Behavior Learning via State Chains from Motion Detector Sensors

Dietmar Bruckner
Institue of Computer Technology
Vienna University of Technology
bruckner@ict.tuwien.ac.at

Brian Sallans
Programm- und Systementwicklung
Siemens AG Austria
brian.sallans@fin4cast.com

Roland Lang
Institue of Computer Technology
Vienna University of Technology
langr@ict.tuwien.ac.at

ABSTRACT

A method for the automatic discrimination of sensor and system behavior and construction of symbols with semantic meaning from simple sensor data 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 minibatch 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. The SCRS's method of representing scenarios can be also used in autonomous agents for decision making processes. An example is discussed.

Keywords

Hidden Markov models, behavior recognition, intelligent environment, intelligent sensor systems

1. INTRODUCTION

Embedded computer systems have seen their computing po-wer increase dramatically, while at the same time miniaturization and wireless networking allow embedded systems to be installed and powered with a minimum of support infrastructure. The result has been a vision of "ubiquitous computing", where computing capabilities are always available, extremely flexible and support, entertain, and assist people in their daily lives.

Up to now, advances in the electronics industry have driven growth in several areas like home entertainment, surveillance, home automation products (e.g. lighting, heating, and power), home appliances (e.g. laundry machines, fridges, etc.), and ad-hoc wireless sensor networks (AWSN), which promise to add a truly ambient intelligent component to the home.

The future home clearly represents an opportunity for the convergence of these different technologies far beyond the level of integration seen today (like e.g. proposed in [1]). In fact, the future smart-home will contain a collection of heterogeneous networked devices, capable of distributed computation, dynamic reconfiguration, high-performance media dissemination, and user/environment awareness.

Future building automation or assistance systems for care and security applications - which is the main focus of this work - will act based on their awareness of system status (see [10], [7]) 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 increasing 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 about what is happening. For example there could be different symbols for a normal daily routine in an appartment, or an afternoon interrupted by a visitor.

The goal of this work is to accomplish the task of reliably inferring simple semantic concepts from sensor data with probabilistic methods, without the need of pre-programmed conditions or user-entered parameters. 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.

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 [5] and software self-awareness [6].

The system is part of the ARS project (Artificial Recog-

Bionetics '07, December 10-13, 2007, Budapest, Hungary

nition System) [8]. It was tested in a building automation system consisting of various sensors installed in one appartment of a retirement home. The sensors consist of motion detectors, door contacts, and preasure sensors. The data was collected over a time period of two months. This paper presents the first results of this trial and suggests future areas of improvement and application.

2. SYSTEM STRUCTURE

The goal of the SCRS model is to automatically discriminate system behavior in a running dynamic system (in this case an appartment in a retirement home). It does this by learning about the behavior of the system and by observing data flowing through the system. The SCRS builds a model of the sensor data in the underlying system, based on the data flow. We use a set of statistical generative models (SGMs) [2] to represent knowledge about the system under consideration. A statistical generative model takes as input a sensor value, status indicator, time of day, etc., and returns a probability between zero and one. The model comprises not only statistical generative models describing the possible sensor values, but also a model of the underlying events that cause a change in system behavior. These underlying dynamics are modeled by a hidden Markov model (HMM) [9]. From this model, the system can identify recurring scenarios - patterns within the sensor values - with slightly varying sensor values. The system is also capable of launching an alarm in case of the occurrence of new scenarios or variations within scenarios with very low probability under the model. These could represent unusual or dangerous situations in an assisted living context (such as in the retirement home). Additionally, the HMM can be queried to deliver the most probable path through the model. In our case the most probable path can be interpreted as a similar (the best matching) situation compared to the current one, which has already been learned.

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 situations. 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 (mis-)behavior, (in our case, an unusual or dangerous situation). Third, the model can be continuously updated to adapt to sensor drift or to slightly changing operation conditions - for example, if the daily rhythm of the user changes with the season.

For the application described in this case study, HMMs were used with Gaussian models as emission probability distributions. It is not necessary to use a mixture of Gaussians for the emissions, because each Gaussian belongs to a certain state of the HMM. Under this point of view, the whole system behaves like a set of mixtures of Gaussians, but the priors of the mixture distribution - coming from the transitions - vary with respect to the past.

The SCRS in this application is used to model daily routines in an elderly home with the use of binary motion detector sensors. These values are then fed into the model and during a procedure of 5 steps (see also [4], [3]) the parameters of the model are updated:

1. Averaging motion information over 30 mins
2. Comparison of the chain's beginning/end
3. Merging of identical states
4. Merging of consecutive states
5. Merging of nonrelevant states

These steps ensure the creation of a HMM with a fair number of states. Too many states make the model very specific to particular situations and assign a very low probability for previously unseen situations whereas too few states correspond with a very general model structure that would not be able to give reliable information about the probability of the current situation.

3. HIDDEN MARKOV MODEL

There exists a variety of models for modelling a data source which is beliefed to obey the Markov property - that new values, 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 sections give an introduction into HMMs and their most useful algorithms.

3.1 Markov chain

The discrete time Markov chain of 1^{st} order has the following property:
$P(Q_{t+1} = q_{t+1}|Q_t = q_t, Q_{t-1} = q_{t-1}, ..., Q_0 = q_0)$
$= P(Q_{t+1} = q_{t+1}|Q_t = q_t)$, 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.

3.2 Hidden Markov model

Under some circumstances the process we want to model is not described sufficiently by modeling sensor values directly. 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. In this case the states cannot be directly observed, they are hidden. Each state has a probability distribution over some or all possible output symbols. 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 Π. 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.

3.3 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.

Forward algorithm.

Consider a problem where we have different models for the same process and a sample observation sequence, and we want to know which model has the best probability of generating that sequence. This task is accomplished by the *Forward Algorithm* in a recursive way.

Viterbi algorithm.

The Viterbi algorithm addresses the decoding problem. We thereby have a particular HMM and an observation sequence and want to determine the most probable sequence of hidden states that produced that sequence.
The solution is to define the Viterbi path probability, which is the probability of reaching a particular intermediate state, having followed the most likely path. The Viterbi path probabilities give us the probability for the best path through the model, but the aim is to find the best path and not only its probability. The solution is to remember the predecessor of each state that optimally provoked the current state, or in other words to store a back pointer for each intermediate state.

Forward Backward and Baum-Welsch 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.

4. MODEL STRUCTURE CONSTRUCTION

In this work we present a system that learns semantic symbols from a binary motion detector sensor. That sensor - a wireless off-the-shelf motion detector sensor - sends a data packet with value 1 and the sensor's ID in case of detected motion followed by a packet with value 0 and the sensor's ID right after no more motion is observed. In our elderly home environment every motion detector sensor produces about 500 data packets per day. During a procedure of 5 steps the structure of the model is learned:

1. Averaging motion information over 30 mins
2. Comparison of the chain's beginning/end
3. Merging of identical states
4. Merging of consecutive states
5. Merging of nonrelevant states

These 5 steps 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). [11] 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 our motion detector sensors.

4.1 Averaging motion information over 30 mins

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. An example of actual sensor data is given in Fig. 1. The chains of 48 values are then fed into the model and used to create states.

4.2 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

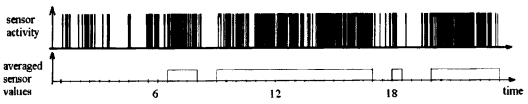

Figure 1: Averaging of sensor values within 30 min time frames. The averaged 48 values are used for further processing.

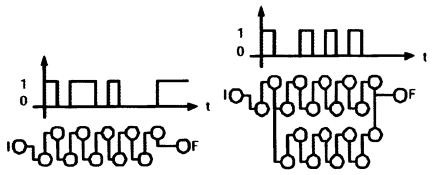

Figure 2: Comparison of chain borders. The top sequence of sensor values is mapped into a chain of states with appropriate emissions and transitions with 100% from state to state. The next sequence of sensor values is compared to the already seen emissions and in case of different values the model splits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

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.

4.3 Merging of identical states

It happens frequently 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. E.g. in the case of the averaged motion detector values a "1" means that the person in the room is quite active. If this is continued over the introduced borders of 30 min we can assume the behavior of the person is continued. The same arguments can be used for merging states with emission of "0". 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, hidden semi-Markov models must be used.

4.4 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 describing a behavior. Each splitting in a path is interpreted as a change in 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

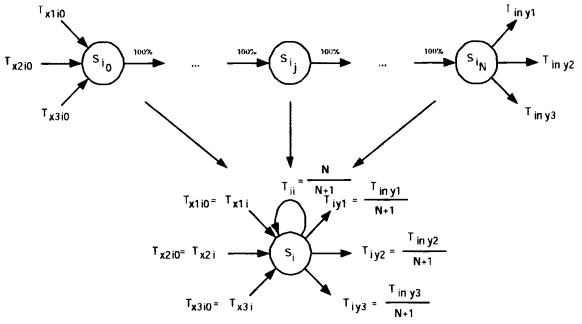

Figure 3: Merging of a state chain part with n identical states into one single state i. The original chain has unity transitions. x's and y's being other states in the HMM.

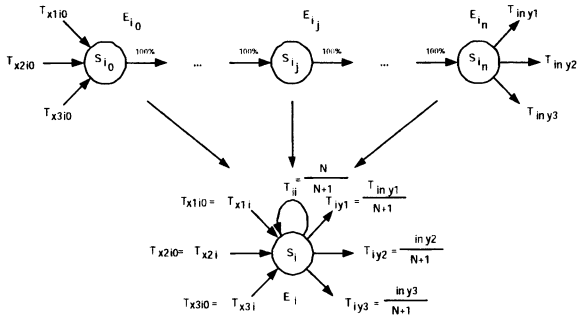

Figure 4: Merging of a state chain part with states with unity transitions in between into one single state i. The new Emissions E_i is computed as Emissions of original states times state's weight normalized to sum to unity.

chain can change their place with other values that happen often in the same time frame during a day. These considerations lead to the third heuristic: states with transitions of 100% are merged into a single state. In other words: if, after the merging of identical states, chains of states with unity transitions remain, then those chains are merged (see Fig. 4).

4.5 Merging of nonrelevant states

Finally, after a model is constructed with principles that compare the data, a last heuristic is applied that simply looks for states with low weight. Those states are probably not the result of earlier merging procedures neither part of often appearing scenarios. Therefore we consider them nonrelevant and merge them with neigbours. If the (nonrelevant) state should be merged with its predecessor or successor depends on the number of possible transitions from and to the state. Whichever number is equal to 1 is used as the connection to merge. If the state has more than 1 prede- and 1 sucessor it keeps untouched because it is considered as rather important splitting point in the model. In this application we used a weight of 3 as merging criteria. In this case it is very unlikely for such a state to have at the same time more than 1 prede- and 1 sucessor.

5. MODEL INTERPRETATION

In these models 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. Examples for states could be "lunch", "time before getting up", or "bathroom visit after lunch" and for (sub-)paths something like "afternoon with low activity".

5.1 Models

Figures 5 - 9 show the models of various motion detector sensors in the mentioned appartement. The drawings are to be read in the following way:

- The circles represent the states of the model. They are labelled with numbers from 0 to N, whereby states 0 is the "initial state" which is part of every chain and state N is the "final state", also part of each chain. These two states have no meaning, except that they "connect" the chains to form a comprehensive model.
- The lines represent possible transitions. Only the drawn transitions are nonzero and learned during the above mentioned procedures.
- The ellipses above some states represent states with nonzero self-transitions. This means that these states can be visited several times in a row.

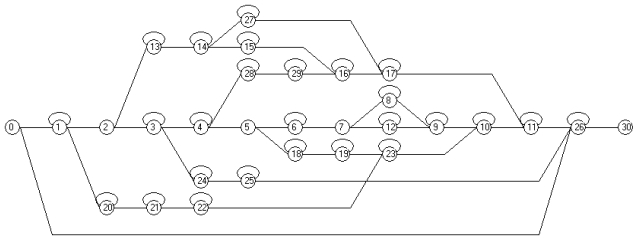

Figure 5: Model of a motion detector sensor located in the living room (right top corner); 29 states

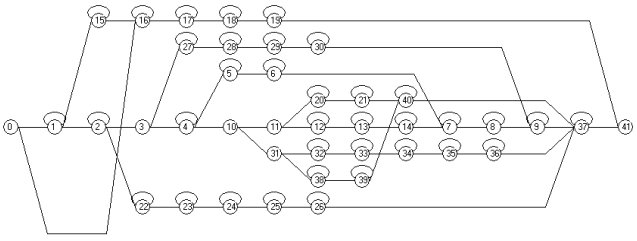

Figure 6: Model of a motion detector sensor located in the living room (left bottom corner); 40 states

5.2 Paths

The following figures 10 - 13 show 4 out of 16 different averaged sensor value chains of whole days that lead to the creation of the model in Fig. 9. The highlighted states mark the most probable path for the value chain computed by the Viterbi algorithm (see section 3.3). The diagram on bottom

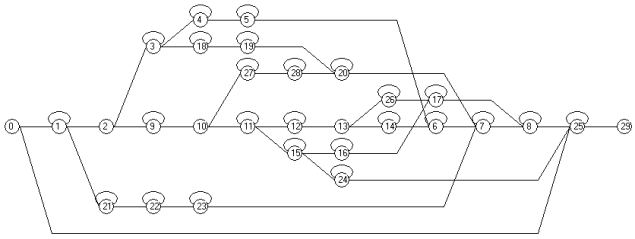

Figure 7: Model of a motion detector sensor located in the living room (right bottom corner); 28 states

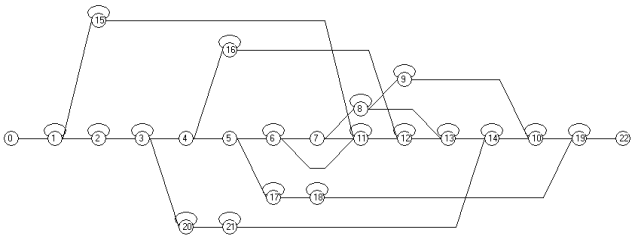

Figure 8: Model of a motion detector sensor located in the kitchen; 21 states

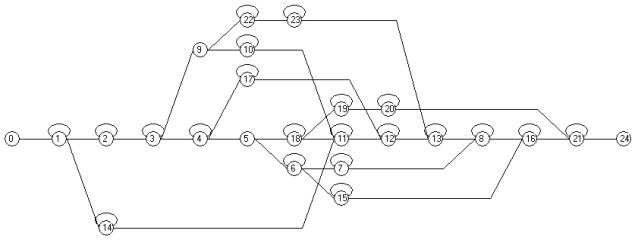

Figure 9: Model of a motion detector sensor located in the bathroom; 23 states

of the figures shows the log-likelihood of the observation over the Viterbi path, the value to the left bottom the overall log-likelihood of the corresponding observation. Table 1 summarizes the log-likelihoods of all the 16 observation sequences used for model structure learning of the model in Fig. 9. The second column gives the log-likelihood of the Viterbi path. These range from -17,4 to -29,8 (a rough calculation to get a feeling on these numbers is: if we assume emission and transition probabilities to be equal for all 48 values we can calculate $P_{Emission=Transition} = e^{\sqrt[96]{log(P)}} = e^{log(P)/96}$ which give 83% for -17,4 and 73% for -29,8). The third column gives the worst matching path through the model[1]. The log-likelihood of the worst paths ranges from -23,8 to -45,5 which translate to average transitions and emission probabilities from 78% to 62%. The last column gives the number of paths that have the possibility to produce the relevant observation sequence. These range from only 3 to all 8.

The numbers above the averaged sensor values in the fol-

[1]The trick how to get these figures was to find out where the actual Viterbi path splits from the other paths the last time and set this transition probability temporary to zero. E.g. in Fig. 10 this would be the transition from state 6 to 7. With this modification the Viterbi algorithm is applied again and gives the next best path.

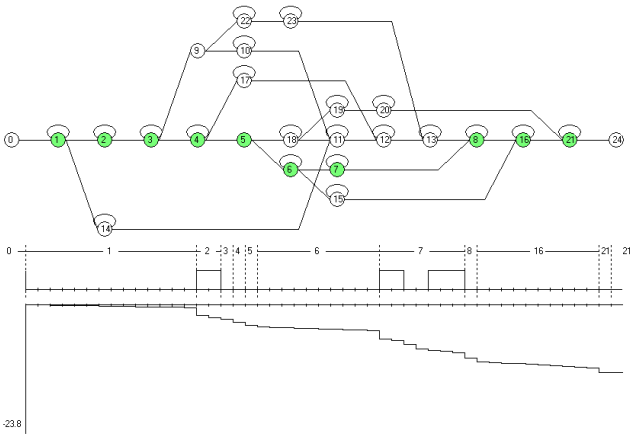

Figure 10: One particular day which was incorporated in the model. The colored states mark the Viterbi path of that day in the model.

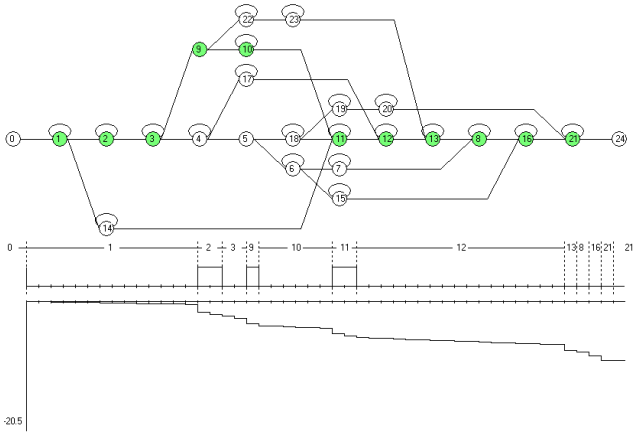

Figure 11: Another particular day for that model.

lowing figures depict the corresponding state in the model on the (x^{th}) Viterbi path. This information is given because the model itself contains no information about time - just implicit, in that states more to the right certainly are visited earlier than those to the left.
One important aspect arising from merging is that not all paths of the model will be Viterbi paths even of the value chains that created them. In the case of the here presented model all 16 value chains - the ones from which the model is learned - produce one of the four presented Viterbi paths.

What can be seen immediately from the Viterbi paths is

- All paths start with state 1 which represents the time until the first motion is detected. Because this motion detector "looks" at the shower and the area in front of it, we probably can deduce that the person is asleep in this state and the next state, 2, represents the morning toilet.
- The path on top in the drawings represents a day with rather much activity around the middle of the day in the bathroom.
- State 20 represents a rather long phase of no activity in the bathroom in the evening while in case of shorter

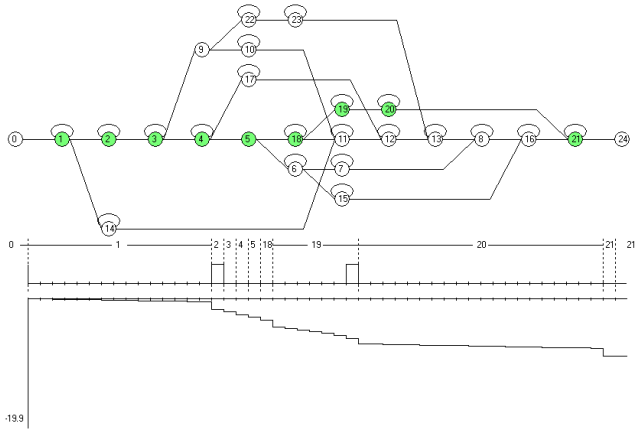

Figure 12: Another particular day for that model.

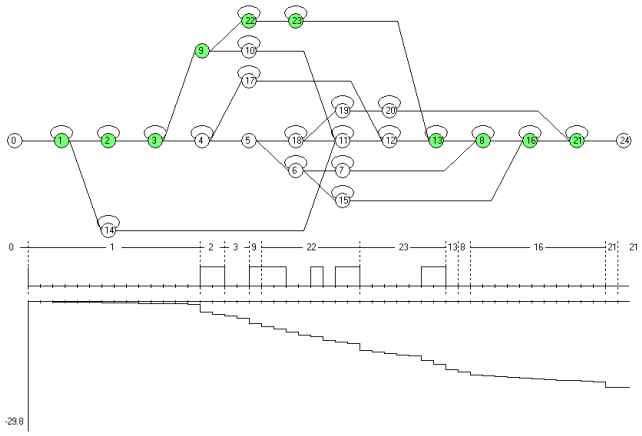

Figure 13: Fourth prototype day for that model.

Day	log-likelihood of best matching path	log-likelihood of worst matching path	number of possible paths
1	-23,8	-37,3	7
2	-20,5	-26,5	4
3	-19,9	-25,9	7
4	-19,5	-31,9	7
5	-17,4	-23,8	6
6	-25,9	-34,8	8
7	-22,6	-34,8	8
8	-29,8	-38,6	3
9	-23,7	-29,4	5
10	-26,5	-29,2	6
11	-27,2	-30,4	6
12	-23,1	-27,8	3
13	-23,3	-31,3	5
14	-28,7	-45,5	7
15	-22,4	-28,2	7
16	-25,1	-30,7	6

Table 1: Log-likelihood range and number of possible paths for the 16 observation sequences

periods state 16 is more probable(notice the flat progression of the log-likelihood in Fig. 12 while in state 20 compared to the "steeper" one in Fig. 10 while in state 16).

- There is still potential for optimization in the model. For example, the second half of the days drawn in Fig. 11 and Fig. 12 are quite similar, the difference of the two days lays in the forenoon. In such a case it would be of advantage to merge states 12 and 20 to have an even better representation of the behavior.

Finally, the figures 14 - 16 show 3 of the seven paths with the highest difference between best and worst matching path from above. The first one has a quite flat progression over the whole day without higher "steps" comming from transitions or emissions. The second takes the way over state 14, which has only low probabilities for representing "motion" but fair transitions. The third figure shows the worst matching observation sequence of the model under investigation. It takes the way over state 15 - which represents the time with no activity very well but lacks an emission probability for activity. From there the only way to the final state is via state 16, which also has low probability for missing activity, therefore the log-likelihood decreases dramatically.

6. CONCLUSIONS AND OUTLOOK

The work presented in this paper describes a test of the HMM approach for the automatic classification of daily routines with respect to already seen ones that serve as a kind of prototypes. The SCRS system can automatically build a model of situations during a pre-defined model constructing phase. During operation afterwards the model can be adjusted with each new value either to adjust the parameters in order to better fit the transitions to the ratios of real occuring scenarios, or to even introduce new states or paths in

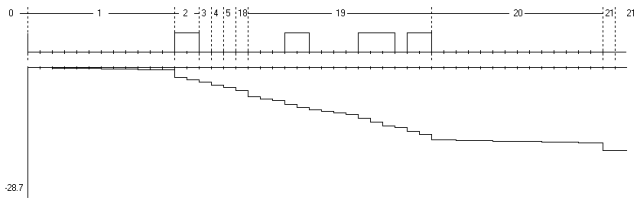

Figure 14: Day 8: Averaged sensor values and the correpsonding Viterbi path in that model with a log-likelihood of -28,4.

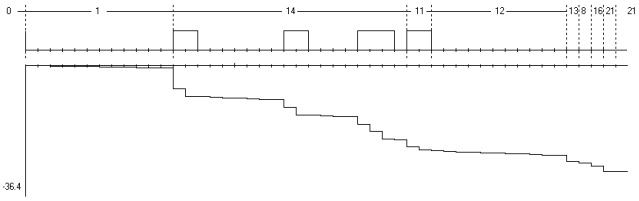

Figure 15: Day 8: Averaged sensor values and the correpsonding 6^{th} Viterbi path in that model with a log-likelihood of -36,4.

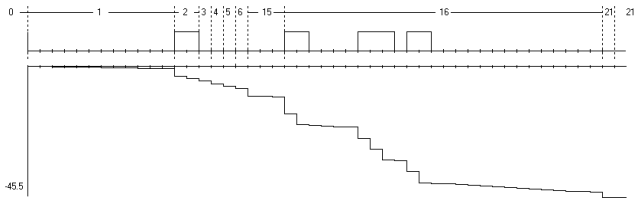

Figure 16: Day 8: Averaged sensor values and the correpsonding worst matching path in that model with a log-likelihood of -45,5.

the model in the case the current seen situations has a very low probability under the model to fit with the incorporated prototypes. Each path through the model represents a particular daily routine in the scope of a motion detector sensor and some of the created states can be labelled by a human.

This work shows that the SCRS is capable of differentiating between various daily routines on a top level view. The system has shown its ability to give a quantitative measure of the likelihood of how well the current situation fits the prototypes stored in the model. The quite similar structure of the various models of the motion detector sensors presented here give indications that it could be possible to fuse all or some of the models to one comprehensive model of the whole flat. Another envisaged future work is to introduce overlapping models which give a better initial (in the beginning of each day) guess on the path to choose in the model based on the last day.

7. REFERENCES

[1] C. R. Baker, Y. Markovsky, J. van Greuen, J. Rabaey, J. Wawrzynek, and A. Wolisz. Zuma: A plattform for smart-home environments. In *Proceedings of the 2th IEEE International Conference on Intelligent Environments*, 2006.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press Inc., New York

NY., 1995.

[3] D. Bruckner. *Probabilistic Models in Building Automation: Recognizing Scenarios with Statistical Methods*. Institute of Computer Technology, Technical University of Vienna, Vienna, Austria, 2007. Dissertation thesis.

[4] D. Bruckner, B. Sallans, and G. Russ. Probabilistic construction of semantic symbols in building automation systems. In *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, 2006.

[5] 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)*, pages 73–82, 1999.

[6] 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)*, pages 11–14, 2004.

[7] G. Pratl. *Symbolization and Prozessing of Ambient Sensor Data*. Institute of Computer Technology, Technical University of Vienna, Vienna, Austria, 2006. Dissertation thesis.

[8] G. Pratl and P. Palensky. Project ars - the next step towards an intelligent environment. In *IEE IE 2005*, 2005.

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

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

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