Tracking Systems for Multiple Smart Home Residents

Aaron S. Crandall
acrandal@wsu.edu

Diane J. Cook
cook@eecs.wsu.edu

March 29, 2011

Abstract

Once a smart home system moves to a multi-resident situation, it becomes significantly more important that individuals are tracked in some manner. By tracking individuals the events received from the sensor platform can then be separated into different streams and acted on independently by other tools within the smart home system. This process improves activity detection, history building and personalized interaction with the intelligent space.

Historically, tracking has been primarily approached through a carried wireless device or an imaging system, such as video cameras. These are complicated approaches and still do not always effectively address the problem. Additionally, both of these solutions pose social problems to implement in private homes over long periods of time. This paper introduces and explores a Bayesian Updating method of tracking individuals through the space that leverages the Center for Advanced Studies in Adaptive Systems (CASAS) technology platform of pervasive and passive sensors. This approach does not require the residents to maintain a wireless device, nor does it incorporate rich sensors with the social privacy issues.

1 Introduction

Smart homes are providing an ever more important suite of tools for home care. From simple assistive tools to complex contextually aware interactive personas, smart home technology has penetrated many parts of the medical community. These tools rely on various sensors, artificial intelligence algorithms and human intelligence to operate. Most often the tools are geared towards recognizing the activities of daily living (ADLs), with the purpose of providing historical and instantaneous feedback about the residents' behavior. Any improvements to these tools in recognizing ADLs is welcomed by care practitioners and the residents themselves because it gives them a more accurate day to day picture of the residents' situation [1].

Currently, the latest in smart home technology has trouble operating with low profile, privacy aware sensor platforms. These sensor platforms are designed to minimize effort on the part of the resident while maximizing privacy at the cost of sensor granularity. The goal of research in this area is to make use of this reduced sensor information to still build systems capable of providing quality assistive technologies.

As an added hurdle, the smart homes are often deployed where there is more than one resident dwelling in the space. Even visitors and care providers make it difficult for the smart home system to determine which person currently in the space caused a given event and attribute it appropriately. Without that ability, ADLs become much more difficult to detect through the noise in the data and individual histories are impossible to obtain.

The tools used to follow an individual through the space are commonly called tracking systems. A tracking system's goal is to determine the current number and location of individuals, as well as their identity if possible. This information is invaluable when dealing with multi-resident situations to provide the computer a method of attributing events to individuals. There are three main strategies for tracking people in smart homes:

1. Carried Devices or Tags
2. Video biometrics
3. Entity detection via dividing events spatially and temporally

Carried devices or tags are commonly done via RFID [2, 3, 4] or a wireless device carried on ones person [5, 6, 7, 8]. The device or a base station of some sort reports the current location of the device to the central system. This has been accomplished using PDAs, cell phones, actigraphs, custom built RF devices, et al. While these kinds of systems work, it does require that every individual in the space keep and maintain their personal device at all times. It is easy for the residents to forget their device, have the batteries run down or not even want to have it. Additionally, guests need to be issued a device whenever they are in the space to ensure they are accounted for. In many environments this is a feasible solution, given the manpower to maintain it. For example,

hospitals and full time care facilities are often able to make use of such systems. In private homes or understaffed situations, it becomes a less feasible solution.

For video biometrics one or more cameras are placed around the monitored space. These cameras capture the current residents for tracking and processing [9, 10]. The goal is to interpret the video data to identify individuals, detect ADLs and give more context to item interaction. While these tools are often very good at meeting these goals, they bear the overhead of expensive cameras and the privacy concerns of the residents. Asking individuals to have 24 hour video monitoring in the homes can be difficult. While some may be willing to accept such an intrusion, many others will not [11, 12].

The last solution, doing Entity Detection by interpreting the sensor network data directly, strives to remove the effort of carried devices and the privacy concerns of cameras in exchange for more complexity in the tracking algorithm [13, 14]. Many smart homes are very sensor rich. By exploiting the physical locality of the sensors with the timing and general behavior of the residents, tools can be developed to determine how many residents there are and attribute events accordingly. This approach is a much more classical artificial intelligence one, and one that will likely get a probabilistic result. Whether or not it is good enough to support the other tools, such as ADL detection, is the question.

The researchers at Washington State University's Center for Advanced Studies in Adaptive Systems (CASAS) have built a set of smart home testbeds to support research into assistive care technologies. After working with the systems and the residents it was hypothesized that an algorithm could be devised to take the full event stream from anonymous residents and exploit the spatial and temporal features to make a tracking system without a carried device or rich sensors such as cameras. This approach was chosen from the outset because of known privacy and acceptance issues with smart home monitoring systems. People are often wary of having a monitoring system in their home, especially older adults who are still living independently. By working with a system that does not require cameras, microphones or a carried wireless device, more people are accepting of having such a system in their home.

This work introduces two algorithms that would be considered entity detection and tracking tools. They are used to divide the events generated by the sensor system into different sets. Each set represents a person currently in the space, and the events they caused on the sensor network. These sets can then be used to identify individuals, detect ADLs and give a much better sense of the behaviors occurring within the smart home. The result is a tracking system that uses passive and unobtrusive sensors to track people as they go about their day within the smart home space.

2 Sensor Platform and Data

The Center for Advanced Studies in Artificial Systems has constructed a number of smart home testbeds. These testbeds are used to record sensor information

from the activities of the residents, both in scripted and unscripted situations. The testbeds were designed to support the detection of resident activities and track individuals in a passive manner. Unto these ends, a number of sensor types have been introduced:

1. Passive Infra-Red (PIR) motion detectors
2. Temperature sensors
3. Power meters
4. Door open/close sensors
5. Item presence platforms
6. Light switch monitors
7. Water flow meters
8. Phone use

The primary role of these sensors is to aid in ADL detection. Out of these sensors only the PIR motion detectors are used in this work for tracking and localization of residents. The rest of the sensors have been found to have a marginal benefit for this purpose. The PIR sensors are commodity off the shelf home security sensors that have been modified with a custom built communications daughter board dubbed the Lentil Board. This daughter board senses a change in the sensor relay state, i.e.: ON vs. OFF, and transmits the change to a Dallas 1-wire bus to be logged by a host computer.

These PIR sensors come in two configuration. The first is an area sensor that is placed in a fairly common home security position. These sensors have a field of view that covers most of an entire room and are used to measure occupancy of the space. The second, and more common type, is a downward facing unit that had had its view occluded to only be able to see about 4' x 4' of the floor below it. These second PIR sensors give a much better sense of the location of motion within the space.

These kinds of PIR sensors are used quite often for smart home implementations. They're inexpensive, robust and accepted by most residents. During the initial design phases of the CASAS project, it was determined that high fidelity sensors, such as cameras, raise significant privacy concerns. When using these low fidelity platforms and more intelligent algorithms, the residents have been very accepting of being monitored by such systems. During several of the in-home deployments of CASAS testbeds the residents expressed significant concerns over having such a system watching them full time. By showing them the very simple form of the data gathered and the simple visualizations that can be created their initial concerns have been assuaged.

For the purposes of the algorithms used in this work only the downward facing sensors are used for the tracking of individuals. The area sensors give a much too general sense of a resident's location for any kind of precise locality.

Normally a PIR security sensor is only used to say someone is in the room, which is enough for intrusion detection. These smart home systems need a much finer-grained tool where an event means someone is within this small area. The more information that can be derived about locality the better.

The events created by the CASAS testbed are very simple. They come in the form of a four tuple:

1. Date
2. Time
3. Location
4. Message

The date and time are the time the event occurred, locally to the testbed. The location value is a named physical spot within the testbed, not an absolute coordinate. By abstracting the serial number of a device from the physical location individual devices may be changed out in the face of hardware failure without impacting the running algorithms. Lastly, the message field is an arbitrary string. For most devices, such as PIR sensors, it is a binary state with either "ON" or "OFF". Other more complicated sensors, such as temperature, power meters and water flow sensors, put a number value or a description of their state in this field.

By leveraging a platform that generates simple and discrete events, the total size of the data sets created are smaller than approaches that use more complex sensor sources such as video or wearable sensors. There is also significantly less pre-processing to do to interpret the data for later classification. For example, to determine resident location using a camera a number of image processing techniques have to be applied first. These techniques can take a noticeable amount of computational resources and induce additional noise that later algorithms need to account for. The CASAS sensor platform and event model make for very clean data to be used by artificial intelligence algorithms.

3 CASAS Testbeds

In this work two different CASAS testbeds were used. In both locations people live, work, and perform daily activities without external guidance or scripting. The Tokyo testbed shown in Figure 1, is located in a WSU Electrical Engineering and Computer Science (EECS) department lab. The room is 12.2m x 10.9m, and is used by the CASAS graduate students as their main work area. Within the room there are a number of separate spaces containing desks, cubicle walls, a conference area, a sitting area and an inner room with engineering work tables. Anywhere from zero to 9+ people will be in the space at any given time, at nearly any hour of the day. The testbed is outfitted with PIR sensors on the ceiling, a front door open/close sensor and power switch sensors. The testbed has now been operational for nearly three years.

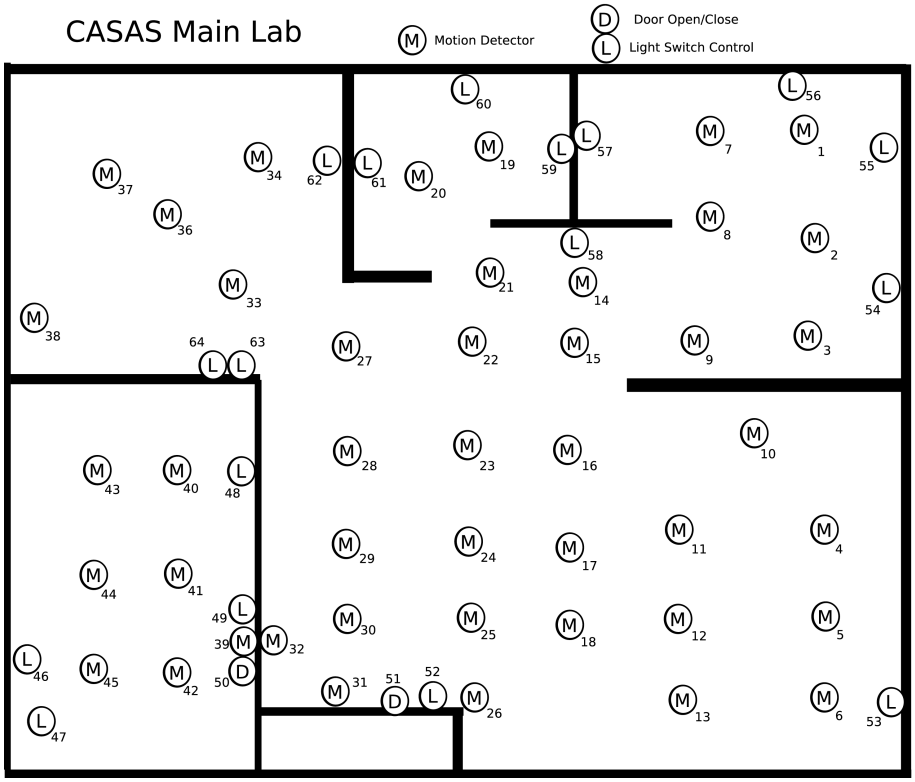

Figure 1: Floor plan for WSU CASAS office testbed, named Tokyo.

Tokyo has 44 motion detectors[1]. They are all placed on the ceiling pointing down at roughly 1.2m intervals. The ceiling is a dropped t-bar ceiling with a nearly uniform height of 3.0m and the sensors are attached to the t-bar surface. An example of this implementation at the Tokyo testbed can be seen in Figure 2.

The Kyoto testbed is a three bedroom campus apartment provided by WSU Housing. The facility has 52 motion detectors along with many other types of sensors. This testbed has been in operation for just over two years, normally with two full time undergraduate residents. In addition to day to day living, the space is also used for a number of smart home studies. These studies will have one or more people moving about the space in scripted or unscripted behaviors during the day.

A map with the Kyoto sensor placement is shown in Figure 3. The sensors that begin with 'M' are the motion detectors used for determining the rough location of motion with the space. The rest of the sensors monitor doors, water flow, lights and items.

The third bedroom of this apartment is unoccupied by a permanent resident.

[1]There is no motion detector number 35 due to a numbering error at installation time.

Figure 2: Sample of PIR motion detectors, as installed in the Tokyo testbed.

Instead, the room is utilized as a control room by experimenters who run short controlled experiments in the Kyoto testbed. To keep the noise in the data set down, this room has a bare minimum of sensors.

4 Approaches

This work introduces two algorithms to track individuals in the smart home space. They both attempt to exploit the physical and temporal features of the events generated by the residents on the CASAS sensor network. The goal is to incrementally build a model of what is likely transpiring, i.e. who is moving where when, and attribute the events accordingly. Because the tools will eventually need to operate in real time, they take events in order and should be able to classify them quickly. This classification would then be used by other tools, such as ADL detectors, to more accurately describe the current activities.

In both tools it was determined that some terminology had to be defined. The researchers use the term 'entity' within the models to represent an individual. This is because not every entity in the model represents a person. Most often they are people, but the studies have included smart home installations with cats, dogs and even robots that cause events. By using the term entity, it allows for a wider understanding of how complex living spaces can be.

The two algorithms are similar in many ways, as the evaluation of one led

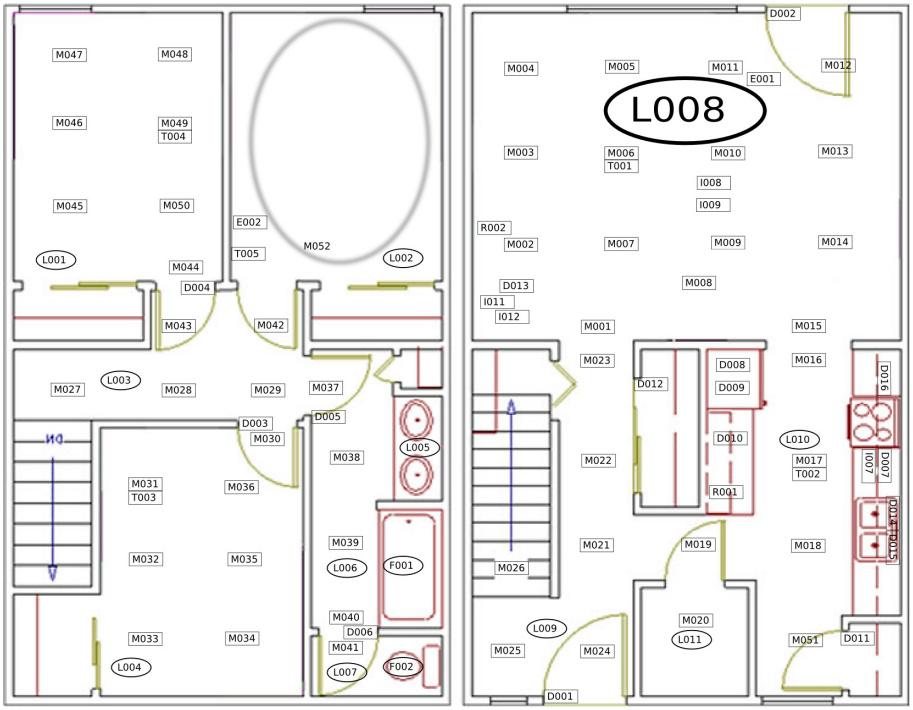

Figure 3: Floor plan for WSU CASAS student apartment testbed, named Kyoto. The various sensors are labeled with their location name (M = motion, L = light, I = item, D = door, T = temperature) and number.

to the creation of the other. The first algorithm is a rule-based tool. It uses a set of simple rules combined with a graph of all possible routes between sensor locations to track individuals. This tool is dubbed 'GR/ED', which stands for Graph and Rule based Entity Detector. The initial results for the GR/ED were promising, but the tool fell down in more complex social situations as well as in poor sensor network environments. The GR/ED is introduced and explored in more depth in section 4.2.

As a means to exploit the available data and create a better tool, a second tool based on Bayesian Updating was created. By using a corpus of training data annotated with the number of residents, a probabilistic transition matrix is built and used to update the world model. This tool is dubbed the 'BUG/ED', which stands for the Bayesian Updating Graph based Entity Detector. By leveraging a probabilistic model, the system is able to handle significantly more issues with the sensor network and perform marginally better in the face of more complex resident behaviors. With these additional successes, the BUG/ED was also tested for its efficacy at improving the performance of a Bayesian Classifier for doing ADL detection. The BUG/ED is discussed in more detail in section 4.3.

4.1 Annotated Data

To use both algorithms two corpora of data were created. A subset of the stored events for both the Tokyo and Kyoto testbeds were taken and annotated by humans. The humans were taught to watch the events as they were replayed using a visualization tool and log the current number of residents in the testbed. This value representing the current occupancy could then be used to determine how accurate the tracking tools were, and in the case of the BUG/ED it was also used to train the transition probability matrix.

The Tokyo data set represents sensor events that were generated while faculty, students, and staff performed daily working routines in the lab over a course of 59 days. To train the algorithm, the data was manually inspected by a person and every event annotated with the current number of residents in the space. In total this made for 209,966 motion sensor events, with a mean of 86.84 events and a standard deviation of 201.11 events per resident instance. The resident count ranged from zero to more than nine during this data gathering window.

Once the testbed had more than six to seven people in it, the annotators noted that there was little available information to identify what was happening in the space. This was anecdotal evidence for the limited resolution of the testbed. Adding more sensors should increase this maximum detectable occupancy.

The Kyoto data was taken from 21 days of the Kyoto testbed. This made for 231,044 motion sensor events, with a mean of 603.67 events and a standard deviation of 843.17 events per resident instance. Again, the sample data was inspected by a person and annotated with the number of people currently in the space. In this set, the number of residents ranged from zero to five and the annotators noted a marked decrease in their ability to interpret individuals' movements as the occupancy reached about four residents.

4.2 GR/ED - Graph and Rule Based Algorithm

The GR/ED algorithm was designed to use the order of events to incrementally track individuals in the CASAS testbed. The core idea is that entities will most likely trip sensors as they cross from one location to another, and multiple entities will often be separated by one or more sensors as they go about their day.

The "graph" part of the tool is based around the physical locations of the sensors within the testbed. The two CASAS testbeds used in this work are shown in Figures 4 and 5[2]. These graphs are made up of only the downward facing PIR motion detectors, which are laid out to cover most of the floor space. Since the sensors are placed to cover the space fairly well, people walking around have an obvious and complete chain of events from one place to another. The graph that represents a given space has vertexes representing the sensors

[2]The edge cutting across the Kyoto graph from M026 to M027 is connecting the sensor at the bottom of the stairs with the one at the top leading to the second story.

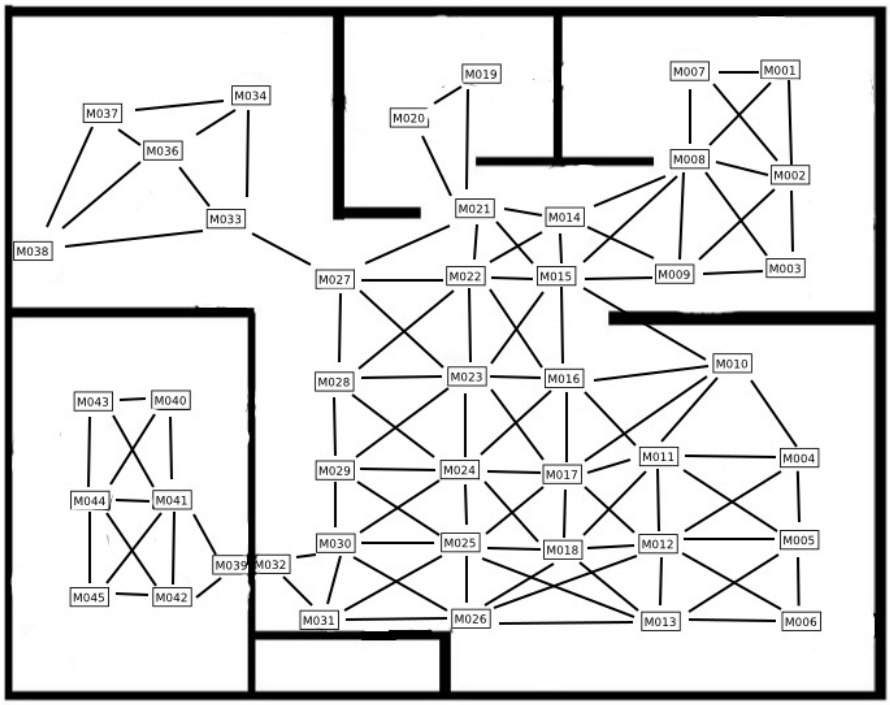

Figure 4: Graph of sensor locations for the Tokyo testbed.

themselves and edges that represent the possible connections between those vertexes.

The rule based part of GR/ED is a simple set of logical rules for creating, destroying and moving entities within the model based on the evidence given by the event series. The first rule is for creating a new entity. With this rule, if an "ON" event occurs at a location with no adjacent entities, a new entity is created. This theoretically means that this event was caused by a heretofore unseen entity. They could have either entered the space, or have been shadowing another one of the residents and only just then been separated enough to have noticed as a separate entity.

The second rule is for destroying entities. An entity is destroyed from the model under two circumstances. First is when they have been determined to leave the sensor network. In the case of the CASAS system, this is when an entity moves to the sensor most adjacent to the exit. Since there is no hardware available to easily determine whether someone has moved through the doorways, it can be assumed that moving next to the door is an exit.

The second way an entity can be destroyed is when they fail to generate new events for a period of time. If the model has an entity that does not actually exist, then it must have a means to recover. Since the PIR sensors do not provide

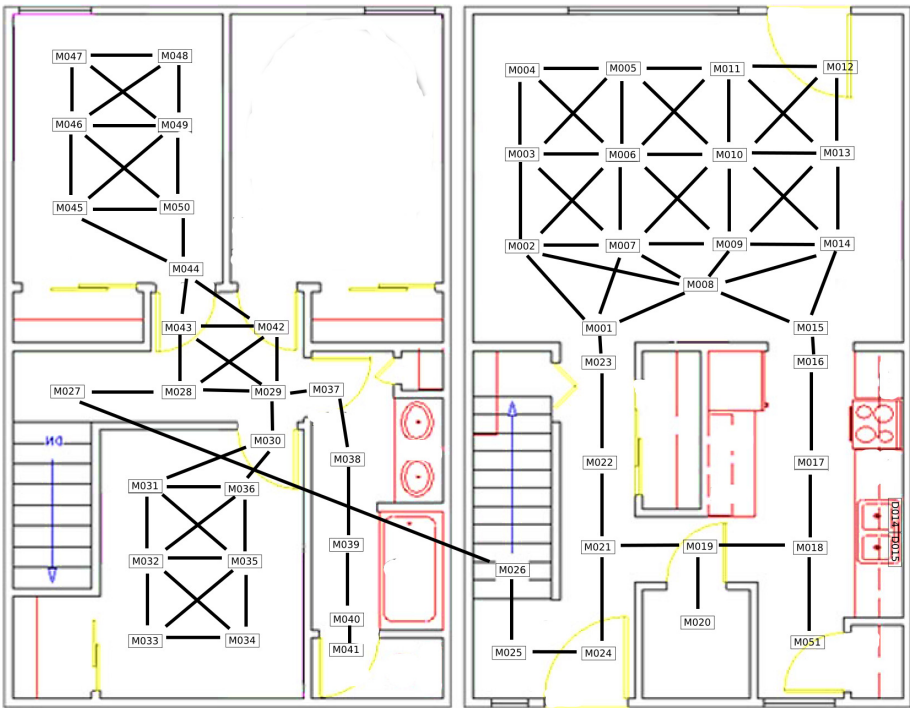

Figure 5: Graph of sensor locations for the Kyoto testbed.

data if an entity does not move, then it becomes difficult to determine if they are still in a given location, or if they have moved away without triggering events. This kind of movement can either occur due to a flaw in the sensor network, or if two entities move to the same location followed by moving together across the space. Since the sensors do not provide a magnitude of the size of an entity, it is easy for multiple people to move as a group and leave old entities in the model that no longer exist. To remedy this, a timeout on entities has been imposed. After trying a wide range of values with the Tokyo data set, it was determined that a timeout of 300 to 600 seconds is the best range, and 300 was used for this work.

The final rules for the GR/ED tool have to do with movement. The first rule for movement is that when an "ON" event occurs and an entity is at a neighbor in the graph, then that entity moves to the location that generated the event. Only one entity can have that event attributed to them, so if more than one entity is adjacent to the new event, then the one that moved most recently takes it. This most recent mover continues rule allows the system to deal with entities moving together in tighter areas.

As the system operated, it was noticed that people could easily fool the GR/ED by walking back and forth. The PIR sensors chosen are from a commodity home security product line. Because the home security hardware is slow,

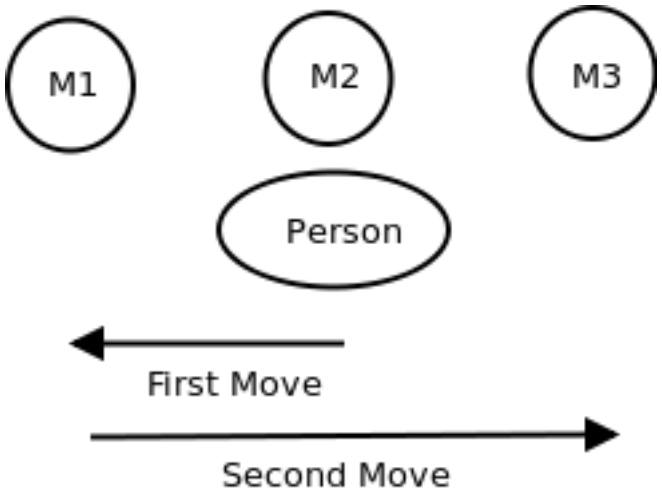

Figure 6: Example of movement that breaks the simple GR/ED algorithm.

the sensors stay in the "ON" state for anywhere from one to five seconds before turning back off once movement stops. Due to this very long time frame, people could walk in the pattern shown in Figure 6, which would move their virtual entity to the node on the left, but the sensor in the middle would stay on long enough that they would then move to the sensor on the right without causing an "ON" event on the middle sensor. This would leave their old virtual entity on the left, and create a new one from the new "ON" event from the right most sensor. At this point the system was out of sync with the space and the false entity left behind would have to time out before the GR/ED would be correct again. To remedy this failing, the Open List of sensors was proposed.

With the Open List, an entity has a set of locations that define their present location instead of only a single one. For every ON event sent by the sensors, there is always an OFF to match it. When an entity is attributed an ON event, that sensor location is placed in their Open List. Once that sensor finally sends an OFF event, the location is removed from their Open List. Now that this list is available, an entity's location is not merely their current vertex in the graph, but the whole of the Open List. If an ON event occurs that is adjacent to any location in this list, it will be attributed to the entity. This technique remedied most problem instances of people walking back and forth. In the previous example, the entity's Open List would be both the center and left sensors. So when they next trip the right sensor they are still considered "adjacent," due to the middle sensor being in their Open List, and would properly be attributed that new event from the right sensor.

Each entity in the model has a list of locations that it has visited in the past. The ordered list of these locations may be used to build a tracklet for the resident. Alternatively, the current number of entities in the GR/ED model is the estimated occupancy of the space.

The resulting system was efficient and operated in near real time, making it feasible for real-world smart home implementations. As an added advantage it takes no training data to operate, only the graph of possible routes between sensor locations. This would allow the GR/ED to be deployed and started once the layout of the sensors is known without having to wait for any kind of annotated training data to be made available.

4.2.1 Testing the GR/ED

The GR/ED tool was tested for accuracy at counting the current number of residents using both the Tokyo and Kyoto data sets. The tool was evaluated using 10-fold cross validation, divided by days. This validation was run 30 times to provide variance for significance values. Once the data sets were run through the tool, the resultant guesses were compared to the human annotated ground truth. The results could then be inspected for total number of event correct, as well as total length of time correct.

For a form of baseline comparison, a weighted random classifier was trained and tested on the same data. The weighted random algorithm was also run 30 times to provide variance.

4.2.2 Results for GR/ED

GR/ED was accurate with one resident as expected, but rapidly fell to a lower rate as the number of residents increased. In Figure 7, the accuracy by number of events on the Tokyo data set is shown. Note that as the resident count increased the accuracy declined, though the GR/ED algorithm was always significantly better than a weighted random guess.

Since the GR/ED tool cannot tell the difference between a single or multiple residents at a given location, while the annotators can, it is often too low in its estimations. Additionally, it can be too high if an entity is a false positive until it times out. Overall, the GR/ED algorithm achieved an overall accuracy of 72.2% with a standard deviation of 25.21% by counting events and an accuracy of 88.9% with a standard deviation of 12.8% for the total time represented by the data set.

The Kyoto data set truly showed the flaws in the GR/ED algorithm. This testbed has significantly more sensor error. People are able to move past sensors in many more places without tripping intermediate sensors. This quickly leads to many false entities being created in the model and a marked reduction in accuracy. Overall, the GR/ED had an accuracy of 16% measured by number of correctly-labeled events and 45% for total correctly-labeled time on the Kyoto data set. These low accuracies placed it well within range of a weighted random guess, so further evaluation on the Kyoto data set was abandoned.

The GR/ED tool has the advantage of not requiring any training data, only the graph itself. If the sensor locations can be determined at installation time, or automatically through some means, then this tool can be used with a new

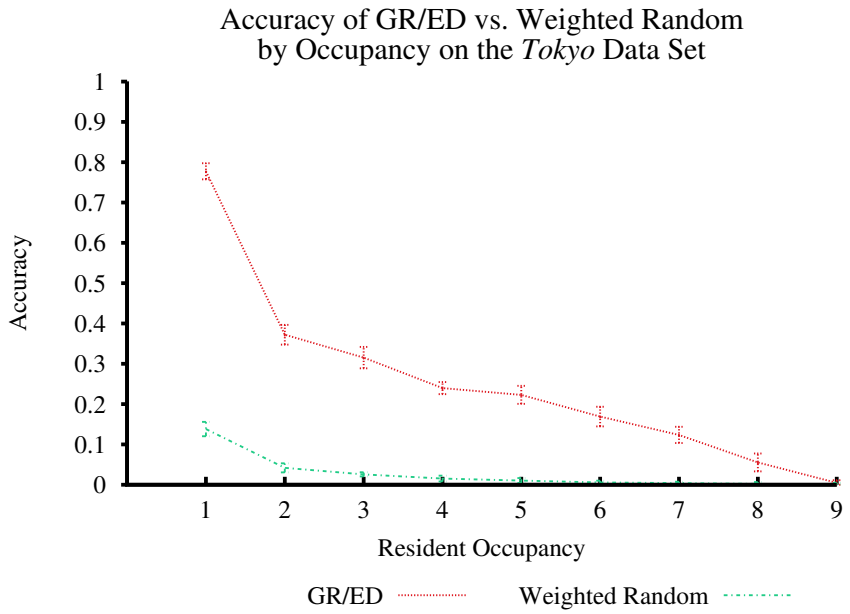

Figure 7: Accuracy by occupancy count for the GR/ED tool on the Tokyo data set. The error bars show three standard deviations.

smart home installation quickly. Depending upon the needs of the other tools within the system, it may be sufficient for the smart home application.

Because the graph used by the GR/ED is so rigid, it was determined that a more probabilistic model might be a better solution. Instead of relying on a human-created set of equal and fixed connections between locations, perhaps a graph of likely connection as derived from the annotated data might serve better. This lead to the application of Bayesian Updating and the creation of the BUG/ED tool.

4.3 BUG/ED - Bayesian Updating Graph

After looking through various algorithms, it was determined that a Bayesian Updating algorithm might be a good choice for a successor to the GR/ED tool. Bayesian Updating is a probabilistic strategy where new evidence provided is used to update the guess at the model of the world. The Bayesian Updating Graph Entity Detector (BUG/ED) proposed here takes the current model of the smart home space, with respect to the current resident locations, and combines it with the new evidence in the form of a sensor event to build the most likely world model for the latest state. The behavior is similar in many ways to the GR/ED, but instead of a simple and uniform graph it has a transition matrix of probabilities. The matrix can also be augmented with other sources of evidence, though the algorithm here was only provided sensor location to sensor location

transition likelihoods.

The biggest advantage of the BUG/ED over the GR/ED is handling failures of the sensor network. Often a person will bypass a sensor in the graph, which caused an immediate problem for the GR/ED tool. It would end up creating a new entity in the model, and abandon the old one improperly. With the BUG/ED, the transition matrix will normally have a likelihood of transition between those two more distant sensors and will often properly move its entity, even if the person it represents skips sensors occasionally. This ability alone increased the robustness of the system in day to day operation.

4.3.1 Training the BUG/ED probability matrix

With Bayesian Updating, there must be some corpus of information for the algorithm to use in estimating the conditional and joint probabilities. Obtaining or generating that corpus is up to the implementation and domain. The data annotated by humans for the *Tokyo* and *Kyoto* testbeds specifying the number of current residents was used to train the BUG/ED transition matrix. This training process is done before operation of the BUG/ED can begin and resembles the GR/ED algorithm with a very important addition.

Since the annotated data has the true count of residents, the training algorithm can make use of that key data for determining when residents entered and left the space . The training algorithm takes the events from the training data one at a time and incrementally builds a model of the residents' locations and transitions between sensor locations, much like the GR/ED. The key difference is that it uses the resident count from the training data to decide when to create, destroy or move entities.

The training algorithm also makes use of the same graph utilized by the GR/ED tool, but only for counting hop count between sensor locations. This graph has one addition for the BUG/ED algorithm, a virtual sensor location called "OUTSIDE." This OUTSIDE location represents all of the universe not monitored by the smart home sensors. It is directly connected via an edge in the graph to any sensor at an exit to the smart home, such as sensors next to the front and back doors. Entities are also moved from OUTSIDE when they are created, and to OUTSIDE when removed. The graph is used in determining which entity is closest to the OUTSIDE location, or which entity is closest to an event that just occurred.

The training algorithm will either create, destroy or move entities by looking at whether the resident count went up, down or stayed the same between events. If the count goes up, a new entity is created at that location by moving them from the virtual location OUTSIDE to the location of the event. If the count goes down, the entity closest to the exit is immediately moved to the virtual location OUTSIDE. If the count stays the same, the closest entity to the event on the graph is moved there.

Every time an entity is created, destroyed or moved, that transition from one location to another is added to a matrix. The matrix represents the number of times entities transitioned between locations, and is the source of probabilities

during the operation of the BUG/ED algorithm on new data. The length of time an entity resides at a given sensor location is also kept. This set of time lengths is used to determine dynamic timeouts for entities, which will be discussed in greater depth later.

4.3.2 Noise reduction in the BUG/ED matrix

The training algorithm for the BUG/ED matrix is not perfect. Inspection of the results shows several instances where the model of the testbed got into a state where taking the closest entity was inappropriate. For example, when the two residents were both in the living-room of the Kyoto testbed and moved upstairs together, one of the virtual entities would be moved with them while the other was left behind. As long as the residents remained together upstairs, the single virtual entity with them would bounce back and forth between their locations. This situation would increase the likelihood of transition between two unrelated locations, but by having a large enough training set this would not normally impact the overall performance of the BUG/ED too greatly.

In some of the training data the human annotators were also incorrect in their resident count. Since that value is very important to the training phase, these bad training files would also impact the overall accuracy of the system.

To overcome these aberrant transitions between sensor locations, a flooring filter was applied to the transition probability matrix. Any transition likelihood below the threshold would be changed to the lowest probability. Setting a flooring value was seen to have a profound effect on the behavior of the system. If too many bad connections were left in, by setting it too low, then the BUG/ED would have too little evidence to create new entities as people entered the space. Alternatively, if it was set too high then too many entities would end up being created. For each data set, the value to floor with was experimentally derived. In future work, a proper outlier detection algorithm for each sensor location will replace this fixed number.

An additional noise reduction tool was implemented to remove training data that was too complex for good use. This was a maximum occupancy limit on the training data. As the number of residents increases within a space, it becomes more and more difficult to determine how many are truly there. This limit is a factor of the sensor density and how mobile the residents are. It was noted by the annotators that once more than five or six people were in the Tokyo testbed, it was nearly impossible to keep their locations perfectly tracked. At that juncture, the annotators watched the entrance for people entering and leaving more than individual events anywhere in the space. Since the training algorithm to build the BUG/ED transition matrix is a simple one, a ceiling value on the number of occupants in the space was implemented. If the training data exceeded that number, it was thrown out. Between removing very unlikely connections and not using training data with too many residents, the BUG/ED tool started to perform much better in day to day use, and the overall accuracy of the system improved.

4.3.3 Dynamic Timeouts in the BUG/ED

In the GR/ED tool, a flat timeout for entities was enforced. This was set at 300 seconds, a figure experimentally derived by running the GR/ED tool on the data repeatedly with different timeout values. The overall accuracy at determining the number of residents was compared for each timeout. The best value of 300 seconds was taken for future work with the tool. This flat timeout of 300 seconds is the default used by the BUG/ED as well, though it is supplanted by the dynamic timeout algorithm described below.

It was noted by the residents that the GR/ED would timeout most often when people sat and worked in a location for a period without moving enough to cause sensor events . Because the training algorithm for the BUG/ED is stateful and remembers an entity's location indefinitely until they move, it could be used to find a more appropriate timeout for every sensor location. It was hypothesized that by making a dynamic timeout system that utilizes the training data, the BUG/ED would be improved when handling situations where entities remain still for long periods of time.

As the BUG/ED transition probability matrix is being trained, the length of total time an entity spends on a given sensor is kept. Once the data has all been used for training, these lists of times are used to calculate a customized timeout value for each sensor location. The mean plus three standard deviations (to capture 99% of all occurrences) of the time lengths in a sensor's list was used for the timeout value at every given location.

Manual inspection of the customized timeouts largely conformed to the expected pattern. Areas such as hallways and kitchens had shorter timeout values, while desks, beds and couches ended up with longer timeouts. This was not always true, but the flaws in the timeout calculations were results of flaws in the simple training rules used to build the transition probability matrix, and not the timeout calculation algorithm.

4.3.4 BUG/ED Bayesian Updating

During operation of the BUG/ED a model of the current entity locations is maintained. This model is modified by motion events with an "ON" message arriving. The only two things that may occur are either an existing entity is moved to the location of the event, or a new entity is created.

The likelihood that an entity e of all existing entities E has moved to the sensor location s_k of the sensor that fired from the entity's old location $e_{s_{k-1}}$ is calculated using Bayes' Rule in equation 1.

$$arg\ max_{e \in E}\ P(e|s_k) = \frac{P(s_k|e_{s_{k-1}})\ P(e)}{P(s_k)} \tag{1}$$

The value of $P(s_k|e_{s_{k-1}})$ is taken from the probability transition matrix. This is the likelihood that the entity transitions from their current location to where the latest sensor event is located based upon the historical training data. If the transition never occurred in the training data, then it was given a

very small minimum value based on the smalled existing value in the transition matrix.

The factor $P(e)$ is considered the same for all entities, as they all have an equal likelihood of moving at any given time. This value could be modified with information about the likely direction, speed or likelihood of movement based on training information and become a serious factor in future versions of the BUG/ED.

The last value in the denominator of $P(s_k)$ is the same for all entities as it is the probability that the given sensor fired. Since this is a constant for all entities being compared, it is only a scaling factor.

Of the existing entities, the one with the highest likelihood (P_{move}) of making the transition to the sensor that fired is chosen to move in the model. This likelihood is compared to a threshold of the probability to create a new entity in the model instead of moving an existing one (P_{create}). If ($P_{move} < P_{create}$), then a new entity is initialized at s_k and the number of active entities in the model increases by one. Otherwise, the most likely entity to move has its tracklet of events increased by adding the most recent event and its location is updated to s_k.

At this juncture the BUG/ED has an updated model from the old model with the new evidence from the latest event. These updates reflect the most likely series of events based on the historical training data.

4.3.5 Testing the BUG/ED

The BUG/ED was tested using the same two data sets as the GR/ED tool. Because the BUG/ED requires training data, a 3-fold cross validation system was implemented. In this case, 2/3 of the available days were used to train the transition matrix, and the last 1/3 was for testing. The days were randomly selected and the model was reset with each new day when testing.

The overall accuracy value was calculated by counting the number of events where the BUG/ED was correct in the current number of residents. The difference between the true value and the current guess by the tool was also calculated to give a sense of how far off the model was from the ground truth. Since this is a probabilistic model, some error is to be expected. Depending upon the final use of the tools, having a roughly accurate guess might be sufficient for the smart home system's needs.

4.3.6 Results of the BUG/ED

The BUG/ED performed better than the GR/ED tool on these data sets. It was noted by researchers watching the BUG/ED operate in real time that it felt more 'stable'. Indeed, the BUG/ED failed less often in the face of skipping sensors and timed out less often when people stayed in one place for a period of time. These results were quantified by higher accuracy rates and measurable benefits to the ADL detection tools.

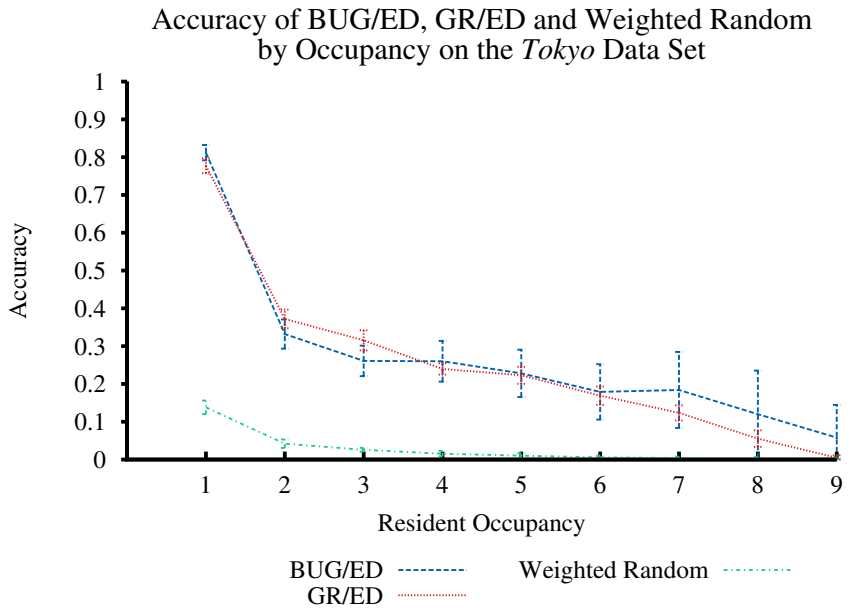

Figure 8: Accuracy by occupancy count for the tools on the Tokyo data set. The error bars show two standard deviations from the mean.

As a baseline comparison, the BUG/ED and GR/ED tools were also compared against a Weighted Random classifier. This classifier was weighted by the occupancy counts and used to guess the current occupancy at every event. The tool was run 30 times for each data set and its variation used to determine significance.

The BUG/ED tool's overall accuracy improved over that of the GR/ED on both data sets. It was a significant improvement on the Kyoto data, mostly due to its ability to handle missed sensors as people moved about. Overall, the BUG/ED classifies 44% of the events correctly, which accounts for 85% of the total time on the Tokyo data set. Where it improves over the GR/ED tool is when there are more people occupying the space. In Figure 8, the accuracies for 2, 3 and 4 residents are noticeably higher than in Figure 7, which showed the GR/ED results for the same data. This new robustness in the face of more residents attests to the efficacy of the BUG/ED approach.

Where the BUG/ED truly performed much better was on the Kyoto data set. While the GR/ED tool routinely failed as people traversed the space, the BUG/ED would much more often track them correctly. In Figure 9, it shows that in the most common state of two residents, the tool performs perfectly accurately just over 60% of the time. Overall, the BUG/ED classified 59% of the events and 67% of the total time for the Kyoto data set correct. This was significantly better than the GR/ED tool on this data set.

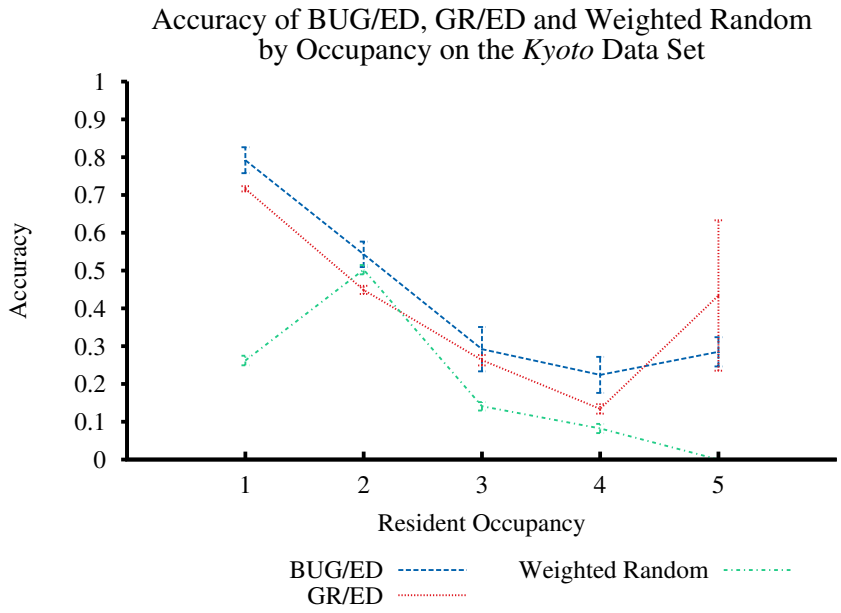

Figure 9: Accuracy by occupancy count for the tools on the Kyoto data set. The error bars show two standard deviations from the mean.

These improvements in behavior and accuracy attest to using a probabilistic model for decision making in this kind of tracking system. There are too many uncertainties with sensor placement, resident behavior and system configuration to have a purely rule based system operate well.

4.3.7 Application of BUG/ED to Activity Recognition

Many of the applications of smart environments that have been explored, such as health monitoring, health assistance, context-aware services, and automation, rely upon identifying the activities that residents are performing. Activity recognition is not an untapped area of research and the number of algorithms that have been used to learn activity models varies almost as greatly as the types of sensor data that have been employed for this task. Some of the most commonly-used approaches are naïve Bayes classifiers, decision trees, Markov models, and conditional random fields [15, 16, 17].

While activity recognition accuracy has become more reliable in recent years, most existing approaches are applied to situations in which a single resident is in the space performing activities. Recognition accuracy significantly degrades when multiple residents are in the same space. We hypothesize that this accuracy can be improved if the data is separated into multiple streams, one for each resident, or if each event is labeled with the corresponding resident identifier.

To validate this hypothesis, we apply the BUG/ED algorithm to data col-

Set Name	#Months	#Residents	#Activities
Set 1	2	2	12
Set 2	2	2	13
Set 3	5	2	25

Table 1: Attributes of the three tested Kyoto data sets.

Set Name	Without BUG/ED	With BUG/ED
Set 1	42%	40%
Set 2	63%	88%
Set 3	54%	63%
Overall	56%	67%

Table 2: Before and after ADL detection accuracies when adding BUG/ED tracking information to Kyoto data.

lected in the Kyoto apartment while two residents lived there and performed normal daily routines. The data used for this experiment actually represents different time frames, different residents, and different activities than were used to train the BUG/ED graph. Attributes that describe these three data sets are shown in Table 1. The activities included in this data set include, but were not limited to, sleeping, eating, cleaning, watching TV, toileting, cooking and showering. None of the activity was scripted and the annotation was done with the help of the residents themselves for accuracy.

Table 2 summarizes the performance of the activity recognition algorithm for each data set with and without entity labeling using BUG/ED. As is shown in this table, the accuracy of activity recognition generally improves when entity detection and tracking are employed.

To demonstrate that the BUG/ED strategy is useful in further smart home tools, it was used to annotate three new sets of Kyoto data. That data was then used to train a naïve Bayesian ADL detector. The results with and without the BUG/ED tracking information were compared and summarized in Table 2.

These three data sets are annotated for 11 different ADLs in an unscripted environment. There are two residents, though one or even more than two might be present at any given time. The data sets cover nearly a full calendar year in total and run all day every day. The overall improvement to complex ADL detection was just over 10%.

5 Conclusion

In this work two different, though similar, tracking tools were introduced and evaluated. The first uses a graph of the sensor network in a smart home environment and a set of rules to determine the current location and history for individuals. The second uses a history of resident occupancy information to

build a set of probabilities to be used by a Bayesian Updating tool for tracking residents. Both tools have benefits and negatives, though overall the probabilistic model provided by the BUG/ED performed better, especially in an environment with poor sensor layout.

There will be places for all kinds of tracking systems in smart home technologies. Choosing the right one for the needs of residents will be important for the continued success of smart homes in multi-resident situations. Continued research into passive tracking systems should improve upon these kinds of tools, allowing smart homes to handle ever more complex behaviors and numbers of residents.

6 Future Work

Both of the tools presented here offer chances for improvement. The BUG/ED especially has opportunities for continued success. First would be a better method of garnering the transition matrix. Changing the algorithm for training the matrix, or even finding ways to reduce the amount of data needed to make a successful set of probabilities would be very beneficial. Second would be incorporating better methods of detecting an entrance or exit of individuals. This could be accomplished by taking door sensor information into account, as well as a more specific kind of sensor at the doorway to report someone entering or leaving. Finally would be an evaluation of the impact of the sensor layout itself. The current CASAS sensors are focused on detecting ADLs, but perhaps some sensors could be placed in key locations to improve the tracking ability of the system.

Acknowledgments

This work is supported by NSF grant IIS-0121297.

References

[1] M. Skubic, G. Alexander, M. Popescu, M. Rantz, and J. Keller, "A smart home application to eldercare: Current status and lessons learned," in *Technology and Health Care*, vol. 17, no. 3. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009, pp. 183–201.

[2] D. J. Cook and S. K. Das, "How smart are our environments? An updated look at the state of the art," *Journal of Pervasive and Mobile Computing*, vol. 3, pp. 53–73, 2007.

[3] U. Naeem and J. Bigham, "Activity recognition in the home using a hierarchal framework with object usage data," *Journal of Ambient Intelligence and Smart Environments*, pp. 335–350, 2009.

[4] J. S. Choi, H. Lee, R. Elmasri, and D. W. Engels, "Localization systems using passive UHF RFID," in *The International Joint Conference on INC, IMS and IDC*, ser. NCM '09. Los Alamitos, CA, USA: IEEE Computer Society, August 2009, pp. 1727–1732.

[5] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, August 2001.

[6] R. C. Luo and O. Chen, "Indoor human dynamic localization and tracking based on sensory data fusion techniques," in *The IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS '09, October 2009, pp. 860–865.

[7] E. Navarro-Alvarez and M. Siller, "A node localization scheme for ZigBee-based sensor networks," in *The IEEE International Conference on Systems, Man and Cybernetics*, ser. SMC '09, October 2009, pp. 728–733.

[8] J. J. M. Diaz, R. d. A. Maués, R. B. Soares, E. F. Nakamura, and C. M. S. Figueiredo, "Bluepass: An indoor Bluetooth-based localization system for mobile applications," in *IEEE Symposium on Computers and Communications*, ser. ISCC '10, June 2010, pp. 778–783.

[9] V. Libal, B. Ramabhadran, N. Mana, F. Pianesi, P. Chippendale, O. Lanz, and G. Potamianos, "Multimodal classification of activities of daily living inside smart homes," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, ser. Lecture Notes in Computer Science, S. Omatu, M. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, and J. Corchado, Eds. Springer Berlin / Heidelberg, 2009, vol. 5518, pp. 687–694.

[10] V. Menon, B. Jayaraman, and V. Govindaraju, "Biometrics driven smart environments: Abstract framework and evaluation," in *Ubiquitous Intelligence and Computing*, ser. Lecture Notes in Computer Science, F. Sandnes, Y. Zhang, C. Rong, L. Yang, and J. Ma, Eds., vol. 5061. Springer Berlin / Heidelberg, 2010, pp. 75–89.

[11] G. Demiris, B. K. Hensel, M. Skubic, and M. Rantz, "Senior residents' perceived need of and preferences for smart home sensor technologies," in *International Journal of Technology Assessment in Health Care*, vol. 24. Cambridge University Press, 2008, pp. 120–124.

[12] P. Klasnja, S. Consolvo, T. Choudhury, R. Beckwith, and J. Hightower, "Exploring privacy concerns about personal sensing," in *Proceedings of the International Conference on Pervasive Computing*, ser. PerCom '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 176–183.

[13] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proceedings of the International Conference on Ubiquitous Computing*, ser. UbiComp '08. New York, NY, USA: ACM, 2008, pp. 114–123.

[14] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Using height sensors for biometric identification in multi-resident homes," in *Pervasive Computing*, ser. Lecture Notes in Computer Science, P. Floréen, A. Krüger, and M. Spasojevic, Eds., vol. 6030. Springer Berlin / Heidelberg, 2010, pp. 337–354.

[15] D. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods of Information in Medicine*, vol. 48, no. 5, pp. 480–485, 2009.

[16] U. Naeem and J. Bigham, "Recognising activities of daily life through the usage of everyday objects around the home," in *International Conference on Pervasive Computing Technologies for Healthcare*, ser. PervasiveHealth, April 2009, pp. 1–4.

[17] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Recognizing independent and joint activities among multiple residents in smart environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, no. 1, pp. 57–63, 2010.