Design and Organization of Autonomous Systems
Person Identification SYStem

Martijn Liem
Thomas Mensink
Markos Mylonakis
Roeland Weve

2nd February 2006

Supervised by Dr.ir. Marinus Maris

`{mliem,tmensink,mmylonak,rweve}@science.uva.nl`
`http://www.science.uva.nl/~rweve/DOAS/`

Abstract

Ambient intelligence applications heavily rely on successfully identifying people in their home environment. Current applications use complex, expensive or intrusive sensors and methods like RFID tags, retina scans or human face identification. In this paper we present the *Person Identification SYStem (PISYS)*. In contrast with previous work, PISYS uses simple, non-intrusive sensors to reason about the presence of persons in particular rooms and calculate probability values with respect to their identities. As part of the project, a home environment simulator was implemented and used to generate train and test scripts. Using this data, we present how the PISYS system can successfully identify people in a range of situations, including multiple people in the same room.

1 Introduction

In this paper we describe our approach in the field of person identification in home environments. This Person Identification SYStem (PISYS) could then be used in Ambient Intelligence (AmbInt) applications. The general idea of AmbInt is to create a supportive domestic system which will help people to feel at home and make them comfortable. One of the important aspects of this is identifying persons and adapt the environment to their preferences, like setting the heating at the right temperature, or switching on the tv on the person's favorite channel.

The field of AmbInt is now one of the hot topics in AI research. Large enterprises like Philips [1], as well as research institutes like MIT [2, 8] conduct research in the field. A key difference between most of the current approaches in person identification and our system is in the types of sensors used. Most researches use complex, expensive and inconvenient sensors like RFID chips, facial recognition, voice recognition or iris scans. These systems most often are seen as intrusive into the person's personal environment [2]. One of the aims that sets this project apart from previous work is that we want to make a system that can identify a person by using non-intrusive, simple and affordable sensors.

These sensors could be placed anywhere around the house to measure basic things like the height or weight of a person, as well as monitor activity (e.g. the operation of appliances, switches etc). By making use of the joint probabilities of persons having certain properties or operating certain things, we will try to infer which person is in which room at which moment. This should result in a system which is almost completely invisible for the people who use it, but still able to assist them in their daily activities.

To solve the problem stated above, we needed a reasoning system that can use the sensor output generated by people in the home environment to calculate probability values with respect to their identities. One of the models that is currently successfully in use in related problems is *Bayesian Networks* [8], and we decided to make use of this method. We also wanted to experiment with a novel method in the field. The idea of solving the problem with autonomous, communicating sensors inspired us to chose the Max-Plus algorithm. Both methods will be explained more elaborately in the next part of this paper.

After the formal explanation of the two methods used, we will continue on the implementation of our system. Here we will mention the design choices we made as well as some methods we used to make the algorithms work for our case. We discuss the design of a home environment simulation engine, as well as explain how we implemented the Bayesian Network model and the Max-Plus algorithm.

In the experiments section, we will present the way in which we tested our system and our findings on the workings of both algorithms for our purpose. At last we will make some final remarks about our system and propose some subjects for future research.

2 Formalization of the reasoning engines

2.1 Bayesian Network

2.1.1 Network Design

Bayesian Networks(BN) are powerful tools to cope with problems of causes and uncertainty. A BN is a directed graph of nodes representing variables. The arcs of the graph represent

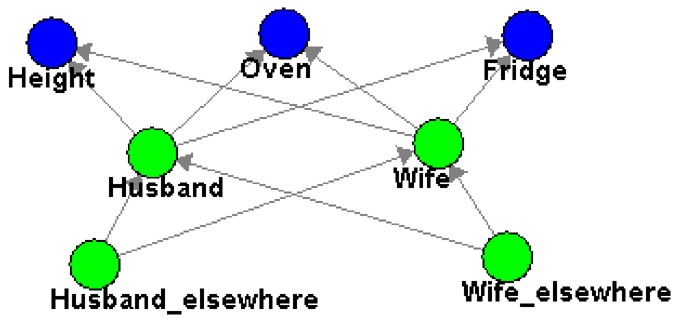


Figure 1: A simple Bayesian Network model of a room

the dependencies between these variables. Nodes that have a known value are called *evidence nodes*. A BN is a representation of the joint distribution over all the variables represented by nodes in the graph. More information on BNs and aspects of their application can be found in [3, 7].

Good design choices might make a certain BN model of a problem robust, easy to train and fast to make inferences on. Complex models that depict all the subtle (but still existing) dependencies of the variables are usually not the best choice. One reason for this is that complex models are difficult to train, as each variable is conditioned on many others and, in order to compute the conditional probability tables, extremely large data sets are needed. Another reason is that the inference procedure on complex BNs can easily get out of hand and become extremely computationally expensive. Finally, complex BNs are difficult to debug and maintain, and usually cannot be automatically generated.

Having these in mind, we designed the model of a house in the following way. A separate BN was created for each room. These networks look like the simplified example in figure 1. The nodes of the network are the variables following the output of the sensors (e.g. the Oven usage sensor), the probability of a person being in the room (e.g. Husband and Wife nodes) and the probability of a person being in another room (e.g. the Husband_elsewhere). In the network design, we can see that sensor nodes are not connected to each other, i.e. the assumption that a sensor's output is independent to other sensor's output. This choice was deliberate, although some might argue that the assumption does not hold (which is true in some cases). The first reason for making this choice is to keep the model simple and avoid having to manually tune it for each application. Because we aim to develop an easy to adapt and deploy scheme for person identification, manually designing associations between each room's sensor should be avoided. Another reason for this choice is to keep the method simple and computationally efficient. Finally, in other studies of similar problems like in [8], it is argued that naïve BNs over sensor outputs such as the one we propose, are equally good at classifying using people behavior as more complicated network designs.

An important aspect that can enhance results in person identification is the reuse of probabilistic information from the other rooms. We introduced the *_elsewhere* nodes, which contain probabilistic information on the presence of a person in another room and form the connecting blocks between the rooms. The reasoning engine fills these probabilities at run time gathering information from other rooms. An example of the usage of this is when there is a doubt between the two persons in figure 1 by using only the sensor readings. Though, if Husband_elsewhere is set to be highly probable, its presence here will be less probable. In practice though, this mechanism proved to be inadequate and caused problems, because of cycles in the procedure. More on this will be discussed in section 5.2.

2.1.2 Training and Inference

All the sensor output is assumed to be discrete. In cases where the sensor output is continuous (e.g. height sensors), their output is transformed to a discrete spectrum, using a series of output ranges. Training in our case must provide the *conditional probability tables (CPT)* for all sensor and people nodes (*_elsewhere* nodes probabilities are set at run time). The way that these CPTs are computed is straight-forward. Our intuition dictates that the probability of performing an activity (and thus firing a sensor) in a house setting is strongly dependent on the time that we spend performing this activity, compared with the other people. Thus, we base the computation of our CPTs on time duration of the sensor states, rather than their number or another metric.

For a person p_1, the prior probability $P(P = p_1)$ of this person being in the room is computed by dividing the total time of this person in the room by the total time spent by all persons in the room. Let $T(P = p_1)$ be a function that returns the amount of time that p_1 is present in a room, then the prior is given by:

$$P(P = p_1) = \frac{T(P = p_1)}{\sum_p T(P = p)} \tag{1}$$

Having computed the priors for all persons in a room, the CPT of their nodes conditioned on their *_elsewhere* parent nodes is given by equally distributing the priors of persons who are known to be elsewhere to the rest.

The training of the sensor CPTs conditioned on the people present in the room is given in the following way: For each sensor state s_1, the conditional probability of sensor's variable S, given the states of parent person nodes X_1,…,X_n, $P(S = s_1|X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$ is computed by dividing the time that the sensor was in this state while these people were in the room, divided by the total time that these people have stayed in the room.

$$P(S = s_1|X_1 = x_1, \ldots, X_n = x_n) = \frac{T(S = s_1, X_1 = x_1, \ldots, X_n = x_n)}{\sum_s T(S = s, X_1 = x_1, \ldots, X_n = x_n)} \tag{2}$$

In run-time, the sensor nodes become *evidence nodes* as sensor output is fed into the BN model. The *Variable Elimination* algorithm is then used to make inferences and compute the marginal probabilities $P(X = x_1|evidence)$ of the persons' variables X.

2.1.3 Smoothing the Probabilities

When working with people behavior, sparse data is always a problem [2]. For example, in the BN architecture mentioned, if a person performs an action that fires a sensor output that was never recorded in the training session (in relation to this person's presence in the room), the person is assigned zero probability of being in the room. A way to resolve this problem is employing smoothing techniques. We implemented a simple smoothing mechanism in the training procedure, which, whereas not sophisticated, highly augmented the accuracy of the results, as will be presented in the experiments section. The smoothing technique used is applying a uniform Dirichlet prior [7], which translates to adding *pseudo counts*, even to

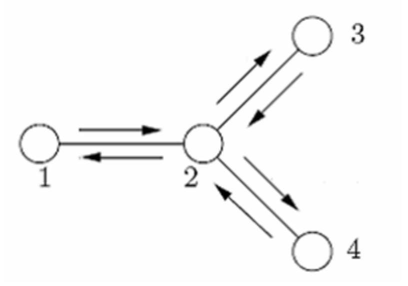


Figure 2: Simple Max-Plus coordination graph

events that have not been recorded during training. Using this technique, the equation (2) changes to:

$$P(S = s_1 | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \frac{T(S = s_1, X_1 = x_1, \ldots, X_n = x_n) + \alpha}{\sum_s T(S = s, X_1 = x_1, \ldots, X_n = x_n) + m\alpha} \tag{3}$$

where α denotes the amount of *pseudo counts* added and m is the count of all s so that the result is normalized. In our implementation α was set to 1 % of the sum in the denominator of the formula plus 1 (so that when the sum is zero, the probability distribution goes to uniform).

2.2 Max-Plus

2.2.1 Introduction to Max-Plus

The second reasoning system we used was the Max-Plus algorithm. One of the differences of this model to the current Bayesian network implementation, is that it assumes the sensors to be able to communicate to each other and reason about their readings[1].

The Max-Plus algorithm has previously been used for decision making in a multi agent system [5]. It calculates the *maximum a posteriori* probability in an undirected coordination graph. It works by iteratively sending messages between agents. These messages can be regarded as locally optimized payoff functions. Using these messages, the agents get to the optimal joint action.

In figure 2, the arrows indicate messages about a certain action being sent between the numbered agents. These messages contain the agent's opinion about whether or not the other agent should execute this action. The higher the value of the reward contained in this message, the more the agent persuades the other agent to execute this action. The method of calculating the values of the messages being sent is discussed in the next section.

Apart from the rewards given by other agents, the agent also computes his own reward for selecting this action. Given the messages received and the value of his own reward, all agents independently select their optimal action. While we want to decide who is where instead of selecting actions, each agent has to select the persons who are at this location at this moment.

In our implementation, we use two kinds of agents. There are location agents which represent each room in the house as well as sensor agents which represent the different sensors in each room.

[1]Networked Intelligent Devices (NID) [6] are an example of such sensors

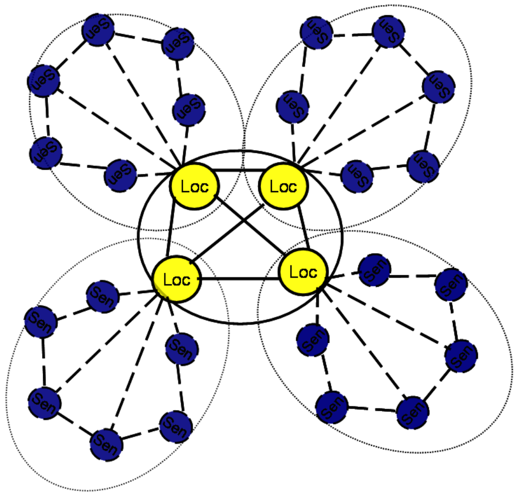


Figure 3: The Max-Plus coordination graph

A novel idea in this research was to divide the communication in two loops. First, there is an ongoing communication loop between all location agents. This should overcome situations where the reasoning engine proposes that one person is in two places at the same time. The second communication loop is between one location and its sensors, it's meant for deciding who is at this specific location. This second communication loop is only needed when new sensor data arrives for this location. An example of this is shown in figure 3. In this figure, the inner circle represents the connected locations, while each outer circle represents a location with its connected sensors.

There are two main advantages of this approach. The first is the fact that it reduces the communication, while not all the agents have to communicate every step. On the other hand, it shows the flexibility of the system. By creating a number of subsystems, the system can easily be extended.

The Max-Plus algorithm is proven to select the optimal joint action in a non-cyclic coordination graph. However, previous experiments [5] also show good results in cyclic coordination graphs.

2.2.2 Formalization of Max-Plus

As described in the previous section, the agents communicate by sending messages. The value of the message $agent_i$ sends to $agent_j$ about person pj is done according to (4). The message sent, should reflect the amount in which $agent_i$ persuades $agent_j$ to select person pj.

The value of the message is the maximized sum of three functions, the OwnReward, the CombiReward and the IncMessage function.

$$\mathbf{Message_{i,j}}(pj) = \arg\max_{pi}\{\mathbf{OwnReward_i}(pi) + \mathbf{CombiReward_{i,j}}(pi,pj) + \mathbf{IncMessage_i}(pi)\} \tag{4}$$

$$\mathbf{Normalize} : \mathbf{Message[i][j][pi]} = \frac{\mathbf{Message[i][j][pi]}}{\sum_p \mathbf{Message[i][j][p]}} \tag{5}$$

$$\mathbf{SelectPerson_i} = \arg\max_{pi}\{\mathbf{OwnReward_i}(pi) + \mathbf{IncMessage_i}(pi)\} \tag{6}$$

The OwnReward function calculates the own reward $agent_i$ gives himself for selecting a specific person. For example a height sensor will give itself a high reward for a person if the sensor reading is the same as the person's length.

To calculate the reward $agent_j$ gets for selecting person pj, in combination with $agent_i$ selecting person pi, the CombiReward function is used. In this function, rules are incorporated like one person can only be in one room at a time and all sensors in one room should choose the same persons.

The IncMessage function sums all incoming messages $agent_i$ received about one person, except for the message received from $agent_j$. This avoids $agent_j$ receiving his own sent message back from $agent_i$.

Because the used coordination graph contains cycles, it is necessary to normalize the sent messages [5]. For this purpose the Normalize function (5) is used.

The structure of the Message function (4) is recursive. This means each message sent will provoke changes in all messages sent by the receiving agents. The communication between the agents will continue until the message don't change anymore. After the communication has finished, every location agent autonomously calculates which persons are in its room according to (6).

3 Implementation

In our implementation we used two separate layers. The first layer is the simulator, which is used to generate data. The other layer consists of two parts, the Max-Plus and Bayesian reasoning engines. Both read the data generated by the simulator. To accommodate this, we designed one MySQL database in which the simulator could save all the training data, as well as the house structure and the household. This enables us to test more than one reasoning engine with the same database.

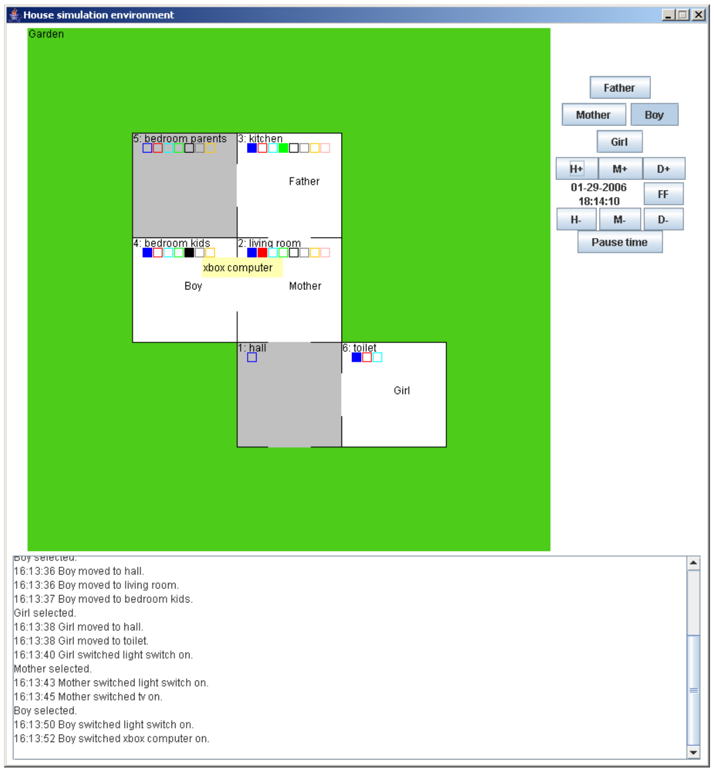
The reasoning engines are created to check the database for changes every once in a while. When a change is noticed they start reasoning about the information gathered and they infer the current state of the system.

3.1 Simulator Environment

Because we have no knowledge of any current projects similar to ours, we faced the problem of gathering data. The project would be far too extensive if we set up a real test environment. This meant we had to develop a simulator which could create the data needed to train and test the system.

This simulator should be able to simulate a house, the sensors in and around the house and the people moving around and interacting with the house. The output of the simulator should be realistic and usable by the reasoning engine.

All data saved to the database by the simulator can be used as a ground truth. This means the reasoning engines can use this data to learn and predict what a certain person is most likely to do.



```
Boy selected.
16:13:36 Boy moved to hall.
16:13:36 Boy moved to living room.
16:13:37 Boy moved to bedroom kids.
Girl selected.
16:13:38 Girl moved to hall.
16:13:38 Girl moved to toilet.
16:13:40 Girl switched light switch on.
Mother selected.
16:13:43 Mother switched light switch on.
16:13:45 Mother switched tv on.
Boy selected.
16:13:50 Boy switched light switch on.
16:13:52 Boy switched xbox computer on.
```

Figure 4: The simulation environment

The simulator has been designed to be as extensible as possible. This means we used a modular structure which can easily be adapted to the specific wishes of different testing environments.

The house consists of a linked list of rooms. Each room has his own parameters for the sensors it contains and people that are in the room. There is also a maximum of four 'doors' for each room, one for each of the four walls.

Each room, by default has a few basic sensors. First of all there are two sensors in each doorway, a height and a weight sensor. When a person moves from one room to another, his height and weight are measured. We also assume each room has a sensor which can compute the number of persons in the room. This last sensor is not really a physical sensor that could be placed in a real room. But we assume that with the combination of certain kind of sensors, like a movement sensor and a height sensor, it could be possible to know how many people are in a room.

At this moment there are six different kinds of sensors. These are the previously mentioned height, weight and number-of-persons sensors together with two cumulative versions of the height and weight sensors and a binary sensor type. This binary sensor can be used to represent any kind of sensor which can be turned on and off like light switches, a tv or drawers that can be opened or closed. Furthermore the physical information about all persons is stored.

In figure 4, the GUI[2] we created to interact with the simulator is shown. In the center the

[2]Graphical User Interface

house with its sensors is shown. Since the simulator has his own time control, time can be manipulated in the simulation environment. As a result of this, the simulator has a day-night cycle, which results in dark rooms at night. Time manipulation makes it easier to create a time-dependent simulation. This means it is easy to create a script in which the father of a family has to go to work at nine, and comes home at five. This can then be used by the reasoning engine to estimate who could be the person coming home at five.

The bottom part of the simulator shows the script being generated while the simulator is used. The text shows all actions that have taken place since the simulator was started. This way all events can easily be retrieved. All actions considering sensor readings are also saved into the database. This final step makes all generated simulation data available for further processing by other programs, like our own reasoning engines.

3.2 Bayesian Network model Reasoning Engine

We implemented the Bayesian Network model of the house as presented in section 2.1, using a mixture of newly developed software models and the BN engine *JavaBayes* [4]. JavaBayes is a software tool in the *Java* language for designing BNs and making inferences using them. By performing minor modifications we used this software package as a run-time inference engine for our model.

The theoretical model itself is engineered in an abstract way into the *HouseIntelligence* software bundle, completely disconnected from our current inference engine (JavaBayes) and data storage design (MySQL database, data tables, etc). Then, a layer of extensions specifically designed for our current implementation provides the interconnection of the system with the inference engine and data storage. This design (seen in figure 5) allows for easy reuse of the theoretical model in other implementations.

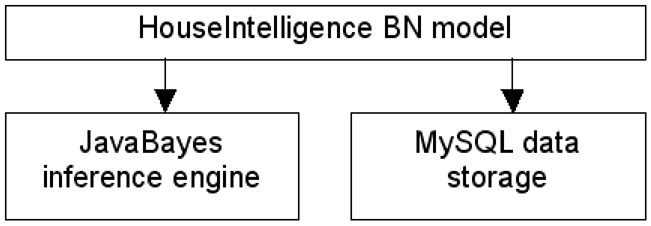


Figure 5: The software design of HouseIntelligence

Implementation parts are provided that model the environment of the house (e.g. sensor, person, room, house, etc.) and dynamically create each BN given the data description of its layout. Furthermore they compute CPTs based on training data, update the model using the stream of sensor output, convert continuous output values to discrete spectrums, perform inference, compute marginal probabilities, etc.

3.3 Implementation of Max-plus

Because we had no knowledge of any existing Max-Plus packages, we wrote our own. The focus of this research was to give a proof of concept for the Max-Plus algorithm to work in the

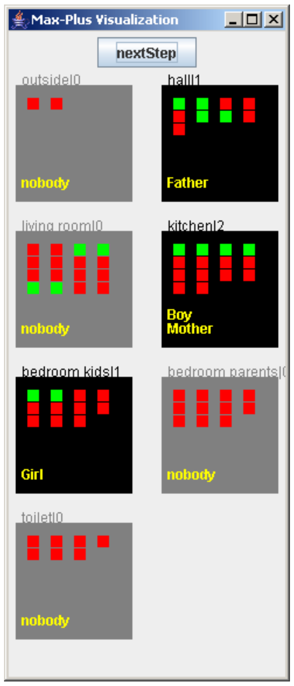


Figure 6: Max-Plus GUI

domain of person identification. For this reason we decided it was unnecessary to implement all sensors.

The algorithm only makes use of the cumulative height and cumulative weight sensors. The number-of-persons sensors are considered as a part of the location agents. To work with the notion of empty rooms, an extra person called 'nobody' is added to the household.

To show the output of the Max-Plus algorithm to the user, we also developed a small GUI which is shown in figure 6. The GUI helps us get more insight into the workings of the algorithm.

The functions OwnReward and CombiReward, are of great importance for the working of the algorithm. As an example the computation of the OwnReward value of a cumulative height sensor is shown below.

```
Pseudo Code for OwnReward(person) for a cumulative height sensor

IF(SensorValue > 0 AND Person = nobody)
    THEN return - 10
ELSEIF(SensorValue = 0 AND Person != nobody)
    THEN return -10
ELSEIF(SensorValue = 0 AND Person = nobody)
    THEN return 10
ELSE
    RETURN 10 - 100*abs(SensorValue-Person.Length)
```

The function above describes the reward a cumulative height sensor will give itself, about a person. For example, if the sensor value is 1.78, and the given person is '*nobody*', this sensor will give a negative reward. However, when the given person is '*father*' who has a length of 1.82, the sensor will reward a value of 6.

The above code will only deal with one person in a room at a time, because the sensor value is only compared to one person's length. Therefore, errors will occur when more than one person is around. As can be imagined, these functions are quite complex to create.

```
Pseudo Code for Max-Plus

WHILE(Messages are not converged)
    FOR ALL Agents ai
        IF(ai == location agent) THEN ai.doCommunicationRound
        FOR ALL Agents aj
        IF(ai == aj || notConnected(ai,aj) THEN continue
            FOR ALL People pj
                Message(i,j,pj)
            END
            Normalize
        END
    END
END
FOR ALL Agents ai
    IF (ai == location agent) THEN ai.selectPerson
END
```

The pseudo code for the Max-Plus algorithm is shown above. At the forth line the communication round between the location and its sensors is initiated. This enables the two loop algorithm we proposed. The algorithm for both loops is the same, however the OwnReward and CombiReward functions for the location agents differ. The goal during the communication between the location and its sensors is to let them all choose the same persons with a high reward. On the other hand, the communication between the locations should make each location select different persons.

4 Experiments

In this part, we will describe the settings we used to test the reasoning engines as well as the results of these tests.

4.1 Definition of Methods & Terms

4.1.1 House and sensors

We created a house with six rooms: a hall, the living room, kitchen, bedroom for the parents, bedroom for the kids and a toilet. The connections between the rooms can be seen in figure 4. Other binary sensors like a microwave, Beertender, coffee machine, washing machine, candy cupboard and freezer are placed in the kitchen. The living room consists of a sensor in the telephone, PC, XBOX computer, radiator, window as well as a sensor in the book shelf, that can 'recognize' if a book is being used or not. The other rooms have similar sensors. The complete structure of the house and its sensors can be found in Appendix A.

4.1.2 Household

We had to create a storyline that represents the family and its behavior. By using this storyline, we can create the training data to train the reasoning engines. It is not useful to generate training data at random, because that would not make any sense for the description of real human behavior. The storyline is also used to create the test data. We used a storyline based on a small family. They are the only people entering and leaving the house.

The household consists of four persons, namely the father, the mother and two kids. Each person has a different weight and height, but the kids have a weight and height that are quite similar. To give an example, the storyline of the father is described below.

> *"The father is usually not at home between ± 8.00 and ± 17.00 o'clock, because he works during the day. Sometimes he has to work on the PC in the living room. He likes coffee, but after drinking coffee, he has to go to the toilet and that takes a while. Also the Beertender is a favorite item of the father; he drinks beer now and then. Especially drinking beer and watching TV is one of fathers favorite combinations. He hates cooking; he leaves that up to his wife (the mother). Father has is own wardrobe in the bedroom. In the morning he uses the razor and the evening he watches TV in the bedroom."*

These storylines are translated to scripts by making corresponding mouse clicks in the simulator. The scripts as well as the storylines used for the description of the household can be found in Appendix B.

Some transitions or sensor activations not described by the storyline were also added to the scripts to make realistic deviations of a person's normal behavior. In total three days of scripts were created.

4.2 Experimental scenarios

As part of the evaluation of our system, we developed a set of test scenarios that model human behavior in the house setting. In most of the common cases, like when a single person moves around the house or two persons are in the same room, both reasoning systems successfully identify these persons.

We noted one difference between the behavior of the two reasoning systems. The BN model is trained using the people behavior data. In this way, the BN model successfully identifies people when their behavior follows the one recorded in the training session. Though, because of the smoothing methods employed, it can also perform reasonably well on many settings not following training patterns. There are still cases when the output is unsatisfactory, like when a person enters a room that he has never entered during the training session. On the other hand, the current implementation of Max-Plus does not use training data and reasons using a comparison of the height and weight readings with the physical data of the people stored in the database. For this reason, it is not affected by the previously mentioned factor of unexpected people behavior.

From the experiments performed, we would like to present three scenarios that highlight different aspects of the system performance. These scenarios are depicted in Figure 7.

- In the first scenario, a single person moves around the house. With this setting, we want to see if the reasoning engines work efficiently in a simple setting, when only height and weight sensor output data are used. Both reasoning systems were able to successfully

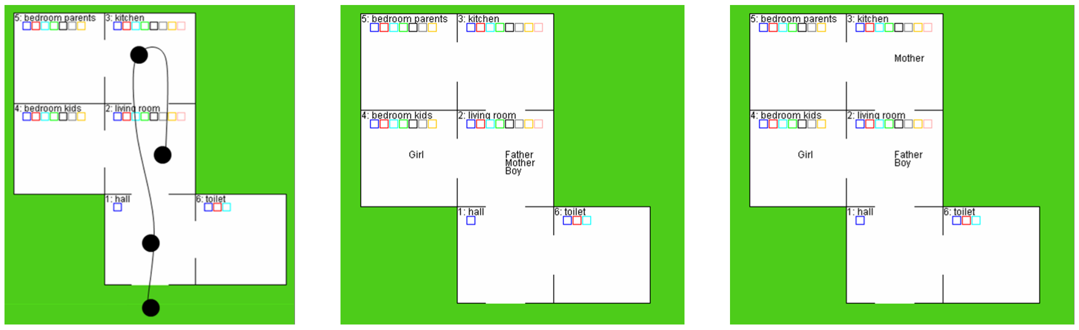


Figure 7: This are some scenarios were we tested the system on. From left to right these are refered to as Scenario 1, 2 and 3 respectively

identify the person with very high probability in most cases. Sometimes, the boy and the girl are confused, because they deliberately have similar heights and weights. In the case of the BN, this confusion is overcome when other sensors are fired, as more sensor output is combined in the inference procedure. In addition, whereas the current implementation of Max-Plus does not use the rest of the sensors to reason, this feature can be introduced to the system.

- In the second scenario, three people are present in the same room. For the BN engine, this scenario presents an interesting situation since the presence of Father, Mother and Boy together in the living room was never recorded in the training session. Still, because of the smoothing technique applied, the reasoning engine can marginally identify the three persons. However, when additional sensors are fired, the engine can deduce with higher confidence the identity of the persons . The Max-Plus engine can also identify the three people. However, when we start moving persons from the room, the engine might be mislead. For example, when the Boy moves out from the living room to the kitchen, the communication between the locations may fail to adapt to the new setting. The Girl is properly identified in the kids bedroom.
- In the third scenario, the Mother moves out of the living room and to the kitchen, leaving the Father and the Boy in the living room. This is another case when something not recorded in the training session happens. This time, the BN engine fails to adapt to the unexpected setting. On the other hand, the Max-Plus algorithm does not face this problem and identifies all people successfully, since it does not rely on training data.

5 Discussion

5.1 Simulator

As mentioned in the simulator description 3.1, it is capable of storing time stamps together with the generated data. This property however, is currently left unused by the reasoning engines. In the future, reasoning engines could make use of this information for extra clues about people behavior.

An addition to the simulator could be to create an interactive interface for creating the simulation environment. Such an interface would enable the user to have an easy way of creating a complete house, including sensors and a household. Another enhancement would be to create an option to save the simulation environment, so a couple of different simulations could be created to test the system on.

5.2 Bayesian Network model

During the experiments performed, the importance of applying smoothing techniques during the training procedure was proven. We believe that gathering enough data to compute probabilities for all possible combinations of people in rooms and sensor output will never be feasible. This problem is also enhanced by the unpredictability of human nature. People do not always do what they are supposed to and this should be anticipated and handled. The smoothing technique employed in this implementation is far from optimal. Still, it made the reasoning engine capable of producing sound results in unforseen occasions and proved the power of the technique. We do think though that more training data combined with a more sophisticated smoothing technique can highly enhance the results in such cases.

A second point worth discussion is the problems encountered while trying to use the *_elsewhere* nodes introduced in section 2.1. What happens is that the use of these nodes in order to combine information from the different rooms creates a cycle in the procedure. In more detail, when the probability of a person being in room A is introduced in a person's *_elsewhere* node in room B, then the person's marginals in room B are changed. This in turn influences the data sent back to room A's *_elsewhere* nodes , which creates a never-ending cycle. Because of this reason, we had to turn off this feature in the final implementation. One alternative to this would be to think of a BN covering the whole house area, instead of separate BNs for every room. For this implementation this was not the choice, since we believe that it could possibly end with a too complicated, slow and difficult to understand model. Though, we do think that an efficient way to link the rooms in one way or another is important. Possibly a combination of the Max-Plus algorithm and the Bayesian Networks could be a solution.

A final note, that is also relevant to the previous paragraph, is the use of other methods, apart from Max-Plus, to enhance the results of BN inferences. A major issue with the BN design that we used is that it does not take the sequence of state changes into account (i.e. it does not understand time). As many human activities are the result of a sequence of actions, it seems as a natural step to use this information to identify people. One way to perform this is by training and using a *Hidden Markov Model (HMM)* or related technique that can model state transitions. Possible candidates for this would be room transitions (as a person moves in the house), sensor output transitions, or even the movement of persons (although this might need a sophisticated object tracking sensor that is not part of the scope of this project). We think for example that an HMM tracking room transitions is directly implementable and could also help to introduce a link between the rooms.

5.3 Max-Plus

One of the goals of this research was to show the capabilities of the Max-Plus algorithm in the world of person identification using a combination of simple sensors. We have shown that Max-Plus is able to do this.

However, the proposed solution could be improved in two ways. The first improvement is to extend the system to use all available sensors. Second, the rules for communication between locations should be revised. It seems quite hard to formulate general rules which work in all cases. We are quite sure these improvements could be made in the future.

Furthermore, research can be done on the combination of different kinds of agents or the design of an adaptive algorithm. For example, Bayesian network agents could be added to the Max-Plus algorithm. Because people have the tendency to change their behavior over time, an ideal system will adapt to these changes.

5.4 Combination of Bayesian Networks and the Max-Plus Algorithm

The main feature of the current implementation of the BN model for person identification is that it forms a complete, trainable, turn key solution, that was proved to work well in most situations. It can readily adapt to any house and sensor setting, and train itself using any available data. The downside of it is the problem of combining the probabilistic information from the BNs of every room, as discussed in section 5.2. On the other hand, the strong point of the Max-Plus algorithm is that it can combine all knowledge from the different rooms in a comprehensive way. However, the functions it uses to compute the reward of a person in a room are rather complex. A lot of time is needed to develop the correct functions, while Bayesian Networks use a straightforward approach for this.

Having these points in mind, we propose as future research, the combination of the BN model for each room with the location agent structure of the Max-Plus algorithm. In this way, a powerful reasoning engine can be formed, that is both adaptable to all settings and can efficiently reuse all probabilistic information to achieve even higher performance.

6 Conclusion

While person identification in home environments is not an easy task, we believe that it forms part of the backbone of Ambient Intelligence applications. In this paper we presented our approach, *PISYS*, which proposes two reasoning systems that can successfully perform person identification in a variety of situations, as highlighted by our experimental results. The addition of an extendable home environment simulator transforms PISYS in a complete package for the task.

The field demands future research and we do not claim of completely 'solving' the problem. Though, overall, we believe that PISYS proves that using simple, ubiquitous sensors, a system can be able to successfully identify people in home settings.

References

[1] E. Aarts. Ambient intelligence: Building the vision. *Philips Research*, 2003.

[2] J. S. Beaudin, E. Munguia Tapia, and S. S. Intille. Lessons learned using ubiquitous sensors for data collection in real homes. *Extended Abstracts of the 2004 Conference on Human Factors in Computing Systems*, pages 1359–1362, 2004.

[3] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[4] Fabio Gagliardi Cozman. Javabayes: Bayesian networks in java, 1998-2001. `http://www.cs.cmu.edu/~javabayes/Home`.

[5] Jelle R. Kok and Nikos Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. *RoboCup 2005: Robot Soccer World Cup IX*, July 2005. Best Scientific Paper Award. To appear.

[6] M. Maris. Lecture of “design of autonomous systems”. University of Amsterdam, 2006.

[7] Kevin P. Murphy. An introduction to graphical models. 2001. `http://www.cs.ubc.ca/~murphyk/Papers/intro_gm.pdf`.

[8] E. Munguia Tapia, S. S. Intille, and K. Larson. Activity recognition in the home setting using simple and ubiquitous sensors. *Proceedings of PERVASIVE 2004*, LNCS 3001:158–175, 2004.

APPENDIX

A House structure

The outside door is attached to the hall. From the hall it is possible to go to the toilet or living room. The living room is connected to the bedroom of the kids and the kitchen. The kitchen is attached to the bedroom parents.

All the rooms in the house have a cumulative weight, cumulative height, light switch and 'number of persons in the room' sensor. Rooms with other sensors attached to the room are printed below.

- Living room - light sensor
  - tv
  - radiator
  - PC
  - telephone
  - window
  - xbox computer
  - bookshelf

- Kitchen - oven
  - microwave
  - freezer
  - Beertender
  - candy cupboard
  - washing machine
  - coffee

- Bedroom kids - Lego box
  - Barby box
  - other toys
  - Xbox computer
  - wardrobe boy
  - wardrobe girl

- Bedroom parents - light switch
  - make-up drawer
  - electric razor
  - TV
  - telephone
  - wardrobe man
  - wardrobe woman

- Toilet - flush toilet
  - washing hands

B Storyline and script

Household consists of four persons. The behavior and the height and weight of the persons is described bellow.

- Father - 1.80cm, 80kg's Not at home between ± 8.00 and ± 17.00 o'clock, because he works during the day. Sometimes he has to work on the PC at home too. He likes coffee, but after drinking coffee, he has to go to the toilet and that takes a while. Also the Beertender is a favorite attribute of the father; he drinks beer so now and then. Especially drinking beer and watching TV is one of fathers favorite combinations. He hates cooking; he leaves that up to his wife (the mother). Father has is own wardrobe in the bedroom. In the morning he uses the razor and the evening he watches TV in the bedroom.

- Mother - 1.50cm, 52kg's
  Not at home between ± 13.00 and ± 15.00 o'clock, because she does some shopping at that moment. The mother is the housewife. She does the cooking and makes breakfast in the morning. She also does the laundry. She doesn't really like coffee, but drinks juice instead. She has her own wardrobe in the parent's room and calls her friends and family multiple times a day. When she goes to the toilet, she mostly doesn't take long. She also cleans up the house, which also means cleaning up everything the children leave behind (clear up toys and switch off stuff left on by the children) In the afternoon, she always goes shopping which will take her an hour or so.

- Girl - 1.15cm, 30kg's
  Not at home between ± 8.30 and ± 16.00 o'clock, because she goes to school everyday. After school she usually plays with her Barbie's or reads a book. She leaves anything behind (= she usually never switches the sensors off). Sometimes she watches TV, but doesn't find this really interesting. She also likes candies, which can be found in the kitchen. The girl has her own wardrobe in the bedroom of the kids.

- Boy - 1.20cm, 35kg's
  Not at home between ± 8.30 and ± 16.00 o'clock, because the boy also goes to school everyday. When he's at home, he plays with the Xbox a lot in his own room, or in the living room. But sometimes he uses the Lego as well. Watching TV is another hobby of the boy. He loves candies too. The boy has his own wardrobe in the bedroom of the kids.

The storylines are translated into scripts. An example of such an script is printed on the next page. At the beginning of the script, everybody was outside

```
Mother selected.
Mouse deactivated because time is paused.
Please unpause time.
Mother selected.
15:06:01 Mother moved to hall.
15:06:11 Mother moved to living room.
15:06:18 Mother moved to kitchen.
15:06:22 Mother switched freezer on.
15:08:35 Mother switched freezer off.
15:08:49 Mother moved to living room.
15:08:58 Mother switched telephone on.
Wrong click.
Mother selected.
15:30:24 Mother switched telephone off.
15:30:30 Mother moved to hall.
15:30:35 Mother moved to toilet.
15:30:36 Mother switched light switch on.
15:33:49 Mother switched flush toilet on.
15:33:51 Mother switched flush toilet off.
15:33:53 Mother switched washing hands on.
15:34:02 Mother switched washing hands off.
15:34:04 Mother switched light switch off.
15:34:05 Mother moved to hall.
15:34:08 Mother moved to living room.
15:34:26 Mother switched tv on.
Boy selected.
16:06:16 Boy moved to hall.
16:06:22 Boy moved to living room.
16:11:36 Boy moved to bedroom kids.
16:11:40 Boy switched xbox computer on.
Girl selected.
16:11:42 Girl moved to hall.
16:11:47 Girl moved to living room.
16:16:55 Girl moved to bedroom kids.
16:16:59 Girl switched barby box on.
Father selected.
17:13:13 Father moved to hall.
17:13:22 Father moved to living room.
17:18:28 Father moved to kitchen.
17:18:30 Father switched beertender on.
17:18:39 Father switched beertender off.
17:18:47 Father moved to living room.
17:19:01 Father switched PC on.
17:25:13 Father moved to hall.
17:25:16 Father moved to toilet.
17:25:17 Father switched light switch on.
17:41:33 Father switched flush toilet on.
17:41:34 Father switched flush toilet off.
17:41:36 Father switched washing hands on.
17:41:40 Father switched washing hands off.
17:41:41 Father switched light switch off.
17:41:43 Father moved to hall.
17:41:45 Father moved to living room.
Boy selected.
17:42:58 Boy switched lego box on.
Mother selected.
17:43:03 Mother switched tv off.
17:43:04 Mother moved to kitchen.
17:43:13 Mother switched freezer on.
17:46:17 Mother switched freezer off.
17:46:19 Mother switched oven on.
17:48:22 Mother switched microwave on.
17:55:47 Mother switched light switch on.
Father selected.
18:06:24 Father switched light switch on.
Boy selected.
18:08:33 Boy switched light switch on.
Mother selected.
18:16:58 Mother switched microwave off.
18:17:07 Mother switched oven off.
Father selected.
18:17:15 Father switched PC off.
18:17:20 Father moved to kitchen.
Boy selected.
18:17:24 Boy moved to living room.
18:17:26 Boy moved to kitchen.
Girl selected.
18:17:30 Girl moved to living room.
18:17:32 Girl moved to kitchen.
Father selected.
19:04:00 Father switched beertender on.
Girl selected.
19:04:05 Girl moved to living room.
19:04:07 Girl moved to bedroom kids.
Father selected.
19:04:09 Father switched beertender off.
19:04:14 Father moved to living room.
Boy selected.
19:04:21 Boy moved to living room.
19:04:24 Boy switched xbox computer on.
Mother selected.
19:04:53 Mother moved to living room.
19:04:57 Mother moved to bedroom kids.
```