International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:4, 2009

Scenario Recognition in Modern Building Automation

Roland Lang, Dietmar Bruckner, Rosemarie Velik, and Tobias Deutsch

Abstract—Modern building automation needs to deal with very different types of demands, depending on the use of a building and the persons acting in it. To meet the requirements of situation awareness in modern building automation, scenario recognition becomes more and more important in order to detect sequences of events and to react to them properly. We present two concepts of scenario recognition and their implementation, one based on predefined templates and the other applying an unsupervised learning algorithm using statistical methods. Implemented applications will be described and their advantages and disadvantages will be outlined.

Keywords—Building automation, ubiquitous computing, scenario recognition, surveillance system.

I. Introduction

TODAY, building automation is mainly concerned with simple monitoring of environments (e.g. temperature) and adjusting them to predefined value ranges targeting comfort and energy preservation. However, as outlined by [8] and [4], in future, this will shift towards applications like safety and self-learning environment control. More and more sensory information will be available for processing. Existing approaches will be challenged by this abundant amount of data. There will be a need for new concepts to handle the challenges of the upcoming future. [5] point out that modern building automation has to deal with very different types of demands, depending on the use of the building (hospital, airport, soccer stadium, office building, etc.) and the persons acting within this building. More and more, ubiquitous computers will become a topic in building automation, to support persons in their actions and with relevant information needed [14].

Nevertheless, without knowing anything about the situation the persons are involved in, the demands that can be satisfied by an implemented system are very limited. Even a simple control variable as simple as the temperature of a room can depend on various other values than only e.g. the time or date. The essential values a building automation system has to be aware of (safety, security, convenience, etc.) can very often depend on a huge variety of diverse sensor information. The data from these multiple sensor sources can be partly redundant, contradicting or inconclusive. The processing of huge amounts of such redundant or ambiguous data requires sophisticated information processing principles and an adequate architecture for information processing [15].

To meet the demands of situation awareness in modern building automation, scenario recognition becomes more and more important in order to detect such demands and react to them [9]. As described in [10], a scenario is a defined sequence of events. To recognize scenarios, information coming from various sensors has to be collected, merged, and interpreted. This process does not only have to be performed for single moment snapshots but also across time.

R. Lang, D. Bruckner, and T. Deutsch are with the Vienna University of Technology, Institute of Computer Technology, Austria.
R. Velik is with Tecnalia - Fatronik, Biorobotics, Spain.
Manuscript received August 12, 2009; revised August 17, 2009.

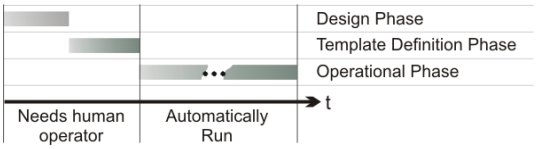

Fig. 1. Three phases of appliance

In the following two chapters two very different approaches of scenario recognition are introduced. The first scenario recognition model is based on predefined perception patterns, called image templates, combines different sensor outputs, and gives them a semantic meaning. In the further, recognized image templates are used as transition conditions between the states of a scenario recognition process based on predefined patterns of possible scenarios. It will be shown how the designed, tested, and implemented scenario recognition model supports the desired demands. The second model for scenario recognition follows an approach based on unsupervised learning of behavior patterns. During a learning phase, the system detects and learns all new scenarios that take place and recognizes them or remarks exceptional scenarios during the operational phase. It learns a set of prototypes for scenarios from the actual data and weights them according to their frequency. In the operational phase, it classifies new data with respect to the prototypes and computes an overall probability for these data. This is very commonly done in modern control engineering in different areas by statistical methods, as most recently described in [13]. Finally the concepts are compared and advantages and disadvantages are shown.

II. Structured Scenario Recognition System

The structured scenario recognition system was designed to detect scenarios that can be predefined before the operational phase is started. As shown in Figure 1, there are distinguished three phases.

The *design phase* is used to predefine the different entities of sensors that are used in the application. During the *template definition phase*, the concept of condensing linked sensor data to symbols and scenario templates using these symbols is predefined. This architecture is sensor independent. By

following the concept of symbolization – defined in [10] – as a standardized interface between sensor data and the structured scenario recognition system, any type of sensor entity can be used.

Based on the level of symbolization, two types of templates are necessary to guarantee scenario recognition: the *image template*, representing a typical set of perceived data within a single moment and second the *scenario template*, representing a perceived sequence in time. Finally, the *operational phase* starts, where the system's output are one or more recognized scenarios. The following sections describe why image and scenario templates are necessary and how they are implemented and defined. The concept of symbolization used is defined in [10] and will not be described here.

A. The Difference between a Play Card and a Royal Flush

During the development of the new scenario recognition model, the authors of this article investigated several models of how the human perceptive system handles data, symbolizes them to a higher semantic meaning, and stores recognized scenarios. The authors came up with a concept using standard poker playing cards as a metaphor for what has been defined as an image template. Like the ace of hearts contains different kinds of information (e.g. color, type, rank, value), several templates of a perceived image were defined that can be compared to the currently ongoing situation. When a specific image template matches the current situation, the state of the system is changed and several actions are taken.

Using the picture of play cards of a standard poker deck, this technical concept was presented to a psychoanalytical advisor – apparently a passionate poker player. Following the explanations of the concept, the advisor summed up: "As a poker player, the fact of looking at an ace of hearts does not mean anything to me. Only if I am in possession of the whole set of cards that complete a royal flush, the ace of hearts becomes meaningful to me".

In the same way as a playing card alone does not have any relevance to a poker game, sensory values are not sufficient to realize modern scenario recognition that needs a broad overview of information, also over larger intervals of time.

B. Image Definition

An *image template* consists of a set of rules defining the perception of specific types of symbols. These symbols can be very simple like the number of persons in a room or the temperature or humidity in a room or they can contain complex information like that a meeting is taking place in the conference room. Because of the generic way of the concept for defining image templates, the name and meaning of symbols shall be abstracted in the further text as symbols S_1 to S_n. As the perception module of such a system produces a constant stream of symbols, every calculation step contains a subset (P) of the set of all symbols (S) including the elements S_1 to S_n.

$S = \{S_1, ..., S_n\}$ ... possible perceived symbols

Name	Symbol	Base
iE		-
iN		iE
iL		iE

Fig. 2. Basic elements in image template definition

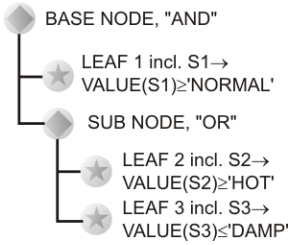

Fig. 3. Example for a generic image template definition accomplished in a tree structure

$P = \{S_3, S_9, S_{17}, ..., S_m\} \subseteq S$ ... perceived symbols

$IT1 = \{w_3 * S_3 \ AND \ w_{17} * S_{17}\} \subseteq P$ ... image template

Since all the definitions of image templates ($_{IT}$) have to be compared with the currently perceived and weighted (w_3 and w_{17} are the weight factors) set of symbols (P), a tree structure is created to define the content of one image template. In Figure 2, the three basic elements of this tree are shown: *image element* (iE), *image node* (iN) and *image leaf* (iL).

As a base class of all elements within the tree, the *image element* holds the following data: For debugging and visualization, every element contains a name and optionally a description of the specific use. Furthermore, every element – no matter if node or leaf – stores the information if this element is optional or mandatory. Additionally, the image node is equipped with the information about its child nodes. These can be further sub-nodes or image leaves. It is also equipped with the boolean composition (AND or OR) between these sub-elements, a negation flag that represents a boolean negation of all sub-elements. The image leaf on the other hand defines the type of symbol that shall be part of the image template detection and a logical operator ($==, !=, <, <=, >, >=$) that compares the value of a symbol to the value defined in this image leaf.

This value depends on the kind of symbol that is generated from the corresponding sensors, which can be e.g. an integer value (counter, temperature, etc.) or a fuzzy value (freezing, cold, warm, hot, boiling in case of temperature) which is mapped to an enumeration. Figure 3 gives an example for a generic image template definition accomplished in a tree structure. Described from bottom to the top, in this image template, either the value of symbol S_2 has to be equal or higher than 'HOT' (as a symbolic output for e.g. a temperature sensor – for whatever that means semantically) OR the value of symbol S_3 has to be equal to or lower than 'DAMP' (as a

MATCH	L1	L2	L3
1.0	X	X	-
1.0	X	-	X
0.5	X	-	-
0.5	-	X	X
0.5	-	-	X

Fig. 4. Possibilities of matches in the example image template

symbolic output for e.g. a humidity sensor).

With this lower part of the illustrated tree, the first half of the image template is defined and matches if the perceived data is within the defined range of values. The second half of the image template matches if the value of symbol S1 equals to or is greater than 'NORMAL'. The image template only matches completely when both conditions – leaf 1 as well as the sub-node – match. This is defined by the logical operator AND in the base node.

C. The Result of the Matching Algorithm

After the comparison of the predefined image template with the actual perceived symbols, a value from zero to one must be generated that describes the quality of the match. To calculate this quality, a simple algorithm is implemented. First, the number of elements within a node, containing the operator AND is determined and summed up with the nodes containing an OR-operator. Applying this algorithm recursively through all branches of the tree, the total weight of the tree is calculated no matter if there is any match or not.

In the example tree of Figure 3, the total weight is two – one for leaf 1 and one for leaf 2 or 3 or both of them. In the next step, there are counted only leaves that have a valid condition match. The number of matching conditions divided by the total weight of the tree gives the quality of the match. Taking again the example of Figure 3, the match would be 1.0 if leaf 1 and one or both of the leaves 2 and 3 match, 0.5 if leaf 1 but neither leaf 2 nor 3 match or only leaf 2 or 3 match. The match is 0 if none of the leaf rules meet the conditions. Figure 4 shows these possibilities without the full and the zero match.

The disadvantage of this algorithm is that every element has the same weight. This can be useful for several applications but driven by a rising demand in different applications, the image element was extended by a weight that can be defined specifically. The algorithm was slightly adapted to the given weight in each node.

As described in [6], the concept of comparing the current perception to a set of templates that have been learned previously or that are predefined is following a bionic approach to find a new way in affective computing as described in [7]. By retracting the layer of image template recognition, a new higher semantic level of symbolization is reached. Based on this level, a scenario recognition was implemented that is described in the following sections.

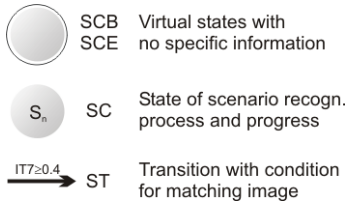

Fig. 5. Basic elements of scenario definition

D. Scenario Definition

Until now, the described concept only dealed with a single moment in time. Every image template and the resulting match did not contain the variable of time. By adding the value of time to the concept, a new step in the hierarchy is made, containing perceived data of the past.

A scenario is defined as a sequence of several recognized images that are perceived (like a royal flush is composed of several play cards). By looking at common sequence diagrams, it was decided to use the concept of state charts to represent such scenarios. From the beginning to the end of a scenario recognition process, several circumstances can occur. For example, there may exist *more than one possibility* for a event that has to be perceived by the image template recognition. *Different paths* have to be covered in the definition of a scenario process. It is necessary to make global *abort conditions* possible, either caused by a timeout or another event that triggers the abortion of the scenario recognition and resets it. Figure 5 depicts the four basic elements of a scenario definition and their purpose: starting scenario state (SCB), ending scenario state (SCE), scenario state (SC) and scenario transition (ST).

The begin and end states only have a virtual meaning in the system. All scenario recognition processes are initially set to the begin state, waiting for the first transition condition to occur. When the end state is reached, the scenario recognition process reaches the end of its lifetime and does not deliver any further information. The scenario states between the begin and end state indicate the process of the scenario recognition defined by their position. Each scenario state (except the end state) holds a list of transitions. These transitions specify the condition when switching to the next state. The match of a selected image template must meet the specified condition. For instance, in figure 5, the image template number 7 must have a perceived match higher than 40 percent.

Figure 6 shows a generic scenario template definition. Here, the scenario recognition process is triggered when the image template (IT) number 1 is perceived with a match of at least 60 percent. The current state is set to the state S_1. This state contains two different transitions. The first transition leads to state S_2 as soon as $IT2$ is perceived with a match of hundred percent. The second transition leads to the state S_3 when $IT3$ is perceived with at least 80 percent. With a match of at least 70 percent, $IT4$ closes the path and the recognition process gets into the state S_2. The final condition for complete scenario recognition is fulfilled when $IT5$ is detected with a match of at least 50 percent.

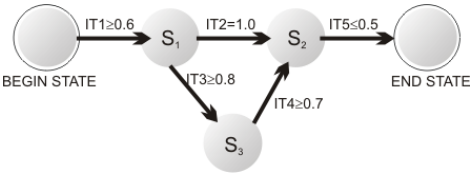

Fig. 6. Generic scenario template definition

The definition of both the image templates and the scenario templates were defined in an XML structured file, representing the knowledge of the system. Images and scenarios that are not defined in this database cannot be recognized.

E. Concept Verificationm and Application

To test the implementation and verify the functional correctness of the image and scenario recognition, a simulation environment was created. This tool offers the possibility to generate symbols and build a stream of perceptions without the necessity of data from physical sensors. The generated data stream was used as an input for the image template matching algorithm. To test the scenario recognition unit, recognized image templates and their match were generated. With this, it could be confirmed that scenarios could be recognized, an abort transition successfully aborted the recognition process, and timeouts lead to an abortion of the process.

The scenario recognition model was implemented in the office kitchen of a research department [3]. Several cheap sensors of different types (tactile sensor, light barriers, motion detector, etc.) were used to get a redundant sensor arrangement. The scenario recognition module was embedded into the system to detect various scenarios happening in the kitchen (meeting, person making coffee, lights left on and nobody present, etc.). Recognized scenarios were visualized on a screen.

During the test phase, the scenario recognition unit was also migrated to a project dealing with simulated, embodied, autonomous agents and their perception and decision units are implemented as outlined by [4]. It was shown that the described scenario recognition model can also cope with these demands, being completely different from modern building automation. Figure 7 shows the visualization of the scenario recognition and the internal values of the embedded autonomous agents at a certain moment of the simulation.

During the test, it was shown that a ”recognition boost” of image templates that are expected by already started recognition processes leads to a better performance in scenario recognition. To ”boost” a recognized image template means that the calculated match is increased by a predefined value which increases the probability that a scenario transition condition is met.

III. Scenario Recognition with Statistical Methods

System that are intended to be aware of the context of persons or systems need the ability to adapt themselves to changing conditions that maybe have not even been foreseen at design time. Therefore, as a supplementary approach to scenario recognition algorithms that rely on pre-defined pattern, an approach based on unsupervised learning [1] of behavior patterns is presented in this section.

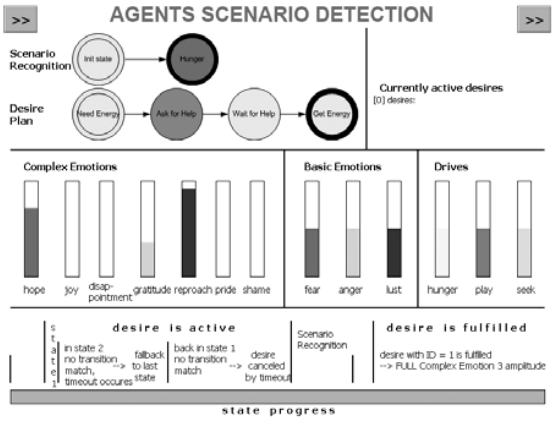

Fig. 7. Visualization of scenario recognition within embodied autonomous agents

Several possibilities have to be distinguished when considering introducing learning methods into scenario recognition: They range from making the definitions of pre-defined scenario building blocks fuzzy in terms of sensor values or allow various sequences of pre-defined events in pre-defined sequence templates to happen and end up with acompletely unsupervised learning of building blocks, sequences, and their interconnections.

In the following sections the principles of unsupervised learning of behavior are presented and a concrete example is given. The ultimate goal of all the algorithms is to create a model of behavior that can be interpreted easily by humans.

A. Scenario Learning Principles

Finding recurring behavioral pattern in sensor data that can be used to model scenarios in a way that humans can interpret heavily depends on the type of sensor data and the characteristics of the data generation.

Therefore, each sensor needs a pre-processing stage to deliver only ”useful” data. For instance *keep alive* messages with identical values in wireless networks or recurring bursts from motion detectors or other presence detection sensors have to be omitted in later learning procedures. Once the preprocessing has been performed for all involved sensor types, no more human operation is necessary in order to let the system learn and later recognize behavior patterns (see Figure 8).

The result of the modeling process is the creation of states. These states are the fundamental building blocks of this kind of scenario learning algorithms. The framework for the model is the hidden Markov model [12]. It offers a set of algorithms for adapting parameters to new data as well as estimating the membership of new data to states in the model.

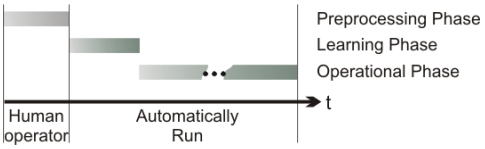

Fig. 8. Three phases of implementation

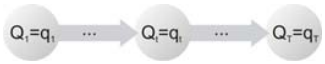

Fig. 9. The Markov chain. The time index goes from 1 to T in case of completed data samples (horizon T)

B. Hidden Markov Model

There exists a variety of models for modeling a data source which is believed to obey the Markov property – new values, no matter if discrete or continuous, somehow depend on old values. In most cases, a first order relationship – that the current value depends somehow on the last one – is assumed. HMMs in particular are used where it is not possible or useful to directly model observation sequences, but rather to model the underlying source for the change in observations. The following paragraphs give an introduction to Markov models and HMMs and their most useful algorithms.

Markov Chain: A discrete time Markov process produces a discrete time Markov chain of values (see Figure 9). The discrete time Markov Chain of 1st order is defined as

$$P(Q_{t+1} = q_{t+1}|Q_t = q_t, Q_{t-1} = q_{t-1}, ..., Q_0 = q_0) = P(Q_{t+1} = q_{t+1}|Q_t = q_t),$$

where Q_t is the random variable at time t and q_t is a variable for some state at time t. This means that the probability for being in some state at some time is only dependent on the previous state.

If the number of possible states in a Markov process (its state space) is assumed to be finite, it can modeled with a Markov model (see figure 10). The Markov model explained here, and all subsequently introduced models in statistics, try to model processes which are assumed to possess the Markov property. Sometimes, this assumption does not hold, but the approximation is still sufficient and convenient. Every state of the Markov model is said to produce some output symbol s, which is an element of the output or symbol alphabet $\sum$. The probabilities of transitions between states – the transition probabilities – together with the Prior distribution π also called initial state distribution vector, formally define a Markov model.

Markov models provide a framework which the user can adapt by supplying transition pdfs (probability distribution functions) of an arbitrary shape. If the basic Markov model is used and by an emission probability distribution to allow the association of a distribution over some output symbols with each state, it is called hidden Markov model HMM (see Figure 11). Additionally adding a duration probability distribution – this allows the modeling of the time, the model stays in a state before leaving – gives a hidden semi-Markov model (see Figure 12). It is also possible to define a model that consists of a Markov model on the top level having HMMs as states.

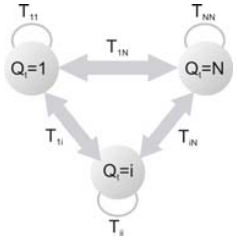

Fig. 10. The Markov model. It consists of N states with a $N \times N$ transition probability matrix T. Depending on the non-zero transitions, the actual state at time t, Q_t can take every value from 1 to N

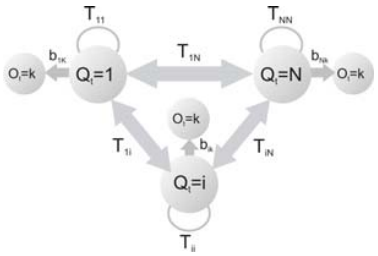

Fig. 11. The hidden Markov model. It consists of N states with transition probabilities between these states, denoted T_{ij} . Each state i also has an emission probability distribution over output symbols b_i. Here, the random variable for the output is depicted as O_t and the random variables for the states as Q_t

Hidden Markov Model: As mentioned above, if the states of a process are directly observable, the process can be modeled with a Markov model. Under certain circumstances, the process that has to be modeled cannot be described sufficiently by a Markov model (e.g., a situation where you can measure – or observe – some value, but you would like to infer from those observations the driving force behind the values). In those cases, hidden Markov models are used. Their states cannot be directly observed, they are hidden. Each state has a probability distribution over some or all possible output symbols. In other words, the hidden Markov model extends the Markov model by emission probability distributions. The complete definition

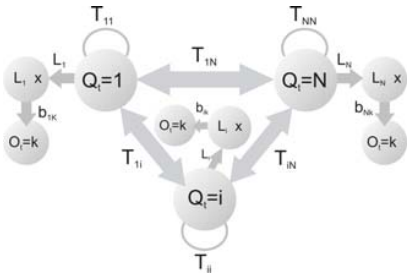

Fig. 12. The hidden semi-Markov model. It consists of N states with transition probabilities, emission probability distributions over output symbols and duration probability distributions that give a number of repetitions every time each state is visited

of a HMM is: A hidden Markov model is a variant of a finite state machine having a set of states Q, a transition probability matrix A, an output alphabet $\sum$, a confusion or emission probability matrix B and initial state probabilities π. The states are not observable and therefore called hidden. Instead, each state produces an output symbol according to the emission probability distribution (B). Characterizing for HMMs are:

- The number of states N
- The number of possible output symbols M
- The transition probability matrix $A = \{P_{ij}\}$, where $P_{ij} = P(Q_{t+1} = j|Q_t = i), 1 \leq i, j \leq N$
- An emission probability distribution in each of the states $B = b_{ik}$, where $b_{ik} = P(O_t = k|Q_t = i), 1 \leq i \leq N, 1 \leq k \leq M$
- The initial state distribution vector $\pi = \pi_i$, where $\pi_i = P\{Q_0 = i\}, 1 \geq i \leq N$

Hidden Markov Model Algorithms: After having selected the HMM to model a specific process, there are three possible tasks to accomplish with it [2]:

- Inferring the probability of an observation sequence given the fully characterized model (evaluation).
- Finding the path of hidden states that most probably generated the observed output (decoding).
- Generating a HMM given sequences of observations (learning).

In case of learning a HMM, *structure learning* (finding the appropriate number of states and possible connections) and *parameter estimatio*n (fitting the HMM parameters, such as transition and emission probability distributions) must be distinguished.

In this scenario recognition application, the following algorithms are used for the three tasks:

- *Forward algorithm:* This algorithm considers problems where different prototypes for the behavior and a sample observation sequence are present and it has to be determined which prototype has the best probability of generating that sequence. Examples in a surveillance application are a set of possible prototype scenarios like "person walking along the corridor" or "person going for lunch". According to a resulting sequence of values, it has to be determines which scenario – and which point in time within the scenario – is the most probable cause for this sequence.
- *Viterbi algorithm:* The Viterbi algorithm addresses the decoding problem. A particular HMM and an observation sequence is given and the most probable sequence of hidden states that produced that sequence is determined.
- *Baum-Welsh algorithm:* This algorithm addresses the third – and most difficult – problem of HMMs: To find a method to adjust the model's parameters to maximize the probability of the observation sequence given the model. Unfortunately, there is no analytical way to accomplish this task. Only local optimization of the probability of the observation sequence can be done of the given models.

C. Applying the Model

Each state comprises an emission distribution (describing its possible sensor values), a transition distribution (describing the connections between states in the model), and – for learning and merging purposes – a weight. The goal is that states represent the behavior of the observed system or person.

Having processed the raw sensor data to represent meaningful statuses or events, a data base with several chains of sensor values forms the base for scenario learning. Ideally, those sensor value chains have their starting point with the start of the desired scenario and their end should coincide with the end of the scenario. Of course, this prerequisite violates the intended unsupervised fashion of learning, but in several cases – e.g., the learning of daily routines or the learning of behavior in a room delimited with opening of the door – the time frame of the scenario (s) is known and can be used as prior knowledge.

After having obtained an initial data set, the value chains are compared. The same scenario is expected to generate roughly the same sensor values in each occurrence and each value chain represents one particular form of the desired scenario. The comparison is undertaken in several steps with the goal to merge the values into states and to reduce the number of states until they reach a degree of expressiveness to be interpreted by a human.

One of the comparison steps looks for equal sensor values at equal times with equal delay times to another value; another step looks for consecutive recurring patterns to merge. For a detailed description of the algorithms see [2].

D. Model Interpretation

For explaining the capacity of this approach, an example located in an office environment is illustrated. The input sensor data come from a motion detector that generates a "1" when it detects motion (at a maximum rate of every five seconds) and a "0" after one minute of no detected motion. For preprocessing the motion detector, values are averaged over 30 min periods in order to represent "more" or "less" activity within these time frames. The model was trained with 15 of those 48 values long sensor value chains. Figure 13 shows the result. The model consists of 14 states (plus initial and final state) as a result of the merging of more than 15,000 sensor values emitted by the motion detector during the observation period of 15 days.

The model identified 6 different daily routines represented by the various paths from the left to the right. One of these paths (0, 1, 5, 13, 4, 15) is discussed in the following paragraph:

Figure 14 shows a path in the graph of the model while figure 11 shows the sensor value chain which created the states and transitions of this path – its Viterbi path. It is important to mention that the graphical representation of the model contains only sparse information about time insofar as the transitions show the possible sequences of states; and states to the right cannot be visited before states to the left.

The paths do not contain information about when during observation the current state of the model changes to the next.

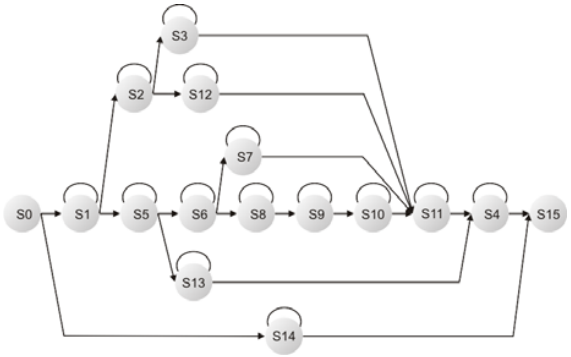

Fig. 13. The model. States are depicted with labeled circles, possible transitions with arrows. The initial and final states (0 and 15) appear at the start and end of every sensor value chain. The ellipses above the states show possible self-transitions

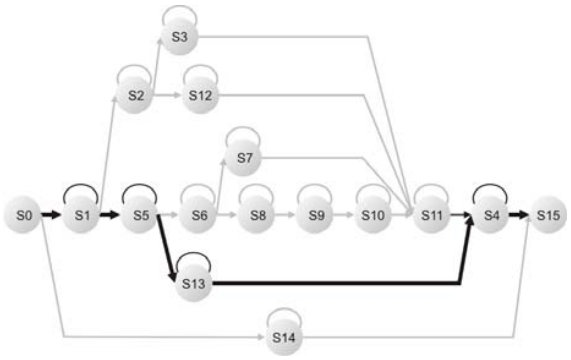

Fig. 14. A path in the learned model. The path (0, 1, 5, 13, 4, 15) represents a particular daily routine from 0:00 to 24:00 o'clock in the observed office

Therefore, the path of states has to be viewed together with the sensor values which created it.

The Viterbi algorithm then determines the sequence of states as can be viewed on top of Figure 15. In this illustration, it is possible to interpret the – fully unsupervised learned – states of the model: State 1 represents the mornings where nobody is in the office and state 4 the evenings. As can be seen in the model graph, all daily routines, except the weekends represented by state 14, share those morning and evening states.

State 5 was a surprise, it represents the cleaning personal! They entered the room nearly every day round 6:00 in the morning and emptied wastepaper baskets, etc. Finally, state 13

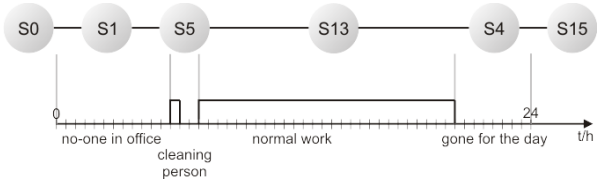

Fig. 15. Sensor values that created the path (0, 1, 5, 13, 4, 15). In the vertical middle of the figure the pre-processed 48 "sensor values" are drawn. On top, the sequence of states that matches best the observed sensor values are shown and, finally, on bottom, the interpretation thereof is given

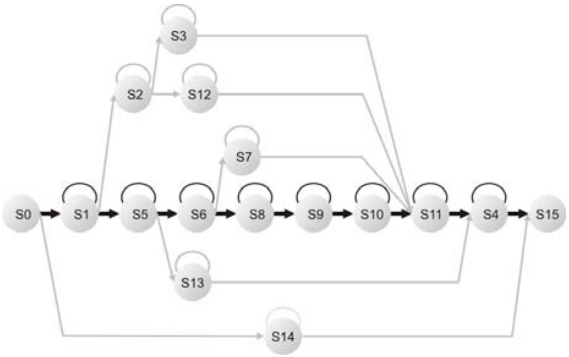

Fig. 16. Another path through the model

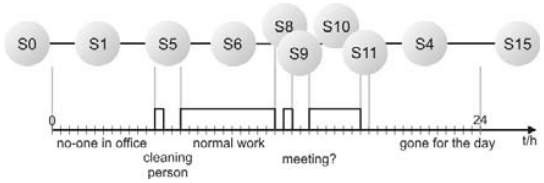

Fig. 17. A day with breaks in activity in the afternoon. Probably a meeting

represents the normal activity in the office which lasts from morning until evening.

The model also saw days with discontinuous appearances of motion. An example is given in Figure 16 and 17 where the afternoon is divided into time frames with and without motion. Each of these time frames are represented by their own states. The model assumes that there is an underlying cause for the change in behavior. In a later parameter update phase, there are at least 2 possibilities for those states and the transitions to them: Either those values were singularities and the probability therefor sinks, or afternoons like this happen more often and maybe persons from that office can interpret these states as their weekly project meeting or the like.

The interpretation of the states 1, 5, and 4 is as it was with the first day, state 6 represents a "shorter" working day and states 8, 9, 10, and 11 cannot be interpreted by us because of lack of particular knowledge, but could represent lunch and meetings.

This little example shows the power of the unsupervised scenario learning approach. It allows learning a model only from sensor data without human supervision in a way that a human operator can interpret the meaning of the model's elements.

E. Application

The described model of statistical based scenario recognition has been implemented in a real time environment within the scope of the project SENSE (Smart Environment for Assisted Living). The goal of this project was to establish security, care, and comfort for elderly people within their homes. Even though elderly people may be handicapped in different ways, it may impossible for them to stay in their usual home instead of moving to a nursery home. However,

specific requirements have to be met so that elderly people can be observed in a way that their daily routines can be supported and safety critical situations can be detected and handled.

Since a private home is not the place for video and web cams, first of all because of the strong request for privacy and second and because of the effort that has to be taken by external human observers or image recognition software, avoiding video observation within the project was mandatory. The system was based on wireless sensor networks to provide an easy and lexible installation within a flat. The sensor network consisted of different sensors, e.g. motion detectors, pressure sensitive switches, simple on/off and touch sensitive switches, and multifunctional sensors for e.g. temperature, light, humidity, etc. Although video cameras were replaced by these simple kinds of sensor types, a surprisingly high amount of information could be extracted to observe safety critical situations in association with the scenario recognitions system based on statistical methods. For example, a pressure sensitive switch was attached to the bed of the room so that the system could detect whether a person lies in the bed or not. Within an automatically learning phase of two weeks that has been declared as 'normal living' without any exceptions, the system learned when the inhabitant of the flat usually wakes up during the week and on weekends. During the operational phase, the system would detect an abnormal behavior if the person stays in bed until ten in the morning although the normal wakeup time was 9 AM and would start an alert procedure. Within the scope of the test applications, the system did support the nursing stuff and was strongly accepted by the elderly people.

IV. Model Comparison

The two models described in this article use different approaches for achieving the same goal. Perception of everyday activities and crude "understanding" of what is happening is vital for future automation systems. Both approaches have certain advantages and a comparison has to consider the different underlying methods and the different applications for the models.

The structured scenario recognition system bases on predefined scenarios and can therefore provide additional semantic information about the perceived scenarios. This is important if a human operator needs to evaluate the information that the system has perceived. In cases where privacy is essential, this additional information, which edited before the operational phase, greatly assists the operator. Privacy is one of the main concerns, when persons with special needs are observed: in retirement homes or in hospitals, such a system can be well integrated and provide constant 24-hour surveillance and only alarm a human user if it detects a predefined scenario that requires assistance. The disadvantage of the model is the initialization phase. Although many scenarios can be predefined and do not change from one installation to another, it will still be necessary to do adaptations depending on the layout and situations at a new venue. This may increase the costs of the whole system.

In constrast, the scenario recognition with statistical methods does not suffer from a long and costly initialization phase. It automatically adapts to the incoming sensor information and learns to tell apart usual from unusual situations. It is in principle possible to assign semantic information to a model the system has learned. A disadvantage is that since the associations between sensor values are learned completely autonomously, the system has no means to provide information about the scenarios it has detected. It can only provide information whether a scenario is unusual. An experienced operator can use this information and attempt to assign "meaning" to the scenarios. However, this task is subject to personal interpretations.

It appears that the best solution is a combination of both systems. This way, the system is equipped with basic knowledge of its whereabouts, but is still flexible enough to adapt to specific situations.

V. Conclusion and Outlook

This article introduced two models for scenario recognition – one based on predefined scenario patterns and one based on unsupervised learned patterns. The work bases on the research results of [11].

The model of the structured scenario recognition system showed an approach for detecting predefined scenarios in sensor-equipped buildings or autonomous agents. After predefining templates of detected data and expected scenarios within an XML knowledge base, the operational phase of scenario recognition is started. The output of such a system are recognized scenarios. On this basis, further processing e.g. taking according actions can be performed. Learning of new scenarios during operational phase is not foreseen. An option is to extend the knowledge base by a human operator.

The model of scenario recognition with statistical methods realizes the unsupervised learning of scenarios during a learning phase and therefore minimizes the effort of adapting the system to new surroundings. The semantic meaning of automatically learned scenarios does not emerge during the learning or operational phase, except if an operator filters redundant or meaningless scenarios and adds a semantic meaning to the detected scenarios.

The two models of scenario recognition are currently implemented in a test run in a modern kitchen and in the living rooms of a home for elderly people both being equipped with a range of different sensor types. While the equipped modern kitchen has to be considered as a prototype implementation that is running as a long time try to be able to collect and verify data for further research, the rooms for elderly people are applied to this real world environment. Here, elderly people are supported in their daily life and observed without affecting their privacy. Looking in this direction, further applications in the field of safety critical surveillance systems (e.g. security in public spaces, airports, modern soccer stadiums) are planned.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, New York NY: Oxford University Press Inc., p. 20, 1995.
[2] D. Bruckner, *Probabilistic Models in Building Automation – Recognizing Scnearios with Statistical Methods*, Ph.D. Thesis, Vienna University of Technology, 2007.

[3] W. Burgstaller, *Interpretation of Scenarios in Buildings*, Ph.D. Thesis, Vienna University of Technology, 2007.
[4] T. Deutsch, R. Lang, G. Pratl, E. Brainin, S. Teicher, Applying Psychoanalytical and Neuro-Scientific Models to Automation. *Proc. International Conference on Intelligent Environments*, pp. 111-118, 2006.
[5] D. Dietrich, G. Russ, C. Tamarit, G. Koller, M. Ponweiser, M. Vincze, Modellierung des technischen Wahrnehmungsbewusstseins fr den Bereich Home Automation, *e&i*, Vol. 11, pp. 454-455, 2001.
[6] M. Dornes, *Der kompetente Sugling – Die prverbale Entwicklung des Menschen*, Fischer Taschenbuch Verlag, 2001.
[7] R.W. Picard R. W, *Affective Computing*, The MIT Press, 1997.
[8] G. Pratl, P. Palensky, The Project ARS – The Next Step Towards an Intelligent Environment, *Proc. International Conference on Intelligent Environments*, pp. 55-62, 2005.
[9] G. Pratl, W. Penzhorn, D. Dietrich, W. Burgstaller, Perceptive Awareness in Building Automation. *Proc. International Conference on Computational Cybernetics*, pp. 259-264, 2005.
[10] G. Pratl, *Processing and Symbolization of Ambient Sensor Data*, Ph.D. Thesis, Vienna University of Technology, 2006.
[11] G. Pratl, D. Dietrich, G. Hancke, W. Penzhorn, A New Model for Autonomous, Networked Control Systems, *IEEE Transactions on Industrial Informatics*, Vol. 1, Issue 3, pp. 21-32, 2007.
[12] L. R. Rabiner, B. Juang, An Introduction to Hidden Markov Models, *ASSAP Magazine*, Vol. 3, pp. 4-16, 1986.
[13] R. Rakotomamonjy, R. Le Riche, D. Gualandris, and Z. Harchaoui, A Comparison of Statistical Learning Approaches for Engine Torque Estimation. *Control Engineering Practice*, Vol. 16, Issue 1, pp. 43-55, 2007.
[14] E. M. Tapia, S. S. Intille, K. Larson, Activity Recognition in the Home Using Simple and Ubiquitous Sensors, *Pervasive*, pp. 158-175, 2004.
[15] R. Velik, G. Pratl, R. Lang, Multi-Sensory, Symbolic, Knowledge-Base Model for Humanlike Perception, *Proc. International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, pp. 273-278, 2007.