# Probabilistic Construction of Semantic Symbols in Building Automation Systems

Dietmar Bruckner
Institute of Computer Technology
Vienna University of Technology
1040 Vienna, AUSTRIA
*bruckner@ict.tuwien.ac.at*

Brian Sallans
Information Technologies
ARC Seibersdorf research GmbH
1220 Vienna, AUSTRIA
*brian.sallans@arcs.ac.at*

Gerhard Russ
Information Technologies
ARC Seibersdorf research GmbH
1220 Vienna, AUSTRIA
*gerhard.russ@arcs.ac.at*

***Abstract*— A method for the automatic discrimination of sensor and system behavior and construction of symbols with semantic meaning in a building automation system is introduced - SCRS (Semantic Concept Recognition System). The automated method is based on statistical modelling of sensor behavior. A model of a sensor's value sequences is automatically constructed. The model's structure and parameters are optimized using an mini-batch model merging and parameter updating algorithm. Incoming sensor values are then conveyed to the model and the most probable path through the model to some particular state is computed. That classification can be interpreted as a semantic symbol or concept. The SCRS can be used as a security and care system for observation of persons and interpretation of scenarios. An example modelling a motion detector is discussed.**

## I. INTRODUCTION

Today's building sensor and control systems are primarily based upon the processing of sensor information using predefined rules. The user or operator defines, for example, the range of valid temperatures for a room by a rule - when the temperature value in that room is out of range (e.g. caused by a defect), the system reacts (for example, with an error message). More complicated diagnostics require an experienced operator who can observe and interpret real-time sensor values.

We will focus on "security and care systems" where the system includes observations of persons and interpretation of scenarios. Such a system could be used for understanding the daily routine in an office or care institution for example. In this case either rules would have to be defined which are few and broad, or a large number of possibilities would have to be specified. However, as systems are targeted more and more towards the end-user in home environments, both possibilities (rule-based systems and expert users) become problematic. The security and care system would require comprehensive prior knowledge of possible operating conditions, ranges of values and possible dangerous conditions. This knowledge will either not be readily available or will be difficult for an unsophisticated user to input. It is also not affordable to have each system customized by an experienced operator during a longer initialization phase in the particular environment.

Future building automation or habitation assistance systems for care and security applications will act based on their awareness of system status (see [1], [2]) and user behavior and preferences. Therefore efficient and reliable scenario detection, pattern recognition and tracking systems and algorithms have to be designed. Automation system awareness is a complex and interdisciplinary field. One step towards inferring the intelligence of systems is to define and recognize simple recurring scenarios with slightly different sensor values. In this context we divide a scenario into semantic symbols or concepts. A semantic concept represents a system or user event that caused the automation system to change its beliefe in what is happening. For example there are different symbols for a normal daily routine in an office environment and an afternoon that is interrupted by a meeting.

The goal of this work is to accomplish the task of reliably condensing simple semantic concepts out of sensor data with probabilistic methods without the need of pre-programmed conditions, user-entered parameters, or experienced operators. The system observes sensor data over time, constructs a model of "normal sequences", and compares and classifies newly arriving sensor values. The result is a system that can produce concepts with semantic meaning of sensor readings, with minimal manual configuration of the system. Further, if sensor readings vary or drift over time, the system can automatically adapt itself to the new "normal" conditions, adjusting its parameters accordingly.

The system is part of the ARS project (Artificial Recognition System) [3]. It was tested in a building automation system consisting of various sensors installed in one floor of an office building. The sensors comprise motion detectors, door contacts, temperature, brightness and humidity sensors. The data was collected over a time period of five months. This paper presents the first results of this trial and suggests future areas of improvement and application.

## II. BACKGROUND

The symbol generation system is based on a framework of statistical "generative" models (SGMs). Statistical generative models attempt to reproduce the statistical distribution of observed data. The model can then be used to determine how likely a new data value is or to "generate" new data (i.e. draw samples from the model).

The symbolization system uses a number of different SGMs to capture the distribution of sensor data, including histograms, Gaussian mixture models [4], and hidden Markov models (HMMs) [5]. The latter can be

1-4244-9701-0/06/$20.00 

seen as a framework in a sense that each HMM has probability distributions that have to be filled with other SGMs. An HMM consists of a number of states with transition probabilities between them. The HMM also has emission probabilities, either specific for each state or for each transition between states. The distributions of those probabilities (and also of duration probabilities for states in case of more complex hidden semi-Markov models [6]) are modeled using lower level SGMs.

Previous work in statistical process monitoring have been applied to a number of systems including chemical processes monitoring [7], monitoring of engines [8], and monitoring of communications networks [9]. Typical statistical methods used include principal components analysis and partial least squares models fit to historical batch [7] or online data [10].

In comparison to previous process monitoring methods that recognize variations in different values, we understand sensor data processing as an event driven type of processing and model the transitions of events. We assume particular events to be the underlying cause of change in system behavior or status. Events are represented by so called symbols or concepts. With this nomenclature we want to express, that the Semantic Cconcept Recognition System (SCRS) encapsulates concepts in symbols analogously to the way that languages encapsulate concepts in words.

For convenience we posit that the transitions between events exhibit the Markov property - that the immediately next temporally consecutive event depends only on the previous $n$ one(s). If the current state depends only on the single past state, we have a first order Markov process. HMMs for system analysis have been used for automated process discovery in software [11] and software self-awareness [12].

Despite their success in other domains, SGMs and especially HMMs have hardly been applied to system monitoring and error prediction for building automation sensor and control data. There are two reasons for this. First, it has only recently become possible (and economical) to collect a wide range of cross-vendor sensor data at a central location in buildings. Second, most algorithms to optimize the structure and parameters of SGMs are quite computationally intensive. Many algorithms have been of only theoretical interest, or are restricted to small toy problems. Only recently have powerful approximation algorithms and powerful computers become available that can handle large quantities of data in real-time.

## III. SYSTEM STRUCTURE LEARNING

The goal of the SCRS model is to automatically discriminate system behaviour in a running automation system. It does this by learning about the behavior of the automation system by observing data flowing through the system. The system builds a model of the sensor data in the underlying automation system, based on the data flow. The model comprises not only SGMs describing the possible output of the sensors but also a model of the underlying events of change in system behavior that produce varying input for the sensors (see Fig. 1). From that model, the system can identify recurring scenarios with slightly varying sensor values represented in the SGMs by emission distributions. The system is also capable of launching an alarm in the case of new scenarios or variations within scenarios with very low probability under the model.

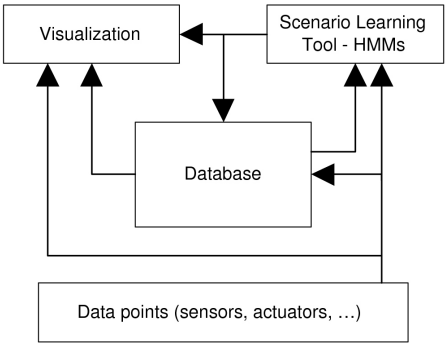


Fig. 1. The Automation System. Sensor and actuator values are stored in a database, and simultaneously analyzed on-line. Connections between the diagnostics system and other system components are either direct, or via the Database.

We use a set of statistical generative models to represent knowledge about the automation system. A statistical generative model takes as input a sensor value, status indicator, time of day, etc., and returns a probability between zero and one. Additionally, HMMs can be queried to deliver the most probable path to the current state. In our case the state that the model is in can be interpreted as a concept with semantic meaning.

Using SGMs has several advantages. First, because the model encodes the probability of a sensor value occurring, it provides a quantitative measure of "normality", which can be monitored to detect abnormal events or jumps between scenarios. Second, the model can be queried as to what the "normal" state of the system would be, given an arbitrary subset of sensor readings. In other words, the model can "fill in" or predict sensor values, which can help to identify the source of system behavior, the underlying event. Third, the model can be continuously updated to adapt to sensor drift or to slightly changing operation of the system.

### A. Statistical Models

For the application described in this paper, HMMs were used with either mixture of Gaussians models or histograms as emission probability distributions.

*1) Modelling process:* The SCRS uses SGMs of automation data points. For any given data value $x$, a model $M$ assigns a probability to $x$: $P_M(x) \rightarrow [0, 1]$.

Note that, for discrete distributions such as a histogram, the value assigned to $x$ by the model $P_M(x)$ is a well-defined probability, since the set of possible assignments to $x$ is finite. For a probability density, such as a Gaussian or mixture of Gaussians, the probability value assigned to $x$ by the model is the probability *density* at that value. In order to convert this density

to a probability, the probability of generating a value within a neighborhood $\pm\delta$ around $x$ is computed as $\int_{-\delta}^{\delta} P_M(x+\phi)d\phi$, and approximated as $2\delta P_M(x)$ for small $\delta$. Alternatively the probability under the model of equaling or exceeding the observed value can be computed: $P_M(x' \geq x) = \int_{\phi}^{\phi+\infty} P_M(x+\phi)d\phi$.

The data $x$ can be a sensor reading such as an air pressure sensor, contact sensor, temperature sensor, and so on. Given a new data value, the model assigns a probability to this value and uses the value to somewhat adjust its parameters.

Given a sequence of sensor readings $\mathbf{x} = \{x_1, ..., x_T\}$ from times 1 to $T$, the SGM must model the sensor readings. The model uses an mini-batch version of the expectation maximization algorithm for maximum-likelihood parameter estimation. Given a model $M$ with parameters $\theta$, the log-likelihood of the model parameters given the data $\mathbf{x}$ is given by:

$$\mathcal{L}(\theta) = \log P_M(\mathbf{x}|\theta) \tag{1}$$

where the notation $P(\mathbf{x}|\theta)$ denotes the conditional probability of $\mathbf{x}$ given the current values of the parameters $\theta$.

The maximum-likelihood parameters are defined as the parameter values which maximize the log-likelihood over the observed data:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \{\log P_M(\mathbf{x}|\theta)\}$$

*2) On-line and mini-batch Parameter Updates:* In order for the system to continually adapt the model parameters, the parameter update algorithm must incrementally change the parameters based on newly observed sensor values. Such "on-line" updates have the advantage that there is no time during which the system is in an "optimization" phase, and therefore unavailable for diagnostics. For the mixture of Gaussians model, the SCRS system uses a simple stochastic estimation method, based on an expectation-maximization algorithm. As each new data value $x_i$ is observed, the parameters are adjusted in a two-step process. First, the posterior probability of each element of the mixture given the data value is computed. Second, the parameters are adjusted so as to increase the expected joint log-probability of the data and the Gaussian mixture component. See [4], section 2.6, for details. The mini-batch version of the expectation-maximization algorithm waits for a number of sensor values and learns them in one step. This version is especially useful when learning the structure of the HMM for value sequences of time series like whole days. We discuss this in depth in the results section.

### B. Hidden Markov models

HMMs 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 sections give an introduction into HMMs and their most useful algorithms.

*1) Markov chain:* The discrete time Markov Chain of $1^{st}$ order is defined as follows:
$P(Q_{t+1} = q_{t+1}|Q_t = q_t, Q_{t-1} = q_{t-1}, ..., Q_0 = q_0)$
$= P(Q_{t+1} = q_{t+1}|Q_t = q_t)$, 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.

*2) Markov model:* If the states of a process are directly observable, then we can model the process with a *Markov Model*. In that case every state is said to produce some output symbol, which can directly be observed and which is dedicated to its state. The probabilities of transitions between states - the transition probabilities - together with the Prior distribution, also called initial state distribution vector, formally define a Markov Model. Markov Models can be arbitrarily complex because they just define a framework that the user has to fill with probability distributions. If one uses the basic Markov model and extends it by an emission probability distribution, it is called a Hidden Markov Model, additionally adding a duration probability distribution gives a Hidden semi-Markov Model.

*3) Hidden Markov model:* Under some circumstances the process we want to model is not described sufficiently by a Markov Model. Consider 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 that cases, *Hidden Markov Models* are used. The difference to the above mentioned Markov models is that the 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 of a Hidden Markov Model is given below:
A Hidden Markov Model is a variant of a finite state machine having a set of states $\mathbf{Q}$, a transition probability matrix $A$, an output alphabet $\mathbf{\Sigma}$, a confusion or emission probability matrix $B$ and initial state probabilities $\Pi$. The states are not observable and therefore called hidden. Instead, each state produces some output symbol according to the emission probability distribution $(B)$. Characterizing for HMMs are:

- The number of states $N$
- The number of output symbols in the output alphabet, $M$
- The transition probability matrix $A = \{p_{ij}\}$
  $$p_{ij} = p\{Q_{t+1} = j|Q_t = i\}, 1 \leq i, j \leq N,$$
  $Q_t$ being the current state at time $t$. The transition probabilities of course must match the two normal statistical constraints
  $$0 \leq p_{ij} \leq 1, 1 \leq i, j \leq N$$
  and
  $$\sum_{j=1}^{N} p_{ij} \ = \ 1, \ 1 \leq i \leq N$$

- An emission probability distribution in each of the states $B = \{b_{ik}\}$
  $b_{ik} = p(O_t = k|Q_t = i), 1 \leq i \leq N, 1 \leq k \leq M,$
  $O_t$ being the output symbol at time $t$. Again, normal stochastic constraints have to be considered.
- Finally, the initial state distribution vector $\Pi = \{\pi_i\}$
  $$\pi_i = p\{Q_0 = i\}, 1 \leq i \leq N,$$
  which elements also have to be positive and sum to unity.

### C. Hidden Markov model algorithms

After having selected the HMM to model a specific process, there are three possible tasks to accomplish with the model.

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

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

*1) Forward algorithm:* Consider a problem where we have different models for the same process and a sample observation sequence and want to know which model has the best probability of generating that sequence. This task is accomplished by the *Forward Algorithm*. As an example in the ARS application we will have a set of possible semantic concepts like "person walking" or more abstract "normal day in an office". Given a sequence of values we want to know which symbol could be the most probable cause for the sequence we see.

*2) Viterbi algorithm:* The Viterbi algorithm addresses the decoding problem. Thereby we have a particular HMM and an observation sequence and want to determine the most probable sequence of hidden states that produced that sequence.

*3) 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. All we can do is locally optimize the probability of the observation sequence given different models.

## IV. SYSTEM STRUCTURE

In this work we present a system that learns semantic symbols from a binary motion detector sensor. That sensor - a wireless off-the-shelf X10 branded motion detector sensor - sends a data packet with value 1 and ID in case of detected motion. When the sensor permanently detects moving objects it sends packets at most every five seconds. After detecting no moving object for more than 1 minute the sensor sends a packet with value 0 and ID. In our office environment every sensor produces about 1000 data packets per day. We supply the SCRS not directly with the sensor values, but with averaged sensor values. We divide the 24 hours of a day into 48 time slots, each 30 minutes long. In those time slots we compute the mean of the sensor values and round them. If no value is available during 30 minutes we set the mean to 0. The chains of 48 values are then fed into the (empty) model and during a procedure of 3 steps the structure of the model is learned:

1) Comparison of the chain's beginning/end
2) Merging of identical states
3) Merging of consecutive states

These 3 steps in combination with the averaging of the sensor values produce HMMs with a manageable number of states. The number of states of HMMs is a compromise between generalization (low number of states, the model is applicable for a wide range of different scenarios, but not able to distinguish between particular ones) and specialization (rather high number of states, not every possible scenario is depicted in the model and quite similar scenarios can have different paths). [13] have shown that model merging from very special models always ends up with better or equal results than specializing very general models. They also introduced a method for merging models and maximizing the posterior probability of the model. The starting point for model merging is to find an application specific heuristic to reduce the number of states dramatically, because the computational effort for their proposed best-first model merging is relatively high. In our application we waive the best-first model merging because the below stated application specific heuristics work fine in case of motion detector sensors.

### A. Comparison of the chain's beginning/end

This part of the algorithm is responsible for comparing the "sensor values" at start and end of the chain (see Fig. 2). As long as the new values match the emissions of the model, we just update the weight of the state in the model. This procedure is accomplished for both the first states in the model and the first new values and the last states of the model and the last new values. This means that for learning the structure of the model some completed sequences of whole days have to be available. The aforementioned weight of the state is important for later merging and will be explained in depth later on.

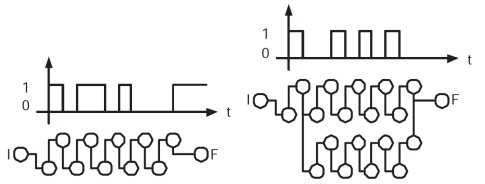


Fig. 2. Comparison of chain borders. The top sequence of sensor values is mapped into a chain of states with appropriate emissions and transitions with 100% from state to state. The next sequence of sensor values is compared to the already seen emissions and in case of different values the model splits and introduces a new path. This is done in forward and backward direction.

### B. Merging of identical states

In case of binary sensors or sensors with discrete sensor values it can happen that a sensor emits a sequence of identical values, so that chains of identical states appear. For modelling the events behind system behavior we can assume that either nothing or the same event happens when a sensor continously emits the same value. The motion detector in our application send every 5 seconds a new value "1" when it sees objects moving. For scenario detection purposes it makes no difference if that motion takes 15 times 5 seconds or only 11 times 5 seconds. This considerations lead to the heuristic of merging identical states. Thereby chains of states with 100% transitions from one to the consecutive next state and identical emissions are sought. These states are then merged into a single one as shown in Fig. 3. The "self-transition" is computed as $T_{ii} = N/(N+1)$, $N$ being the number of states merged. It is important to note that this way of creating the new state produces a geometric duration probability distribution. If it is desired to use a different duration probability distribution, HSMMs must be used.

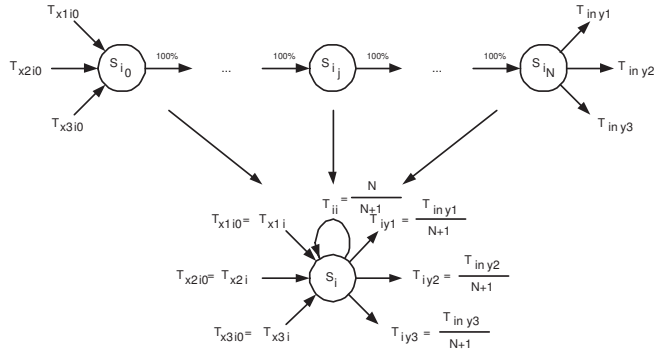


Fig. 3. Merging of a state chain part with n identical states into one single state i. The original chain has unity transitions. x's and y's being other states in the HMM.

### C. Merging of consecutive states

In the model's learning phase a model with initial state, final state and a number of paths in between is created. Each of the paths has the potential to be a scenario and each state has the potential to represent a semantic concept. Each splitting in a path is interpreted as a change in system behavior. During model merging each of those paths has to prove its value for the model. We assume that scenarios can vary their order. This means that values in a chain can change their place with other values that happen often at the same time. These considerations lead to the third heuristic: states with transitions of 100% are merged into a single state. If, after the merging of identical states, chains of states with unity transitions remain, then those chains are merged (see Fig. 4).

## V. RESULTS

The semantic concept construction system was used to analyse sensor values from a building automation system in an office environment. In this paper the results of modeling a motion detector sensor are presented.

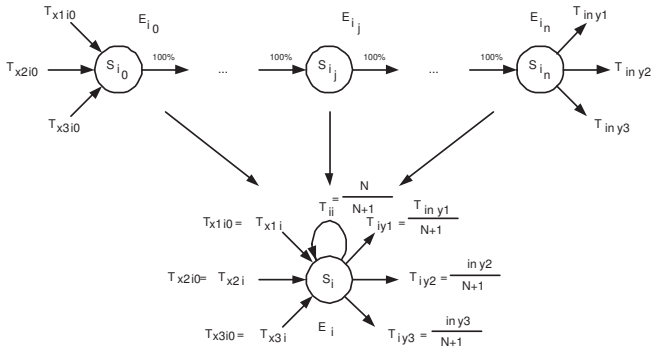


Fig. 4. Merging of a state chain part with states with unity transitions in between into one single state i. The new Emissions $E_i$ is computed as Emissions of original states times state's weight normalized to sum to unity.

### A. Model Construction

The model consists of a HMM as a framework, emission and transition probability distributions being multinomial probability distributions. The motion detector sensor emits approximately 1000 values per day. Those were averaged during 30 min periods. If no value was emitted, 0 - being the value for "no motion" - was assumed. The obtained 48 "sensor values" were then fed into the model. As described in section IV for the first 15 days state chains are created. Afterwards, the two types of merging were applied. Fig. 5 shows the result. In this example, on weekends no sensor value was emitted. In this cases we decided not to input a chain of 48 0's, but an empty chain. Those weekend days are represented by the transition directly from the initial to the final state.

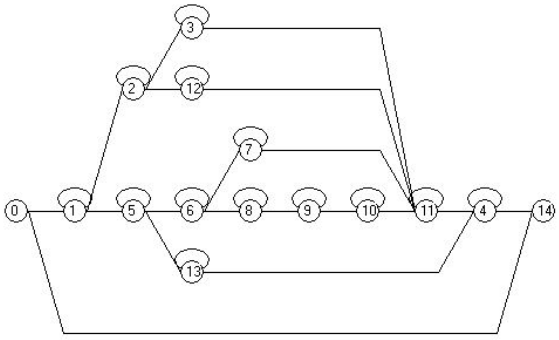


Fig. 5. The model. States are labeled with numbers, 0 being the initial state and 14 the final one. The initial and final states appear at the start and end of every sensor value chain. The ellipses above the states show the self-transitions. Ellipses and lines represent transitions with a probability greater than 0.

### B. Model Interpretation

In this model every path through the model represents a particular daily routine. In this context we can talk of a semantic concept for a whole day. But, moreover, some of the states themselves also represent particular - by humans identifiable - parts of a daily routine. In this model, all but one paths go through state 1 and end up in state 4. The only exception is the direct transition from initial to final state, which represents the weekends (and has a transition probability of 28.6%). Along with the following figures of particular daily routines state 1

can be interpreted as the morning until the first motion is detected and state 4 represents the evening after everybody already left the office (i.e. no more motion is detected).

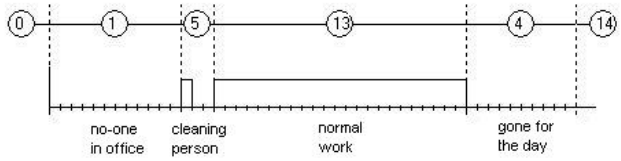


Fig. 6. A normal day in an office. The figure shows the Viterbi path through the model and the 48 averaged sensor values for that day. Dotted lines mark changes in states.

Fig. 6 shows a normal day in the "observed" office. One comment concerning the "sensor values": In this office the cleaning person comes every day in the morning to empty the wastebasket. We can see that state 5 covers a short motion followed by a longer "break" with no motion, temporally located in the morning. This state thus represents the cleaning person. Finally state 13 represents the period of constant activity during the day.

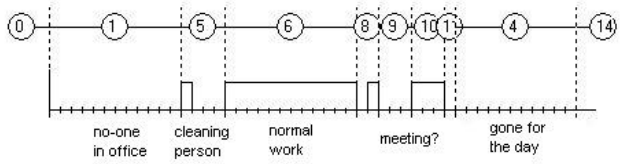


Fig. 7. A day with pauses in activity in the afternoon. Maybe a meeting?

The model also saw days with discontinuous appearances of motion. An example is given in Fig. 7, where the afternoon is divided into time frames with and without motion. Each of these time frames are represented by its own state. 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 thereof 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 states 1, 5 and 4 is as it was before, state 6 represents a "shorter" working day and states 8, 9, 10 and 11 cannot be interpreted by us because of missing particular knowledge, but could represent meetings.

## VI. CONCLUSION

This paper describes a test of the utilization of HMMs to model sensor values in an office building and the attempt of constructing symbols with semantic meaning, semantic concepts. The SCRS system can automatically build a model of various daily routines in an office environment. It does this by using a batch learning algorithm to create the model and an on-line mini-batch version during normal operation. While learning incoming value sequences - daily routines - are used to create a new path through the model or - if parts of it already exist - just to create a new smaller piece of a model. After a predefined number of examples are seen, some state merge heuristics are applied, that produce a model with a manageable number of states to be interpreted by humans.

We found that the system does create models with paths and states that do have a meaning. In some cases a state has a meaning that can directly be related to some particular event like the appearance and disappearance of the cleaning person, the whole morning and evening as well as state 13 that represents the normal working day. These states, or even whole paths like 1-5-13-4, can be seen as high-level semantic concepts, and can be easily interpreted by human operators. The learned semantic concepts divide the daily routine in the office environment into similar segments as would be intuitively used by a human operator.

In future work models with more than one sensor per model and hierarchical models will be investigated.

## VII. REFERENCES


[1] G. Russ, *Situation Dependent Behaviour in Building Automation*. Vienna, Austria: Institute of Computer Technology, Technical University of Vienna, 2003, dissertation thesis.

[2] C. T. Fuertes, *Automation System Perseption*. Vienna, Austria: Institute of Computer Technology, Technical University of Vienna, 2003, dissertation thesis.

[3] G. Pratl and P. Palensky, "Project ars - the next step towards an intelligent environment," 2005.

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York NY.: Oxford University Press Inc., 1995.

[5] L. R. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE ASSAP Magazine*, vol. 3, pp. 4–16, January 1986.

[6] M. Russell and R. Moore, "Explicit modelling of state occupancy in hidden markov models for automatic speech recognition," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Tampa, 1985, pp. 5–8.

[7] B. Wise and N. Gallagher, "The process chemometrics approach to chemical process fault detection and supervision," *Journal of Process Control*, vol. 6, no. 6, pp. 329–348, 1996.

[8] N. Pontoppidan and J. Larsen, "Unsupervised condition change detection in large diesel engines," in *IEEE Workshop on Neural Networks for Signal Processing*, C. Molina, T. Adali, J. Larsen, M. V. Hulle, S. Douoglas, and J. Rouat, Eds. Piscataway, New Jersey: IEEE Press, 2003, pp. 565–574.

[9] C. S. Hood and C. Ji, "Proactive network fault detection," in *INFOCOM (3)*, 1997, pp. 1147–1155.

[10] P. Goulding, B. Lennox, D. Sandoz, K. Smith, and O. Marjanovic, "Fault detection in continuous processes using multivariate statistical methods," *International Journal of Systems Science*, vol. 31, no. 11, pp. 1459–1471, 2000.

[11] J. E. Cook and A. L. Wolf, "Automating process discovery through event-data analysis," in *Proceedings of the 17th International Conference on Software Engineering (ICSE'95)*, 1999, pp. 73–82.

[12] J. M. R. J. F. Bowring and M. J. Harrold, "Tripewire: Mediating software self-awareness," in *Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS'04)*, 2004, pp. 11–14.

[13] A. Stolcke and S. Omohundro, "Hidden Markov Model induction by Bayesian model merging," in *Advances in Neural Information Processing Systems*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds., vol. 5. Morgan Kaufmann, San Mateo, 1993, pp. 11–18.