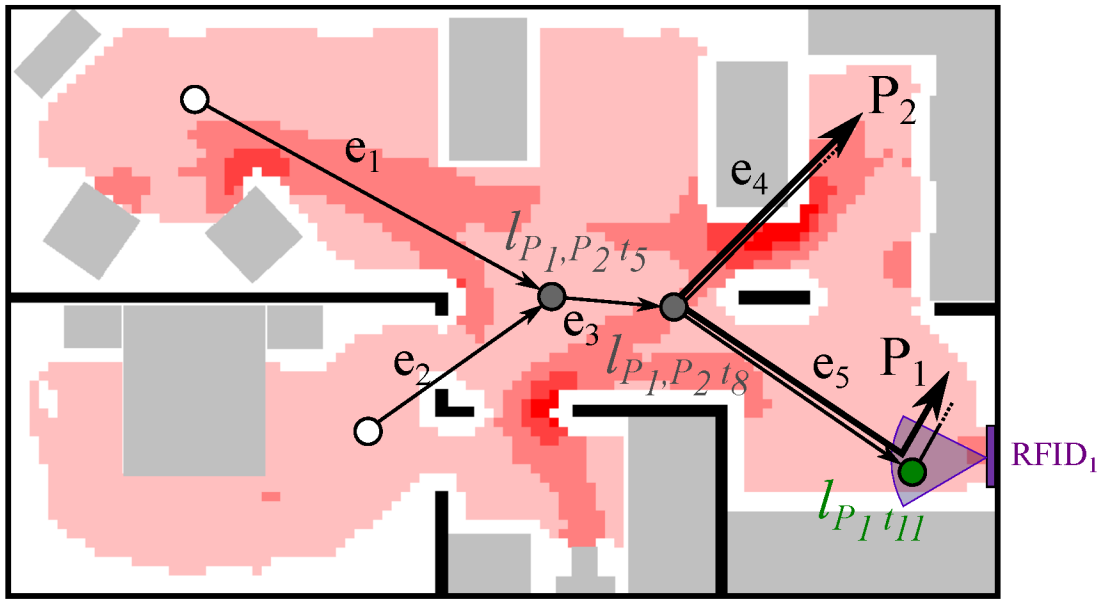


Figure A.19: Using the EM Graph for disambiguation.